Q/A Sheet #4 – Agile software development

Date: 3/31 number: 2016311821 name: 한승하

Questions from Prof

1. Discuss and write your own ideas on why rapid development and delivery is important from a business and technology perspective

   ⇨ 이전 강의 들에서 Prototype이 필요한 이유와 일맥 상통할 것 같다. 빠른 development와 delivery는 Stakeholders의 Requirement를 더 명확히 하하는 것에 도움이 되며, Error을 찾는 것 에도 유리하다. 이는 Business 적으론 이후 Operation중의 Rework Cost을 줄일 수 있으며, 더 Customer들이 만족하는 결과물을 도출하는 것에 유리하다. Technology 측면에서도 빠른 개발과 배포에서 오는 명확한 Requirement들은 Subsystem 들을 분산 계획하기에 더욱 용이 하며, 이는 Parallel development에 큰 이점이 된다.

2. Explain the principle of agile methods with each reason

   A. Customer involvement

   ⇨ Customer는 development process에 포함되어 있어야 하며, 그들의 역할은 새로운 system requirement들을 정의하고, 우선순위를 정하며 system을 평가하는 일이다. System development라는 것이 궁극적인 목표가 Customer의 needs를 충족하는 것이 목표인 만큼, Customer가 그들의 needs를 명확히 제공하는 것이 무엇보다 중요하다. 따라서 Customer는 system개발에 포함되어 있어야 한다.

   B. Incremental delivery

   ⇨ Software는 점진적으로 개발되어야 하며, 각 개발과정은 Customer가 개발로 인해 추가된 requirements에만족하도록 진행되어야 한다. 이는 Agile method의 핵심으로 보이며, 매 Subsystem 혹은 개발과정이 끝날 때 마다 빠른 배포로 Customer에게 feedback을 받아 가는 과정으로 이루어 져야 Incremental development가 의미가 있을 것이다.

   C. People not process

   ⇨ 각 팀원들의 Skills 들은 공개되어지고 모두가 알고 있어야 한다. 각 팀원들은 각각의 개발을 해야 하는데, 이때 각 개발된 Subsystem들의 interface을 맞추기 위해, 또 하나의 System으로 합치기 위해 각 팀원들에 대한 파악과 거시적인 개발 그림이 필요할 것이다.

D. Embrace change

⇨ System requirements 들이 바뀔 것 이라 가정하고 개발 해야 하며, Design system 은 이러한 변화들을 수용하는 방향으로 개발되어야 한다. Agile development가 빠른 배포와 빠른 Feedback을 목표로 하는 만큼, System은 언제든지 변화될 것이라 가정하는 것이 일반적일 것이다.

E. Maintain simplicity

⇨ Software와 development process는 가능한 simple하게 구성되어야 하며, 이를 위한 activity들이 수행되어야 한다. 많은 변화를 수용하는 개발과정인 만큼, 쉽고 간단한 system이 더 변화를 수용하기 쉬울 것이다.

3. Discuss the strengths and problems of agile methods

⇨ Customer가 Project에 오래 참여하게 하기가 어렵다.

팀 내에서 Intensive involvement를 오래 유지시키기가 어렵다.

여러 개의 요구사항에서 또 여러 stakeholder들 사이에서 우선순위를 정하기가 어렵다.

Refactoring 자체도 추가적인 시간과 작업을 요구한다.

명확한 순서가 있는 개발이 아니기 때문에 계약조건을 정하기가 어렵다

개발과 배포가 빠르기 때문에 위의 1번 문제에서의 강점들을 가지고 있다.

Customer의 요구사항을 보다 잘 받아들일 수 있으며, 변경 또한 용이하다.

팀 생산성이 증가한다.

프로젝트의 가시성 (간단, 명료함) 이 개선된다.

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

⇨ System이 Detailed한 명세서 혹은 Design, 문서가 필요할 경우 agile로 접근하기가 어렵다.

Customer가 Development에 지속적으로 참여할 수 가 없다면, agile로 접근하기가 어렵다.

Agile방법론은 Small scale system, Small scale team에 좋기 때문에 Global 혹은

넓게 분산된 팀의 작업, 혹은 매우 큰 System에 경우에 agile방법론을 적용하기가 어렵다.

