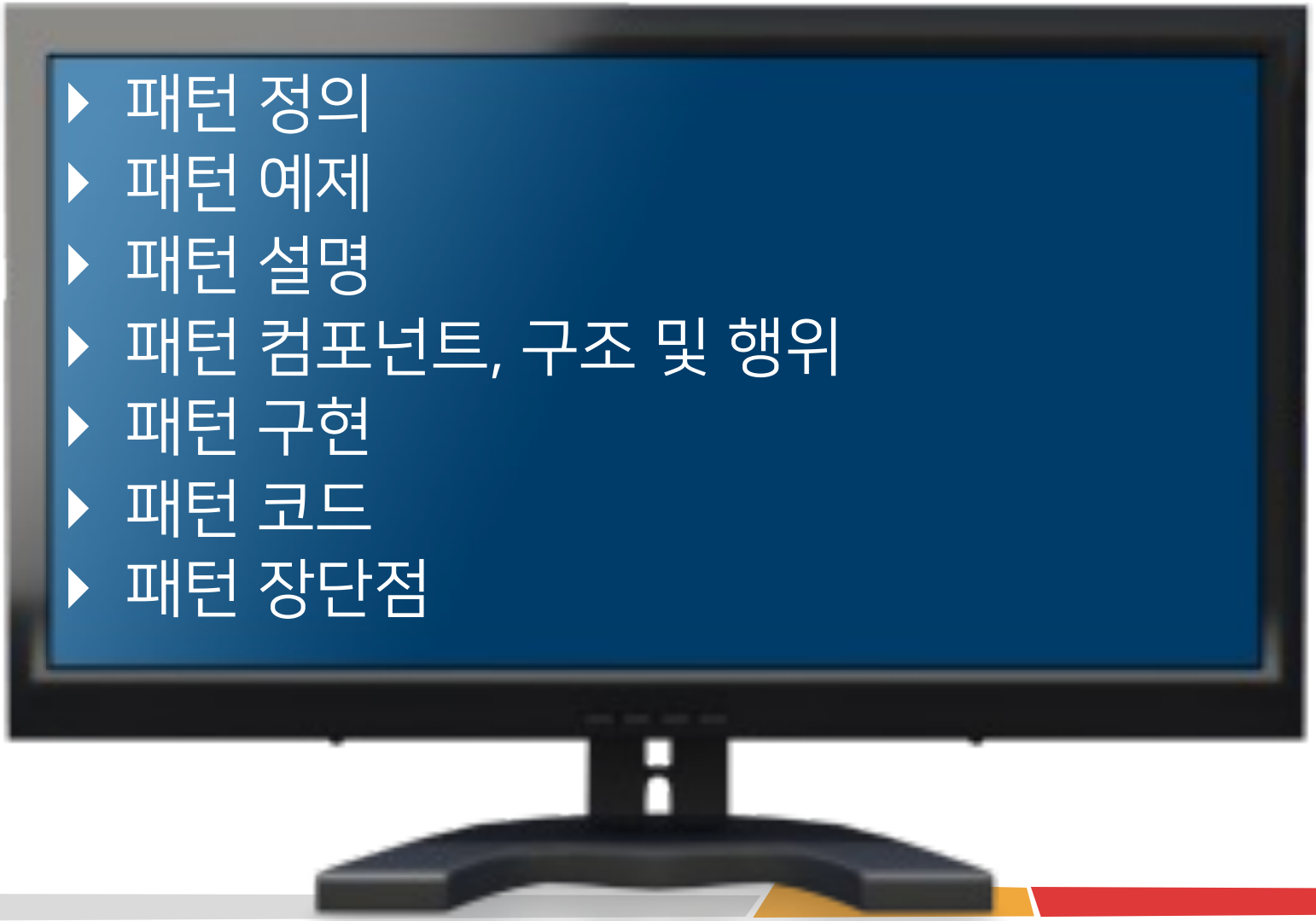
Spring 2023

소프트웨어 아키텍처 패턴: Blackboard

Seonah Lee

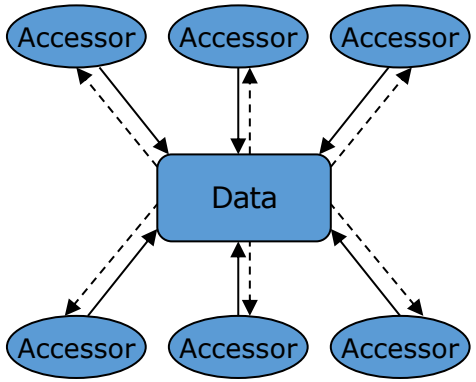
Gyeongsang National University

Blackboard 패턴



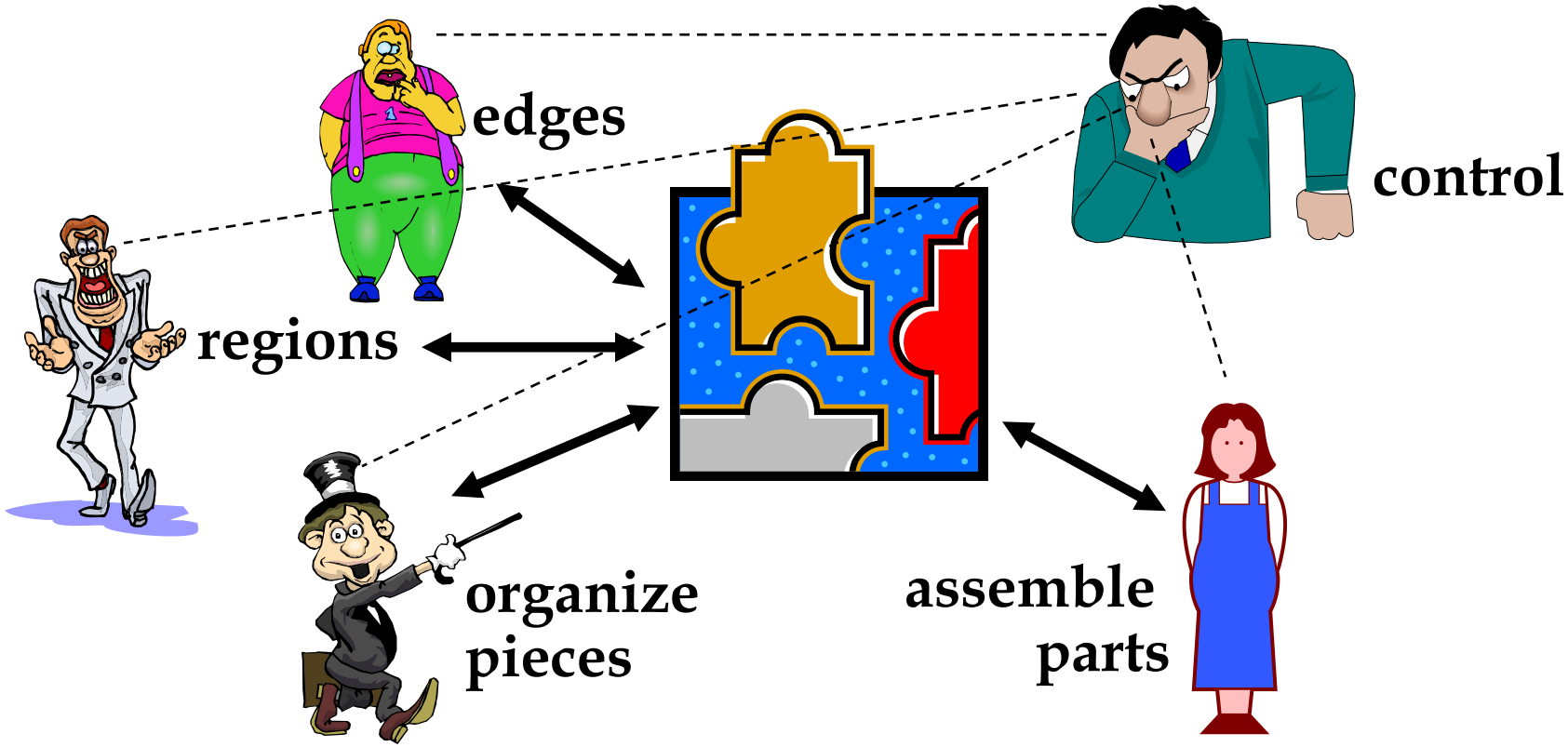
