Home exam 1DV600, 2017:1

Name: Sarpreet Singh Buttar
Personal ID: 950131-0271
Lnu email: sb223ce@student.lnu.se

March 19, 2017

Task 1 - Planning

a) Process models plays an important role in planning. If process model is not relevant to the project needs it can lead to failure because each process model has its work style. Process models are generic means they does not directly help us to solve the problem, instead they reduce the unwanted risks, manage our activities, reduce communication problem and guide us on how to work on the inherited activities(Requirements, Analysis, Design, Implementation, Testing etc) in software development. In addition, selecting an appropriate process model does not guarantee the success because they only manage our work flow, not provide solutions. For instance: after selecting the process model, we still have to work on the inherited activities which means if we are not good in these, process model will not save us from failure. There are many types of process models such as waterfall, Agile, Spiral etc. We must not forget that most of the times we need to tailor the model from abstract to specific model for our project.

b)
1) Early Assumptions: We do not have clear goals when we plan the project. This problem leads to assumptions(such as resources assumption) which further leads to unwanted functionality in the system or complete failure. With the help of iterative and incremental development we can reduce this risk by verify the assumptions at an early iteration.
2) Refinement of product requirement: It is not possible to get a complete definition of system in the beginning of development, so it is more productive to gather the requirements in iterative cycles.
3) Complexity: It is very hard to provide the description of software entity by constructing models because complexity is the essential property which cannot be ignored. From the complexity many problems comes out such as difficulties to communicate with mates, delays, going over budget, product flaws, less understanding and many more. Incremental/Iterative development helps to growing the software system because it grants painless backtracking and early prototypes.

Task 2 - Process models

a)

SCRUM	**Open UP**
Role	
It has 3 major roles named as Product Owner, Scrum Master and Development team	It has many roles such as Business designer, implementer, Tester, Project manager etc.
Artifacts	
It have Product increment(finished part of product), Product backlog(list of prioritized ideas) and Sprint backlog(detail plan for next sprint)	It have many artifacts (such as Use cases, Risk list, Test plan, Software development plan) which are created by responsible roles. Information can be saved formally or informal.
Activities	
It have Sprint Planning, Review, Retrospective and daily meetings with team and scrum masters	It has many activities, normally a role is either responsible for the activity or providing help/information to complete it
In General	
It demands experienced team members	Beginners can easily work with it

b) Assume that I will use Open UP process model for implementing 'Library System' given in the course. We know that Open UP have many roles, artifacts and activities but I do not need all the them because I must work individual. In this case, I will tailor(cut-off) the roles(such as

Business analyst, Database designer etc) that are beyond my scope. In addition, it also demand many artifacts which means I will again tailor the invalid artifacts such as Physical data model, UI design specification etc. For the activities, I have to assign myself both roles (executor and responsible person) rather than playing one of the role. In result, I will tailor the activities which are out of my scope. In conclusion, I tailor down the generic Open UP process model into my specific project model.

Task 3 - Requirements

a) I assume that, we have already identified the functional requirement/s from scenarios. Lets take 'Get Books' function requirement from 'Library System'. Before modelling use case we must know its title, description, initiation, pre-post conditions, flow of events(primary, alternative, exceptional), primary-secondary actor/s, relationships to other use cases. Below is the example of our use case which describes all the required information.

Title	View a list of books
Description	User request all the available books
Primary actor	User
Pre-conditions	Server must be running on port 9090
Post-conditions	User must view a list of book/s
Initiation	User send GET request URI(/api/books)
Primary flow	1. API looks for all books 2. Book/s found 3. Return JSON array of book/s
Secondary flow	2a) No book found 2b) Return an empty JSON array

b)

Non-Functional/Quality Requirements	**Functional Requirements**
These requirements defines how the system should perform a certain function	These requirements defines what the system should perform
These requirements are the left over usability, performance, scalability etc but does not describes business rules	These requirements describes business rules
These requirements are constraints placed all over the system	These requirements are very specific and demanded by the end users

For modelling quality requirements, I will use FURPS+ (Function, Usability, Reliability, Performance, Supportability) technique because these requirements are all over the system and hard to model in the use cases. With the help of FURPS+, I will create a complete list of quality requirements and prepare one or more questions related to it. Later, I will ask these questions to stakeholders in order to find the impact and priority of these requirements. The answers will help me to identify the impacted use cases as well as actors. Then I will add the dependencies to those use cases.

Task 4 - Testing

a)

ID: 1	**Name:** Empty array	**Date:** 19 March 2017
Scenario: User provide an empty array	**Test Data:** []	
Reason: Method should throw exception when array is empty		
Motivate: This is one of rare case, where sorting is not possible		
Steps	**Expected Result**	
Provide 'Test Data' as input	Method should throw an exception	

ID: 2	**Name:** Array is null	**Date:** 19 March 2017
Scenario: User provide a null instead of an array	**Test Data:** null	
Reason: Method should throw exception when null is provided		
Motivate: This is one of rare case, where sorting is not possible		
Steps	**Expected Result**	
Provide 'Test Data' as input	Method should throw an exception	

ID: 3	**Name:** Unsorted array	**Date:** 19 March 2017
Scenario: User provide an unsorted array	**Test Data:** [oranges, grapes, bananas, apples]	
Reason: Method should sort the array		
Motivate: The given array is unsorted, we check if method can sort it		
Steps	**Expected Result**	
Provide 'Test Data' as input	[apples, bananas, grapes oranges]	

ID: 4	**Name:** Sorted array	**Date:** 19 March 2017
Scenario: User provide a sorted array	**Test Data:** [apples, apples, bananas, grapes oranges]	
Reason: Because it is already sorted, so method should return as it is		
Motivate: This is one of rare case, where array is already sorted		
Steps	**Expected Result**	
Provide 'Test Data' as input	[apples, apples, bananas, grapes oranges]	

ID: 5	**Name:** Unsorted mixed array	**Date:** 19 March 2017
Scenario: User provide a regular array	**Test Data:** [TV, Speakers, iphone, *Mac, speakers]	
Reason: Method should sort the array according to standard alphabetical order not ascii		
Motivate: The given array is unsorted, we check if method can sort it		
Steps	**Expected Result**	
Provide 'Test Data' as input	[*Mac, iphone, Speakers, speakers TV]	

ID: 6	**Name:** Array with 1 item	**Date:** 19 March 2017
Scenario: User provide a regular array with 1 item	**Test Data:** [TV]	
Reason: Method should return the array as it is		
Motivate: This is one of rare case where there is no need of sorting		

Steps	**Expected Result**
Provide 'Test Data' as input	[TV]

b) Executing tests automatically with the help of testing tools such as XUnit is called test automation. In test automation, we set the expected result and the tool which we use compare the actual result with it. The automated test only pass if the actual result is same as expected result. In cases of failure, we get an error message from test tool.

Usage: Creating automated tests are widely used during unit testing because it saves time and we can repeat it as many times as needed in a very short time. Apart from unit testing, we can also create it in the Regression testing, Smoke testing, Static and repetitive testing etc. In addition, where result is not clearly visible to human eye such as testing minor color difference, automated tests are better than manual testing.

Non-Usage: Creating automated tests for a system functions that are still under development may be a wastage of time because if functionality changed, automated tests will be invalid. Moreover, they are also not worthy in subjective validation such as in some cases (where result is user friendly) if expected result is 8.1 but the actual result is 8.0000000011 then testing tool will fail it. However, as humans we can pass it because its not completely wrong. In addition, manual tests are better than automated in the case of testing usability.