QA Sheet 8

2017310695 이주민

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

- Architectural design : 소프트웨어가 어떤 sub-system들로 구성되어 있는지 이해하고, sub-system들끼리 control, communication하는 전체적인 구조를 디자인하는 과정으로 결과물은 sub-system들이 상호작용을 하면서 어떠한 형태로 조직되어 있는지를 나타내는 software architectural임
- Requirement engineering과 architectural design process에는 중복되는 부분이 존재하는데 그 이유는, architectural design이 requirement engineering 단계에서 진행이 되어야만 하는 부분이 존재(링크)하기 때문임.
- Requirement engineering 과정 : requirement gathering → requirement partitioning, requirement grouping → requirement clustering
- Architectural design 하는 과정 : subsystem을 식별 → 식별된 subsystem에 requirement를 배정 → subsystem을 정의 → subsystem간의 인터페이스를 정의

구매 조달할 때도 러프한 architecture가 필요함 (초반에는 참고할만한 architecture을 가져옴)

2. Explain the advantages of explicit architecture in slide 6, and add an another advantage.

- Stakeholder communication : 이해당사자들이 architecture를 기반으로 하여 시스템에 대해 소통함
- System analysis : Non-functional requirement에 대해 평가할 수 있는 베이스라인을 제공함
  A > B> C처럼 A architecture가 B, C보다 Non-functional requirement를 잘 충족한는지 비교할 수 있음
- Large-scale reuse : Design level(=Abstract level)에서 재사용이 가능하면 design에서 code까지(중간의 interaction까지) 재사용 가능성이 높아짐.
- 점진적인 prototype 개발을 가능하게 한다 : Architecture가 결정되면, 분석 가능한 prototype 형태의 골격 시스템으로 개발이 가능해짐.

architecture 정보가 잘 유지되어 있으면 유지/보수할 때 편리함. 유지/보수의 파급 효과의 근거가 architecture 정보이기 때문임.

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

- Performance를 높이기 위해서는 components간의 통신을 줄이기 위해 그 크기를 크게 해야 함.
- Security를 높이기 위해서는 layered architecture를 사용하여 중요한 것을 내부 layer에 보관하고 보안 인증을 높은 level에서 처리함
- Safety를 높이기 위해서는 안전에 관련된 operations를 하나의 서브시스템 내부로 국한시킴
- Availability를 높이기 위해서는 여분의 components를 두고 fault tolerance mechanism을 가짐
- Maintainability를 높이기 위해서는 작고 독립적인 components로 설계하여 변경하기 쉽게 함

Trade-off : performance-maintainability, availability-security, safety-performance

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

: layered Architecture이다. 전체적인 구조를 계층적으로 나눌 수 있기 때문이다.

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

- Repository model : 큰 데이터 이동 및 큰 데이터 공유가 필요한 프로젝트
  단점 : 데이터의 중복, communication의 overhead
- Client – server model : 공유된 데이터를 멀리 있는 여러 사람이 공유하기 위한 프로젝트
- Layered model : 존재하는 기능에 시스템을 올려놓는 시스템 개발 프로젝트 logical architecture
  top layer일수록 사람/다른 시스템과 인접한 layer임

6. Summarize the strengths and weaknesses of object-oriented and function-oriented models for modular decomposition.

① object–oriented modular decomposition

- 장점 : strong cohesion & weak coupling적인 특징때문에 수정변경으로 인한 다른 object에 대한 영향을 별로 주지 않음. object를 통해서 real world entities를 반영할 수 있음(시스템이 복잡해지면, class를 도입해야함. 하지만 거의 불가능) object-oriented implementation languages는 이미 많이 쓰여지고 있음. 유지보수 시에 비용이 적게 듦
- 단점 : object interface가 변경되었을 때 문제가 발생함. 복잡한 entities는 object로 표현되기 어려움
  분석/설계에 비용이 많이 듦.

② Function-oriented pipelining

- 장점 : function들이 reuse 될 수 있음. 직관적인 구성을 가짐. 새로운 function을 추가하기 쉬움. concurrent/sequential한 flow 둘 다 표현할 수 있음
- 단점 : data transfer를 위한 common format이 필요하므로 이에 대한 사전 협의가 필요함. data를 중심으로 표현하기 때문에 control system에서 등장하는 event를 기반으로 하는 interaction을 표현하기 어려움

7. Compare the call-return model with the manager model.

- call return : 어느 한 시점에서 하나의 component가 제어권을 가짐. 이때 제어권은 계속 이동함
- manager model : controller가 고정되어 있고, system controller가 subsystem/module을 제어함
- manager model은 call하고 결과를 기다리고 다른 call을 내리는 process이므로 hardware real time system보다 느림

8. Compare the broadcasting model with the selective broadcasting model.

- broadcasting model : event가 발생하면 모든 subsystem에 보내서 해당 event에 interest가 있거나 반응할 수 있는 subsystem들이 반응하는 구조임. 따라서 시스템 부하가 큼
- selective broadcasting model : 사전에 각 subsystem들이 어떤 event에 관심이 있는지 event handler에 등록을 해놓기 때문에 sub-system들이 계속 event를 감시할 필요가 없으므로 부하가 작아짐.
- 하지만, 둘 다 real-time system은 어려움. 왜냐하면 각각의 sub-system이 event에 응답을 할지/말지, 언제 할지도 알 수 없음. 또한, 각 sub-system들이 event에 대한 응답이 다를 때, 어떻게 처리할지도 어려움

9. Compare the reference model with the generic model.

- generic model : 해당 도메인에서 사용되는 system들을 분석하여 공통된 속성을 모아 architecture를 만들어가는 bottom up 방식 일종의 domain knowledge
- reference model : 특정 도메인에서 사용해야 하는 표준 모델로 top down 방식 반드시 따라야 함

MY Question

1. broadcasting과 selective broadcasting 둘 다 control policy가 있는 것인가?
- 있다.
- broadcasting은 전체에 다 event를 주고, 응답이 오는 순서대로 반응하는 것
- selective broadcasting은 handler에 사전에 등록하고, 응답이 오는 system들을 어떻게 반응할 것인지 미리 정해줌

2. top-down 방식과 Bottom-Up 방식의 차이점은?
- Top-Down : 시스템의 skeleton을 개발 후 컴포넌트를 붙여줌
- Bottom-Up : 구조 기반 컴포넌트를 통합 후 기능 컴포넌트를 붙여줌