Blackboard Pattern: Definition

- 정의
  - **Shared data, database**와 같은 데이터 중심 패턴 중에 하나
  - 명확히 정의된 문제 해법이 없을 때 문제를 풀어가는 하나의 방식을 정의한 패턴
  - 대략적으로 해법을 수립하기 위해 특수한 서비스 시스템의 지식을 조합하는 패턴
- 예제
  - **AI system**
  - **Signal processing**
  - **Radar/sonar**
  - **Vision processing**
  - **Speech processing**



Blackboard Pattern: Example

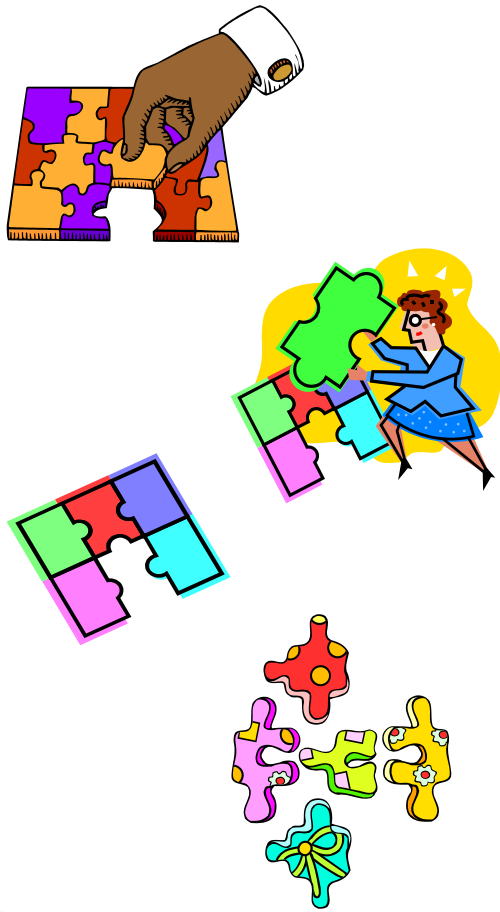
- How do you solve a puzzle?



