Advanced Software Engineering, COMP3C05, 2002

Answer Question 1 and two further questions.

Marks for each part of each question are indicated in square brackets

Calculators are NOT permitted

1. Your 3C05 Presentation

   Write an essay of no more than 500 words about the subject of the 3C05 presentation that you prepared and delivered. Address the research results for that particular area that you presented as well as the open problems that are currently being investigated.

   Every student gave a seminar as part of the course. The following topics were covered.

   - Software Engineering Economics
   - Software Metrics
   - Performance Engineering
   - Real-time Systems
   - Reliability and Dependability
   - Safety-critical Systems
   - Security Engineering
   - Software Engineering for Mobile Systems

[Total 34 marks]

2. Version and Configuration Management

a. Define the following concepts used in version and configuration management:

i. Version
A snapshot in the development history of a software engineering artefact
[2 marks]

ii. Revision
A sequential version
[2 marks]

iii. Variant
A concurrent version
[2 marks]

iv. Configuration
A set of artefacts where one particular version is selected for each artefact such that they are consistent
[2 marks]

v. Check-in and Check-out
The principle operations of version and configuration management. Check-out creates a new revision (or variant if another version is concurrently checked out. Check-out creates a private copy of the version. Check-in writes back the private copy into the version management repository.
[4 marks]

vi. Workspace
The local storage area of developers and the version selection rules.
[2 marks]

[Subtotal 14 marks]

b. Give five reasons for using a version and configuration management system in a software development process.

1. continuous availability of stable baseline supports regular builds
2. management of developed / delivered increments
3. availability of private developer workspaces
4. ability to undo changes to versions.
5. understanding the change history of artefacts
6. provision of raw data for collection of software engineering metrics

[10 marks]

c. Which artefacts developed during a USDP iteration have to be kept under version control?

All. They should all be kept under version control as they need to be revisited during later increments.

[9 marks]

[Total 33 marks]

3. Development Process

a. Explain the meaning of the following Unified Software Development Process concepts:

i. Iteration
A workflow of requirements, analysis, design, implementation and testing.
[2 marks]

ii. Increment
The software release that is produced during an iteration
[2 marks]

iii. Inception
The first phase of the USDP during which the feasibility of the development is established and a business case for the new system is being made.
[2 marks]

iv. Elaboration
The second phase of the USDP during which all main development risks and uncertainties about requirements are resolved in iterations that develop prototype increments. It is during this phase that an architectural vision for the system is established and tested and that most of the requirements are clarified.
[2 marks]

v. Construction
The third phase during which iterations add the required functionality to the system in a series of increments. Only minor requirements changes are made during this phase and components of increments are unit tested and increments are integration tested.
[2 marks]

vi. Transition
The final phase that establishes the transfer of operational increments into use. Involves beta testing in production environment and user training.
[2 marks]

[Subtotal 12 marks]

b. In no more than 200 words, discuss the advantages and disadvantages of the Waterfall process model and compare it to iterative and incremental development as suggested by the Unified Software Development Process.
Waterfall process is quite simple and suitable for processes with little or no risk and when the requirements are clear, unambiguous and complete. The disadvantage of the Waterfall process model is that it cannot really cope with situations of high risk and unclear requirements. USDP improves that situation through iterations that are used to add certainty about requirements and resolve risks at an early stage.

[10 marks]

c. During the inception phase of a project developing a mission-critical world-wide distributed trading system for an investment bank the project team has constructed the business case and suggested the use of a 3-tier architecture to develop the system. The team has also identified the following set of risks.

1. Foreseen hot standby solution fails to deliver required level of fault tolerance.
2. Available network bandwith is too small for the traffic that is generated by the client hosts.
3. Performance on remote clients is too poor for traders.
4. Database does not scale to increased number of client hosts.
5. Windows client application does not work due to use of different versions of the operating system at different sites.
6. Traders might be reluctant to participate in requirements elicitation meetings as this work would not contribute towards their bonuses.

Produce the iteration plan for the elaboration phase in such a way that these risks are being mitigated.

First prioritize these risks. Then classify them according to different classes, e.g. non-functional requirements not met, functional requirements cannot be defined. Then plan risk mitigation through prototyping in two-three iterations. Tackle highest risk first. For example:

First iteration: Mitigate high risks 2,3,6: Elicit use cases from traders (and find ways to remunerate them if they do not participate), develop vertical slice through architecture to measure performance of use of client application run at remote site. During this test also measure required network bandwidth and analytically establish that the bandwidth is sufficient for the anticipated distribution.

Second iteration: Mitigate medium risk 1,5: Add replication to the first increment to implement hot standby. Test by switching off one machine during operation. Distribute first increment to all sites that have different operating system platforms and perform a test installation.

