Florida Institute of Technology

Department of Computer Sciences

Comprehensive Exam (Fall 2007)

SOFTWARE ENGINEERING

Friday, October 26th, 2007; 10:00am – 11:30am

Instructions

- □ Write ~~the last four digits of~~ Full your student identification number in the space below.
- □ This exam consists of 17 pages (including this cover).
- □ **The exam consists of two parts:**
  - **You must answer Questions 1 and 2.**
  - **Answer any three (3) of Questions 3, 4, 5, 6, 7**

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

    3 4 5 6 7

    (If you leave this blank, Questions 3, 4 and 5 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.	General Knowledge – True/False	10	
2.	General Knowledge – Multiple Choice	15	
3.	Design	25	
4.	Construction	25	
5.	Testing	25	
6.	Maintenance & Evolution	25	
7.	Process	25	
	Total	**100**	

1. General Knowledge - True/False (10%)

Grading: 1% for each correct answer; total 10 questions.

Note: Circle either T (true) or F (false) for each question. Circling *both* T and F will result in 0 points for that question.

(a)	Software engineering is only concerned with the technical activities of the software life cycle, such as requirement, design, construction, testing, and maintenance & evolution.	T	F
(b)	It is generally accepted that one cannot have weak software processes and still create high-quality software products.	T	F
(c)	The use of assertions in source code is an effective means of implementing black-box testing.	T	F
(d)	Functional requirements are more important than non-functional requirements. As long as functional requirements are met, meeting non-functional requirements are optional.	T	F
(e)	One can overload whitespace as an operator in C++.	T	F
(f)	In UML diagrams, use-case actors always refer to people; they can never be hardware devices.	T	F
(g)	Requirements Specification refers to a complete list of what the customer expects the system to do, from the user's point of view.	T	F
(h)	Linked lists can be implemented using arrays.	T	F
(i)	The three most common types of software maintenance are contemplative, adaptive, and perfective.	T	F
(j)	Extreme Programming (XP) is a heavyweight software process model.	T	F

2. General Knowledge - Multiple Choice (15%)

Grading: 1% for each correct answer; total 15 questions.

Note: For each question, please select only *one* answer that best suits the question. Selecting no answer or more than one answer will result in 0 points for that question.

1. The aim of software engineering is to produce software that is:
   - (a) Fault-free
   - (b) Delivered on time
   - (c) Delivered within budget
   - (d) Satisfies users' needs
   - (e) All of the above

2. Software deteriorates rather than wears out because:
   - (a) Software suffers from exposure to hostile natural environments
   - (b) Defects are more likely to arise after software has been used often
   - (c) Multiple change requests introduce errors in component interactions
   - (d) Software spare parts become harder to order
   - (e) All of the above

3. The waterfall model of software development is:
   - (a) A reasonable approach when requirements are well-defined
   - (b) A good approach when a working program is required quickly
   - (c) The best approach to use for project with large development team
   - (d) An old fashioned model that cannot be used in a modern context
   - (e) None of the above

4. If a specification statement is to be testable then one of the properties it must have is:
   - (a) Tenacity
   - (b) Probability
   - (c) Usability
   - (d) Traceability
   - (e) None of the above

5. Which of the following is not a software life-cycle model?
    (a) Waterfall
    (b) Rapid prototyping
    (c) Synchronize and stabilize
    (d) Evolutionary
    (e) All are software life-cycle models

6. In which of the following circumstances might the waterfall model be an appropriate software life-cycle model to use?
    (a) Large-scale, new products
    (b) For products utilizing an open architecture with a complex user interface
    (c) Real-time systems
    (d) Mission-critical systems
    (e) Follow-on releases on mature systems

7. Which of the following software process models specifically address risk analysis and risk resolution?
    (a) Waterfall
    (b) Rapid prototyping
    (c) Synchronize and stabilize
    (d) Revolutionary model
    (e) Spiral model

8. A common solution used in real-world situations by knowledgeable programmers to deal with a component that encounters an error during processing is to:
    (a) Return an error value to the callee
    (b) Throw an exception
    (c) Set a status flag
    (d) All of the above
    (e) None of the above

9. A common metric for estimating the effort to develop a software product is:
   (a) Function Points (FP)
   (b) Lines of Comments (LOC)
   (c) COCOMO
   (d) Bytes
   (e) None of the abov