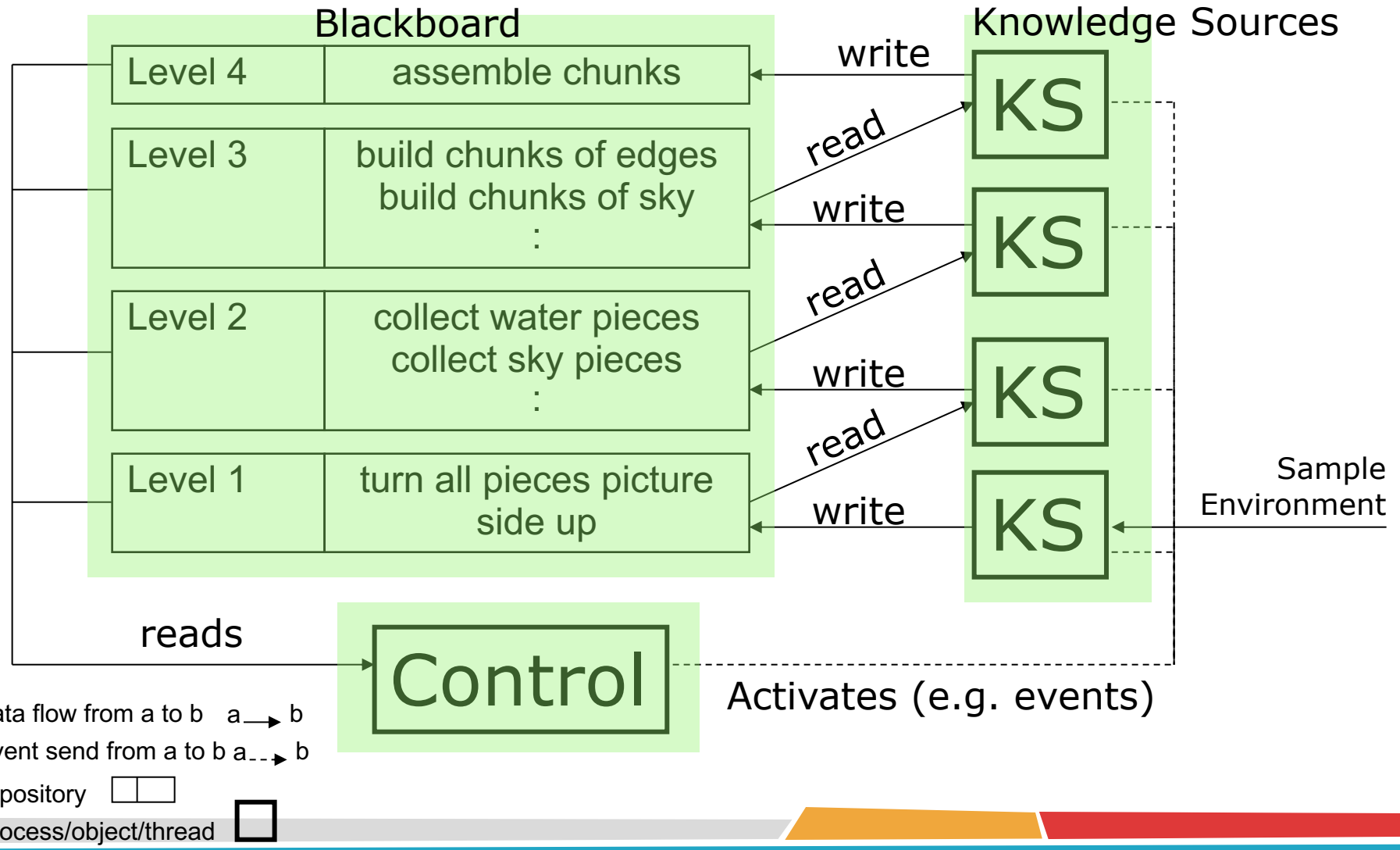
Blackboard Pattern: Example

- How do you solve a puzzle?

