Software engineering Analyzing architecture

2017030055 윤상현

1. Blackboard Architecture

1-1) What is Blackboard Architecture?

Blackboard Architecture는 해결해야할 문제에 대한 솔루션이 알려지지 않은 문제에서 유용하다. Blackboard Architecture는 문제를 찾고 초기 데이터를 Blackboard에 쓰는 것으로 시작한다. 전문가들은 솔루션을 발전시키기 위해 blackboard를 주시하고, 기회가 생기면 Blackboard에 최종적인 문제 해결을 위해 사용될 만한 내용을 기록하며 솔루션을 찾아 나간다.

모든 컴포넌트는 blackboard에 접근하고, 컴포넌트는 블랙보드에 추가되는 새로운 데이터 객체를 생성할 수 있다. 컴포넌트는 블랙보드에서 필요한 데이터를 찾으며, pattern matching과 기존의 resource를 활용하여 데이터를 찾는다.

1-2) Components & Connectors of Blackboard Architecture

- Components

1) Blackboard: Blackboard는 각 분야의 전문가들이 함께 일하는 곳이다.

Blackboard의 공식적인 정의는 "globally accessible database" 로 문제해결의 중간,

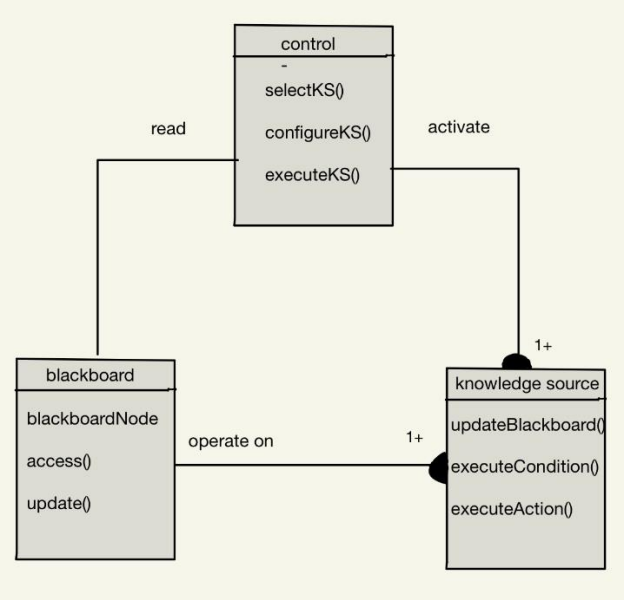
그리고 부분적인 결과에 사용된다.

2) Knowledge source: Knowledge source는 문제해결에 필요한 지식을 담은

독립적인 모듈이다. Knowledge source는 현재 문제에 대한 솔루션에 대한 진전에 중점을 둔다.

3) Control component: Controller는 최적의 문제해결을 위하여 다음으로 실행할 후보 Knowledge source를 결정한다. 한가지 영역에서 여러 expert가 있을 경우, Controller는 어떤 expert가 참여할지 결정합니다.

1-3) Diagram of Blackboard Architecture



1-4) Pros & Cons

-Pros

1) 문제에 대한 다양한 접근법을 찾을 수 있다.

2) 다양한 expert들의 참여로 유지보수성이 뛰어나다.

3) 재사용 가능한 지식자원의 창출이 가능하다.

-Cons

1) 테스트를 진행하기 어렵다.

2) 문제에 대한 완벽한 해결책이 보장되지 않는다.

1-5) Real life example

- 음성인식 기술
- 차량 식별 및 추적
- 음파 탐지기의 신호 해석

2. Client-server Architecture

2-1) What is Client-server Architecture?

Client-server Architecture는 하나의 컴퓨터, 혹은 호스트를 서버로 지정하고, 나머지 컴퓨터들을 클라이언트로 지정하여 하나의 서버와 다수의 클라이언트를 구성한다.

다수의 클라이언트는 하나의 서버에 필요한 서비스를 요청하고, 서버는 요청을 받기 전까지 대기하고 요청이 들어오면 해당하는 서비스를 제공해 준다.

2-2) Components & Connectors of Client-server Architecture

- Components

1) Server: 중심에서 클라이언트가 요청하는 서비스를 제공하는 컴퓨터이다.

다수의 클라이언트로부터 요청이 들어오면 처리해줘야 하기에 좋은 성능과 거대한 용량을 가지고 있다.

2) Client: 다양한 서비스를 사용하는 사용자들, 혹은 그들이 사용하는 기기.

요구할 서비스가 있다면 서버에 요청을 보내 필요한 서비스를 제공받는다.