Real-time system과 같은 implementation이전에 많은 분석이 필요한 system의 경우 agile방법론을 적용하기가 어렵다.

Lifetime이 긴 System의 경우 더 정교한 design 문서가 필요하기 때문에 agile로 접근하기가 어렵다.

Agile방법론에 필요한 기술적인 환경 (지속적으로 design을 수정하고 보안하기 용이한 좋은 툴들) 이 부족한 경우 Agile방법론을 적용하기가 어렵다.

Development team이 독립적으로 분산되어 있거나, development의 일부를 outsourcing 할 경우 Agile 방법론으로 적용하기가 어렵다.

Traditional engineering organization이 존재할 경우 새로운 방법론 (Agile)을 적용하기가 어렵다.

팀의 구성원들이 충분히 실력이 있지 않다면 agile방법론 보다 plan-based 방법론이 훨씬 좋다.

System이 외부적인 규제를 크게 고려해야 하는 상황이라면 Agile 방법론을 적용하기가 어렵다. (더 상세한 문서가 요구될 수 있기 때문)

5. Discuss the release cycle of XP and compare it to the waterfall model

   ⇨ XP는 Select user stories for release -> Break down stories to tasks -> Plan release -> Develop / integrate / test software -> Release software -> Evaluate system을 반복하는 Cycle을 가진다.

   Waterfall model과의 비교를 하자면, XP는 한번의 Cycle로 개발을 끝내는 Waterfall model과 다르게 Stories가 동날 때 까지 반복되는 model이라는 점.

   시작시에 Requirements를 모두 명세화하고, 전체 System을 고려해 Plan을 짜는 Waterfall model과는 다르게 전체적인 Design과정이 존재하지 않으며, 각 Requirement의 개발 시작시에 각각의 Plan을 수립한다는 점.

   Release 과정이 개발 중에 반복적으로 있다는 점. 등을 이야기 할 수 있을 것 같다.

6. Summarize the important features on practices of XP

A. Incremental planning

⇨ Requirements 들을 User Story로 관리하며, 각각의 Story에 대한 개발이 필요할 때에 Task로 나누어 Planning 하는 것

B. Small release

⇨ 최소의 business적 가치가 있는 useful set of functionality들이 먼저 개발되어 지며, 선 배포 후 통합의 과정을 거친다.

C. Simple design

⇨ 현 개발에 필요한 Design은 충분하게 하지만 그 이상을 Design을 하지는 않는다.

D. Test-first development

⇨ Test Code를 먼저 작성을 하고, Application code / function code를 작성을 하여 Test code를 사용하여 작동시키는 방식을 말한다.

E. Refactoring

⇨ Code를 가능한 간단하고 가시적이도록 Refactoring 과적을 거친다.

F. Pair programming

⇨ 두 사람이 Driver와 Observer로 역할을 나눠 Pair로 작업하는 것

G. Collective ownership

⇨ Pair programming의 장점 중 하나로써 조직에 존재하는 모든 코드를 조직에 있는 모든 작업자가 알 고 있게 되며, 이는 중간에 담당자 중 한사람이 빠지게 되더라도 양질의 개발 작업을 이어갈 수 있다.

H. Continuous integration

⇨ Task가 완료될 때 마다 전체 System에 통합된다. 통합 이후엔 system의 모든 unit이 test를 통과하여야 한다.

I. Sustainable pace

⇨ 큰 추가작업은 허용되지 않는다. 이는 이것이 Code quality 저하와 medium term productivity를 불러올 수 있기 때문이다.

J. On-site Customer

⇨ 개발을 요청한 Customer는 늘 XP team과 작업이 가능한 상태여야 한다.

