University of Washington
CSE 403: Software Engineering
Reading Summary Description and Sample Reading Summary

Each summary should contain the following:

- a topic sentence that states the name, author, and chapter(s)/section(s) (if applicable) of the reading
- a paragraph containing the main idea and several of the most important points from the reading (be specific! cite specific examples or ideas used in the paper)
- a paragraph of your own analysis of the paper (Does the author present valid points? Do you agree with the paper? What criticisms do you have, or what changes would you make? How does this material relate to what has been covered so far in class?)

The summary should be typed. It should occupy no more than one page (expected length: approximately 1/2 to 2/3 page). Assume that the reader of your summary is a competent computer scientist who has *not* read the article before.

Summaries will not be accepted late. The summaries will be graded on the following four-point scale:

4: All main concepts and skills mastered and all major questions answered, with only minor errors.
3: Important points made, but contains some significant omissions or errors.
2: Substantial missing concepts or errors.
1: Effort shown, but not a significant amount of relevant or correct content.
0: Not turned in, or almost no effort or understanding demonstrated.

Sample Summary:

The following document could be turned in as a reading summary for this course:

Billy Jones (bjones@cs.washington.edu), CSE 403, Winter 2048 (Instructor: Smith)
Reading Summary #1

In "Rules about Copying and Sharing Java Code," author Josh Smith believes that code copied from others should be cited as such, otherwise it is plagiarism. Another important idea that Smith discusses is that most discussions of plagiarism are with respect to "works in written and spoken language", and hence he wants to discuss how to cite the work of others within computer programs. He supports this latter idea by specifying that "due credit" is given to others by specifying the original author, the source where the code was obtained, and any alterations that the current author is making to the original code. The author provides examples citations whose source is from a textbook, an instructor, the Internet, from multiple sources, and from code that is "common knowledge" in order to show how one can always clearly identify the author of each code unit in a variety of situations. Another important point made by Smith is that code should never be transferred between students electronically, because this would imply unsuitable sharing of work and plagiarism.

Smith's target audience is computer science students, as it is likely that either they are unaware of plagiarism in general, or they are aware of plagiarism in other fields but have not considered how it applies specifically when writing code. This material relates to the current course material because it comes after the design process and during the implementation process, when the most code is being written and would be most available for potential copying. Smith's guidelines for copying and reusing code are accurate and useful; however, he forgets that sometimes a great deal can be learned by examining code written by others. It would have been nice if he had left some provision where it was okay to do this under the right circumstances.

Summary Grading Notes:

The following are common errors students mistake in writing their summaries that can lead to score deductions:

1) Meta-summarizing the paper without going into detail ("MS"):

   "After briefly introducing the topic, the author describes the pros and cons of various approaches to project management."

   Like what?

2) Regurgitation without showing evidence of deeper understanding ("REG"):

   "And then the author says... and then... and then..."

3) Excessive quotation ("QUOT"):

   You aren't required to back up all your claims with quotes. A few are okay, but you don't need to back up every statement with a lengthy quote. In fact, you often shouldn't.

4) Bad balance of specific and general ("DETAIL"): e.g. If the paper is about lifecycle models, writing great detail about only a few models, but ignoring the rest; or only discussing general principles of model application.

5) Vague critical analysis ("CRIT"):

   "In general I thought it was pretty good. Sometimes certain statements were a little unclear, but it didn't really detract from the argument he made. ...."

   Why was it pretty good? What was unclear? What argument did he make?

6) Criticism with no supporting evidence or logical backing, etc. ("RANT"):

   "The study of this kind of thing is just a waste of time. Every argument boils down to obvious project traits and downright common sense."

   "This class is stupid!"