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

date: 11/7 number: 2018312280 name: 이상수

**Questions from Prof.**

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

시스템의 전체적인 구조를 설계하고 디자인하는 과정입니다.

Requirement engineering은 구조에 있어서 주요한 components와 그 관계를 정하는 단계이기에 이 architectural design과 중요한 관련이 있습니다.

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

Stakeholder communication

명확한 구조가 만들어지면, stake holder에게 설명할 때 주요한 자료로서 사용됩니다.

System analysis

Non-functional requirements 인 Performance, reliability 등이 시스템에 잘 적용 되는지 판단하기 좋습니다.

Large-scale reuse

다른 시스템을 만들 때도 비슷한 구조를 쓰기 좋을 경우 재사용할 수 있습니다.

추가 : 시스템 제작할 때 구성원들에게 업무 분배. 시스템 개발에 어느 부분이 우선적으로 개발되어야 하는지도 판단하는데 도움이 되고, 이후에 시스템을 유지보수할 때도 사용할 수 있습니다.

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

보안을 위해 어떠한 에셋에 접근할 때 layer를 여러 개 둘수록 보안은 좋아지지만 performance는 낮아질 수 있습니다.

결함을 대비해서availability를 높이려고 주요 파트를 2배로 만들면, 결함으로부터 안전하지만, safety나 security를 두개 다 관리해줘야합니다.

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

코딩테스트 학습 사이트. 코드를 입력하면 그에 대한 결과와 분석이 나와서 점수를 알려줍니다.

저희의 프로젝트도 코드를 입력하면 그에 대한 결과와 분석 결과가 나와 점수 대신 탄소 발생량을 알려 줍니다.

1. Explain the architectural design decisions on Slide 14.
2. 이미 이 시스템과 비슷한 구조가 존재하는가?
3. 시스템이 어떻게 분산 되어야 하는가? 물리적으로 분산 해야하나, 아니면 시스템적으로 분산되면 되나?
4. 구조적 패턴, 스타일을 무엇을 써야하나?
5. 시스템 설계에 있어 기본적인 내용은 무엇인가?
6. 시스템의 동작과 조작에 어떤 전략을 쓸것인가?
7. 시스템의 구성요소들을 어떻게 서브 시스템으로 나눌 것인가?
8. 어떠한 구조적 구성이 non-functional requirements를 전달하기 가장 좋은가?
9. 어떻게 시스템 구조를 문서화 할것인가?
10. Explain the 4+1 view model on Slide 18.

Logical view :

최종 사용자의 관점. 클래스 다이어그램, state diagram. 등을 사용하고 시스템 기능과 관련이 있어 functional view라고도 불립니다.

Physical view :

시스템 엔지니어 관점에서 컴포넌트 다이어그램, deployment diagram 등을 쓰고, 시스템의 하드웨어와 소프트웨어 컴포넌트가 프로세서에 어떻게 분배되어 있는지 나타냅니다. Deployment view라고도 불립니다.

Development view :

프로그래머의 관점. 소프트웨어가 개발을 위해서 어떻게 구성되어있는지를 신경씁니다. Implementation view라고도 합니다.

Process view :

시스템의 런타임 동작에 관점. 시스템의 동적인 동작, 통신 방법, 동시성, 성능, 확장성이 관련되었습니다. Runtime view라고도 불립니다.

System architecture (Use case) : use case에 있어서 위에 관점들을 검토합니다.

1. Explain the architectural patterns such as MVC, repository, client/server, layered and pipe/filter about when to use, advantages, and disadvantages.

MVC :

3가지 컴포넌트로 나누어 시스템을 나타내는 방법으로

Model component는 시스템 데이터를 관리하는 컴포넌트,

View component는 데이터가 사용자에게 표시되는 데이터를 관리하고,

Controller component는 사용자의 상호작용을 관리합니다.

사용처 : 데이터에 view와 interact할 방법이 여러가지 거나, 데이터에 대한 상호작용과 표시에 대한 미래의 요구사항을 알 수 없을 때 씁니다.

장점: 데이터를 표시하는 방식이 독립적으로 변하거나 그 반대의 경우가 가능하게 해주고, 같은 데이터에 대해 다른 표시방법을 지원하여 한가지의 변화가 모든 경우에 보일 수 있게 해줍니다.

단점: 데이터 모델이 단순하여도 추가적인 코드를 요구합니다.