10. The aim of the synchronization stage of the synchronize-and-stabilize software life-cycle model is to:
    (a) Repair faults found in any earlier releases
    (b) Freeze change requests so that the build can be stabilized
    (c) Draw up the specification document
    (d) Release the current version(s) of the product to the clients based on the versions and releases installed at their various sites
    (e) Put the partially completed components together and test and debug the resulting product

11. The use of traceability in requirements tools helps to:
    (a) Debug programs following the detection of run-time errors
    (b) Determine the performance of algorithm implementations
    (c) Identify, control, and track requirements changes
    (d) All of the above
    (e) None of the above

12. Methods of requirements gathering include:
    (a) Interviews using structured techniques and close-ended questions
    (b) Interviews using structured techniques and open-ended questions
    (c) Questionnaires
    (d) An analysis of forms used by clients
    (e) All of the above

13. Use-case diagrams:
    - (a) Describe what the system should do
    - (b) Display object interactions arranged in a time sequence
    - (c) Are collections of objects with the same characteristics
    - (d) All of the above
    - (e) None of the above

14. A finite state machine consists of five parts: a set of states J, a set of inputs K, a transition function T that specifies the next state given the current state and the current input, the initial state S, and the set of final states F. What is required to determine the next state?
    - (a) The initial state and the transition function T
    - (b) The final state and the transition function T
    - (c) The current state, the current input and the transition function T
    - (d) The current state, the transition function T and the final state.
    - (e) Sets J, K and T

15. Which of the following is not the task for requirements validation and verification?
    - (a) Check for completeness
    - (b) Check for consistency
    - (c) Check for feasibility
    - (d) Check for traceability
    - (e) Check for adaptability

3. Design (25%)

Design is a key component of the overall software lifecycle. Good design contributes to the construction of elegant and bug-free software. There are several timeless guidelines that have been used by software designers over the years, including:

1. Iterative enhancement
2. Stepwise refinement
3. Information hiding

Problem:

(a) Describe software design by placing it in context of the overall software lifecycle. Your answer should include a discussion of issues such as the different types of design (e.g., high-level versus low-level), different design paradigms (e.g., object-oriented versus functional), and different design representations (e.g., UML).

(b) Explain the three timeless software design guidelines shown above.

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

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

4. Construction (25%)

(a) There are several common solutions that are often used in real-world situations by knowledgeable programmers to deal with a software component that encounters an error during processing. (19%)

Problem:

i) Describe three (3) of these solutions. Clearly explain their relative advantages and disadvantages.

ii) Give an example of two (2) of these solutions by providing a representative code fragment in C/C++.

Grading: (i) 5% each (15%) ; (ii) 2% each (4%).

(b) **Problem:** The programmer is trying to produce a random number of messages. However, it's taking a long time. Explain what is wrong with this C++ program by highlighting the offending statements and describing the problem(s). (3%)

```
#include <iostream>
#include <stdlib.h>
#define Random() rand

int main(void) {
  int n = 0 ;
  do {
    cout << "hello " << ++n << "\n";
  } while( Random() != 0 );
  return 0;
}
```

(c) **Problem:** This function, which is intended to count the vowels in the string provided, is taking a long time to do so. Explain what is wrong with this C++ code fragment by highlighting the offending statements and describing the problem(s). (3%)

```
int count_vowels(char *s) {
  int sum = 0;
  for(;;)
    switch( *s++ ) {
      case 'a':
      case 'e':
      case 'i':
      case 'o':
      case 'u':
        sum++; continue;

      default: continue;

      case '\0': break;
    }
  return sum;
}
```

5. Testing (25%)

There are many different types of strategies commonly used for software testing. However, the strategies can be broadly classified into two distinct categories: "Black Box" testing and "White Box" (aka "Glass Box") testing.

Problem:

(a) Describe "Black Box" testing and "White Box" testing. Include in your description an explanation of the difference(s) between the two categories, and the advantages and disadvantages of each.

(b) Comment on who should do "Black Box" testing: the developer or someone else.

Grading: (a) Description of each software testing category: 10% each; (b) 10%.

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

6. Maintenance & Evolution (25%)

Maintenance is the act of modifying a program after system deployment. Following good software practice doesn't negate the need for maintenance—just its severity. In fact, maintenance is the most common form of software evolution.

Problem:

(a) Describe the three (3) most common types of software maintenance. Be sure to provide examples of each type of maintenance using realistic scenarios.

(b) Discuss how software maintenance differs from software construction.

Grading: (a) 5% for each description (15%); (b) 10%.

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 software process improvement model.

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

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