**Software Engineering Q/A Sheet (#8)**

date:2020/11/07 number:2016310936 name: 우승민

**Questions from Prof.**

1. Define the architectural design process and explain the link between requirement engineering and this process.

System을 구성하는sub-systems과 sub-system의 제어와 소통을 위한 framework를 식별하기 위한 설계 process이다. System을 구성하는 주요 components와 이들 사이의 관계를 식별하기 때문에 requirement engineering과의 중요한 link이다.

1. Explain the advantages of explicit architecture in slide 7, and add an another advantage.

Stakeholder communication : stakeholder들이 architecture를 discussion의 focus에 사용할 수 있다.

System analysis : System이 non-functional requirement을 충족할 수 있는지 분석할 수 있다.

Large-scale reuse : 비슷한 requirements를 가진 여러 systems에서 재사용 가능하다..

1. System architecture affects system characteristics. Find a new example of architectural conflict.

Performance를 주요 목표로 하면 system 전체를 빠르게 하기 위해 가볍게 만들기 때문에 security와 availability가 약해질 수 있다.

1. Find a generic application architecture related with your team project.

제 project의 대상인 ‘songsterr’ application은 시스템에서 유저가 요청한 악보와 수정이 필요한 부분을 바꾸는 과정을 처리하고 database에서 계속 정보를 업데이트 하기 때문에 Transaction processing applications이라고 생각합니다.

1. Assume a project situation that is appropriate for each of the repository, client-server, and layered model.

Repository : system을 개발하는 개발자들이 code를 공유하는 상황에 사용할 수 있다. 예를 들어 github 사이트를 통해 팀원들이 code를 저장하고 서로 보면서 수정할 수 있다.

Cliend-server : 이 또한 개발자들이 멀리서 함께 개발할 때 사용할 수 있다.

Layered model : 개발을 여러 팀에서 나눠서 할 경우에 사용할 수 있다. 예를 들어 가장 기본적인 것을 한 팀에서 개발하고 나면 다른 팀이 받아서 이어서 개발할 수 있다.

1. Summarize the strengths and weaknesses of C2 and Weave respectively.

C2 : 공통된 특성을 가고 있는 components들의 상호 관계를 알기 쉽게 표현하는데 좋다. 하지만 같은 layer에 존재하는 components간의 관계는 표현하기 어렵다.

Weave : 각 Components끼리의 관계를 보여주는 것이 좋다. 하지만 C2에서 표현하기 쉬웠던 공통된 특성을 가진 components 사이의 관계를 보여주는 것은 어렵다.

1. Explain the architectural design decisions on Slide 14.

1. 설계되고 있는 system의 template에 맞는 generic architectural가 존재하는가?

2. System이 hardware cores나 processors 사이에서 어떻게 분산될 것인가?

3. 어떠한 architectural patterns, style을 적용할 것인가?

4. 어떠한 접근방식으로 system의 structure을 구성할 것인가?

5. System components의 operation의 통제는 어떠한 방식으로 이루어 질 것인가?

6. System을 sub-components로 어떻게 나눌 것인가?

7. System의 non-functional requirements에 가장 적합한 architectural organization은 무엇인가?

8. 문서화 작업은 어떻게 진행될 것인가?

1. Explain the 4+1 view model on Slide 18.

Logical view : System의 주요 abstractions을 objects나 objects class로 보여준다.

Process view : System이 작동하는 runtime에 process간 interaction 구성 방식을 보여준다.

Development view : software가 development때 어떻게 decomposed되는지 보여준다.

Physical view : 하드웨어, 소프트웨어 components가 system processors에 어떻게 분산되는지 보여준다.

Use cases와 scenarios들이 +1로 사용된다.

1. Explain the architectural patterns such as MVC, repository, client/server, and pipe/filter about time points of use, advantages, and disadvantages.

MVC : data를 보고 interact하는 많은 방법이 있을 때, 또는 data의 interaction이나 표현에 대한 향후 requirements를 알 수 없을 때 사용한다.

Data가 각각의 표현에 독립적으로 변경될 수 있고 반대도 가능하다. 같은 data를 여러 방법으로 표현하는데 도움이 된다.

Data model과 interactions이 simple할 때 추가적인 code나 code 복잡성을 포함해야한다.

Repository : 장기간 저장해야 하는 많은 양의 정보가 생성되는 system일 때 사용한다. Repository에 data를 포함하면 action이나 tool을 triggers하는 data-driven system에서도 사용 가능하다.

Components가 독립적이다. 하나의 component가 변하면 모든 components에 전파된다. 모든 data가 하나에 장소에 있는 것처럼 일관되게 관리가 가능하다.

Repository에서 발생한 문제가 전체 system에 영향을 끼친다. Repository를 통한 모든 communication을 구성하는데 비효율적일 수 있다. Repository를 몇 개의 computers로 나누는게 어렵다.

Clien/server : 공유 database의 data가 다양한 위치에서 접근될 때 사용한다.

Servers가 network를 통해 나눠질 수 있다. 일반 function을 모든 clients에서 구현할 필요 없이 이용 가능하다.

각각의 service가 허점이기 때문에 service 공격이나 실패 거부에 취약할 수 있다. Performance가 network와 system 둘 다에 달려있기 때문에 예측이 어렵다.

다른 조직에서 servers를 소유하면 관리가 어렵다.

Pipe/filter : inputs이 별도의 stages에서 처리되어 관련 outputs를 만드는 data processing applications에서 사용된다.

이해하기 쉽고 transformation reuse를 지원한다. Workflow style이 많은 business processes 구조에 어울린다. Transformation을 추가하여 evolution하는 것이 쉽다. Sequential, concurrent system 두 가지 방식 모두로 구현 가능하다.

Data 전송 format은 communicating transformations간에 협의되어야 한다. 각 transformation은 협의된 form으로 input을 parse하고 output을 unparse해야한다. System overhead를 증가시키고 적합하지 않은 data 구조를 사용하는 functional transformations 재사용이 불가능하다.

Questions from your ownself

1. Sub-system과 module의 차이점

Sub-system은 하나의 독립된 system으로 동작할 수 있지만, module은 독립적으로 동작할 수 없어서 다른 module이나 sub-system의 도움을 받아야 한다.