SENG 371 – Software Evolution

Spring 2024 (Sections A01 & A02)
CRN 22933 & 22934
Final Exam: Friday, 19 April 2024
Duration: 3 (three) hours
Instructor: Roberto A. Bittencourt

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.

- **All answers are to be written on this exam paper.**
- We will not answer questions during the exam. If you feel there is an error or ambiguity, write your assumption and answer the question based on that assumption.
- The exam is closed book. No books or notes are permitted.
- When answering questions, please do not detach any exam pages!
- ***Electronic devices are not permitted. The only exception is a simple calculator***.
- Cellphones must be turned off.
- The marks assigned to each section are within parentheses. Partial marks are available.
- There are sixteen (16) printed pages in this document, including this cover page and an answer bubble sheet in the final page.
- Please do NOT remove pages from the exam.
- We strongly recommend you read the entire exam through from beginning to end before starting on your answers.
- **Please have your UVic ID card available on the desk for inspection.**

Section A (40 marks): For each question in this section, **mark only one answer on the answer bubble sheet on the last page of this exam.** Answers marked only on the question pages will not be graded. Twenty multiple-choice questions follow, each one worth 2 (two) marks.

Question 1: Mark the only incorrect assertion below about software life span models.

a) During evolution, new features are added, and mistakes and misunderstandings about the problem domain are corrected.
b) During servicing, changes are limited to patches and wrappers.
c) During phase-out, no more patches are serviced, and users must work around known deficiencies.
d) For close-down, users are directed toward a replacement product, and an exit strategy is chosen.
e) The versioned staged model is different from the simple staged model in that version control is used to record software changes.

Question 2: During the software concept location phase of Rajlich's software change process, some activities are performed. Mark the one activity that is not usually performed in this phase.

a) One or more concepts are extracted from a change request.
b) Programmers may formulate a query to locate the concept(s) in the source code.
c) After locating the concept in one or more files, the potential impact of the change in other files is analyzed.
d) After a query returns the results, they are manually analyzed by programmers.
e) If an initial search fails, programmers try a new search or a different strategy.

Question 3: During code refactoring, some code smells are found and removed from source code. About code smells, it is incorrect to say that:

a) A *Large Class* code smell means a class that has too many responsibilities.
b) A *Long Method* code smell may make it hard for developers to understand what the method does.
c) A *Feature Envy* code smell happens when a method is invoking methods on another object too many times.
d) A *Middle Man* code smell happens when a class has various functionalities plus the functionality of delegating work to other classes.
e) A *Code Duplicated* code smell means using the same code structure in more than one place in the source code.

Question 4: Mark the incorrect assertion in the context of change impact analysis and actualization.

a) True positives are the files predicted to change that were changed.
b) True negatives are the files predicted to change that were not changed.
c) Precision indicates how many files in the estimated impact set were correct.
d) False negatives are the files predicted not to change that were changed.
e) Recall is the rate of files predicted to change by the files that were changed.

Question 5: Mark the correct answer about what the JUnit `setUp()` method is.

a) A method to exercise the assertions of the test case.
b) A method to clean up the test fixture after the test.
c) A method to provide a common fixture for running different test cases.
d) A method to provide a fixture for running only one test case.
e) A method to prepare a test suite as a sequence of test cases.

Question 6: Mark the incorrect assertion about the change conclusion phase of Rajlich's software change process.

a) A new issue/change request is written to the issue tracker.
b) Developers may either commit their changes to version control or make a pull request for their change to be accepted.
c) Baseline testing will try to assure that the new baseline is as bug-free as possible.
d) Code reviews may be performed during change conclusion.
e) Conflicts between the main repository code and the changed code may need to be solved.

Question 7: About social barriers faced by newcomers in open source systems (OSS), mark the one example below that is not a social barrier.

a) Some newcomers do not understand the documentation of an open source system.
b) It may take a long time to receive an answer in the OSS communication forums.
c) Some newcomers do not send meaningful messages in their posts.
d) It may be hard to get mentoring for newcomers in OSS projects.
e) Some newcomers feel the need of initial contact with a real person, which is not always available.

Question 8: Mark the only wrong answer about software reverse engineering.

a) Typically, it starts from source code and produces higher-level abstractions.
b) The recovered high-level design is changed to better fit new technologies and/or customer requests.
c) A low-level representation of source code is created.
d) A low-level representation of code is abstracted into a high-level model or description.
e) High-level recovered views are usually presented as either graph or matrix visualizations.

Question 9: Mark the option below that is not a feature of a distributed version control system.

a) Relies on one or more distributed repositories.
b) Allows different users to evolve in different directions (branches) from the same common point.
c) Offers tools for merging different branches back into the main branch.
d) Requires Internet connection to send commits to a repository.
e) Allows different workflow strategies, such as Central Repository, Integration Manager or Dictator and Lieutenants.

