CS3043 Assignment 1

Sample Answers

1. What are the most important differences between generic software product development and custom software development? What might this mean in practice for users of generic software products?

   The essential difference is that in generic software product development, the specification is owned by the product developer. For custom product development, the specification is owned and controlled by the customer. The implications of this are significant – the developer can quickly decide to change the specification in response to some external change (e.g. a competing product) but, when the customer owns the specification, changes have to be negotiated between the customer and the developer and may have contractual implications.

   For users of generic products, this means they have no control over the software specification so cannot control the evolution of the product. The developer may decide to include/exclude features and change the user interface. This could have implications for the user's business processes and add extra training costs when new versions of the system are installed. It also may limit the customer's flexibility to change their own business processes.

2. Based on your own knowledge of some of the application types discussed in the textbook (chapter 1, section 1.1.2), explain, with examples, why different application types require specialized software engineering techniques to support their design and development.

   Different application types require the use of different development techniques for a number of reasons:

   a) Costs and frequency of change. Some systems (such as embedded systems in consumer devices) are extremely expensive to change; others, must change frequently in response to changing requirements (e.g. business systems). Systems which are very expensive to change need extensive upfront analysis to ensure that the requirements are consistent and extensive validation to ensure that the system meets its specification. This is not cost effective for systems that change very rapidly.

   b) The most important 'non-functional' requirements. Different systems have different priorities for non-functional requirements. For example, a real-time control system in an aircraft has safety as its principal priority; an interactive game has responsiveness and usability as its priority. The techniques used to achieve safety are not required for interactive gaming; the extensive UI design required for games is not needed in safety-critical control systems.

   c) The software lifetime and delivery schedule. Some software systems have a relatively short lifetime (many web-based systems), others have a lifetime of tens of years (large command and control systems). Some systems have to be delivered quickly if they are to be useful. The techniques used to develop short-lifetime, rapid delivery systems (e.g. use of scripting languages, prototyping, etc.) are inappropriate for long-lifetime systems which require techniques that allow for long-term support such as design modelling.

3. Explain why there are fundamental ideas of software engineering that apply to all types of software systems.

   Because of all software systems have common quality attributes, including availability, modifiability, performance, security and safety, testability and usability, the fundamental software ideas provides common solutions or tactics to support those qualities.

4. Giving reasons for your answer based on the type of system being developed, suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:
   a) A system to control anti-lock braking in a car
   b) A virtual reality system to support software maintenance
   c) A university accounting system that replaces an existing system
   d) An interactive travel planning system that helps users plan journey with the lowest environment impact

   a) Anti-lock braking system This is a safety-critical system so requires a lot of up-front analysis before implementation. It certainly needs a plan-driven approach to development with the requirements carefully analysed. A waterfall model is therefore the most appropriate approach to use, perhaps with formal transformations between the different development stages.

   b) Virtual reality system This is a system where the requirements will change and there will be an extensive user interface components. Incremental development with, perhaps, some UI prototyping is the most appropriate model. An agile process may be used.

   c) University accounting system This is a system whose requirements are fairly well-known and which will be used in an environment in conjunction with lots of other systems such as a research grant management system. Therefore, a reuse-based approach is likely to be appropriate for this.

   d) Interactive travel planning system System with a complex user interface but which must be stable

and reliable. An incremental development approach is the most appropriate as the system requirements will change as real user experience with the system is gained.

5. Explain why incremental development is the most effective approach for developing business software systems. Why is this model less appropriate for real-time systems engineering?

Business software systems usually complex, software intensive, and frequently being changes when business goals or processes are changed. So incremental development is better.
Real-time systems usually involve many hardware components which are not easy to change and cannot be incremental. Also real-time systems usually safety critical which needed be built based on well planned process.

6. Suggest why it is important to make a distinction between developing the user requirements and developing system requirements in the requirements engineering process.

There is a fundamental difference between the user and the system requirements that mean they should be considered separately.

a) The user requirements are intended to describe the system's functions and features from a user perspective and it is essential that users understand these requirements. They should be expressed in natural language and may not be expressed in great detail, to allow some implementation flexibility. The people involved in the process must be able to understand the user's environment and application domain.

b) The system requirements are much more detailed than the user requirements and are intended to be a precise specification of the system that may be part of a system contract. They may also be used in situations where development is outsourced and the development team need a complete specification of what should be developed. The system requirements are developed after user requirements have been established.

