Software Architecture

Data-Centered (DC) Software Architectures

Blackboard Architecture Style

Eunmi Choi
Kookmin University

Types of Data-centered Architecture
Repository
Blackboard

데이터 센터
레파지토리와 비교가 됨.

Blackboard Architecture Style

Blackboard Architecture Style

결정론적인 결과가 나오지 않는 문제를 푸는데 좋다!
(결정적으로 output이 항상 동일하게 나오지 않는 문제)

결정적인 절차를 가지고 문제를 푸는 것이 아니라 절차가 정해지지 않은 문제를 적절한 솔루션으로 받아들인다.
결론 자체가 결정적으로 나오지 않음
어떤 상황에 누가 먼저 시작하느냐에 따라 솔루션이 달라지기도 함.
tm토리지인데 데이터가 변화되면서 원하는 해답으로 도출하는 문제

공유 데이터가 블랙보드에 써있고 independent한 exports들이 각자 문제를 풀어나감.
블랙보드 안에 raw data가 있음. 솔루션은 결정적이지 않은 형태로

- Synopsis
  - Similar to the classroom blackboard used in solving problems without deterministic outcome.
    - share data in solving classroom problems via a blackboard
    - play the role of agents to contribute to the problem solving
    - in parallel, and independently, trying to find the best solution

소

- The Blackboard pattern is useful for problems
  - **Transformation of raw data into high level data structures** (eg: diagrams, tables or English phrases) **that have no feasible deterministic solution**
  - **Collection of independent programs** that work cooperatively on a common data structure
    - Each program is **specialized for solving a particular part** of the overall task
    - **Specialized programs** are **independent** of each other
    - All programs **work together** on the solution

결정적으로 풀어나갈 수 없음(8~8:20)

Blackboard Architecture Style

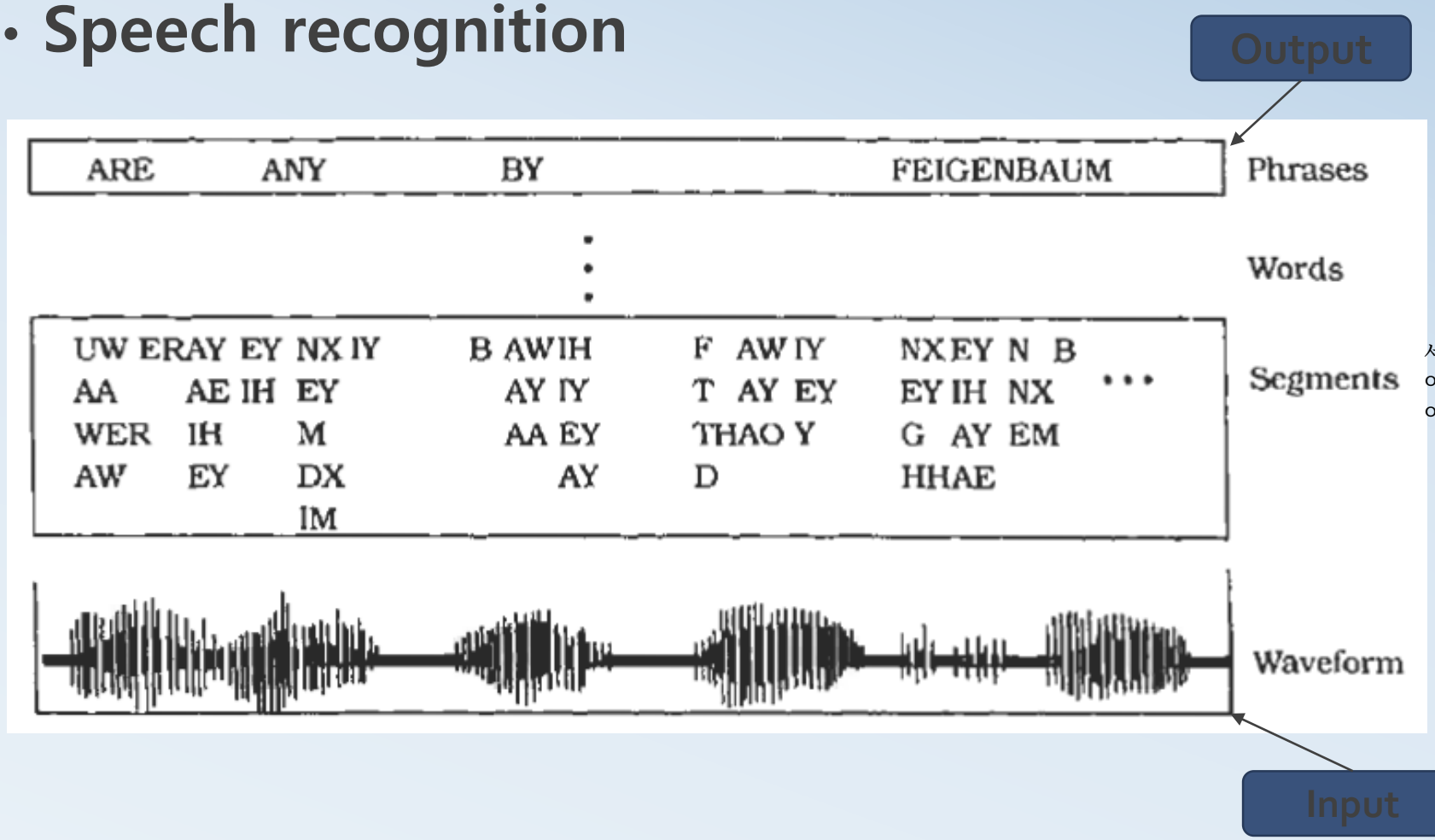
- Application Examples
  - AI community shows the idea to be seen in a wider context

