QA sheet 3

2017310695 이주민

1. Explain the strengths and weaknesses of the Waterfall model

- 장점 : plan-driven한 방법으로 진행되어 parallel development 가능함

  미리 전 단계를 계획하므로 문서화하기가 쉬움

- 단점 : 피드백에 대한 반복 단계가 어려움.

  일정 시간 내에 feedback 은 수정할 수 있지만, 테스트 단계에 발견된 중요 결함에 대한 대응이 어려움.

그럼에도 large project에서 쓰기 위해서는 Component와 component들 사이에 어떤 관계가 있는지 설계되어 있는 architecture 필요

2. Explain the strengths and weaknesses of the incremental model

- 장점 : 고객의 요구가 바뀌었을 때 부분적으로만 재작업을 하므로 드는 cost가 적음

  고객 입장에서 더 빠른 feedback을 얻을 수 있음

  부분적으로 만들어서 고객에게 빨리 배포할 수 있음

- 단점 : visibility는 documentation에서 얻을 수 있는데 incremental model은 specification, development,

  validation이 계속 반복되므로 이때마다 문서를 생성하기 힘들어서 process가 invisible하여 evolution

  과정에서 문제가 생길 수 있음

  새로운 increment가 추가될 때마다 structure가 무너짐

만약에 대규모 프로젝트인데 요구사항이 명확하지 않다면,

폭포수 모델과 진화적인 개발 모델을 혼합해서 사용 (HYBRID-DEVELOPMENT)

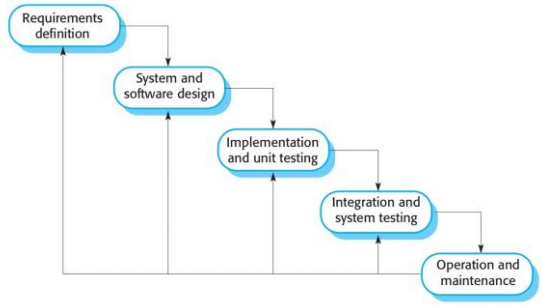
: 팀마다 시스템의 서로 다른 부분을 동시에 개발하여 대형이면서, 복잡하고, 생명이 긴 시스템의 경우 진화적 방법을 사용하면 팀의 산출물을 통합하는 것이 어렵고 안정적인 시스템 아키텍처를 갖기가 어렵다. 요구사항이 간단한 형태로 되어있으므로 폭포수 모델만으로는 불충분하고 risk manager 의 부재로 spiral model 도 사용이 어렵기 때문이다. 시스템 명세서의 불확실성을 해결하기 위해서 진화적인 방법을 사용하는 쓰고 버리는 프로토타입의 개발을 포함한다. 그리고 좀 더 구조적인 방법을 사용하여 시스템을 재구현한다. 잘 이해되는 시스템 부분은 폭포수 기반 프로세스에 의해 분석하고 개발한다. 미리 분석하기 어려운 사용자 인터페이스와 같은 시스템의 다른 부분은 실험적인 프로그래밍 방법을 사용하여 개발한다.

3. Look for refactoring concepts and find examples of them.

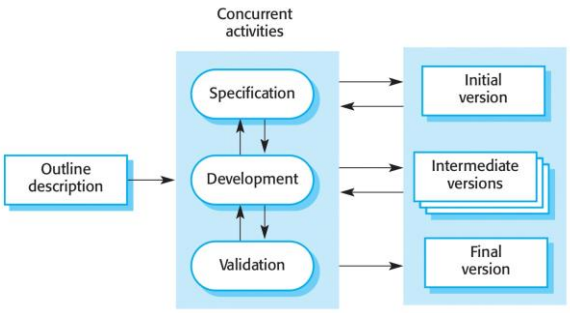
refactoring이란 적은 비용으로 수정할 수 있도록 겉으로 보이는 동작(입출력은 같음, 다른 프로세스랑 연결되는 것은 유지됨)의 변화없이 내부 구조를 변경하는 것으로 이해하기 쉽고 유지 보수하기 쉽게 만드는 것이 목적임