7. Explain how the principles underlying agile methods lead to the accelerated development and development of software.

The principles underlying agile development are:

a) *Individual and interactions over processes and tools*. By taking advantages of individual skills and ability and by ensuring that the development team knows what each other are doing, the overheads of formal communication and process assurance are avoided. This means that the team can focus on the development of working software.

b) *Working software over comprehensive documentation*. This contributes to accelerated development because time is not spent developing, checking and managing documentation. Rather, the programmer's time is focused on the development and testing of code.

c) *Customer collaboration over contract negotiation*. Rather than spending time developing, analyzing and negotiating requirements to be included in a system contract, agile developers argue that it is more effective to get feedback from customer's directly during the development about what is required. This allows useful functionality to be developed and delivered earlier than would be possible if contracts were required.

d) *Responding to change over following a plan*. Agile developers argue (rightly) that being responsive to change is more effective than following a plan-based process because change is inevitable whatever process is used. There is significant overhead in changing plans to accommodate change and the inflexibility of a plan means that work may be done that is later discarded.

8. When would you recommend against the use of an agile method for developing a software system? Explain your answer.

Agile methods should probably not be used when the software is being developed by teams who are not co-located. If any of the individual teams use agile methods, it is very difficult to coordinate their work with other teams. Furthermore, the informal communication which is an essential part of agile methods is practically impossible to maintain.

Agile methods should probably also be avoided for critical systems where the consequences of a specification error are serious. In those circumstances, a system specification that is available before development starts makes a detailed specification analysis possible.

However, some ideas from agile approaches such as test first development are certainly applicable to critical systems.

9. Discover ambiguities or omissions in the following statement of requirements for part of a ticket-issuing systems:

An automated ticket-issuing system sells rail tickets. Users select their destination and input a credit card and a personal identification number. The rail ticket is issued and their credit card account charged.

When the user presses the start button, a menu display of potential destinations is activated, along with a message to the user to select a destination. Once a destination has been selected, users are requested to input their credit card. Its validity is checked and the user is then requested to input a personal identifier. When the credit transaction has been validated, the ticket is issued.

Ambiguities and omissions include:

a) Can a customer buy several tickets for the same destination together or must they be bought one at a time?
b) Can customers cancel a request if a mistake has been made?
c) How should the system respond if an invalid card is input?
d) What happens if customers try to put their card in before selecting a destination (as they would in ATM machines)?
e) Must the user press the start button again if they wish to buy another ticket to a different destination?
f) Should the system only sell tickets between the station where the machine is situated and direct connections or should it include all possible destinations?

10. Rewrite the above description using the structured approach described in chapter 4 of the textbook. Resolve the identified ambiguities in an appropriate way.

11. Write a set of non-functional requirements for the ticket-issuing system, setting out its expected reliability and response time.

Possible non-functional requirements for the ticket issuing system include:

a) Between 0600 and 2300 in any one day, the total system down time should not exceed 5 minutes.
b) Between 0600 and 2300 in any one day, the recovery time after a system failure should not exceed 2 minutes.
c) Between 2300 and 0600 in any one day, the total system down time should not exceed 20 minutes.

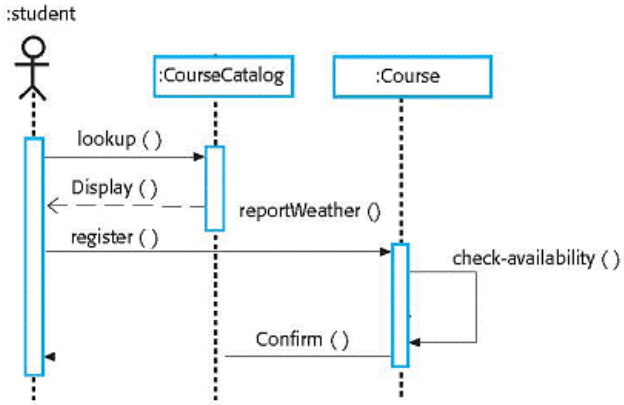
All these are availability requirements – note that these vary according to the time of day. Failures when most people are traveling are less acceptable than failures when there are few customers.

d) After the customer presses a button on the machine, the display should be updated within 0.5 seconds.
e) The ticket issuing time after credit card validation has been received should not exceed 10 seconds.
f) When validating credit cards, the display should provide a status message for customers indicating that activity is taking place. This tells the customer that the potentially time consuming activity of validation is still in progress and that the system has not simply failed.
g) The maximum acceptable failure rate for ticket issue requests is 1: 10000.