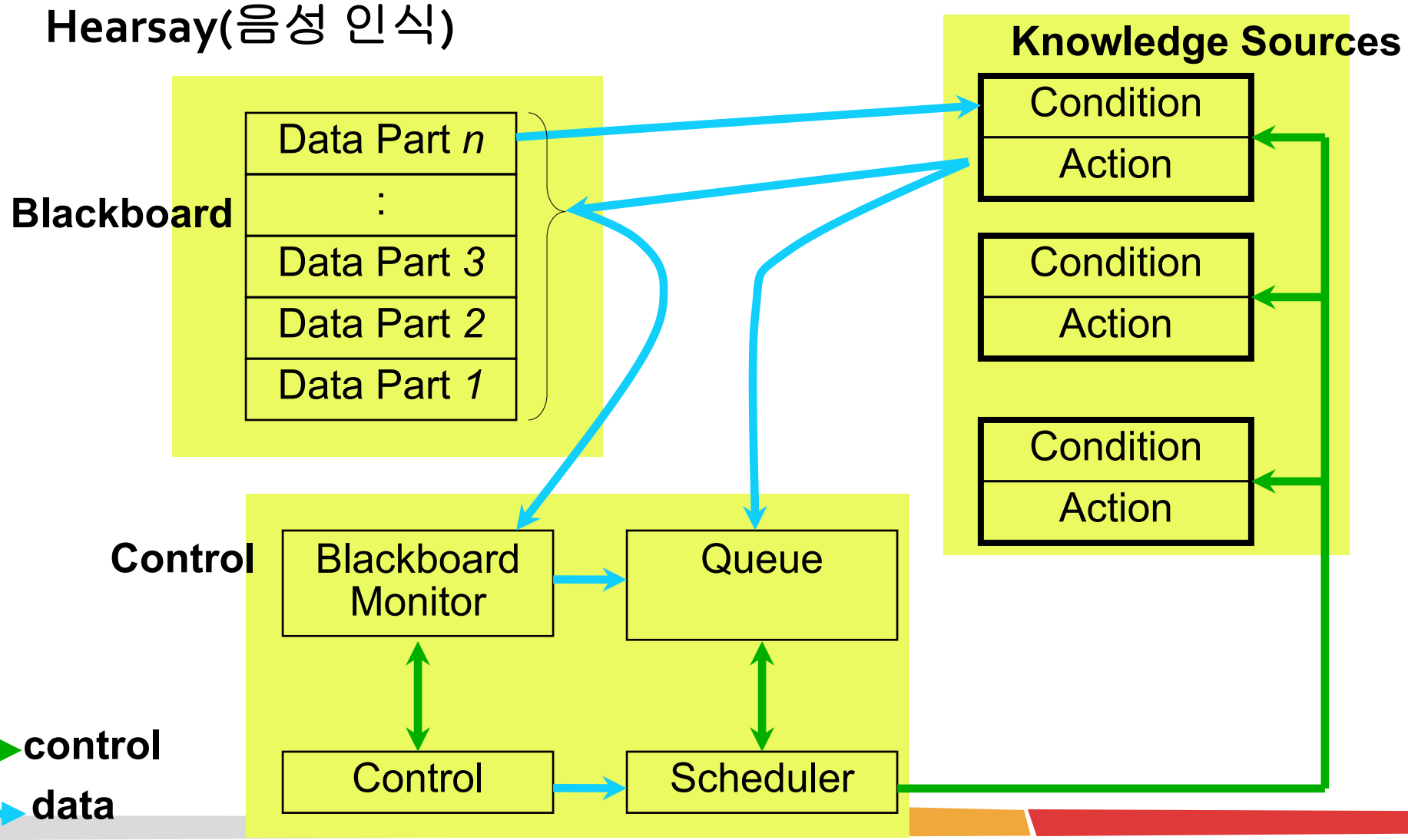
Level 4	assemble chunks
Level 3	build chunks of edges build chunks of sky :
Level 2	collect water pieces collect sky pieces :
Level 1	Turn all pieces picture side up



