University of Toronto

Faculty of Arts and Science
Dept of Computer Science

CSC302F – Engineering Large Software Systems

December 2008
Instructor: Steve Easterbrook

No Aids Allowed
Duration: 2 hours
Answer all questions.

Make sure your examination booklet has 10 pages (including this one). Write your answers in the space provided.

This examination counts for 35% of your final grade.

Name:____
(Please underline last name)

Student Number: ______________________________________

Question Marks

1 ______________/20

2 ______________/20

3 ______________/20

4 ______________/20

5 ______________/20

Total___________/100

1. [Short Questions; 20 marks total]

(a) [Verification and Validation – 5 marks] Explain the difference between verification and validation. Give an example of a technique that can be used for each.

Verification is the process of checking that software meets its specification (i.e. did we build the system right). One way to do this is by writing test cases that check the program behaves in the way specified.

Validation is the process of checking that we understood the stakeholder's problem domain correctly (i.e. are we building the right system). One way to do this is by building models that capture our understanding of the problem, and discussing the properties of these models with the stakeholders to check that they accurately represent the problem.

(b) [Software Aging – 5 marks] What are the *causes* and *symptoms* of software aging? What steps can be taken to reduce the problems associated with software aging?

Software ages because the world continually changes – technology improves and users demand more functionality. If the software is not updated continually, it becomes steadily less useful. Symptoms of software aging include growing size, growing complexity, and deteriorating structure, because it gets harder to accommodate changes within the original design.

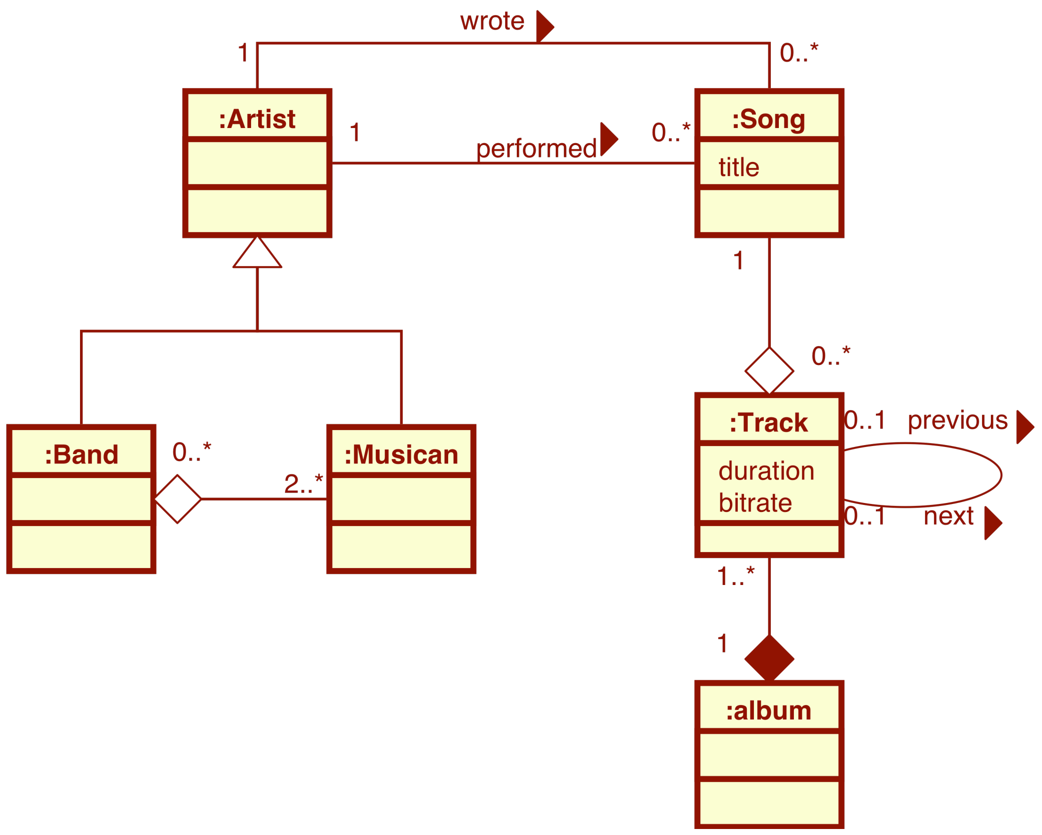
To reduce the problems associated with software aging, we can attempt to restructure the software by re-factoring it, or we can re-engineer it, through a process of reverse engineering a design model, redesigning these to improve modularity, and then re-implementing the system.