    Ex) **Vision, Image recognition, Speech recognition**
  - speech recognition applications(created in 1970s)
  - image pattern recognition
  - weather broadcast systems
  - Hearsay-II speech recognition expert system
  - CRYSTALIS molecular structure analysis system

AI 문제 :
아날로그 음성 인식 문제
날씨 브로드캐스트
이런 결정될 수 없는 것들을 블랙보드로 품

Blackboard: Example

- **Speech recognition**



세그먼트를 계속 찾아가면서 어느 phrases인 것 같다는 어그리게이션에 쓰임

Blackboard Architecture Style

데이터가 저장된 구조를 가짐. 이 데이터는 pasial solution이 계속적으로 나오는, 변형되는 컨텐츠를 유지
전체 시스템 구조도는 블랙보드라는 데이터를 가진 구조가 있고, 놀리지 소스(전문가)를 가진 구조가 있고, 컨트롤러가 있음.

- Property
  - a data-directed and a partially data-driven architecture.
- Structure
  - The entire system is decomposed into two major partitions.
    - One partition, called the *blackboard*, is used to store data (hypotheses and facts),
    - The other partition, called *knowledge sources*, stores domain-specific knowledge.

이 도메인에 아주 전문적인 지식을 가진 에이전트

  - The third partition, called the *controller*

적합한 문제를 풀수 있도록 관리자 역할을 함.

    - initiate the blackboard and knowledge sources and that takes a bootstrap role and overall supervision control.

Blackboard Architecture Style

- Structure
  - The connections between the blackboard subsystem and knowledge sources are basically implicit invocations from the blackboard to specific knowledge sources, which are registered with the blackboard in advance.
  - Data changes in the blackboard trigger one or more matched knowledge source to continue processing.
  - Data changes may be caused by new deduced information or hypotheses results by some knowledge sources.
    - This connection can be implemented in *publish/subscribe* mode.

블랙보드에 있는 데이터가 바뀌어지면 어떤 export가 알 경우 접근을 함.
trigger를 해서 다른 놀리지 소스가 풀 수 있도록 함.

r ㅏ정 : 어떤 경우는이 답이다, 어떤 경우느 ㄴ 이 답이다 하면서 데이터를 변경해나감.

publish/subscribe mode : 서로 연결됨. 트리거는 항상 리슨하는 서브스크라이버를 놀리지소스가 갖고 있다가 .. 하면 트리거 되어 사용할 수 있게됨.

Blackboard: Forces

- **Domain is immature**
  결정적으로 ~ 하지 않은..
- Different partial problems need to have **different resolution algorithms** 엑스퍼트 아레아가 다 다름.
- **Input, intermediate and final results have different representations** account for the implementation algorithms are complex 로우 데이터는 아날로그인데 중간은 파셜 데이터가 텍스트로 나올 수 있고 어떤 경우는 다른 데이터 포맷으로 바꿀 수 있음.
- Algorithm usually **works on the results of other algorithms**
- **Uncertain data and approximate solutions**
  근사 솔루션. 정확한 답에 가까운 솔루션을 접근한다.
- Employing disjoint algorithms induces **potential parallelism**