Third iteration: Mitigate low risk 4: Use second increment that provides replicated middle tier services to perform a scalability test to see whether the central database scales.

[11 marks]

[Total 33 marks]

4. Testing and Inspection

a. Define the following concepts:

i. Black-box testing

Testing a function, module, class or component without considering how it has been implemented.

[3 marks]

ii. White-box testing

Considering the internals of units when defining the test cases. In particular, ensuring appropriate coverage of code.

[3 marks]

iii. Coverage-based testing

Represent program as control-flow graph. Identify paths. Choose data to maximise paths executed under test conditions. Paths not always finite. Some paths infeasible. Coverage metrics can be applied. Can be partially automated.

[3 marks]

iv. Stress testing

Form of black-box testing, which establishes that a component or function behaves appropriately at the boundaries of its input

[3 marks]

[Subtotal 12 marks]

b. Which techniques can you use to decide whether you "have tested enough"? Explain for each of them how they work.

A team has tested enough if their test data is sufficient. To know that, they can perform

statistical analysis Seed known errors into code so that their placement is statistically similar to that of actual errors.

mutation analysis It is assumed that a set of test data that can uncover all simple faults in a program is also capable of detecting more complex faults. In mutation analysis a large number of simple faults, called mutations, are introduced in a program one at a time. The resulting changed versions of the test program are called mutants. Test data is then be constructed to cause these mutants to fail. The effectiveness of the test data set is measured by the percentage of mutants killed.

[9 marks]

c. The testing techniques we discussed during the course were aimed at establishing the functioning of a system. How can you test whether a system meets the following non-functional requirements

i. Scalability
Generate load similar to anticipated future loads and see how an executable architecture behaves.

[3 marks]

ii. Security
Hire a bunch of security experts or hackers and ask them to break the system security.

[3 marks]

iii. Performance
Instrument the source code (if available) or use instrumentation wrappers to with time measurement that is logged into a file during execution and then analyze those logs.

[3 marks]

iv. Tolerance against environmental or hardware failures
Reproduce these environmental failures (switching off air-conditioning, electricity) or simulate hardware failures (by switching off power, disconnecting the machine from the network, or disconnecting the harddrive) to see whether the system remains operational.

[3 marks]

[Subtotal 12 marks]

[Total 33 marks]

5. eXtreme Programming (XP)

a. Define the meaning of the following XP concepts:

i. Pair programming

Two programmers team up to write code. One programmer writes the code, while the other looks up specifications and inspects/reviews the first programmer's code while it is being written.

[2 marks]

ii. Test first

Write the unit test as a specification for the unit. Use of JUnit as a specification/testing tool.

[2 marks]

iii. User story

Things that the system needs to do for them (similar to use cases) In the format of about three sentences of text written by the customer in the customers terminology.

[2 marks]

iv. Refactoring

Refactoring is the removal of redundancy, elimination of unused functionality, and rejuvenation of obsolete designs in order to keep the design simple and avoid needless clutter. It helps to make a design easier to understand, modify, and extend.

[2 marks]

v. Spike solution

A spike solution is a very simple program to explore potential solutions. Build a system which only addresses the problem under examination and ignore all other concerns.

[2 marks]

vi. Project velocity

Development divided into about 12 iterations of about 1 to 3 weeks in length. Project velocity=number of user stories per iteration.

[2 marks]

[Subtotal 12 marks]

b. Assume XP is used in the NewRes project to develop a new hotel reservation system for the UnterContu chain of hotels. The users have written their user stories, some of which are included below:

- NewRes supports direct enquiries of guests for room availability over the Internet using a Web browser. Guests can select a particular hotel, dates, room preference (single, double, twin, suite, non-smoking, smoking) and the system will inform about room availability and prices.
- If rooms are available, guests can reserve a room use a Web browser. They provide their credit card details over a secure connection and receive a confirmation that they can print.
- Guests can amend a previous reservation over the Web. They provide their reservation number and can then change the dates and room category for which the booking is held.

Begin the design of the system based on the above user stories.

XP uses CRC cards for design purposes

Room	
Check availability	Guest
Reserve room	Hotel
Give room details	
Give guest details for date	

Hotel	
Give hotel details	Room
Find free room	Guest
Reserve room	Hotels

Guest	
Record credit card	Hotel
Add charge	

Hotels	
Find hotel by name	Hotels
Find hotel by location	
Find hotel by proximity	

[11 marks]

c. Would you consider introducing techniques of eXtreme Programming into the Unified Software Development Process? Which ones would you choose and why?

- Pair programming in the construction phase
- Test First for the detailed specification of classes
- Refactoring to adjust problems between iterations if increments are not discarded (particularly for later elaboration and construction increments).
- JUnit for unit tests during construction phase

[10 marks]

[Total 33 marks]

END OF PAPER