(c) [Requirements Analysis – 5 marks] It has been suggested that UML models can be used in software engineering as either *sketches*, *blueprints*, or as a *high-level programming language*. Explain each of these different ways of using UML, and describe the limitations of each.

(d) [Program Comprehension – 5 marks] What are the factors that make program comprehension so difficult? What kinds of tools can help overcome these difficulties?

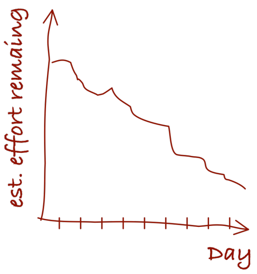
2. **[Black Box Testing – 20 marks]** You have been asked to test a web page intended to set up secure access to a banking system. The web page consists of a form that requests the user's date of birth, 9-digit social security number and an answer to a secret question, and stores these in a database. How would you design a thorough test suite for this web page, including black box, white box, and stress testing strategies? Be sure to indicate what coverage criteria you would apply, and give example tests for each strategy.

3. **[Domain Modeling – 20 marks]** Draw a UML Class Diagram representing the following elements from the problem domain for digital music players: An artist is either a band or a musician, where a band consists of two or more musicians. Each song has an artist who wrote it, and an artist who performed it, and a title. Assume a “song” means a *recording* of a piece of music, so that if a piece of music is recorded more than once (say, by different artists), we treat them as different songs. Therefore, each song is performed by exactly one artist, and written by exactly one artist. An album is composed of a number of tracks, each of which contains exactly one song. A song can be used in any number of tracks, because it could appear on more than one album (or even more than once on the same album!). A track has a bitrate and a duration. Because the order of the tracks on an album is important, the system will need to know, for any given track, what the next track is, and what the previous track is (if there is one). Draw a class diagram for this information, and be sure to label all the associations with appropriate multiplicities.



4. **[Project Management – 20 marks]** Describe three different tools that a manager can use to plan and/or track the progress of a project, and discuss the strengths and weaknesses of each. Explain the actions a manager could take if each of these tools indicates the project is not running according to plan.

Many possible answers. Must have three reasonable tools, and fully discuss strengths, weaknesses and actions that can be taken for each tool.
Example:



The burn down chart, which plots effort remaining (sum of effort estimated for all unfinished tasks) against time. This shows day-by-day how the project is doing, compared to the plans. Advantages are that it gives daily feedback on progress, and concentrates on small tasks for which it is easy to assess completion (i.e. mainly tasks that produce working code). Also, it shows re-estimates (rather than original estimates) for remaining tasks each day. Hence is good at capturing up to date information. The disadvantages are that it doesn't readily show progress on other types of task, and so the trendline might not be a good indicator of the real amount of work to complete. If the chart shows the project will miss the target release date, the obvious actions a manager can take are to move some of the remaining tasks to a subsequent release, or shift the planned release date.

Other possible tools covered in the course:
Gantt Charts
Pert Charts
Test-Cast trend charts
Meetings (as a tool for managers to assess progress!)
Etc.

5. **[Process models – 20 marks]** A ***process model*** provides a detailed prescription of the sequence of activities needed to develop software, giving guidance about when each activity should be used, and what each activity should produce. Describe an example of one software engineering process model that you used on your course project this term, and one that you did ***not*** use. For the first process model, explain how you applied the process, and how well it applied to your project. For the second process model, describe the factors that caused you not to use it. Your answer should take into account whether each process model was suitable for the type of software you were building, and whether you had the right expertise to use it properly.

Many possible answers. For full credit, must explain both process models properly – the one that was used and the one that was not. Must also address suitability and relevant expertise. Plus must say clearly whether the process model used was effective, and give clear reasons for not using the other. Answer must show good insights and reflection on the software development processes to get full marks.

[scratch paper]

[scratch paper]

[scratch paper]