Question 10: Mark the option below that does not describe DevOps' best practices at the Netflix case study.

a) The use of a microservices architecture where each service is owned by a team.
b) Tradeoff between velocity and stability handled by the same group of people, since they handle development and operations together.
c) A mix of different tools for CI and CD, most of them proprietary. Whenever needed, they outsource tool development.
d) Observation of their operations by means of tools that provide telemetry and dashboards.
e) Skills related to troubleshooting operations and coordinating their work are learned during incidents and later shared.

Question 11: Mark the only incorrect assertion below about "floss" refactoring.

a) It is part of everyday programming practice.
b) It is performed in small bursts.
c) There is no need to set aside time to perform it.
d) It is usually a global change that affects large parts of the code.
e) It is usually small in terms of code impact.

Question 12: Mark the only incorrect assertion below about static elements of a mental model for software comprehension.

a) Text-structures are useful to gain control flow knowledge.
b) Chunks enable programmers to create higher level abstractions from lower-level constructs.
c) Schemas are particular knowledge structures meant to describe algorithms.
d) Plans are special types of schemas with a slot type and a slot filler.
e) Hypotheses are conjectures that may hypothesize both the purpose of program elements and the methods for realizing program goals.

Question 13: Mark the only incorrect assertion below about dynamic elements of a mental model for software comprehension.

a) Chunking may be used to replace a block of code by a label representing the functionality of the code block.
b) Cross-referencing links elements from different abstraction levels.
c) Strategies are planned sequences of actions to reach a specific goal.
d) Chunking is a way to overcome the limitations of short-term memory.
e) Cross-referencing identifies goals and actions in source code.

Question 14: Mark the only incorrect assertion about the information fragments model.

a) It allows developers to compose different information fragment types.
b) It shows query results by means of projections of composed information fragments.
c) The id matching operator may be used to match different information fragment types in composition operations.
d) It is formally described by means of trees and predicate logic.
e) It may be used to answer questions related to software development.

Question 15: Mark the only incorrect reason for the end of software evolution.

a) Cultural change in development practices.
b) Team conflicts due to generational gaps.
c) Management decision to retire the system.
d) Software stabilization due to known domain.
e) Code decay, making software changes harder.

Question 16: Mark the only incorrect assertion about incremental reengineering.

a) Avoids disruption of users' routines.
b) Wrong reengineering choices may be easily reversed.
c) Expenses on individual reengineering tasks are limited.
d) Results are more immediate and concrete.
e) Two working systems have to be maintained in parallel.

Question 17: Mark the only incorrect reason for performing code refactoring.

a) To fix bugs in source code.
b) To improve the design of a software system.
c) To increase software comprehension.
d) To reduce costs of software maintenance.
e) To facilitate future changes.

Question 18: Some tools for software architecture and design recovery such as Structure101 are used by part of software industry. Mark the option that describes a feature not available in such tools.

a) Fact extraction of source code entities and relationships.
b) High-level abstraction of source code entities and relationships.
c) Software metrics related to module quality (e.g., fat code and tangled code).
d) System and acceptance test suite automation.
e) Architecture module view description and checking.

Question 19: Mark the only wrong assertion about the use of UML diagrams in software development practice.

a) UML is used by all the software industry, being a *de facto* standard for modelling.
b) Class, sequence and activity diagrams are the most popular UML diagrams.
c) UML diagrams may be used to communicate with stakeholders.
d) UML representations are useful when they are understandable and fit for purpose.
e) UML adds an overhead in stakeholder communication to understand its notation.

Question 20: Mark the only wrong assertion about the workflows for version control.

a) The Central Repository workflow is easily implemented in *git*.
b) The Integration Manager workflow allows more social interaction among developers when compared to the Central Repository.
c) The Dictator and Lieutenants workflow is easily implemented in *Subversion*.
d) There are scaling issues with the Integration Manager workflow when the integration manager is a human.
e) The Dictator and Lieutenants workflow offers better scalability for large projects.

Section B (60 marks): Six open-ended questions follow, each one worth 10 (ten) marks. When answering those questions, you may use your own opinions, as long as they are grounded on technical or scientific knowledge.

Question 21: (10 marks). Compare Rajlich's software change process with the test-driven development process to perform changes in software systems in terms of advantages and disadvantages (describe at least one advantage and at least one disadvantage for each process).

Rajlich's software change process:

- **Advantages:**
  - **General process that encompasses a wide variety of changes.**
  - **Split between change design and change implementation.**
- **Disadvantages:**
  - **Not all phases need to be performed for every change.**
  - **Some developers see the phase of impact analysis as unnecessary.**

Test-driven development process

- **Advantages:**
  - **Tests drive the whole change process.**
  - **The process is simple and independent of change type.**
- **Disadvantages:**
  - **It may be hard to estimate the change cost without impact analysis.**
  - **Some developers prefer to develop first, then test.**

The lists above are not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each minimum required part of the answer.