12. You have been asked to develop a system that will help with planning large-scale events and parties such us weddings, graduation celebrations, birthday parties, etc.

(a) Modeling entities and their relationships which are involved in planning events and parties using a UML class diagram.
(b) Modeling the process context for such a system that shows the activities involved in planning a party (booking a venue, organizing invitations, etc.), using a UML activity diagram.

13. Develop a sequence diagram showing the interactions involved when a student registers a course in a university. Courses may have limited enrolment, so the registration process must include checks that places are available. Assume that the student accesses an electronic course catalog to find out about available courses.



1. Draw a UML state diagram of the control software for ONE of the follows:

- An automatic washing machine that has different programs for different types of clothes.
- The software for a DVD player.
- A telephone answering system that records incoming messages and displays the number of accepted messages on an LED. The system should allow the telephone customer to dial in from any location, type a sequence of numbers (identified as tones), and play any recorded messages.

2. Explain why design conflicts might arise when designing an architecture for which both availability and security requirements are the most important non-functional requirements.

3. Draw a UML class diagram showing a conceptual view and draw a UML sequence diagram showing a process view of the architecture of ONE of the following systems:
   - An automated ticket-issuing system used by passengers at a railway station
   - A computer-controlled video conferencing system that allows video, audio and computer data to be visible to several participants at the same time.
   - A robot floor cleaner that is intended to clean relatively clear spaces such as corridors. The cleaner must be able to sense walls and other obstructions.

4. For each of the following architectural patterns described in the textbook, describe a real-world or example application system which should be designed using the pattern.
   - MVC
   - Layered
   - Repository
   - Client-server
   - Pipe and filter

5. Suggest an architecture for a system (such as iTune) that is used to sell and distribute music on the internet. What architectural patterns are the basis for this architecture? Explain your answer.

6. Using the basic model of an information system, as presented in Figure 6.16 of the textbook, suggest the components that might be part of an information system that allows users to view information about flights arriving and departing from a particular airport.

7. Using the UML graphical notation for object classes, design the following object classes, identifying attributes and operations. Use your own experience to decide on the attributes and operations that should be associated with these objects:
   - A telephone
   - A printer for personal computer
   - A personal stereo system
   - A bank account
   - A library catalog

8. Identify possible objects in ONE of the following systems and develop an object-oriented design for them. Using a UML class diagram and associated explanation to show your design. You may make many reasonable assumptions about the system when deriving the design.
   - A group diary and time management system is intended to support the timetabling of meetings and appointments across a group of co-workers. When an appointment is to be made that involves a number of people, the system finds a common slot in each of their diaries and arranges the appointment for that time. If no common slots are available, it interacts with the user to rearrange his or her personal diary to make room for the appointment.

   - A filling station (gas station) is to be set up for fully automated operation. Drivers swipe their credit card through a reader connected to the pump; the card is verified by communication with a credit company computer, and fuel limit is established. The driver may then take the fuel required. When fuel delivery is complete and the pump host is returned to its holster, the driver's credit card account is debited with the cost of the fuel taken. The credit card is returned after debiting. If the card is invalid, the pump returns it before fuel is dispensed.

9. For the Observer design pattern described in the textbook, describe a real-world or example application which should be design using the pattern.

10. A small company has developed a specialized product that it configures specially for each customer. New customers usually have specific requirements to be incorporated into their system, and they pay for these to be developed. The company has an opportunity to bid for a new contract, which would more than double its customer base. The new customer also wishes to have some involvement in the configuration of the system. Explain why, in these circumstances, it might be a good idea for the company owning the software to make it open source.

    The key benefits of open source are is that it opens up development to a wide range of developers and so accelerates the development and debugging of the product. Doubling the customer base places immense strains on a small company of they have to take on a lot of new staff and so going open source means that the costs of expansion are reduced.

    In this case, because the product is specialized to the needs of different users, the company that own the software can still charge these users to make the changes to the system. Hence the loss in revenue from selling the software is compensated by the additional effort available to service more customers.

    Furthermore, large companies are often reluctant to buy from small companies who may go out of business, To some extent, open source provides reassurance to customers that, even of the original owners of the software are unavailable, they can get access to the source code and hence continue to maintain their system.

Finally, open source may increase knowledge of the company's product and so attract more customers.