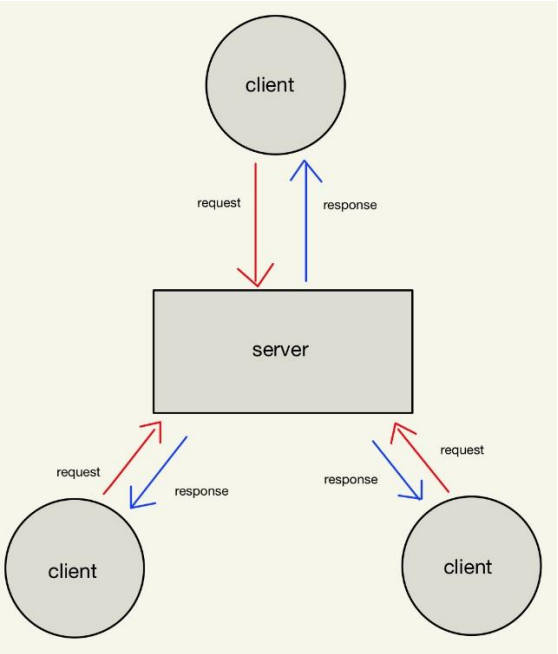
- Connector

1) Request: 클라이언트가 서버에게 필요한 서비스를 요청하는 것을 나타낸다.

2) Response: 서버가 클라이언트로부터 요청을 받은 뒤 해당하는 서비스를

클라이언트에게 제공해주는 것을 나타낸다.

2-3) Diagram of Client-server Architecture



2-4) Pros & Cons

- Pros

1) 프로그래머의 관점에서 유지보수가 용이하다.

2) 서버에서만 데이터를 다루기에 관리 포인트가 적어져 보안에 유리하다.

3) 서버나 클라이언트 컴포넌트를 추가할 때, 구조에 큰 영향을 미치지 않으며 쉽게 추가가 가능하다.

- Cons

1) 서버의 역할을 수행하는 컴퓨터의 성능이 매우 좋아야 하기에 지출이 크다

2) 서버에 문제가 생기면, 모든 클라이언트가 서비스를 제공받는 것에 제약이 생긴다.

3) 성능이 매우 좋더라도 클라이언트가 지나치게 많다면 서버에 부하가 올 수 있다.

2-5) Real life example

-이메일

-은행

-월드 와이드 웹

2. Peer-to-Peer Architecture

3-1) What is Peer-to-Peer Architecture?

Peer-to-Peer에서 Peer 란 동료 라는 뜻으로써, 이 Architecture에선 말그대로 대등한 관계에 놓인 동료의 입장으로 각 컴퓨터들이 연결 되어있다. 즉, 각 컴퓨터들은 서로 데이터를 주고받을 때 어떠한 중계기간을 거치지 않고 직접 주고 받을 수 있다. 그 구현 방식에는 기존 serv-client 모델을 벗어나지 않는 Hybrid Peer-to-Peer 방식, 중앙서버를 전혀 사용하지 않는 Pure Peer-to-Peer 방식, 그리고 앞선 두 방식의 장점을 결합해 놓은 Super-Peer 방식이 있다.

3-2) Components & Connectors of Peer-to-Peer Architecture

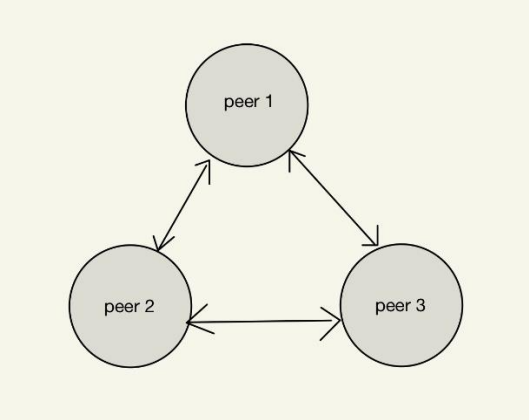
-Component

1) Peers: 각 Peers는 서비스를 제공하는 server의 역할을 함과 동시에 서비스를 제공 받는 client의 역할을 한다.

-Connector

1) invokes procedure: client-server Architecture와 다르게, 양방향 화살표를 통하여 양방향 관계를 표현합니다. 각 Peer가 서로 협력하여 데이터를 전달하고, 데이터를 제공 받는 관계를 표현한다.

3-3) Diagram of Peer-to-Peer Architecture



3-4) Pros & Cons

- Pros

    1) 분산 컴퓨팅 어플을 구축할 시 유연성을 제공한다.

    2) 중앙 서버를 사용하지 않기에 정보 유출의 위험이 적다

    3) 중앙 서버에 집중되는 부하를 분담하여 비용이 Client-server에 비해 상대적으로 적다.

- Cons

    1) 모든 클라이언트의 컴퓨터가 서버의 역할을 수행해야 하기에 다른 클라이언트에게 자신의 IP가 노출될 위험이 있다.

    2) Peer의 수가 적어지면 전송속도가 느려진다.

3-5) Real life example

- 파일 공유 네트워크
- 멀티미디어 프로토콜
- 독점 멀티미디어 애플리케이션

Reference

https://m.blog.naver.com/pcmola/222092887306

https://gelos.tistory.com/m/4

https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Blackboard.pdf