Mini-Project: Tool or Analysis Practicum

17-654: Analysis of Software Artifacts
Jonathan Aldrich (jonathan.aldrich@cs.cmu.edu)

150 points total

The goals of this mini-project are to gain an in-depth practical experience with an analysis tool or technique and reflect on the experience.

The expected scope of effort for this project is around 14 hours per student; plan your project accordingly.

Groups. Students may work on this project in groups of 4-6 (smaller groups are permitted, with scaled expectations, but should discuss this with the instructor). Group projects will be given a single grade. You are free to choose your own groups, subject to the constraint that the instructors reserve the right to assign groups in the case that some students are unable to find partners. Expectations for the project will be scaled (within reason) to the size of the group.

Collaboration Policy. Since different groups will be working on different projects, the only collaboration policy is that your work must be your own (as always).

1 Presentations

Each group will prepare a presentation approximately 8 minutes long. The presentation should describe the tool or analysis technique if it was not presented in class; describe how the tool or technique was applied, describe qualitative and quantitative data gathered, give a summary of lessons learned, including the benefits, drawbacks, and scope of applicability of the tool or technique.

Presentations will be worth 50 points. You will be graded both on content (what you did) and clarity of presentation. You may use the Powerpoint template provided on the course website, or your own.

2 Project Reports

The final 100 points will be based on the project report. Grading will be based both on content and clarity of communication. The paper should be around 4-5 pages of text (diagrams, screenshots, examples will likely take extra space), but this is a general guideline only; clarity is more important than length. The contents of the report are discussed in each section, below.

3 Tool or Analysis Application and Evaluation

Choose an analysis tool from the list available on the course website, or an alternative tool or relevant manual analysis technique by agreement with the course instructor. Apply the tool or technique to one or more realistic programs. Focus on assessing the strengths and weaknesses of the tool or technique, both in quantitative and qualitative terms.

The assessment must be written with other members of the class as the intended audience. The writeup should briefly describe the tool or technique, describe the experimental setup (for example, how was the tool or technique applied and to what subject), and describe both qualitative and quantitative data gathered in the experiment (some of the requirements for reporting are listed below, but you should be creative and report additional data that you see relevant as well). Based on your experience, discuss the lessons you learned, including the benefits, drawbacks, and scope of applicability of the tool or technique.

The course website includes examples of model tool evaluations done in the past, to give you an idea of what is expected. MSE students are encouraged, but not required, to apply an analysis tool or technique with

relevance to their studio project; your Bored Games application is another good potential target.

You must report the following:

1. How you customized the tool (or would recommend customizing it based on your experience). For example, if your tool reports different categories of warnings, which categories would you or did you turn on or off? Justify your choices.

2. For each project, how many true positives the tool found which were relevant to the project (ideally after customization). Across all projects, give two concrete examples of the warning and relevant code, and an explanation of what the error was and why it was an issue (i.e. two total examples, not two per project).

3. For each project, how many true positives the tool found which were not relevant to the project. Across all projects, give two concrete examples of the warning and relevant code, and an explanation of why it was a technically correct warning but was not of interest in the project.

4. For each project, how many false positives the tool reported. Across all projects, give two concrete examples of the warning and relevant code, and an explanation of why the warning was a false positive.