CSE 503: Program Analysis
Winter 2014

Syllabus

Course website: `http://www.cs.washington.edu/503/`

Class meetings: Tue & Thu 10:30-12:00, room CSE 403

Lecturer: Michael Ernst
`http://www.cs.washington.edu/homes/mernst/`
Office hours: TBA
Even outside the official office hours, I welcome discussions with students. Please don't be shy about contacting me if you wish to meet!

TA: Darioush Jalali

1 Structure of the course

In many years CSE 503 is a broad course that covers all aspects of software development. This year's offering is more narrow but deeper.

For a list of specific topics that may be covered, see the course webpage.

2 Grades

Grades will be assigned based on the project (80%), class participation (20%), and instructor discretion (variable). As this is a graduate class, the class is likely to be A-centered, but students are not guaranteed a grade of A.

3 Papers

CSE 503 is a graduate paper-reading seminar. Each class session will begin with a brief discussion and presentation of material, after (or during) which the floor will be open to rebuttals, discussion of related work, criticism, brainstorming about follow-on research, etc. We will all learn from the conversation, even the instructor. We will all have questions and confusions about the material, even the instructor. You are required to be not a a passive listener to lectures but an active participant in the discussion.

To help you prepare, you will write a one-paragraph commentary on each paper, and submit it at least 24 hours *before* the class meets to discuss the paper. You will post your commentary to the course webpage for viewing by the instructor and by other students. The commentary should reflect your understanding and analysis of the issues raised by the paper, and should also help direct (both your and others') preparation for in-class discussion.

You may write the commentary in whatever style you prefer that meets the goals listed above. One good format for the commentary is to critique the paper, listing the following three points: its biggest contribution (and, briefly, why that result was not already obvious), its biggest mistake (in motivation, methodology, algorithm, data analysis, conclusions, or some other area), and the biggest question that it raises (or the most interesting and important follow-on work that it suggests). Another acceptable format is to summarize the paper, describing its thesis, approach, and conclusions, and stating why it is significant. The commentary should also list questions that you have about the paper, such as about technical points or connections to related work. For other ideas about how to critique a paper, see `http://www.cs.washington.edu/homes/mernst/advice/review-technical-paper.html`.

It's OK if you read the paper and there are issues you do not understand. Please ask questions about those issues — both in your summary and in class — and we will all gain by the discussion. It's best to explain *why* something makes no sense to you. For example, don't just say, "I didn't understand section 2", but state where there is a logical

fallacy or a conclusion that does not follow or a term that is not defined. The lecturer will use these questions to help shape the lectures.

I also encourage you to use the summaries to ask questions. If you have a question, it is likely that many other people have the same question but are too shy (or vain, or insecure) to ask it; they will appreciate your raising the point. However, do come to class prepared: carefully read the paper and get as much as you can out of it on your own. Doing so will make the class time that much more productive.

You will have access to all the other students' submissions. Please read them, because reading the other summaries is a good way for you to get perspective. You can see what you missed (or what other people missed), and whether you agree with their take on the key ideas. It will help to make the class sessions more productive.

Please submit your summary in PDF form, and include your name.

Each student will present one paper during the semester. The presentations will be done by pairs of students if there are enough students registered. Presenters will meet with the lecturer a week before the class meeting to receive feedback and improve their presentation. You should come to that meeting prepared with questions, not just expecting to run through all of your slides. The goal of your presentation should be that the class understands the main ideas afterward. (The goal of your presentation is *not* to prove that you know the material!)

4 Projects

The centerpiece of the course is a semester-long project, done in groups of 2–4 students. The project should make a research contribution in the area of tool support for programming tasks. As a research contribution, it should answer a question whose answer no one previously knew.

A non-exhaustive list of types of acceptable projects are:

- proposing and evaluating a fundamental new technique
- developing and assessing new algorithms to replace currently-used ones
- translating a methodology to a new problem domain
- applying known techniques to new problem domains, such as operating systems, networks, embedded systems, security, biology, aerospace, etc.
- evaluation of existing techniques or tools, via case studies or controlled experiments
- port an existing tool to a new domain (e.g., a new programming language or a new version of one)
- implementation of a proposed but never implemented technique

A list of specific potential projects will be distributed at the second class meeting. (It is intentionally not being distributed until after you have completed Assignment 1.) You may select (or modify) a project from among the list of suggested ones, from your answers to Assignment 1, from other students' answers to Assignment 1, or from other sources. The lecturer will help you choose, refine, and scope a project.

For instance, you might start from a programming problem that has vexed you and devise a solution (or demonstrate that it is a broader problem than previously recognized). You might take a paper of particular interest to you and attack one of the problems listed in its future work section. You might take a promising paper that lacks experimental evaluation, or whose methodology is flawed, and perform a proper experimental evaluation.

You should also consider building on your strengths and finding ways to leverage other work that you are doing. For instance, if you do work in AI, you might consider how machine learning could be applied to problems of program comprehension or to filtering output from program analysis tools. You also might consider using a project that you are working on, or that you worked on in the past, as a test case for a tool. Synergies with your other work are welcomed!

