# Q/A Sheet #8 – Architectural Design

Date: 4/29 number: 2016311821 name: 한승하

**Questions from Prof**

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

   A. Architecture design은 system의 sub-system ( major component )을 식별해 내는 작업, sub-system간의 communication, control의 framework을 만드는 작업을 말한다. Requirement engineering 과 architecture design은 서로 overlapping 되는 부분이 존재한다. Architecture design은 design process임에도 불구하고 requirement engineering과 더불어 진행되어야 한다. Requirement engineering 은 sub-system을 식별해내고, assignment, sub-system definition, interface definition등의 작업들이 이루어지고, 이는 architecture design에 해당이 된다.

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

   A. Stakeholder communication

      i. 수십, 수백개의 function에 대한 나열만이 존재했을 때 보다 explicit architecture가 존재하는 경우 stakeholder에게 전달하는 것이 훨씬 수월하다.

   B. System analysis

      i. Explicit architecture로 정의된 system은 평가할 수 있는 요소가 된다. Non-functional requirement은 system에 있어 굉장히 중요한 요구사항이기 때문에 이를 평가할 수 있다는 측면은 굉장한 이점이 된다.

   C. Large-scale reuse

      i. 재사용시 가장 많이 재사용되는 요소는 Code의 abstract한 요소들이다. 따라서 explicit architecture은 이에 대한 high level abstraction을 가지고 있고 이는 높은 재사용율을 가질 수 있다.

   D. Explicit architecture을 design함으로 대규모 system development의 parallel development가 더욱 용이할 것으로 추정된다. System을 sub-system으로 나누는 작업을 진행하므로 각 요소들에 대한 병렬적 개발, 또한 이를 합치는 과정에 대한 interface의 설계 개발이 더욱 용이할 것으로 추정된다.

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

   A. 높은 security는 system에 더 복잡하고, 추가적인 computational resource를 요구하기 때문에 performance와 maintainability에 영향을 줄 수 있다.

   B. 높은 Availability는 높은 독립성을 요구한다. 이를 위해 Sub-system들 간의 Data-transfer가 어려워지고, 더 많은 Hardware가 추가될 수 있으며 이는 Performance에 영향을 줄 수 있다.

   C. 높은 Performance를 타겟으로 한 Architecture는 System 전체의 경량화가 중요하다. 이 때문에 Security에 누수가 생길 수 있고, Availability도 약해질 수 있으며, 많은 HW측면의 Resource를 사용하도록 설계되었다면, HW lifetime이 줄어들어 System의 Maintainability가 어려워질 수 있다.

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

   A. 우리의 Team project는 Application User가 제품들을 추천, search, 혹은 즐겨찾기 로 설정한 제품군들을 categorize 하여 구매결정 혹은 원하는 제품에 대한 정보를 보다 더 효율적으로 이해하고 얻게 하기 위한 development이다. 이를 제공하기 위해서는 Sub-system에 사전에 만들어 진 Database에서 client가 열람하기 원하는 data들을 제공해 주어야 하며, Client에 의해 혹은 Client를 위한 Change가 전체 Database에 빠르게 적용되어야 하므로 하나의 중앙 Database에 Data를 저장하고, Sub-system들이 중앙 Database를 사용하는 Repository Model에 가장 적합한 것으로 보인다.

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

   A. Repository model

      i. 은행의 System이 가장 대표적인 repository model의 예시일 것 같다. 각각 client의 정보, 혹은 거래정보를 각 sub-system이 아닌 중앙 repository에 보관하는 방식 쓰는 것이 효율적으로 보인다.

   B. Client-server model

      i. 요즘 핫한 cloud gaming이 대표적인 예시가 될 수 있을 것 같다. High graphical game들을 다양한 platform에서 제공하기 위해 graphical computing을 server에

서 처리하는 방식이다.

C. Layered model

i. 컴퓨터의 OS system이 가장 대표적인 layered model을 사용해 개발해야 하는 system일 것 같다. OS system은 privileged level에 따른 service를 제공하는 system에 대한 modeling이 필요하므로 layered modeling이 적합할 것이다.

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

A. Object-oriented models

i. 객체지향은 타 Object와의 연관성이 적기 때문에 수정시에 타 object에 영향을 크게 주지 않는다.

ii. 객체지향은 real-world entities들을 반영하여 설계할 수 있으나, 복잡한 system이 되면 이는 어려워진다. 따라서 Logical class가 반드시 만들어져야 한다.

iii. 많은 객체지향 언어들이 많이 이용되고 있다.

iv. 객체와 객체사이의 message passing을 통해 service를 제공한다. 따라서 이러한 interface에 대한 정보가 없거나 모르는 사이에 수정이 생길 경우, 큰 문제로 연결될 수 있다.

v. User Interface 중심의 interactive system에 적합하다.