7. Read the story card on slide 22 carefully and look for tasks 4 or above

   A. New medication

      i. Finding a drug => The system should list the possible drugs starting with the letters typed

      ii. Medication check => If the customer select a medication, the system will show a page to check if the selection is correct

   B. Formulary

      i. Drug search => System should provide a search box to search the right drug and add it to the formulary

      ii. Formulary confirmation => The system should provide a page to check if the write medication is in placed

8. Explain test-driven development and think about the impact on SW quality

   Test-first development라고도 하는 TDD기법은 Application/product code 작성 이후에 test case / code 를 통해 testing하는 기존의 기법과 다르게 Test code를 먼저 작성하여 data중심이 아닌 code중심의 test로 모든 test가 자동화 가능하다 라는 특징을 가지며, Incremental development기법에서 TDD를 사용하면 매번 새로운 development에 대한 testing이 자동화 되어 매 개발 이후에 testing과정이 편리 해진다.

   Programmer들은 testing보단 programming에 더 집중하는 경향성이 있기 때문에 TDD는 programmer들이 더 testing에 신경 쓰도록 할 수 있으며, 이는 SW quality를 높이는 효과를 가져올 수 있다.

   다만 이러한 방식의 Testing은 개발 과정에서 혹은 subsystem을 합치는 과정에서 생기는 의문점들, 우려되는 부분들, 등을 반영하여 혹은, acceptance testing은 사전에 testing code를 작성하기에 어려운 부분들이 있기에 SW quality는 일반적인 testing 보다 떨어질 것으로 우려된다.

   또한 TDD는 interface에 대한 testing, 혹은 전체 system에 대한 testing이 어렵다는 단점이 있다.

9. Summarize the advantages of pair programming

   A. Pair programming의 장점 중 하나로써 조직에 존재하는 모든 코드를 조직에 있는 모든 작업자가 알 고 있게 되며, 이는 중간에 담당자 중 한사람이 빠지게 되더라도 양질의 개발 작업을 이어갈 수 있다.

   B. Observer는 process을 계속 관찰하기 때문에, 개발 과정에서의 review가 가능하고 이로 인해 개발 과정에서 refactoring또한 이점이 될 수 있다.

10. Explain the SCRUM process on slide 40

   ⇨ Agile 방법론은 Agile에 특화되어 있는 프로그램 관리 방법론이 필요하며 대표적인 방법론이 SCRUM이다.

   SCRUM은 Outline planning phase -> A series of sprint cycles -> Closure phase 3개의 phase로 이루어져 있다.

   A. Outline planning phase는 여러 장의 Story card중 Story card를 명세화하고 Project의 목표와 Software architecture을 design하는 phase이다.

   B. Sprint cycles은 앞에서 만들어진 Story cards들을 Priority에 따라 선택, 개발, review, 평가를 하는 단계로, Story cards이 모두 사라질 때까지 반복된다.

   C. Closure phase는 sprint cycles가 모두 끝나면, 마무리와 필요한 문서를 만들어 내는 작업이다.

11. Summarize the benefits of the SCRUM

   ⇨ 전체 Product이 관리가능한 조각들로 나눠져 관리가 편하다.

   불확실한 Requirement들은 작업을 요구하지 않는다.

   전체 팀이 모든 Progress에 대해 인지할 수 있으며, Team간 소통이 향상된다.

   Customer들은 increment한 개발과정을 배포 받아 확인할 수 있으며, product에 대한 feedback을 축적할 수 있다.

   Customer와 developers간의 신뢰성이 향상된다.

12. Summarize the strategies for scaling up and scaling out of agile methods

⇨ Scaling up 은 large software system개발에 있어 agile 방법론을 적용하기 위한 방법으로 Small team이 agile방법론으로 large software system을 개발하기 위한 방법이고

Scaling out은 large organization에서 agile방법론을 사용하기 위한 방법이다.

Questions from Your ownself

1. Explain Product backlog

   A. Product backlog은 Project에서 해결해야 하는 Product list이며, 해야 되는 일들의 list이다. 이를 Sprint backlog로 분해하는 작업을 해야 한다.

2. Explain about Scrum master

   A. Sprint cycle은 외부와의 간섭이 없는 것이 원칙이며, 이때 전체적인 팀 관리와 외부와의 interface역할을 하는 것이 Scrum master이다. Sprint의 말미에는 customer들과 소통을 통해 review가 되어야 하는데 이러한 일도 Scrum master의 업무이다.

   Daily meeting, tracking backlogging, recording decisions, measuring progress또한 Scrum master의 업무이다.