ex) 코드명을 바꾸는 것, 조건문의 단순화, 함수명을 겹치지 않도록 하는 것, architecture refactoring

4. Compare the waterfall model with the incremental model for the workflow



< waterfall model >



< incremental model >

5. Describe four fundamental process activities.

- Specification : 요구사항 명세화, 시스템에서 요구되는 서비스는 무엇이고, 기능들이 무엇이 있는지, 시스템 운영과 개발에서 반드시 만족시켜야 하는 제약 사항은 무엇이 있는가를 명확하게 정의하는 단계

- Software design and implementation : design은 명세서를 실현하기 위한 소프트웨어 구조를 설계하고, implementation은 직접 코드로 실행가능한 프로그램으로 변환하는 단계

- Verification & Validation : Verification은 만든 시스템이 명세서와 일치하는가를 확인하는 작업으로 checking, review, inspection으로 가능함. Validation은 만든 시스템이 고객의 실제 요구사항을 충족하는지 확인하는 작업으로 testing으로 가능함. 이때 test data는 (input data, expected output data)로 구성됨.

- Evolution : 고객에게 전달 후 실제로 사용하는 단계로 operation, maintenance가 포함됨.

Constraint를 제대로 찾고 분석하는 것이 중요한 이유는 고객의 요구사항이 이와 같을 때에는 문제가 심하게 생길 수 있기 때문임

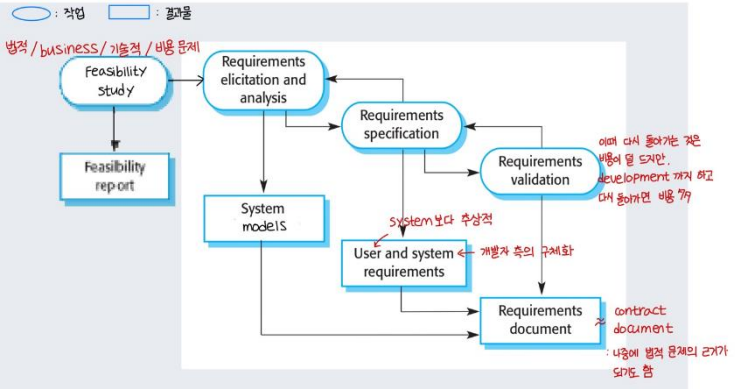
6. Describe the requirement engineering process with key activities and their work products.

- feasibility study : 비즈니스적으로 가치가 있는지, 기술적으로 타당한지, 예산이 충분한지, 법적으로 문제가 없는지 등의 관점에서 할 만한 가치가 있는지 타당성 조사를 함

- requirements elicitation and analysis : 이해당사자, stakeholder로부터 요구사항 추출 후 분석함.

- requirements specification : 요구사항을 명세화함

- Requirements validation : 빠진 것은 없는지 실현 가능한 것인지 등의 관점에서 요구사항을 확인하는 과정



7. Describe the differences between component, system and acceptance tests.

- component(=unit =module) testing : component 개발자에 의해 개별적으로 이뤄짐

- system testing : component들이 모여서 처음으로 emergent properties인 비기능적인 요구사항을 Test data를 사용하여 처음으로 테스트 할 수 있음.

- acceptance testing(인수시험) : 고객이 인수할지 안 할지 테스트하는 작업으로 고객의 real data를 사용함

8. Slide on page 28, discuss when to plan each test with why.

윗단이 개발, 아래단이 test가 이루어지는데, 요구사항 명세서를 만족하도록 제품이 만들어졌는지 알아보기 위한 test를 알맞게 설계하기 위해서 개발 단에서 이루어져야함. 또한, 개발 단에서 다음 단계로 넘어갈 때 test를 설계하는 것이 도움이 됨.

9. Change is inevitable in every large software project. There are two related approaches you can use to reduce rework costs. First, explain the concept of rework and describe two approaches.

