**Software Engineering Q/A Sheet (#4)**

date: 2023-10-02 number:2018312280 name: 이상수

**Questions from Prof.**

1. Describe why rapid development and delivery are important from a business and technology perspective.

오늘날 사업은 빠른 속도로 변화하는 requirements에 맞춰야 합니다. 이 때 기존의 안정적인 software 개발 방식은 이에 대응하기 어렵습니다. 또한 software는 변화에 빠르게 대응해서 진화해야 합니다.

1. Explain the principles of agile method with reasons.

Customer involvement : 고객은 개발과정에 참여하고 요구사항 우선순위를 정하고 매 시스템을 평가합니다.

Incremental delivery: incremental하게 개발하고 배포되며, 여기에 고객의 각 요구사항이 들어갑니다.

People not process: 특별한 개발 과정보다는 개발자 개개인의 개발능력이 더 중요시 됩니다.

Embrace change: requirement가 변화되면 이에 대응할 준비를 합니다.

Maintain simplicity: software 개발시 software는 항상 간결하게 유지되어야하고 되도록 복잡한 작업은 안해야합니다.

1. Explain the strengths and problems of agile methods.

장점 : 변화에 빠르게 대응할 수 있습니다. 소프트웨어가 간단하게 유지되기에 변화에 대응하고 좋습니다.

소프트웨어를 간단하게 유지하다보니 refactoring도 자연스럽게 하게되고, 소규모팀이기에 팀원들간에 소통도 원활합니다.

고객이 개발에 참여하여 고객에게 신뢰도를 높이기 좋습니다.

단점 : 고객이 개발에 계속 참여하기 어렵습니다.

팀원들의 개발 역량이 중요합니다.

stakeholder가 많아지면 priority를 정하기 어려워집니다.

simplicity를 계속 유지하려면 추가적인 작업을 해야합니다.

초기 계약에서 iterative에 필요한 내용을 모두 정하기 힘듭니다.

1. Summarize the determinants when deciding on a plan-driven or agile approach to system development.

1. specification , design이 detail 해야하는 경우는 agile보다 plan-driven이 어울린다.

2. 점진적 배포를 하고 손님의 feedback을 빠르게 수용하기가 현실적이라면 agile method가 좋다

3. 시스템의 크기 고려, 소규모의 소통이 쉬운 가까운 팀인가 ? 큰 팀, 큰 규모의 시스템이면 plan-driven이 어울린다.

4. 개발전에 많은 사전 분석이 필요한가? 그렇다면 plan-driven

5. 오래 사용될 시스템이라면 문서가 많이 필요하기에 이럴때는plan-driven이 좋다.

6. 어떤 기술들이 사용가능한가? 기술이 agile에 충분히 좋아야한다.

7. 개발 팀들이 분산되어있거나, outsourced이면 이러한 사람들끼리 소통할 문서가 필요하다.

8. 조직의 문화가 plan-driven이 어울릴 수 있다. 특히 오래된 조직

9. 개발자들의 능력이 아주 좋아야한다.

10. 외부 규제들이 많으면 이를 고려한 설계를 해야하기에 agile을 쓰기 어렵다.

1. Explain the release cycle of XP and compare it to the waterfall model.

1. 우선 user stories를 정함 (일정 기간동안 무엇을 개발할지)

2. 이 story를 task 단위로 나눔 : 개발부터 평가까지의 과정을 단계로 나눔

3. 어떻게 배포할지 정함

4. 실제 개발, 통합, 테스트

5. 배포

6. 평가

다시 1로 반복.

waterfall 은 모델상으로는 한번에 개발을 끝내는 것을 목표로 하고 요구사항을 파악하여 개발이 모두 끝나면 바로 maintenance로 넘어가는데 반해, XP모델은 개발을 한번에 끝내지 않고 한번 개발이 끝나면 다시 처음으로 돌아가서 다음 개발을 진행합니다.

1. Explain the important features on practices of XP.

incremental planning : requirements는 story로 만들어지고, 이를 개발진은 tasks로 나눈다.

small releases : 작은 단위로 개발하여 빠르게 자주 배포한다.

simple design : 현재 requirement만 만족하는 만큼의 design만한다.

test-first development : 자동화된 test를 function개발전에 만들어 새로 개발하는 function은 이 테스트를 통과하도록 개발한다.

Refactoring : 모든 개발자들은 지속적으로 최대한 빨리 refactoring을 하여 코드를 간단하고 유지보수 쉽게 유지한다.

Pair programming : 두명이 1개의 프로그램 작성하며 서로 협력한다

Collective ownership : 모든 사람이 모든 개발에 참여한다. 그래서 ownership이 생기고 어떤 코드이든 수정할 수 있다.

