Software Engineering
PhD Qualifying Examination
March 18, 2011

Rules:

- Answer six of the following nine questions including questions 1 and 2.
- Include citations for relevant literature where appropriate, including items from the reading list.
- Include a copy of the question with each of your answers.

1. Your Research Interests – You must answer this question

a) What area of research most interests you and why?
b) Describe three important problems in the area along with the current state-of-the art addressing those problems.
c) What is the state of the practice for those problems—i.e., how do practitioners currently deal with the problems?
d) List five people that are top researchers in your area and discuss, for each of them, what was his/her contribution to the field, why that contribution is important, and how it improved both the state of the art and the state of the practice.
e) What problem or problems in the area do you currently plan to pursue, and what contributions do you plan to make?

2. Research Impact – You must answer this question

In *No Silver Bullet*, Brooks distinguishes accidental and essential complexity.

a) What did Brooks mean by these terms? Give an example of each and justify your answer.
b) Is this a helpful distinction for dealing with the problem of complexity? Describe at least two alternatives ways to think about complexity.
c) Give three important Software Engineering advances that have significantly mitigated accidental complexity in actual practice. Justify your choices.
d) What is the single most important Software Engineering advance that has significantly mitigated essential complexity in actual practice? Justify your answer.
e) Propose at least two other research directions for tackling essential complexity. For each describe at least three research studies you would propose for exploring, elaborating and validating that direction.

3. Testing

a) Types of testing
    i. Random testing
        01. Define random testing.
        02. Discuss the advantages and disadvantages of it for unit testing, integration testing, and systems testing.
    ii. Nonfunctional testing

01. Define three types of testing that are nonfunctional.
02. Select two of these and discuss how you would design a testing strategy to test these nonfunctional properties.

iii. Integration testing vs. beta testing
01. What are the differences between integration testing and beta testing?
02. Suppose the company for which you are doing the testing decides not to do integration testing but to do beta testing instead. What would be the advantages and disadvantages of this approach?

b) Testing in the cloud
i. Describe one existing testing strategy that you think will be appropriate for testing cloud software. Justify your answer.
ii. It is assumed that there will need to be new strategies for testing in the cloud because of the new paradigm. Describe one area for which a new strategy will be needed, discuss why you think it is needed, and propose an approach for developing it.

4. Requirements

Jackson, van Lamsweerde, and Gunter, et al. provide definitions and discussion of requirements engineering and requirements.

a) Compare and contrast the definitions provided by each of these three papers for requirements engineering and requirements.
b) Describe two types of analysis that are used to go from the physical system being modeled to the new system to be created in software.
c) What are two current methods of producing high-quality specifications?
d) In the system/software-development process, requirements engineering is often thought to be the most important phase.
i. If this is true, why are high-quality requirements not written more often?
ii. Discuss the advantages and disadvantages for the entire software-development process of requiring high-quality specifications.
e) Select four properties of software specification. For each of them
i. Explain what it means.
ii. Describe how that property can be achieved in the specification of the requirement.
f) Given an example of a type of system for which high-quality specifications are usually written, and describe how such a systems can be kept current when the system evolves (i.e., is changed).

5. Experimentation

Tichy, in his *IEEE Computer* paper, answers the question he poses: *Should Computer Scientist Experiment More?* Instead of the broad category of computer science, these questions will concern Software Engineering.

a) Do you think that experimentation applies more strongly to Software Engineering than to computer science in general? Justify your answer.
b) Select two recent and well-known techniques that have been proposed for Software Engineering (you can select these in the area in which you are working).

These should be well-known techniques for which there appears to be confidence in their effectiveness (in the Software-Engineering community), not obscure techniques or those that were recently developed, and for which there has been little time for experimentation. For each of these two techniques, answer the following:

iii. Describe what experimentation has been performed for the technique.
iv. Discuss whether you think the experimentation is sufficient to show that it has desirable characteristics--effectiveness, efficiency, usefulness, etc.
v. If the experimentation is insufficient, describe what additional experimentation needs to be done to show that the technique has the desirable characteristics.

c) Researchers have often referred to sets of subjects that they use for experimentation as “benchmarks". Tichy gives a definition for benchmarks.
   i. Do you consider the Siemens suite to be a benchmark? If so, explain why and for what types of experiments. If not, explain why not.
   ii. What would you propose as the required characteristics of a benchmark for performing automatic test-case generation?
d) You have written at least one paper on a technique that you have developed.
   i. Describe the technique and the experimentation that you performed to validate it, including subjects, implementations, and results.
   ii. How does (or does not) your experiment adhere to the characteristics of good experiments that Tichy describes? If it does, justify it. If it does not, what additional experimentation would need to be done?

6. Modeling

Models are frequently used to specify and communicate the structure and behavior of software systems, and a variety of notations have been devised for expressing them.

a) List and describe at least three previously existing and used modeling notations. At least one of these should be rigorously formal, at least one of these should be informal, and the third should be somewhere in between.
b) For each modeling notation, list factors that would encourage you to use that notation. Factors may concern the nature of the project, the people working on the project, etc.
c) Describe how the two goals of specification and communication play off against each other. That is, in what way does a good modeling notation aid with both, and how do the two goals compromise each other?

7. Responsibility

As software systems become ubiquitous, they take on more decision-making responsibility.

a) Give an example of a currently deployed system that makes important daily decisions that affect the lives of (at least) thousands of people. Describe its scope and the kinds of decisions that it makes.
b) What tangible and specific extra steps should you as a Software Engineer take to ensure that the decisions the system is designed to make are in the best interests of the people effected? Justify your answer.

c) What tangible and specific extra steps should you as a Software Engineer take to ensure that the implemented system makes the specified decisions correctly? Justify your answers.
d) What future societal or technical factors do you foresee that would prevent or discourage computer software and hardware systems from taking over a greater and greater share of societal and personal decisions in the future? Justify your answer.

8. Software Architecture

a) Your manager approaches you and tells you that the company has standardized on UML as its software-modeling notation and, for consistency, will not be using any other notations. Is this a good or a bad idea, in your opinion? Justify your answer with at least three well-thought-out reasons.
b) How can modeling use cases influence the design of your architectures? What particular properties can doing this help you realize about your architectures?
c) Similarly, discuss how architecture limits or enhances the ability to model certain use cases. To illustrate this point, explain how certain use cases are not possible with a client–server architecture and how some use cases are naturally best captured in a client–server architecture.

9. Symbolic Execution

Symbolic execution, a technique that was introduced more than 30 years ago, is becoming increasingly popular in recent years.

1. Discuss what are the factors, in your opinion, that led to this renewed interested in symbolic execution.
2. Despite improvements in symbolic execution techniques, symbolic execution is still an approach with many limitations. Do you agree with this statement? If so, what are these limitations, what causes them, and what are their consequences?
3. Which limitations of symbolic execution would have to be eliminated for it to be able to generate inputs that can cover any branch in any program?
4. Dynamic symbolic execution (DSE) techniques, such as CUTE, aim to improve traditional symbolic execution. What issues of symbolic execution are DSE techniques trying to solve? How?
5. Provide an example of code, and target in such code (i.e., a branch), for which DSE would be able to successfully generate inputs, whereas traditional symbolic execution would fail. (Do not use examples from the CUTE paper or from other papers on DSE.) Can you provide an example of the opposite situation?
6. Provide another example of code, and target in such code, for which DSE would not be able to successfully generate inputs and explain why this is the case.