Blackboard Pattern: Example

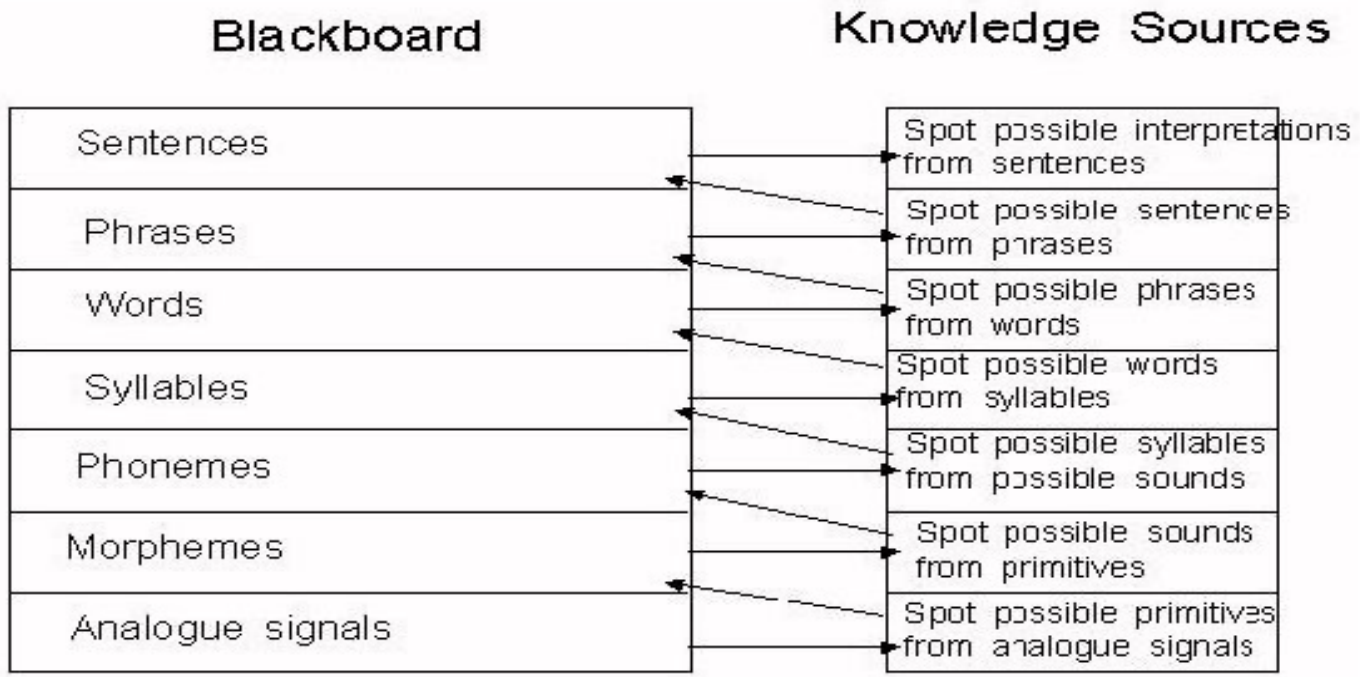


Blackboard Pattern: Example



Blackboard Pattern: Example

- 음성 인식 시스템
  - **Input:** 파형 형태의 음성
  - **Output:** 시스템이 인식한 문장



Blackboard Pattern: Description

- 정황(Context)
  - 도메인의 해법이 명확하지 않은 경우
- 문제(Problem)
  - 컴퓨터 비전, 음성 인식, 화상 인식 등 도메인 분야와 같이 가공하지 않은 데이터(raw data)를 상위 수준의 데이터 구조로 변환하기 위한 결정적인 해법이 존재하지 않음
  - 하나의 문제를 분해했을 때 여러 전문 분야들이 관련된 여러 하위 문제들로 나뉨. 따라서 제각기 다른 표현 방식과 패러다임으로 접근해야 함
  - 하위 문제들을 해법이 있다 해도 그 결과를 다시 조합하여 상위 수준의 문제를 포괄적으로 해결하는데 어떤 전략도 기존에 정의되어 있지 않은 경우.
    - 몇몇 해법 단계를 정렬해서 그 동작을 하드코딩 하는 것이 불가능

Blackboard Pattern: Description

- 해법(Solution)
  - 일반적인 데이터 구조(**blackboard**)를 가지고 각각 독립적으로 동작하는 프로그램들(**Knowledge sources**)을 모음
    - 각 독립적인 프로그램은 전체 해법의 일부분을 해결하기 위해 특수화된 작업을 가짐
    - 각 프로그램은 서로 호출하지 않고, 미리 정의된 순서에 따라 일을 처리하지 않음
  - 제어 컴포넌트 (**Control**)에 의해 모든 독립적인 프로그램들은 하나의 해법을 찾기 위해 서로 협력하여 동작함
    - 제어 컴포넌트는 현재의 진행 상태에 따라 다음 일을 작업할 프로그램을 결정
    - 제어 컴포넌트는 현재의 진행 단계를 평가하고 독립적인 프로그램들을 조율