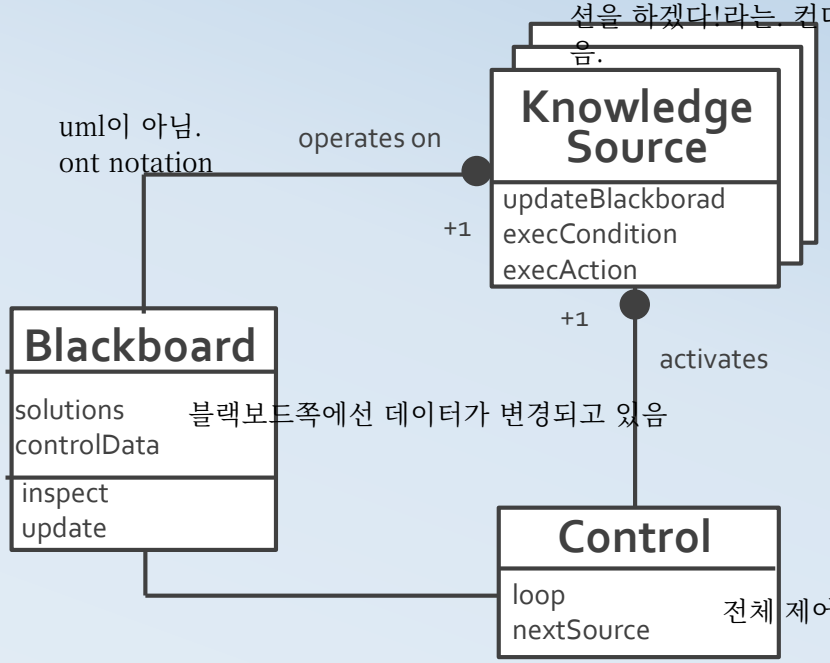
Blackboard: Solution - Structure

- Blackboard:
  - call inspect to check the current solutions on the blackboard.
  - update is used to make changes to the data on the blackboard.
- Knowledge Source:
  - determines when the system should halt, and what the final result is.
- Control:
  - runs a loop that monitors changes on the blackboard and decides what actions to take next.

어떤 상황에 문제를 풀 수 있는 컨디션과 액션이 있음.

하는 일 : 내가 풀 게 없다는 것을 알아야 함. 컨디션에 만족한다 안한다를 알 수 있어야 halt할 수 있음.

전체가 미팅이 잘 되도록 함.



전문가에 대한 결정을 해줌. 어떤 컨디션은 어떤 액션을 하겠다!라는. 컨디션과 액션을 오브젝트로 넣음.

전문가 : 여러개가 있을 수 있음
컨디션에 따라 액션
블랙보드를 바꿀 수 있음.

uml이 아님.
ont notation

블랙보드쪽에선 데이터가 변경되고 있음

1:다 노테이션

전체 제어

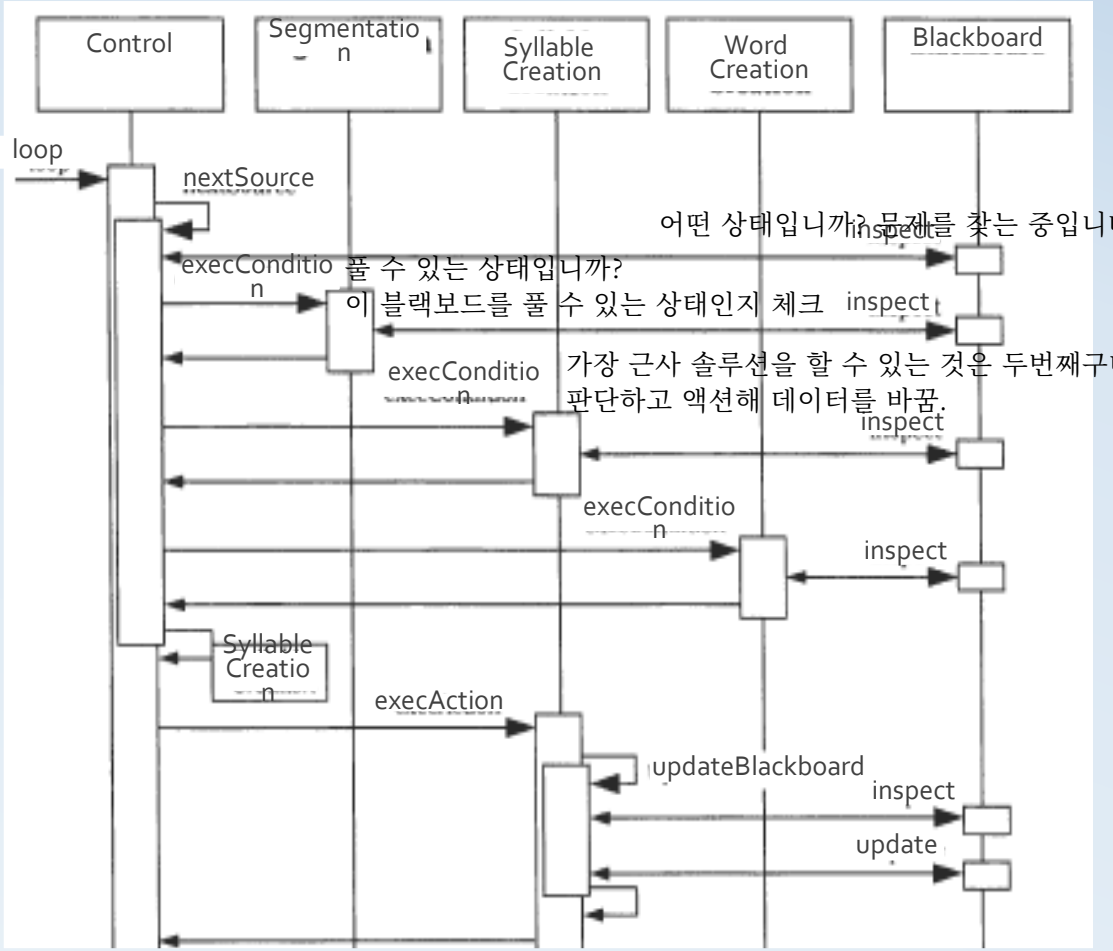
halt : knowledge source
terminate : control이 함

