NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY, KARACHI

MASTER OF (COMPUTER SCIENCE AND INFORMATION TECH.) SPRING SEMES EXAM -2007

(EVENING PROGRAMME)

SOFTWARE QUALITY ASSURANCE - (CT-534)

Dated: 19-05-2007

Time: 3 Hours **Max Marks: 70**

Instructions:

- This paper consists of two sections : Section A and Section B
- Section A consists of 2 Case Studies
- All 10 questions should be attempted.
- Read both the case studies thoroughly before answering.

SECTION A: CASE STUDY (I)

1. What is Extreme Programming? Explain some of its basic features?

[5]

2. What does the term "collaborative approach" means in the case study? Explain.

[5]

3. Discuss the following models
   a. Waterfall model
   b. Prototyping model, as described in the case study

[10]

4. *"The lack of documentation coming from the team began to cause strife (trouble)".*

Why was there a lack of documentation by the team? In your opinion how then was the team documenting the project?

[5]

5. *"How could XP ever work in this kind of scenario?"*

In your opinion what methodology would have worked best? Explain your answer by explaining the methodology in detail you suggest.

[5]

SECTION A: CASE STUDY (II)

6. What were the problems faced by the clients in the first phase of the project? And how, in opinion could they have been avoided?

[5]

7. What is Pair Programming? What advantages and disadvantages of Pair Programming were felt in the project?

[5]

8. *"Pair Programming, coding standards, and test first were all contributors to the improvement in software quality"*

To improve the quality, can Pair Programming exist without coding standards and test first? Explain your answer.

[10]

SECTION B

9. What do you understand by Software Configuration Management? Explain briefly how you can achieve it.

[10]

10. What are black box and white box testing techniques? Explain them in detail

[10]

SECTION A: CASE STUDY (I)

The Importance of Keeping Your Selling and Delivery Methods Synchronized

This case study confirms the importance of keeping your selling and delivery methods synchronized. The way we sell XP is a free sample of the way we will work with XP. This case study is an example of how mixing a noncollaborative sales approach can hinder attempts to have meaningful collaboration with your customer during delivery.

The customer was a public-sector agency that required a system to aggregate data from partner agencies and present a single view of that data. The agency went to the market through a Request for Proposal (RFP) cycle and, as is normally the case with such tenders, limited communication to formal channels. The consulting firm that won the work presented a modified version of XP as their delivery approach. The response itself contained no mention of XP, per se, instead referring to a "***collaborative approach***." The idea was that the core work of the project (software development) could be wrapped within a standard sign off, ***Waterfall method***. The sales cycle did not involve any form of collaboration or discussion with the customer beyond clarification of technical and contractual issues.

The customer was very excited by the concept of collaborative development even though he had no clear understanding about what this meant in real, day-to-day terms. It became clear that his experience of collaboration was limited to rudimentary ***prototyping***. In fact, they were expecting a prototype-driven approach where the team effectively "made up" the initial interfaces, and then presented them to the client for feedback. The customer had no interest in interacting with the team during development and was content to communicate requirements through these prototypes.

On one side the development team was struggling to use XP-like methods, while the customer refused to commit time to working with them. ***The lack of documentation coming from the team began to cause strife(trouble)*** because the customer had an expectation that standard specifications and designs would be included. Frustration continued as the customer complained about the developers' use of automated unit testing with no formal test plans. Documentation was contained in the Wiki Wiki Web, and the idea of a paperless project was hard for the customer to comprehend.

The software engineering rigor that the development team was applying to the work was impressive to the customer, yet the customer felt uncertain about the process aspects of XP. In the end, a compromise was reached—the developers continued to pair and write their unit tests, but requirements were managed through a mixture of functional specification and prototypes.

It was easy to see the correlation between the way the customer interacted with the team during delivery, and how the RFP process was managed. As is often the case in the public sector, controlling risk and conformance to internal project management standards were

key drivers. The project was deemed a success, and rightly so, as the system met client expectations and budgets. From the development teams' viewpoint it was exasperating(annoying) to work with a noncollaborative customer, and on the whole it was a poor XP experience.

This was a fixed-price contract with payment intervals linked to deliverables, most of which were documents not software. The team was forced to retrospectively write documents to get paid! ***How could XP ever work in this kind of scenario***?