Blackboard Pattern: Components

Blackboard
• 중앙 데이터를 관리

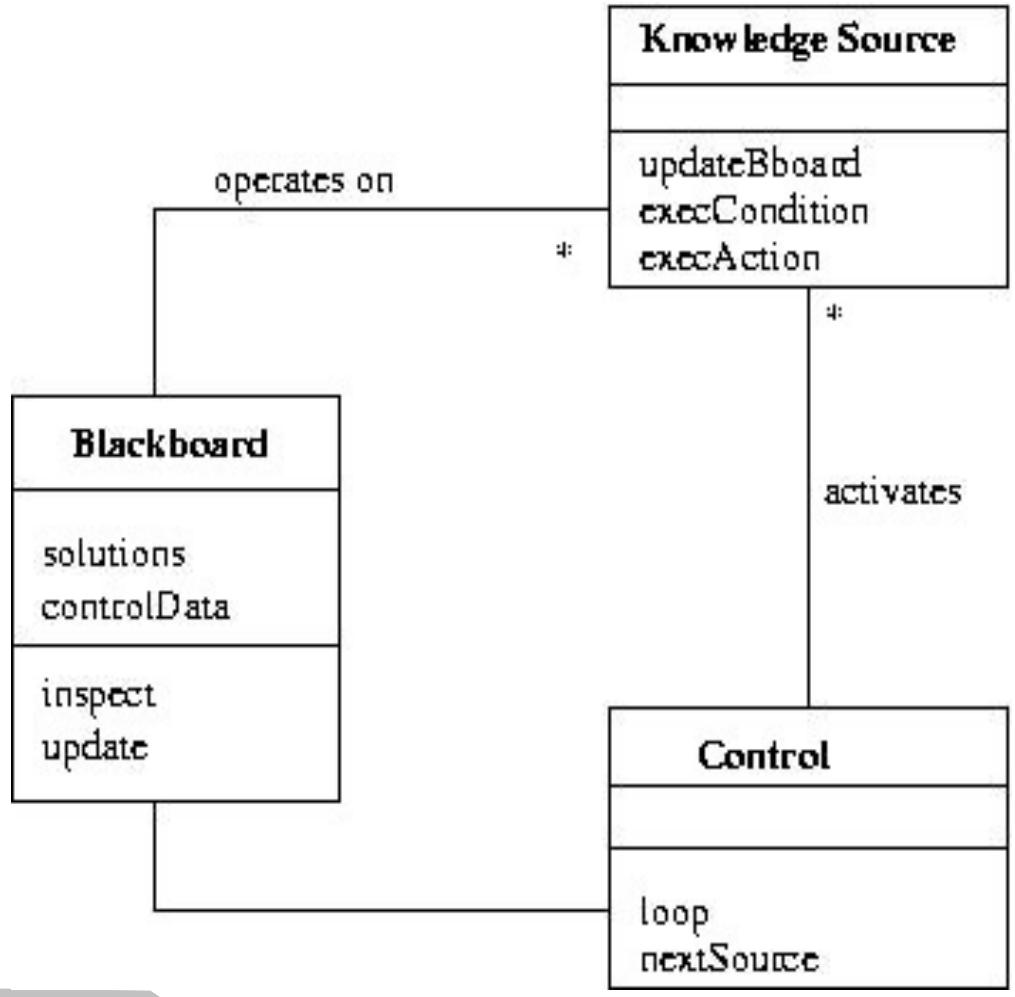
Knowledge Source
• 적용이 가능한지 자체적으로 평가 • 결과를 계산 • Blackboard 컴포넌트를 업데이트

Control
• Blackboard 컴포넌트를 모니터링 • Knowledge Source 컴포넌트 동작을 스케줄링

Blackboard Pattern: Components

- **Blackboard**
  - 중앙 데이터를 관리: 해법 공간(**Solution space**)과 제어 데이터(**Control data**)를 저장
  - 문제의 현재 상태를 보임: **Knowledge Source**들이 읽고 쓸 수 있는 인터페이스 제공
- **Knowledge Sources**
  - **Blackboard** 상태를 업데이트
  - 특정 도메인의 해법을 제시
  - **Blackboard**에 그 해법을 적용 (적용이 가능한지 자체적으로 평가)
- **Control**
  - **Blackboard**의 변경사항들을 모니터링(다음에 어떤 동작을 수행할지 결정하는 루프 반복)
  - **Knowledge Sources** 스케줄을 관리하고 실행시킴

