QA sheet 1

2017310695 이주민

1. Why is the cost distribution pattern of the iterative model and component based model different from the waterfall?

- waterfall model(specification : 15%, design : 25%, development : 20%, intergration&testing : 40%)
- component-based model(specification : 20%, development : 30%, integration&testing : 50%)
  : 초기에 어떤 component가 필요한지 명확해야 하므로 specification이 늘었고, itegration and testing이 component가 명세서와 일치하는지, 다른 component와 함께 예상대로 잘 동작하는지 보장해야 하므로 늘었다. + reuse만 하는 것이 아니라 더 발전해야하므로 증가함
- iterative model(specification : 10%, iterative development : 60%, integration&testing : 30%)
  : 요구사항이 명확하지 않을 확률이 높으므로 waterfall model을 보안하여 만든 모델이므로 specification이 줄었고, user의 요구사항을 만족할 때까지 development를 반복하므로 development 비율이 늘었다.
  + 중간에 계속 testing을 하므로 마지막 testing의 비중은 낮아짐

2. Explain the trade-off between the essential attributes of good software(maintainability, dependability and security, efficiency, acceptability)

- efficieny는 software가 hardware를 효율적으로 쓰는가를 확인하는 요인이다. 이 안에는 responsiveness, processing time, memory utilization이 포함된다.

: Maintainability를 위해서는 초기에 component의 크기를 작게 하여야 한다. 작게 하면 수정 및 추가, 삭제가 용이하다. 하지만 responsiveness는 떨어진다. 왜냐하면 responsiveness는 사용자의 반응성도 있지만 시스템 상호간의 반응속도도 포함한다. 그런데 component가 작다면 그 안에서 서로 통신할 것이니 속도가 떨어질 것이다. 따라서 responsiveness를 위해서는 component의 크기를 크게 하여야 하는데 이러면 maintainability가 떨어진다.

3. Compare the validation and verification activities

- validation : 결과물이 customer가 원하는 것을 충족했는가를 testing을 통해서 확인
- verification : 결과물이 만들어지는 과정에서 작성된 문서들과 일치하는가를 review, code inspection을 통해 검증

4. Explain four fundamental software engineering activites.

- software specification : customers and engineers define the software that is to be produced and the constraints on its operation
- software development : the software is designed and programmed
- software validation : the software is checked to ensure that it is what the customer requires
- software evolution : the software is modified to reflect changing customer and market requriement

5. Discuss the types of applications on slide 18-20 and find more than two examples for each type.

① Stand-alone applications

: 개인용 컴퓨터 또는 모바일 기기에서 실행되는 애플리케이션 시스템으로 필요한 모든 기능이 장치 내에 포함되여 네트워크에 연결할 필요가 없음

ex) office applications on a PC, CAD programs, photo manipulation software, travel apps, productivity apps

② Interactive transaction-based applications

: 원격 컴퓨터에서 실행되며 사용자가 자신의 컴퓨터, 전화 또는 태블릿에서 액세스할 수 있는 애플리케이션

ex) web application : ecommerce application(상품/서비스를 사기 위해 원격 시스템과 상호작용), business application : web browser나 special-purpose client program과 메일/사진 공유와 같은 cloud-based services을 통해 시스템에 대한 access 제공

③ Embedded control systems

: 하드웨어 장치를 제어하고 관리하는 소프트웨어 제어 시스템으로 다른 시스템에 비해 많이 사용됨

ex) 휴대 전화 software, 자동차 잠김 방지 브레이크 제어 software, 전자레인지 조리 과정 제어 software

④ Batch processing systems (일괄 처리 방법)

: 데이터를 대규모로 처리하도록 설계된 비즈니스 시스템으로 많은 개별 입력을 처리하여 출력을 생성함

ex) 주기적인 청구 시스템 : 전화 요금 청구, 급여 지급 시스템

⑤ Entertainment systems

: 사용자를 즐겁게 하기 위한 개인용 시스템으로 special-purpose console hardware에서 실행되는 게임이 대부분임. 제공되는 사용자 상호작용의 품질이 중요 요소임.