Continuous integration : 지속적으로 개발하고 task가 통과되면 통합하고 테스트한다.

Sustainable pace : 지속적으로 일 할수 있게 업무량 조절 등을 함

On-site customer : 실시간으로 손님이 개발에 참여하여 요구사항을 가져다준다.

1. Read the story card on slide 22 carefully and look for another task.

task drug 찾아 고르기 : ‘new medication’을 누르면 해당 화면으로 이동. 새로운 drug의 이름 앞에서부터 검색한 글자가 순서대로 포함이 되는지 확인하여 해당하는 drug들을 밑에 띄워 알려주기. 원하는 drug를 고르면 원하는 drug가 맞는지 한번 더 확인하고 최종선택을함

1. Explain test-driven development and think about its impact on SW quality.

개발전에 test코드를 만들어서 이를 자동화하는 방법입니다. 이렇게 만들어진 test를 통과하도록 개발을 하고, 새로 추가되는 test는 이전의 test들과 합쳐지며 매번 이전 시스템의 test도 같이 진행하는 개발방법입니다.

이렇게 진행을 하면 test가 잘 만들어진다는 가정하에 requirements를 놓치지 않고 만족할 수 있으며, test케이스를 잘 만들면, error가 일어날 경우를 미리 방지할 수 있습니다. 또 한 테스트를 자동화하였기에 개발과정에서 test에 드는 시간이 줄어들고 test에 집중하여 불필요한 코드를 작성할 일이 줄어들기에 개발속도도 빨라집니다.

1. Explain the advantages of pair programming.

pair programming은 pair의 프로그래머가 하나의 코드를 보며 매번 다른 코드를 돌아가면서 확인하기에 common ownership이 생깁니다.

많은 사람이 코드를 review해주기에 refactoring을 자연스럽게 하게 됩니다.

1. Explain the SCRUM process on slide 40.

Review work to be done에서 해야 할 일을 review하고,

to do list같은 Product backlog가 작성됩니다.

이번 cycle에 필요한 item을 선택하고 plan을 합니다.

그리고 해당 sprint cycle에서 해야 할 sprint backlog를 만듧니다.

그걸 기반으로 sprint가 돌아가고 이렇게 진행되는 sprint도 매일 review를 합니다.

이 sprint를 review하다가 sprint에 추가, 수정해야 할 일이 생기면 맨 앞으로 돌아가서 해야할 일을 review하고 또 한번 sprint cycle이 진행됩니다. 이렇게 product backlog의 모든 작업이 완료 될 때 까지 반복적으로 진행이 됩니다.

이렇게 하나의 sprint가 잘 수행이 되면 shippable한 software가 만들어집니다.

1. Summarize the benefits of the SCRUM.

product를 여러 개의 관리되고 이해쉬운 chunk로 나눠서 진행한다.

불명확한 requirements 에 진행이 영향을 적게 받는다.

모든 팀원들이 대부분의 작업에 visibility를 가지게 되고, 이는 팀 communication도 향상 시킨다.

고객은 실시간으로 increment의 배포를 받아 이에 대한 feedback을 줄 수 있다.

고객과 개발진의 신뢰가 상승한다.

1. Explain the characteristics of large systems development.

분할된 팀들이 communicate하여 소통이 어렵다.

'brownfield system' 이미 존재하는 여러 system과 interact하기에 개발에 고려해야할 사항이 많아져 incremental development 의 flexibility를 이용하기 어렵다.

system 구성을 잘 살펴야 한다. 개발해야 할 시스템중에 시스템의 구조에 영향을 많이 받는 시스템들이 있다.

여러가지 rule, regulation 이 개발방법에 제한을 가져온다.

개발 시간이 오래 걸린다. 이에 따라 팀원들이 바뀔 확률이 높고 그러면 개발에도 차질이 생긴다.

다양한 stakeholder가 있다. 그렇기에 개발에 모든 고객들이 참여하기가 어렵다.

Questions from your ownself

1. 반대로 test를 개발후에 만들면 장점이 무엇일까?

개발을 하면서 test에 어떠한 내용을 넣을 수 있을지 생각해볼 수 있다. test를 잘못 만들어서 개발에 악영향을 끼칠 일이 없어진다.

2. 고객이 개발에 참여하면 문제점은 없을까?

대표적으로 게임 산업에서 고객들이 개발에 직접적으로 개입하며 예전에 소통하는 게임이 유행했습니다. 그러나 각 고객별로 게임에 원하는 점이 다 다르고, 어떠한 경우는 오히려 자기가 원하는 요구를 들어주지 않는다고 게임사에 부정적인 여론을 조성에 게임의 인기 자체에 악영향을 끼치는 경우들도 있었습니다.