B. Function-oriented model

i. Business system에는 매우 적합하나, User interface중심의 interactive system들에는 적합하지 않다.

ii. 하나하나의 Function들이 Transformation unit으로서 재사용 될 수 있다.

iii. Stakeholder들이 이해하기에 더 좋아 stakeholder들 과의 communication에 유리하다.

iv. 새로운 Transformations들을 추가하기가 쉽다

v. Concurrent혹은 Sequential 양쪽 모두 쉽게 사용할 수 있다.

vi. Data transfer를 위한 common format에 대한 사전 약속이 반드시 필요하다.

vii. Event-based interaction을 표현하기가 어렵다.

7. Compare the call-return models for modular decomposition ( 문제에 오류가 있는 것 같아, Real-time system model과의 비교도 추가 하였습니다. )

   A. Call return model은 Centralized control의 일종으로서 제어권이 Call과 Return에 의해 계속 이동되는 형식을 말한다.

   B. Modular decomposition은 Sub-system을 module단위로 분해하는 것을 말한다. 이 modular decomposition의 대상이 되는 단위는 Object단위와 Function단위 두가지가 존재한다.

   C. Real-time system control은 manage model이라고도 하며 Centralize control을 이야기할 때 주로 말하는 model이다. 하나의 중앙 controller가 전체 module을 control하는 형태를 말한다. 이 경우에 실시간 system control이 가능하다고 이야기한다. Real-time은 Soft real-time system을 말한다.

8. Compare the broadcasting model with the selective broadcasting model

   A. Broadcasting model은 Event가 발생하면 그 Event를 연결되어진 다른 Sub-system들에게 보내어 Event에 interest가 있거나, 반응할 수 있는 sub-system이 반응하게 하는 model을 말한다. 이러한 방식은 지속적인 sub-system에 대한 관리와, 모든 event에 대한 전송 등 전체 system에 부하가 커지는 단점이 있다.

   B. Selective broadcasting model 은 각 sub-system이 자신이 관심있는 혹은 반응해야 하는 event를 event handler에 등록을 하여, 등록된 event에 대한 interrupt를 받는 방식이다. 선별적인 event 전송이 가능하며, 일반적인 Broadcasting에 비교하면 네트웍 부하도 줄이고, Computing resource도 줄이는 등 효율적인 방식이다.

   C. Interrupt driven control은 Interrupt이 발생하게 되면 하던 일을 중지하고, 미리 정의되어 있는 interrupt handler에 의해 해결하게 되는 구조를 말한다. 이런 구조가 가능하기 위해서는 사전에 interrupt type에 대한 정의와 각각의 type에 대한 handler가 정의되어 있어야 한다. Interrupt을 Interrupt handler로 연결시켜주는 Interrupt vector는 속도가 빠른 HW switch를 이용한다.

9. Compare the reference model with the generic model.

   A. Generic model은 Domain에서 Real world에 존재하는 공통적인 속성들을 모아 Bottom-up적으로 Generic한 system을 만들어가는 방식으로 개발되어지는 model을 말한다.

   B. Reference architecture는 OSI 7layer model과 같이 특정 Domain에 있어 표준처럼 사용되어지는 model을 말하며, 참조 모델이라고 한다. Domain의 전문가 혹은 표준기구 등에서 미리 표준을 정의해둔 Model이며, 반드시 따라야 한다. 또한 Top-down형식의 model이다.

**Questions of yourself**

1. 다양한 Architectural style에 대해 알고 있는 것의 이점은 무엇인가?

   A. Generic architectural model은 많은 development 상황에 쓰이는 기본 모델들이다. 이러한 모델들에 대한 지식이 많을수록, 다양한 상황에 대해 불필요한 개발 과정을 줄일 수 있으며, 이러한 모델들은 기존에 다양하게 사용되어 검증이 되어진 model들으로, 높은 안정성을 가질 수 있다. 또한 Large system development에선 필연적으로 다양한 model에 대한 이해가 필요할 수 있다.

2. Sub-System과 module의 차이가 무엇인가?

   A. Sub-system과 module의 가장 큰 차이는 독립성이다. Sub-system은 하나로 독립된 system으로서의 가치를 가질 수 있다. 다른 어떠한 system이나 module의 도움을 받지 않고도 활동할 수 있는 것을 sub-system이라 하며, module은 이에 비해 독립성이 떨어져서 다른 sub-system 혹은 module의 도움을 받아야 하는 것을 말한다.