11. It states that "testing can only detect the presence of error, not their absence". Do you agree with this statement? If yes, explain why. If not, use an example to show how?

    Assume that exhaustive testing of a program, where every possible valid input is checked, is impossible (true for all but trivial programs). Test cases either do not reveal a fault in the program or reveal a program fault. If they reveal a program fault then they demonstrate the presence of an error. If they do not reveal a fault, however, this simply means that they have executed a code sequence that for the inputs chosen is not faulty. The next test of the same code sequence with different inputs could reveal a fault.

12. You have been asked to test a method called "catWhiteSpace" in a "Paragraph" object that, within the paragraph, replaces sequences if blank characters with a single blank charater. Identify testing partitions for this example and derive a set of tests for the "catWhiteSpace" method.

    Testing partitions are:
    Strings with only single blank characters
    Strings with sequences of blank characters in the middle of the string
    Strings with sequences of blank characters at the beginning/end of string
    Examples of tests:
    The quick brown fox jumped over the lazy dog (only single blanks)
    The quick brown fox jumped over the lazy dog (different numbers of blanks in the sequence)
    The quick brown fox jumped over the lazy dog (1st blank is a sequence)
    The quick brown fox jumped over the lazy dog (Last blank is a sequence)
    The quick brown fox jumped over the lazy dog (2 blanks at beginning)
    The quick brown fox jumped over the lazy dog (several blanks at beginning)
    The quick brown fox jumped over the lazy dog (2 blanks at end)
    The quick brown fox jumped over the lazy dog (several blanks at end)
    Etc.

13. What is regression testing? Explain how the use of automated tests and a testing framework such as JUnit simplifies regression testing.

    Regression testing is the process of running tests for functionality that has already been implemented when new functionality is developed or the system is changed. Regression tests check that the system changes have not introduced problems into the previously implemented code.

    Automated tests and a testing framework, such as JUnit, radically simplify regression testing as the entire test set can be run automatically each time a change is made. The automated tests include their own checks that the test has been successful or otherwise so the costs of checking the success or otherwise of regression tests is low.

14. As a software project manager in a company that specializes in the development of software for offshore oil industry, you have been given the task of discovering the factors that affect the maintainability of the systems developed by your company. Suggest how you might set up a program to analyze the maintenance process and discover appropriate maintainability metrics for your company.

    This is a very open question, where there are many possible answers. Basically, the students should identify factors which affect maintainability such as (program and data complexity, use of meaningful identifiers, programming language, program documentation etc.). They should then suggest how these can be evaluated in existing systems whose maintenance cost is known and discuss problems of interaction. The approach should be to discover those program units which have particularly high maintenance costs and to evaluate the cost factors for these components and for other components. Then check for correlations.

15. Briefly describe the three main types of software maintenance. Use examples to explain why it sometimes difficult to distinguish between them?

    The three main types of software maintenance are:
    a) Corrective maintenance or fault repair. The changes made to the system are to repair reported faults which may be program bugs or specification errors or omissions.
    b) Adaptive maintenance or environmental adaptation. Changing the software to adapt it to changes in its environment e.g. changes to other software systems.
    c) Perfective maintenance or functionality addition. This involves adding new functionality or features to the system.

    They are sometimes difficult to distinguish because the same set of changes may cover all three types of maintenance. For example, a reported fault in the system may be repaired by upgrading some other software and then adapting the system to use this new version (corrective + adaptive). The new software may have additional functionality and as part of the adaptive maintenance, new features may be added to

take advantage of this.

16. Under what circumstances might an organization decide to scrap a system when the system assessment suggests that it is of high quality and of high business value?

Examples of where software might be scrapped and rewritten are:

a) When the cost of maintenance is high and the organisation has decided to invest in new hardware. This will involve significant conversion costs anyway so the opportunity might be taken to rewrite the software.
b) When a business process is changed and new software is required to support the process.
c) When support for the tools and language used to develop the software is unavailable. This is a particular problem with early 4GLs where, in many cases, the vendors are no longer in business.

17. Do software engineers have a professional responsibility to produce code that can be maintained and changed even if this is not explicitly requested by their employer?

Yes, this is general software quality requirements which all software engineers should follow, unless it is specifically un-requested by the employer with good reason such as rapid development or prototyping.

Write your answers in a MS Word file with name “cs3043a1.doc” or a PDF file with name “cs1043a1.pdf”. Submit the file through the course website.