그다음 놀리지소스를 찾아가는것을 컨트롤이 함.

Blackboard: Solution - Dynamics

- **Main loop** of the Control component is **started**
- Control **calls the nextsource** ( ) procedure to **select the next knowledge source**
- nextsource() **invokes the condition-part** of each candidate knowledge source
- The Control component **chooses a knowledge source to invoke**, and **a hypothesis or a set of hypotheses to be worked on**

Blackboard: Dynamics Example



어떤 상태입니까? 문제를 찾는 중입니다.

풀 수 있는 상태입니까?

이 블랙보드를 풀 수 있는 상태인지 체크

가장 근사 솔루션을 할 수 있는 것은 두번째구나 판단하고 액션해 데이터를 바꿈.

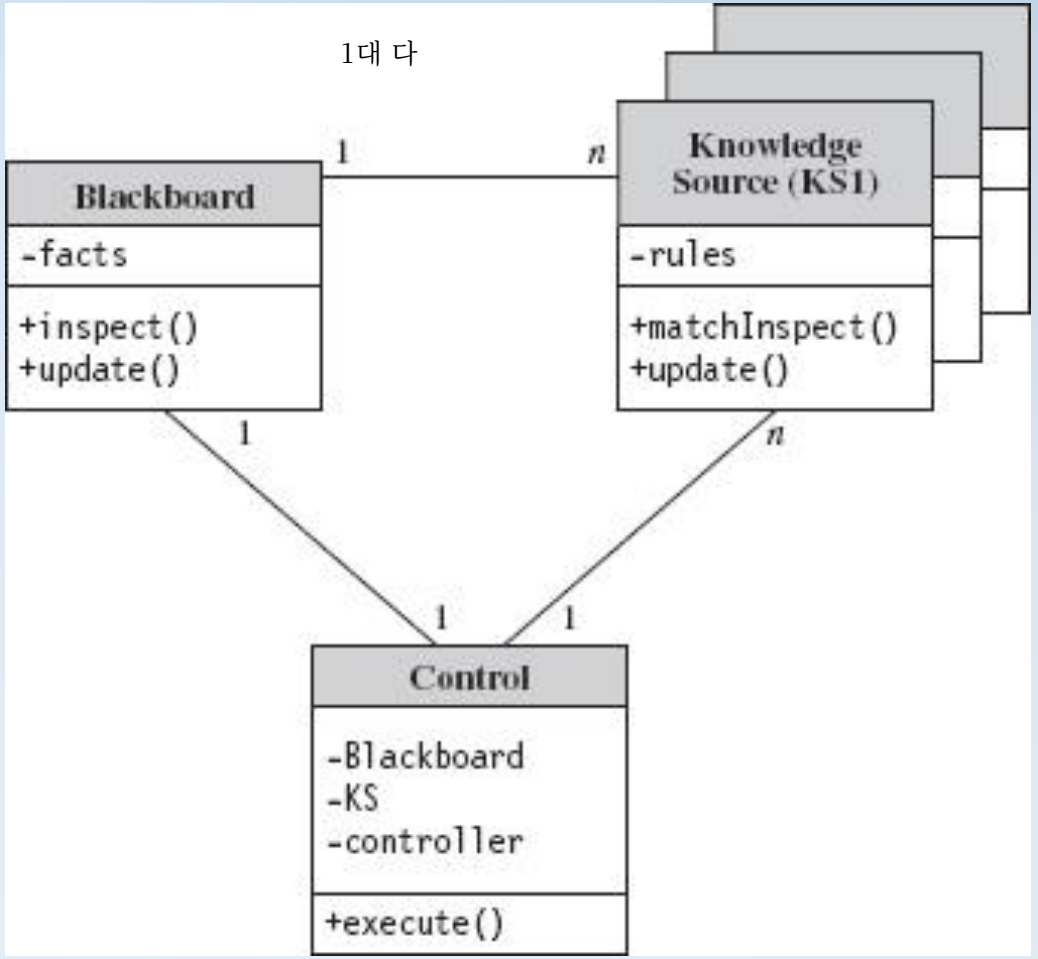
모든 놀리지 소스들이 값을 다 업데이트를 원하는대로 하고 블랙보드엥는 원하는 솔루션을 도출할 수 있을 때 끝남.