Marks may be deduced when each stated advantage or disadvantage is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

Question 22: (10 marks). Describe at least two pros and at least two cons of using pull requests instead of committing changes straight to the trunk of a software version control repository.

Pros:

- **Opportunity to review a change before making it permanent.**
- **Increased communication between developers.**
- **Allows creating a hierarchy of contributors to the system.**
- **Allows developers with no writing access to systematically submit their contributions.**

Cons:

- **If a person is responsible for approving the pull request, that person may be a failure point in the process.**
- **One level of indirection is added to the change process, making it more bureaucratic for some developers.**

The lists above are not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each minimum required part of the answer.

Marks may be deduced when each stated pro or con is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

Question 23: (10 marks). Describe the at least four differences between the Cathedral (closed source) and Bazaar (open source) practices of software development.

Differences:

- **While closed source is developed in a controlled, coordinated way, open source is developed with a lot less control.**
- **While closed source is developed by a relatively small, tightly-knit group, open source is developed by various volunteer developers that may live in various different regions of the world, coordinated through the Internet.**
- **While closed source requires secrecy, sometimes with NDAs, open source is based on full disclosure of software content in open repositories.**
- **While closed source gets less bug reports for they are only reported by the group that has access to the source, open source receives fast feedback from many users that may report bugs over the versions available in the open repository.**
- **While closed source maintains quality based on the team experience and internal checks, open source keeps its quality by releasing very often and getting immediate feedback from a large user base.**

The list above is not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each difference.

Marks may be deduced when each stated difference is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

Question 24: (10 marks). Practitioners and researchers point out various disadvantages of using UML in software development. Describe at least four disadvantages of using UML from the point of view of practitioners.

Disadvantages of UML:

- **Some developers consider that UML introduction is based on an ideology that puts too much emphasis on software models.**
- **Lack of context (about, e.g., requirements, decisions, risks), dealing primarily with software architecture and design.**
- **Overhead to understand notation, i.e., the complexity of the notation limits its utility in discussions with stakeholders.**
- **Some developers mention that UML has an excess of diagram types (14), and that they never use state machine, use case and communication diagrams.**
- **Synchronization between the various diagrams and between diagrams and other software artifacts (e.g., requirements, source code) is a barrier for wholesale adoption of UML.**

The list above is not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each disadvantage.

Marks may be deduced when each stated disadvantage is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

Question 25: (10 marks). Describe at least two advantages and at least two disadvantages of using software architecture checking techniques in the context of large-scale software systems.

Advantages:

- **Implementation follows architectural rules reinforced by checking.**
- **Better understanding of architectural rules, since they are explicitly stated and frequently reminded by the periodic checks.**
- **Reduces or postpones architectural erosion (software aging).**

Disadvantages:

- **Sometimes the violations to architecture in source code are so many that it may be hard to care for checking results.**
- **Some checking techniques such as Reflexion Models may lead to false negatives in the architecture checking reports.**
- **Some developers think that architecture checking is one more bureaucracy added to the software process.**

The lists above are not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each minimum required part of the answer.

Marks may be deduced when each stated advantage or disadvantage is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

Question 26: (10 marks). After 1997, different studies questioned the validity of Lehman's laws of software evolution. Explain at least four different ways that different software development practices affected the assumptions under which such laws relied on.

- **Releases are more frequent today, which reduces a little the typical pressures of a large release, making the effort per release more variable.**
- **Software architecture emerged and brought with it better defined module interfaces, lessening the amount of knowledge developers must understand about other parts of a software system.**
- **Software systems are not monoliths any longer. They rely on external libraries, frameworks, run-time environments, virtual machines and services. Thus, complexity of contemporary software cannot be measured only by traditional software metrics.**
- **Open source development and agile processes affected the way software systems are developed, making developers both contribute to and reuse existing software as part of their development of a software system.**
- **The Internet now acts as a catalyst of emergent uses of software, speeding up those new uses, and affecting the feedback loops of software systems.**

The list above is not exhaustive. Students may come up with different answers, which will be considered if they make sense and are based on sound technical or scientific knowledge.

2.5 marks for each different development practice.

Marks may be deduced when each stated different practice is not based on sound technical or scientific knowledge.

You may use the space below for scratch work.

END OF EXAM

EXAM SECTIONS

Section	**Value**
A – Multiple-Choice Questions (20 questions, 2 marks each)	40
B – Open-Ended Questions (6 questions, 10 marks each)	60
TOTAL	**100**

(This page intentionally left blank for scratch work. Any answers here will not be marked.)