*Florida Institute of Technology*

Department of Computer Sciences

# Comprehensive Exam (Fall 2008)

# SOFTWARE ENGINEERING

**Friday, October 24th, 2008; 10:00am – 11:30am**

## Instructions

- Write the last four digits of your student identification number in the space below.
- This exam consists of 16 pages (including this cover).
- Answer any four (4) of the seven (7) questions. Each question has equal value.

  ➔ **Circle the four questions that you want graded:**

  1 2 3 4 5 6 7

  (If you leave this blank, Questions 1 - 4 will be graded.)
- Use a pen to write your answers in the space provided.
- When a question asks you to "describe," "discuss," or "explain" something, it means you must provide a convincing, clear, and reasonable answer; simply stating a fact without any supporting argument is insufficient.
- No study aids (notes, books, etc.) are permitted during the exam.

Good luck!

**ID Number:**

## For Grading Use Only

| Question | | Worth | Grade |
|---|---|---|---|
| 1. | Requirements | 25 | |
| 2. | Design | 25 | |
| 3. | Construction | 25 | |
| 4. | Testing | 25 | |
| 5. | Maintenance & Evolution | 25 | |
| 6. | Management | 25 | |
| 7. | Process | 25 | |
| | **Total** | **100** | |

## 1. Requirements (25%)

Getting software requirements right is notoriously difficult. One of the main problems is getting everyone to agree to the same thing. In other words, developing a common understanding of the problem, from both a user (requirements definition) and an engineering (requirements specification) perspective.

**Problem**:

(a) Describe three techniques for representing requirements specifications.

(b) Give an example of two of the three techniques for the same hypothetical system.

**Grading**: (a) 5% for each clear explanation (15%); (b) 5% for each example (10%).

**Note:** Use the blank sheet of paper on the next page as needed.

## 2. Design (25%)

Design is a key component of the overall software lifecycle. Good design contributes to the construction of elegant and bug-free software. During high-level design, control models are created to represent the control flow between sub-systems. To work as a system, sub-systems must be controlled so that their services are delivered to the right place at the right time. Structural design models should not include control information. Instead, the software engineer should organize the sub-systems according to some control model that supplements the structural model used.

**Problem:**

(a) Describe the *centralized control* design model. Clearly identify the advantages and disadvantages of this approach. Given an example of centralized control for a hypothetical system.

(b) Describe the *event-based control* design model. Clearly identify the advantages and disadvantages of this approach. Given an example of event-based control for a hypothetical system.

(c) Which of these two models would you choose for a batch processing system that takes information about hours worked and pay rates and prints salary slips and bank credit transfer information? Clearly explain why. Be sure to document the assumptions underlying your answer.

**Grading:** (a) 10%; (b) 10%; (c) 5%.

**Note:** Use the blank sheet of paper on the next page as needed.

## 3. Construction (25%)

Linked lists are one of the essential data structures used in software construction.

**Problem**: Consider a doubly-linked list with alphanumeric data values in its nodes:

```
struct Node {
  int value;
  Node *prev, *next;   // assume circular
};
```

(a) Construct an elegant and efficient **recursive** C/C++ function that takes such a linked list and returns the number of nodes in the list with data values that are lowercase letters.

(b) What is the runtime performance of your recursive solution from part (a) in O(*n*) notation? Be sure to explain your answer. Document all assumptions.

(c) Redo part (a) but make your C/C++ function **iterative**.

**Notes**:

- Make sure your solution is constructed clearly and idiomatically, so that it adheres to the commonly accepted definition of good coding style.
- Your solution cannot make use of the Standard Template Library (STL).
- Be sure to properly comment your program, explain how the solution works, and why you selected the particular algorithm(s) and data structure(s).
- Include defensive programming techniques in your solution where appropriate.
- Use the other side of the paper as needed.

**Grading**: (a) 10%; (b) 5%; (c) 10%.

## 4. Testing (25%)

Software testing plays a key role in ensuring overall product quality and effectiveness. One of the difficulties in testing is knowing when to stop. While one would like as complete a testing procedure as possible, like all software engineering activities, testing must take place within a realistic context of limited resources.

**Problem**:

(a) Explain the definition of "coverage" in software testing.

(b) Describe three (3) different types of coverage that can be measured. For each one, describe a type of bug that you would be certain to find with this type of coverage and describe a type of bug that you might miss even if you achieved 100% of this type of coverage.

**Grading:** (a) 7%; (b) 6% for each description (18%).

**Note:** Use the blank sheet of paper on the next page as needed.

## 5. Maintenance & Evolution (25%)

Evolution is an important activity of the software lifecycle. In the 1980s, Lehman and Belady examined the growth and evolution of a number of large software systems. Based on their measurements from these studies, they proposed a set of five hypotheses, known as "Lehman's Laws," concerning system change. They claim that these "laws" are invariant and widely applicable.

**Problem:** Describe all five of Lehman's Laws. Include commentary on the validity of each of these laws as applied to today's software-intensive systems.

**Grading:** 5% for each description and commentary.

**Note:** Use the blank sheet of paper on the next page as needed.

## 6. Management (25%)

Estimating the cost and effort required for a particular task in a software project is an important management activity.

**Problem:**

(a) Describe two *classes* of productivity measures that are commonly used to aid cost and effort estimation. Note that in this context, a *class* refers to a generic category of productivity measure, not to a specific type.

(b) Give specific examples of each class of productivity measure that are commonly used in practice.

(c) Comment on the advantages and disadvantages of each example.

**Grading:** (a) 5% for each class description (10%); (b) 5% for each example (10%); (c) 5% for a discussion of the relative advantages and disadvantages of each example.

**Note:** Use the blank sheet of paper on the next page as needed.

## 7. Process (25%)

Of the various software process models that have appeared in the literature, it can be argued that the Software Engineering Institute's "Capability Maturity Model for Software" (SEI SW-CMM®) has had the most impact for large organizations.

**Problem:**

(a) Describe the SW-CMM. Explain each level. Provide a diagram.

(b) Discuss the advantages and disadvantages of this process improvement model.

**Grading:** (a) 15%; (b) 10%.

**Note:** Use the blank sheet of paper on the next page as needed.