Lessons Learned

Here is a summary of lessons learned from this project:

- Clarify exactly what your XP approach will mean to the customer. The more common the word (such as, collaboration) the more likely that assumptions will be made and a communication breakdown and confusion will occur.
- Start the way you intend to precede; collaborate early with your customer.
- Write the contract so it will fit your development approach. In our case study, the contract tied payments to deliverables that don't exist in XP!

SECTION A: CASE STUDY (II)

The Result of Changing to XP in Midstream

You can begin using XP at almost any point in your project lifecycle. We'll spend some time analyzing how a consulting firm switched to XP halfway through its development. The change didn't pass without some degree of pain, but the end results were quite remarkable.

In this case study, our client was a large business-to-business (B2B) aggregator. The role of this organization was to provide a centralized meeting place for business across various industry verticals. They had a reputation for innovation, which definitely helped when XP was introduced.

The client commissioned a new Web site that would support their model for industry innovation—matching buyers and suppliers. The contract was signed on a fixed-price basis, with delivery through a series of discrete phases. The initial price was fairly much "back of the envelope" in nature, based on high-level assumptions.

The first phase of the development continued along standard Waterfall or Staged Delivery lines, with time expended at the start on infrastructure and planning issues. Daily builds were a foreign concept to the team, which resulted in the usual mad panic at release time. In reality, the deadline was artificial in nature and exposed poor estimations. The team did meet the deadline at the cost of both morale and lower-than-expected quality. With more than 100 defects found during acceptance tests, the team spent the next few weeks really finishing the release. Not a very pleasant experience!

Phase two saw the introduction of XP into the development cycle. Senior management from the supplier-side spent focused effort to soften up both their own management and the customer to using XP. This was time well spent, cementing the support from the customer and management before exposing the team to XP. The name "Extreme Programming" was downplayed with locally acceptable words used instead.

The XP implementation was biased toward the programming practices as opposed to the process components (such as the Planning Game). The XP champion took the stance that immersion was the best approach, and rollout to the team moved quickly when management was on board. Practices used by the team included

- Pair Programming
- Test First (unit testing)
- Collective Code Ownership
- 40-hour week
- Small Releases
- Planning Game (variation on this)
- Coding Standards

The use of Pair Programming was seen as being essential to the success of this XP project; so much so that the team leader mandated its use with no exceptions. It was either commit to Pair Programming or leave the project. The downsides to this dictatorial method were self-evident: Junior developers generally shrugged their shoulders in acceptance, while more senior members were shown the door. In one case the developer's disaffection with the whole process led to him leaving the company along with his experience and skill. On paper the use of XP practices, such as Pair Programming, had a high-acceptance rate, but behind closed doors there was a degree of grumbling and complaining. Interviews with various team members confirmed this.

Pair Programming, coding standards, and test first were all contributors to the improvement in software quality. One developer commented, "at the start you could tell who wrote what piece of code, but by the end of the phase it seemed like one mind was at work." This homogeneity in source code helped with source code quality as well as code adherence; there was one way of doing this now and that was the team way. To give you a sense of software quality; by the end of the three-month cycle there were three small defects uncovered in acceptance testing. Quite a change from the hundred or more in phase one!

The lack of defects at release was even more remarkable if you consider the poor working conditions the team was forced to endure. Classic software development thinking directly links quiet, peaceful working environments to better quality software. It would appear that the other XP practices used by the team had counteracted the lack of suitable workspace.

A funny endnote is that some developers missed the panic of a standard project! On the final day, the team went home at around 3:00 p.m. and that was that! Panic and chaos were not things missed by the customer. For their part, customers loved the control they were given by the XP process and the visibility of the software. Both customer and supplier characterized the project as a great success. Developers were split on whether it was a good experience or not, we can surmise that perhaps the heavy-handed implementation approach was a factor.

Lessons Learned

Here is a summary of lessons learned from this project:

- Take time to educate both your management and customer; let them know exactly what they're getting into.
- Realize that your people are your greatest resource and try to be gentle as you rollout XP.
- Work very hard to get your first XP project right from both customer and management views. Plan to ask your customer to act as a reference site.
- Expect and accept your own local variations of some of the XP practices.