Your final project report should be written in the style of a conference paper, and you will present it in a CSE 503 "conference" at the end of the semester in the final exam slot (Thursday, December 15 at 10:30am). You are not required to submit your work to a conference or workshop, but you might choose to do so — this is the standard of work expected, and which the instructor will help you achieve. Please use the ACM proceedings style

(http://www.acm.org/sigs/publications/proceedings-templates), but *not* the tighter alternate style since the class does not impose a page limit.

You need not close the book on the area of research you choose, but you should do a good job of discovering and clearly communicating new knowledge in that area. Do not feel overwhelmed by the final project or final report. The instructor will meet with you repeatedly for discussions, feedback, and assistance on all aspects of your project. For tips about writing a technical paper, you should read (among other resources) http://www.cs.washington.edu/homes/mernst/advice/write-technical-paper.html.

4.1 Timetable

For exact dates, see the class calendar.

Week 2: Proposal

Select a group, select a project, and write a short (1–3 pages) project proposal that includes its thesis and technical approach.

Be sure to motivate your project. Your paper should start with the problem. You also must give a use case — essentially, a story about how the tool is used. For example, a software developer wants to perform a particular, realistic task, but cannot. Be specific about the problem and about why current techniques do not address it. Your tool solves that problem. Indicate exactly what inputs the developer must supply, exactly what the developer gets as output, and how that is useful in practice. Include an architectural diagram of how the pieces of your system fit together.

Include evaluation criteria. The philosopher of science Karl Popper states that a theory is scientific only if it is falsifiable — if some experiment exists that would disprove it. You need to clearly state an experimentally refutable thesis or hypothesis. You should also state your evaluation metric, which should be, at least in part, quantifiable. Another way to think of this is: how will you convince a skeptical reader that you succeeded? They aren't going to just take your word that your project was a success.

Clearly state the scientific question or contribution. For example, how is your project an (incremental) advance over previous work? It's good (and in some cases sufficient) that it enables developers to do something they have never been able to do before. But you can also extend a technique to a new domain, or expand a problem statement. And ideally, your results will be transferable to other domains: they will change the way that others think or act in the future, or provide guidance to programmers or to tool developers or both. Oftentimes, much of your project is not novel. That's OK — engineering is necessary in order to do experiments or to build on previous work — so long as you can point out the part that is novel.

Week 2: Approval of proposal

Meet with the instructor for feedback on and approval of your project proposal.

Week 5: Related work and methodology complete

Submit a version of your report that includes the introduction, description of the technical approach (e.g., algorithmic details), related work, and experimental methodology (i.e., evaluation plan). These sections should be in final form: you would be happy to have this text appear in your final paper without any subsequent editing.

It is okay if this description changes as you discover flaws in your initial ideas about the methodology, subject programs, and so forth. However, it is not acceptable to simply skip this assignment by submitting something that is vague or incomplete. The purpose of this milestone is to make you think about these issues and to enunciate them in time to get feedback from your teammates, from yourself when you see what you have written, and from the instructor. Working hard on this assignment will save you a lot of time in the long run.

Your experimental methodology section should not contain vague descriptions or missing aspects of the methodology. It should specify specific subject programs, experimental protocols, specific tasks for any user evaluation, specific metrics and how they are computed, etc. The paper should contain enough detail that any reader could replicate your results, without the need for guesswork or non-trivial design decisions.

The evaluation plan should also justify your choices by explaining why your chosen metrics are the right ones, and what insights they reveal about your work. (For example, do they address the novel part of your technique, or are they end-to-end measures? Can they be directly compared to previous work, or not?) It should also indicate your criteria for success. (You should lay those out now, so that you have an objective measure when the research is complete.) This version of the paper should include all the tables or graphs (if any) that will appear in the final paper, and the structure of any textual sections of the evaluation. The tables or graphs will be empty for now, but you should know what they are.

You should have already divided the work among members of your group.

You should be able to reuse most of the text from your proposal, but you will also have to augment it. Each version of the paper should address all previous comments from the course staff.

Week 8: Code complete and initial results

Submit a version of the report that includes partial results. This draft should also include a description of the interesting parts your implementation — enough information to enable a reader to re-implement it.

If you are building a tool, you should have a prototype implementation that can run correctly end-to-end on a small example program. (For example, a compiler might be able to compile the factorial program.) Having a working implementation now will enable you to enhance it and to run experiments during the next several weeks; if you do not even yet have working code yet, you are unlikely to be able to complete a quality project in time.

As a rule of thumb, by this point you should have one row or column in each table filled in. Or, if you are doing a case study or qualitative experiment, then you should have completed a pilot study. You will find that, besides these results and the implementation description, your document did not change much in this submission. Be sure to also address previous comments and to reflect any changes in your methodology.

Week 10: Final report

Final exam week: Project presentations

You should plan to talk for 20–25 minutes, followed by 5–10 minutes for questions (though the questions may come during rather than after the talk). As with any talk, be sure to practice it before you present in class. Be sure to include plenty of examples! (Even after receiving this advice, most groups fail to include enough to make their ideas comprehensible.) For more tips, see `http://www.cs.washington.edu/homes/mernst/advice/giving-talk.html`.