Blackboard: Rule-based

얘는 룰이 있어서 룰을 매치하면 파이어 함.
룰을 풀겠다!라는 거징.. 거기에 따라 업데이트를 함.

- Rule-based blackboard

데이터들(facts)를 가짐



직;기 풀 수 있는 룰들을 가짐

다 읽음.

Blackboard: Rule-based

- Rule-based blackboard
  - Each knowledge source helps to solve problems in its expertise area
  - Knowledge can be stored in different knowledge representation formats depending on the reasoning strategy
  - A knowledge source stores all related rules and provides activation mechanisms for the blackboard to trigger in rule-based expert system
  - Each individual knowledge source may have its own problem solving strategy and use its own knowledge expertise to contribute to a partial solution which will lead to a final solution.
  - There are many other problem solving strategies that can be applied including Fuzzy set theory, probability and statistics, neural network, data mining, and heuristic searching.
  - The blackboard class holds the current data state, and the final problem solution will be placed in the blackboard for the controller to pick up and use to generate a final report.

추론 : ~이면 ~이다.

모든 룰들을 갖고 있어서 룰에 따라 파이어가 일어남(트리거가 일어나면서) 룰이 실행될 수 있도록 해서 일들이 진행될 수 있게

이런 데에서 알고리즘을 많이 씀.
(여러 AI)

한문제를 푸는게 잘되어 있는데
여러 문제를 푸는 것은 사람이 정의해줘야 함.

Blackboard Architecture Style

스스로 activating을 함.
로우 데이터 파셜 데이터 스스로 트리거 트리거 하는 능력

- Since the blackboard architecture is basically a self-activated system, the controller subsystem in the architecture only acts at the beginning of the process to initiate blackboard and all knowledge sources;
- It also periodically inspects the current state of the blackboard to determine whether to terminate the processing if the solution is acceptable or optimal enough.

터미네이팅 하는 문제는 어려움.
28:30~29:10

Blackboard: Example

지식 기반 시스템

- Example: animal identification knowledge-based system (KBS)
  - The goal is to recognize the animal
  - A set of rules:

~31분까지 좀

    - R1: IF animal gives milk then animal is mammal
    - R2: IF animal eats meat then animal is carnivore
    - R3: IF animal is mammal and animal is carnivore and animal has tawny color and animal has black stripes then animal is tiger
  - Set of facts
    - F1: animal eats meat
    - F2: animal gives milk
    - F3: animal has black stripes
    - F4: animal has tawny color
  - Forward Reasoning vs. Backward Reasoning

