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 C2 and Weave respectively

A. C2는 connector와 component를 의미한다. 단순히 Component군과 Component군의 Connector를 사용한 연결을 표현한다. C2는 공통된 특성을 가고 있는 Component군들의 상호 관계를 알기 쉽게 표현하는데 굉장히 유리하다. 하지만 같은 Layer에 존재하는 Component간의 관계는 표현하기 어렵다는 약점이 있다.

B. Weave는 각 Component가 다른 Component와 Port를 통해 연결된 모습을 보여준다. Weave는 각 Component끼리의 관계를 보여주는 것에 있어서는 큰 강점을 보이나, C2에서 표현하기 쉬웠던 Component군 사이의 관계를 보여주는 것은 어려울 수 있다.

7. Discuss with your colleagues about architectural design decision on Slide 14

Architectural design을 위해 고려해야 하는 대표적인 사항들이다.

A. Generic한 Application이 존재하는 경우에는 가능한 Generic한 Architectural을 사용하는 것이 더 효율적이고 안전하다.

B. System이 분산되어야 하는가, 분산된다면 Hardware core나 Processors 사이에서 어떻게 분산될 것인가?

C. 어떠한 Architectural Style을 적용할 것인가? 어떤 Style이 system에 적당한가?

i. Repository, Client-server, Pipe and Filter, 등등

D. System의 통제는 어떠한 방식으로 이루어 지게 할 것인가?

i. Call-return or Real-time system control, 중앙통제 혹은 Event-driven

E. 어떠한 접근방식으로 System의 Structure을 구성할 것인가?

i. Object 혹은 Functional oriented

F. System을 Sub-components들로 어떻게 쪼갤 것인가? 어떠한 단위로 modulization할 것인가?

   i. Object단위 혹은 Function 단위

G. System의 Non-functional Requirements에 가장 적합한 architectural organization은 무엇인가?

H. 문서화 작업은 어떻게 진행될 것인가?

8. Discuss with your colleagues about 4+1 view model on Slide 14

   System의 Architectural을 바라볼 수 있는 4+1가지의 서로 다른 관점이다

A. Logical view는 system의 object혹은 object classes단위의 핵심 요약본을 말한다

B. Process view는 system이 작동할 때, process간 interaction이 어떻게 이루어지는지를 말한다.

C. Development view는 개발 작업을 위해 Decomposed된 System을 말한다.

D. Physical view는 System의 hardware와 Software component들이 Processor간에 어떻게 분산되었는지를 말한다.

E. Use Case와 Scenarios들에 관련된 사항들이 +1로 적용된다.

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

A. MVC pattern은 Model-View-Controller의 약자로써 System을 3개의 logical components로 쪼개어 설명하는 방식이다. Model Component는 System의 Data, 그리고 Data와 관련된 Operations들의 관리를 담당하는 Component이고, View Component는 data가 사용자에게 어떻게 보여지는가를 관리하고 정의하는 Component이다. Controller component는 User의 Interaction을 관리하는 Component이다.

   Data를 보거나, Data와 interact할 수 있는 방법이 다수 존재할 때 사용한다. 혹은 Data를 보거나 상호작용할 수 잇는 방법이 명확하지 않을 때 사용된다.

   Data가 Representation과 무관하게 독자적으로 바뀔 수 있으며, 반대도 가능하다.

   Supports presentation of the same data in different ways with changes made in one

representation shown in all of them (해석 불가).

Data model과 interaction이 간단할지라도, 추가적인 Code와 Code의 복잡함이 요구될 수 있다.

B. Repository Model은 하나의 중앙 Repository에 모든 Data가 관리되는 형태의 Structure을 말한다.

System이 대량의 Data를 오랫동안 보관해야 하는 경우에 주로 사용된다.

Component들이 독립적이며, 서로 다른 Component들에게 주는 영향이 적다. 한 Component가 만든 변화가 다른 모든 Component들에게 적용되기 쉽다. 모든 Data가 한번에 관리되기 쉬우며, 최신상태를 유지하기 쉽다는 장점이 있다.

중앙 Repository의 허점이 전체 System에 적용될 수 있다는 단점이 있으며, Repository를 component들에게 분산 적용시켜야 되는 경우 굉장히 까다롭다.

C. Client-server Model은 다수의 Server에서 오는 Service로 이루어진 System을 표현한다. 이 Service들의 사용자는 Client이다.

공유되어지는 Database가 여러 곳에서의 접근이 필요할 때 주로 사용한다. 또한 system에 걸리는 부하가 꾀 클 때 사용된다.

Server들이 Network에 분산되어 있을 수 있다는 장점이 있으며, Service와 무관하게 모든 Client들에게 General Functionality를 제공할 수 있다.

Data Interchange가 원활하지 않을 수 있다. 따라서 정해진 표준이 필요하다. Service를 제공하는 서버에 대한 Register혹은 정해진 이름이 없다면 service를 찾기가 어렵다.

D. Pipe/filter은 각 filter가 한가지 타입의 data transformation을 가지는 유닛으로 system의 data processing을 나누는 model이다. Data는 한 component에서 다른 component로 움직이며 processing된다.

Input들이 여러 Stage를 거쳐 output을 도출해내는 형태의 Data processing application에 주로 사용된다.

이해가 쉽고 Transformation 재사용이 가능하다. 많은 Business processes와 workflow가 동일하다. Transformation을 추가하기가 용이하다. Sequential, 혹은 Concurrent system으로 적용될 수 있다.

Data transfer의 format이 사전 협의되어 있어야 한다. 각 Transformation은 format에 맞추어 input을 parsing하고, output을 unparsing하는 과정이 필요하다. System overhead가 증가하고, incompatible data structure를 가진 functional transformation

들은 재사용이 불가능 할 수 있다.

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.