- Rework : 비즈니스 환경의 변화, 새로운 기술, 플랫폼의 출현, 경쟁사의 제품 출시 등으로 인해 생기는 요구사항에 대한 재분석, 새로운 기술 적용 등 소프트웨어에 필요한 changes를 때문에 다시 일하게 되는 비용

- change avoidance: changes를 피하는 방법으로 명확하지 않은 부분이 시스템에 들어가면 나중에 change를 하게 되는 이유가 되므로 개발 단계에서 많은 부분을 보다 명확하게 해야한다는 접근. 이를 위한 방법으로 prototype system을 만들어 빠른 피드백을 받아 불명확한 요소를 제거함

- change tolerance: changes는 어쩔 수 없이 생기기 때문에 그것을 어떻게 효율적으로 수용할 수 있게 만드는가를 생각하는 접근. 이를 위한 방법으로 incremental development를 통해 부분적으로 나눠서 개발하고 평가하는 작업을 반복하는 것을 통해 cost를 줄일 수 있음. (또 다른 change가 생겼을 경우 다음 increment에서 보충을 하면 되기 때문에)

10. Discuss the benefits of prototyping. And explain the reasons for improved maintainability on page 38 of the slide.

Prototyping도 돈이 꽤 많이 들며, 인력이 필요한 일임. 그럼에도 불구하고 개발 초기 단계에 user interface와 같은 것을 할 때는 시장에 빠르게 보여주고 user들이 작동법을 보고 feedback을 얻기위해 사용.

Maintainability 향상 이유 : uncertain한 부분을 줄이기 때문에 나중에 수정할 부분이 줄어들어 maintainability가 향상됨.

11. Compare throw-away prototyping with exploratory prototyping.

- Throw-away prototyping: 명확하지 않은 부분을 명확하게 하고 사용자의 interface를 통해 사용자의 preference를 얻는 목적을 이룬 프로토타입은 더 사용하지 않고 버리는 방식

- Exploratory prototyping: 프로토타입을 조금씩 보완하고 개발하면서 프로토타입으로 final product까지 완성해 나가는 방식

12. With incremental delivery, customers can feel more confident about the system. Guess why.

User의 우선사항이 가장 높은 component 부터 개발하고 다른 component를 개발할때마다 같이 test하게 되어 User가 중요하다고 생각함에도 test가 충분히 이루어지지 않는 상황을 예방할 수 있음. 따라서, 사용자의 만족감도 높일 수 있으며, 개발 도중 바뀌는 요구사항을 반영하기 쉽기 때문에 프로젝트 실패 확률이 낮아짐

- single delivery : waterfall model에서 쓰이는 incremental하게 development한 것을 다 모아서 final system을 시장에 내는 것

13. Complete testing is virtually impossible. Discuss cost-effective testing strategies in development environments where time and cost are inadequate. Hint. Take advantage of available statistics.

Cleanroom SW development model의 operational profile : testing이 완벽할 수 없으므로 어딘가에 선택과 집중을 해야함. 따라서 operational profile (input class, frequent) 형식의 데이터를 사용하여 user의 사용빈도가 높은 task를 test하는 것이 중요. 이를 통해 user가 한번도 사용하지 않는 코드 라인을 수정하는 것을 방지하여 testing 과정이 cost effective해짐

MY QUSETION

1. effort = man month인 이유

   : 맨먼쓰(M/M, Man Month)는 엄밀히 men per month의 뜻으로 프로젝트에 투입되는 월 인원을 나타내는 수로 프로젝트의 크기를 표현할 때 쓰임

   1 M/M 이면, 1명이 한달간 하면 끝낼 수 있다, 2명이면 보름이면 끝낸다는 뜻

2. Prototyping에서는 왜 error checking과 recovery가 안되는가?

   : Prototyping 은 error checking과 recovery가 안되는데 그 이유는 code가 존재해야 에러가 존재하고 찾아야하는데 prototype은 입출력만 있고 그 안에 코드가 없음. 따라서 error checking할 방법도 없고 대상이 아님.