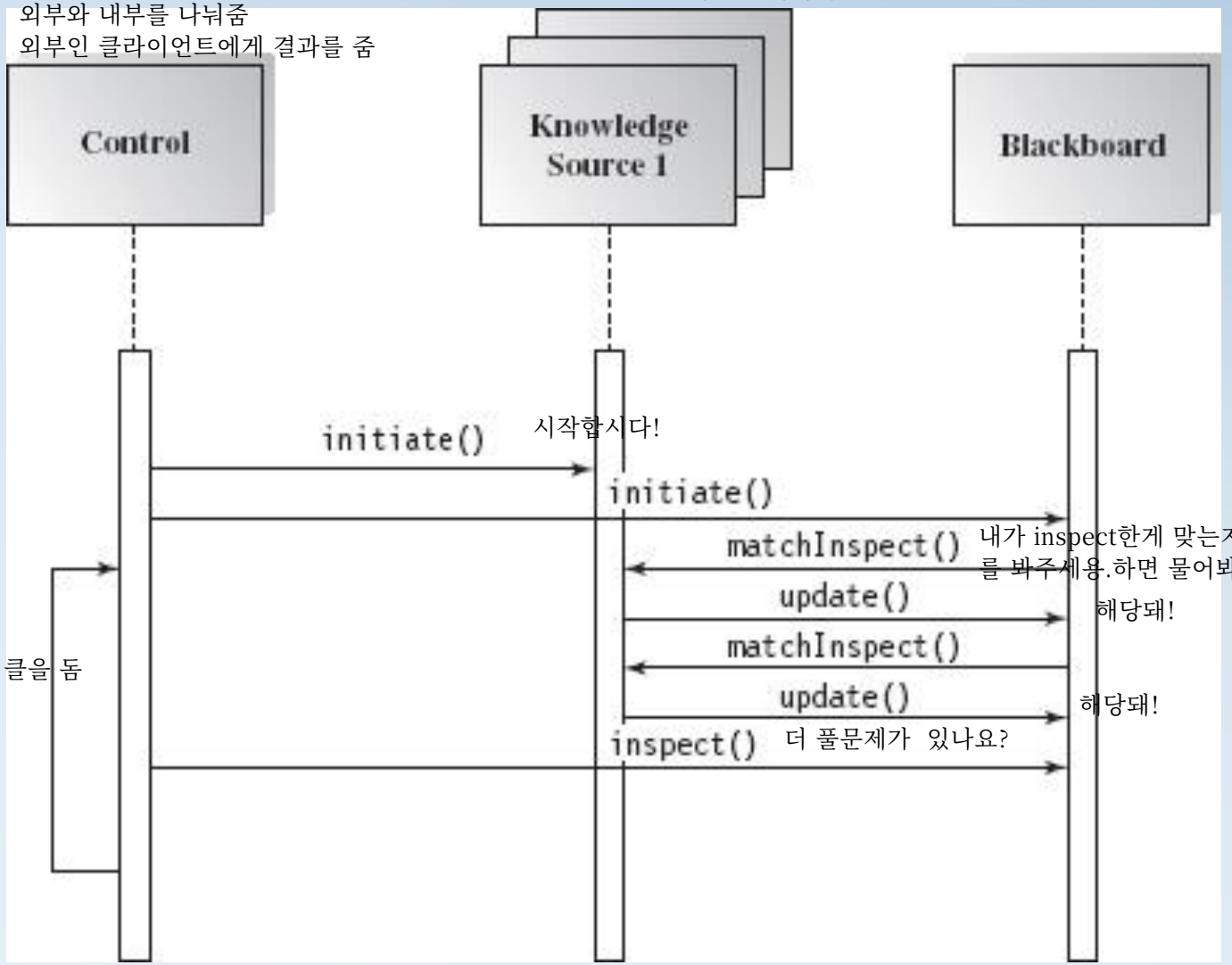
Blackboard: Example

- Sequence diagram of blackboard architecture

전 예시와 비교!

적절한 룰이 없을때까지 반복함.
놀리지 소스가 풀게 없다고 halt 들을 하면 그때 블랙보드가 우리가 원하는 결과가 됨.

외부와 내부를 나눠줌
외부인 클라이언트에게 결과를 줌



앞의 것은 컨트롤러가 하나씩 물어봐줌.
컨트롤이 회의를 진행.