Layered :

연관성이 있는 기능끼리 레이어로 묶어서 표현합니다. 각 레이어는 상위 레이어에게 서비스를 제공하기에 최하단의 레이어는 코어 시스템입니다.

사용처 : 이미 존재하는 시스템 위에 새로운 기능을 추가할 때, 여러 팀이 각각 레이어를 맡아서 개발할 때, 여러 단계의 보안을 요구할 때 사용합니다.

장점 : 인터페이스가 유지되는 한 레이어 전체의 교체도 가능합니다. 불필요한 기능이 각 레이어에서 추가되어 dependability를 증가시켜 줍니다.

단점 : 현실적으로 깔끔하게 분류된 레이어를 만든다는 것은 어렵고, 상위 레이어가 중간 레이어 없이 바로 하위 레이어와 상호작용 해야할 수 있습니다. 다수의 레이어는 성능을 저하시킬 수 있습니다.

Repository:

시스템의 모든 데이터가 Central repository에서 관리되고, 이 중심 repository를 통해서만 component들이 interact하는 구조.

사용처 : 오래 유지 되어야 하는 방대한 양의 데이터가 만들어질 때, 또는 data-driven시스템 같이 데이터의 추가가 이벤트를 유도하는 시스템을 디자인할 때 쓰입니다.

장점 : component들이 독립적이게 됩니다 (서로 존재를 알 필요가 없다). 한 component의 변화가 다른 component에게도 전달됩니다. 모든 데이터가 한 장소에서 일관성 있게 유지됩니다.

단점 : repository에 문제가 생기면 시스템 전체에 문제가 생깁니다. 하나의 repository로 모든 통신을 관리하면 비효율적입니다. Repository를 여러 컴퓨터로 분배하는 것이 어려울 수 있습니다.

Client-server :

각 서비스는 서로 다른 서버에서 제공되고, 클라이언트는 이 서버에 접속해서 서비스를 제공 받습니다.

사용처 : 공유 데이터베이스에 데이터를 먼 장소들에서 접속해야할 때, 시스템에 부하가 커질 수 있을 때, 서버를 여러 개로 복제해서 사용합니다.

장점 : 서버는 네트워크를 통해 전파 될 수 있고, 일반적 기능들은 모든 클라이언트가 사용 가능하며 각 서비스가 이를 가지고 있지 않아도 됩니다.

단점 : 각 서비스가 서버 장애나 공격에 취약합니다. 성능이 네트워크 상황에 따라 바뀝니다. 서버가 다른 조직에게 운영될 경우 운영상 문제가 발생할 수 있습니다.

Pipe-filter :

각 필터가 데이터를 처리하고, 이 데이터들은 파이프를 통해 각 필터로 이동 되는 패턴입니다.

사용처 : 입력이 여러 단계로 처리되어 출력되는 데이터 처리 어플에서 사용됩니다.

장점 : 이해하기 쉽고, 데이터 변화에 재사용을 지원해줍니다.많은 비즈니스 프로세스의 작업 스타일과 비슷합니다. 변화방식의 추가에 따른 발전이 직관적입니다. 절차적, 또는 동시다발적 시스템에 둘다 적용 될 수 있습니다.

단점 : 데이터 전송의 방식이 통신하는 필터간에 합의가 있어야합니다. 각 필터는 input을 자신이 이해하게, output은 합의된 형태로 변환해야합니다. 이러한 조건들이 시스템의 부하를 증가시키고, 서로 공존하기 어려운 데이터 구조들의 기능의 재사용이 어려워질 수 있습니다.

1. Explain the difference between the centralized control and event-based control.

Centralized control은 하나의 서브시스템이 나머지 서브시스템들의 시작과 끝을 조정합니다.

Event-based control은 내/외부의 이벤트에 따라 서브시스템들이 동작하는 방식입니다.

1. Explain the difference between the broadcast model and selective BC model.  
   broadcast model은 이벤트가 모든 서브시스템에 broadcast되고, 각 서브시스템은 자신이 처리해야 할 이벤트일 때 그 이벤트를 처리합니다.  
   Interrupt-driven model은 사전에 interrupt처리 할 수 있는 목록을 만들어 두고, interrupt가 발생하면, 해당 interrupt handler를 통해 필요한 component로 전송해서 이를 처리합니다.

Questions from your ownself

1. interrupt handler의 사용 예시는 무엇이 있을 까?

시스템의 exception handler. OS의 interrupt handler