⑥ Systems for modeling and simulation

: 물리적인 프로세스나 상황을 모델링하기 위해 엔지니어가 개발한 시스템으로 종종 계산 집약적이고, 실행을 위해 고성능 병렬 시스템을 요구함

ex) Ansis, Nastern

⑦ Data collection and analysis systems

: 데이터를 수집하여 다른 시스템으로 전송하여 분석하는 시스템으로 software는 센서와 상호 작용해야 할 수 있으며 위험한 환경(엔진 내부, 원격 위치)에 설치되기도 함

ex) 빅데이터 분석 : 통계 분석을 수행하고 수집된 데이터에서 관계를 찾는 cloud-based system이 포함됨

⑧ Systems of systems

: 기업과 같은 대규모 조직에서 사용되는 시스템으로 여러 다른 software system으로 구성되어 있음.

6. Explain why you should choose different software engineering techniques, methods, and tools depending on the context of the software project.

: project 상황

-〉 유사한 project를 진행해본 경험이 있는지, requirement가 잘 요구되는가, 기술적 변화가 있는가 등에 따라 바뀔 수 있음

① Heterogeneity : system network에서 컴퓨터와 모바일 장치의 다른 형태를 포함하는 distributed system으로 운영되어야 하는 일이 많아졌음. general-purpose computer뿐만 아니라 휴대전화, 태블릿에서도 실행되는 software 필요함. 따라서 다른 programming language로 작성된 legacy system들을 통합하여 새로운 software를 만들 때가 있다. 이때 이 heterogeneity를 대처할 수 있을 만큼 유연하고 신뢰할 수 있는 software를 구축하는 기술을 개발해야함

② Business and social change : 기업과 사회는 새로운 기술이 보급됨에 따라 빠르게 변화하고 있음. 이때

전통적인 software engineering 기술은 시간이 많이 소요되며, 새로운 system을 제공하는데 시간이 오래 걸림. 따라서 user에게 제공하는데 필요한 시간을 줄일 수 있는 software 개발 필요

③ Security and trust : web service interface를 통해 access되는 원격 software system과 같이 software는 우리 삶에 얽혀 있으므로 신뢰할만한 해야함.

④ Scale : 웨어러블 기기에 들어가는 작은 embedded system부터 글로벌 커뮤니티에 서비스를 제공하는 Internet-scale의 cloud-based system까지 광범위한 범위의 software가 개발되어야 함.

7. Think about some fundamental principles that can apply to all types of software systems.

- **managed and understood development process**을 사용하여 개발해야함. 개발 과정을 계획하고, 무엇을 생산하고, 언제 완성할 것인지와 같은 것들을 명확히 해야함. 이때, 사용해야 하는 process는 개발 중인 software 유형에 따라 다름. (가시화)
- **Dependability and performance** : software가 외부의 공격에도 안전해야하며, 효율적으로 수행되어 자원을 낭비해서는 안됨.
- **software specification and requirements**(what the software should do)을 managed하고 understood하는 것이 중요함. 다양한 user가 기대하는 바를 알아야하고 user의 기대치를 관리하여 예산 범위 내에서, 일정에 따라 유용한 시스템을 제공해야함.
- you should **reuse software** that has already been developed rather than write new software (opensource)

8. Describe the distinctive changes in the software development process when using web or web services as a technical platform.

- web-based system 구축 시에 software reuse는 중요함. 이때 기존의 software 구성 요소 및 system을 어떻게 조립해야 하는지 알아야함.
- specification 단계에서 user의 requirement를 모두 아는 것은 비현실적인데 web-based system은 항상 점진적으로 개발 및 제공된다.

9. Investigate the fatal consequences caused by unethical behavior or decisions by software engineers.

: The Volkswagen scandal, which erupted in September 2015, made a lot of noise: Authorities found out that the firm's engineers had been falsely programming their cars to meet US environmental standards in laboratory testing, when they were actually emitting up to 40 times more CO2 in real life. The illegal software had been implemented in about 11 million cars worldwide.

10. Discuss with your colleagues what knowledge and efforts you need to be a competitive software engineer.

: 여러 opensource를 이해할 수 있는 능력 & 사용자 니즈에 맞게 변형하기 위한 코딩 실력

MY QUESTION

1. 고객의 니즈가 명확하지 않은 상태에서도 waterfall model을 사용하는 이유

- 프로젝트의 규모가 작고 난이도가 낮다면, 각단계에서 실수의 확률이 낮음.
- 프로젝트 진행자의 경험이 많다면 각 단계에서 나올 오류를 경험에 의해 예방 가능 함
- 각 단계 종료 후 산출물과 결과가 명확하다면 단계 종료에 대해 확신 하고 다음 단계 진행이 가능해짐

2. legacy system이란

: 조직에서 새로운 것으로 대체하고 싶지만 새로운 시스템 도입시 위험성이 매우 높은 시스템