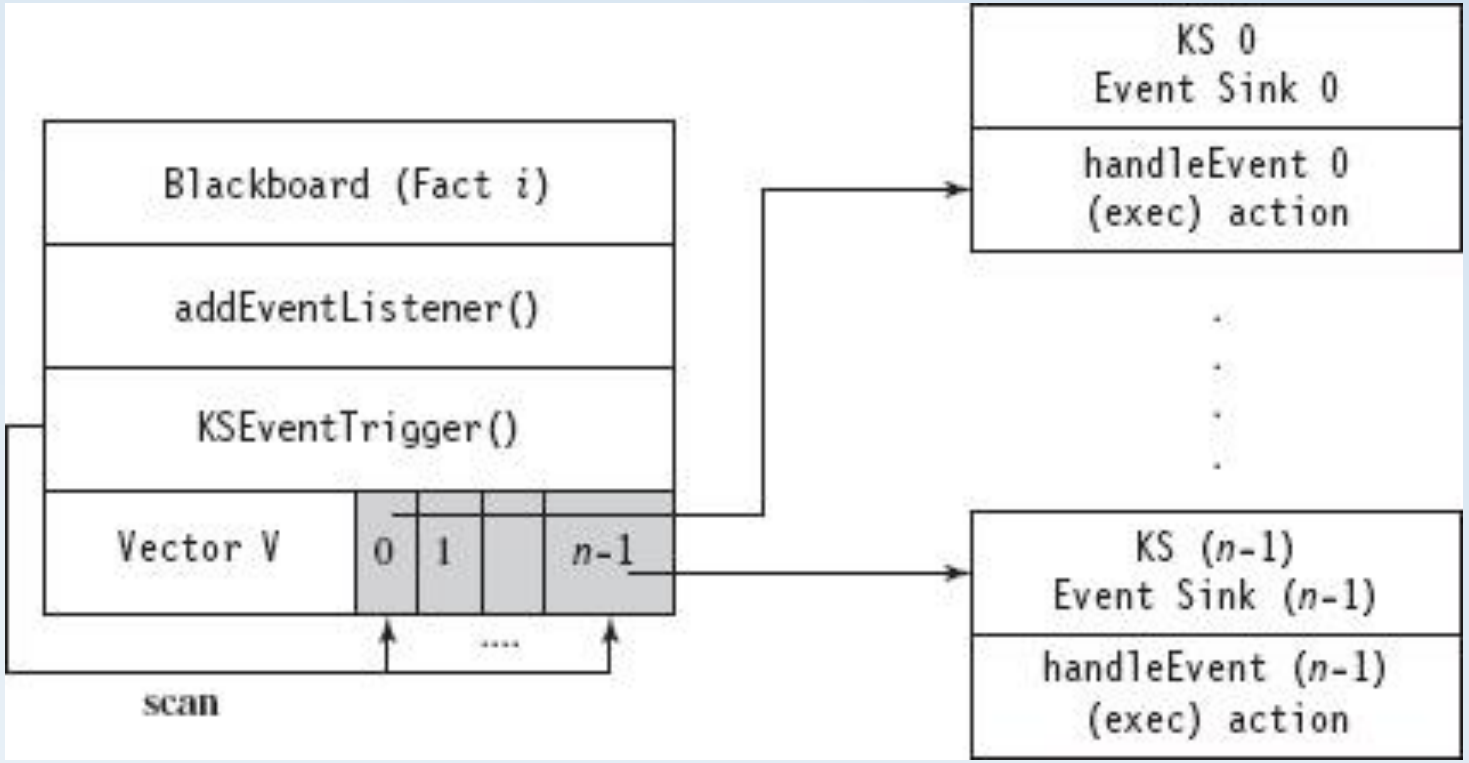
이 케이스는 블랙보드가 그 일을 함. 정의 자체는 이 케이스가 맞음.
왜냐면 스토리지가 스스로 액티브해서 트리거해서
스스로 바꿀 수 있도록 함.(definition은 이 케이스가 유사)
시스템마다 쓰는 방식이 다르다

없다고하면 사이클을 돔

Blackboard: Example

32분~33분 30초

- Publish/subscribe relationship between blackboard and knowledge sources



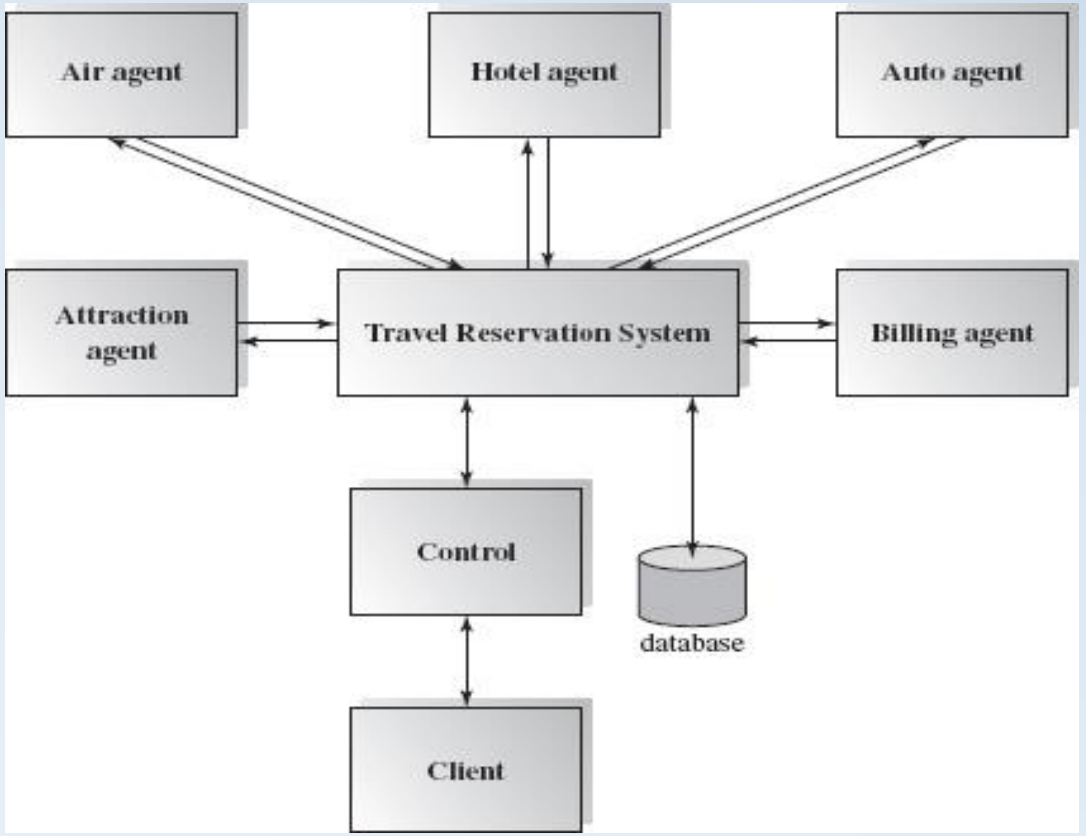
Blackboard: Example

- Application Examples
  - smart travel consulting system
    - involves a data store of budget, available time, and other facts to be shared by all agents

<여행을 가고 싶은데 아직 결정된 것은 아무것도 없고 걍 컨설팅 받고 싶음!>

자기상황을 주고 컨설팅을 받음

이런 것들도 엑스퍼트 시스템으로 충분히 가능하당!



Blackboard: Application

AI 부문에 있어 문제가 오픈되어있고 결정적으로 풀지 못하는 복잡한 문제를 푸는데 적합함.
뉴럴 네트워크를 쓰는 것에 있어서는 문제 접근법과 프리딕션이 정확히 나옴 - 엑스퍼트 시스템이 뉴럴 네트워크를 따라가지 못함.
엑스퍼트 - 룰이 정확하지 않아서 정확도가 떨어짐
블랙보드의 좋은 점은 잘 모르는 도메인에 익스퍼트가 있고 모든게 룰에 따라 움직인다는게 장점
크리티컬 패스(가장 긴 경로)는 차라리 룰에 따라 하는 것이 적합함.

- Applicable domains
  - Suitable for solving open-ended and complex problems such as artificial intelligence (AI) problems where no preset solutions exist.
  - The problem spans multiple disciplines, and each problem involves completely different types of knowledge expertise and problem-solving paradigms that require cooperation.
  - Partial or approximate solution is acceptable to the problems.
  - Exhaustive searching is impossible and impractical since it may take forever because available knowledge and even data and hypotheses may not be complete or precisely accurate.

Blackboard: Benefits & Limitations

- **Benefits:**
  - *Scalability:* easy to add or update knowledge source. 놀리지소스를 필요하다면 더 붙일 수 있음
  - *Concurrency:* all knowledge sources can work in parallel since they are independent of each other. 놀리지소스는 동시 다발적으로 일을 처리할 수 있음.
  - Supports experimentation for hypotheses. 가정 위주로 일을 처리
  - Reusability of knowledge source agents. 재사용성이 좋아짐. 예를 들어 신발공장인 것을 모자공장으로 바꿀 때도 유사한 놀리지소스를 적용할 수 있음.
- **Limitations:** 블랙보드와 놀리지소스가 같은 도메인. 키포인트는 아님, 속성이다라고 이해하면 됨.
  - Due to the close dependency between the blackboard and knowledge source, the structure change of the blackboard may have a significant impact on all of its agents.
  - Since only partial or approximate solutions are expected, it can be difficult to decide when to terminate reasoning. 언제 끝날지 모름. 홀트가 다 되는 것이 마지막 포인트긴 한데 빨리 결정을 해야할 때도 있음. 리즈닝을 추론해나가는, 옵티말 솔루션을 찾는 경우 언제 터미네이션을 할지 알기 어렵다
  - Synchronization of multiple agents is an issue. Since multiple agents are working and updating the shared data in the blackboard simultaneously, the preference or priority of executions of multiple agents must be coordinated. 놀리지 소스가 저마다 일처리를 하는데 멀티플 에이전트가 스스로 동기화 하는 것이 오픈되어 있음. 이런 동기화가 고려해야하는 문제
  - Debugging and testing of the system is a challenge. el버깅과 테스팅이어려움 . 어떤 파셜 솔루션부터가 문제인지 알아내기 쉽지 않음.
- Related Architecture
  - Implicit invocation architecture such as event-based, MVC architecture

Types of Data-centered Architecture

- Repository
- Blackboard

Summary of Data-Centered (DC) Software Architectures

엑스퍼트 시스템에서는 블랙보드 시스템을 사용

Summary of DCR

- Repositories are often used in layered architecture, client-server architecture, data tier in multi-tier architecture, and many other architecture designs.

클라이언트 - 서버는 먼티 티어에서 티어를 넣어줬을때 사용
에이전트들이 자신의 플로우를 따라 접근하게 된다.
레파지토리는 패시브하다.

- Agents of the data store in data-centered repository architecture control the logic flow.
- The data store is passive in the repository architecture, and agents control and trigger all operations on the data store.
- There are many expert systems for pattern recognition, voice or speech recognition, or other similar systems with this architecture.

데이터가모인 구조는 블랙보드 엑스퍼트 시스템은 다른 형태로 접근을 함.