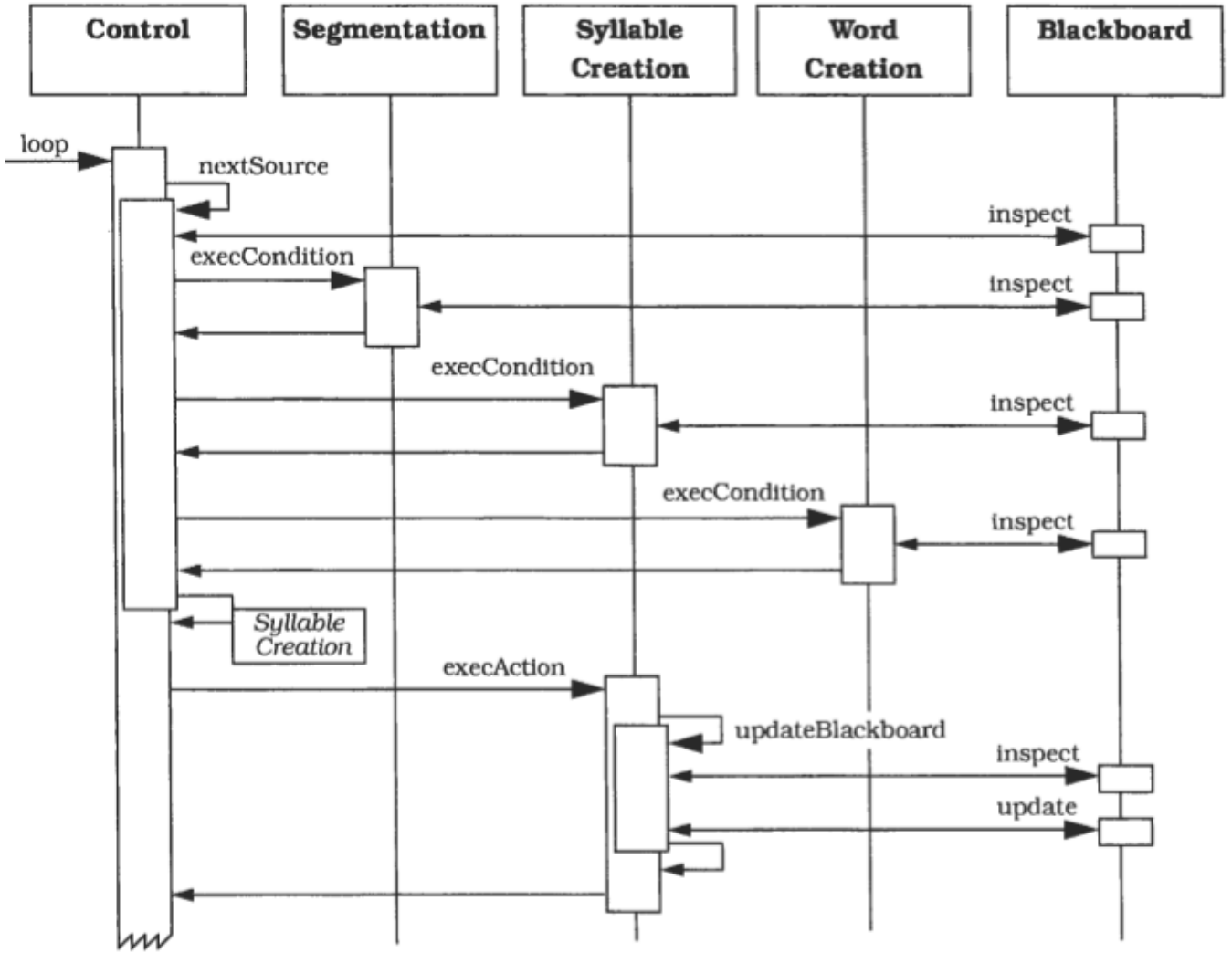
Blackboard Pattern: Structure



1. Start Control::loop
2. Control::nextSource
3. determine potential knowledge sources by calling Blackboard::inspect
4. Invoke KnowledgeSource::execCondition of each candidate knowledge source
5. Each candidate knowledge source invokes Blackboard::inspect to determine if/how it can contribute to current state of solution
6. Control chooses a knowledge source to invoke by calling KnowledgeSource::execAction
7. Executes KnowledgeSource::updateBlackboard
8. Calls Blackboard::inspect
9. Calls Blackboard::update

Blackboard Pattern: Behavior

- 음성인식에 기초한 예제



Blackboard Pattern: Realization

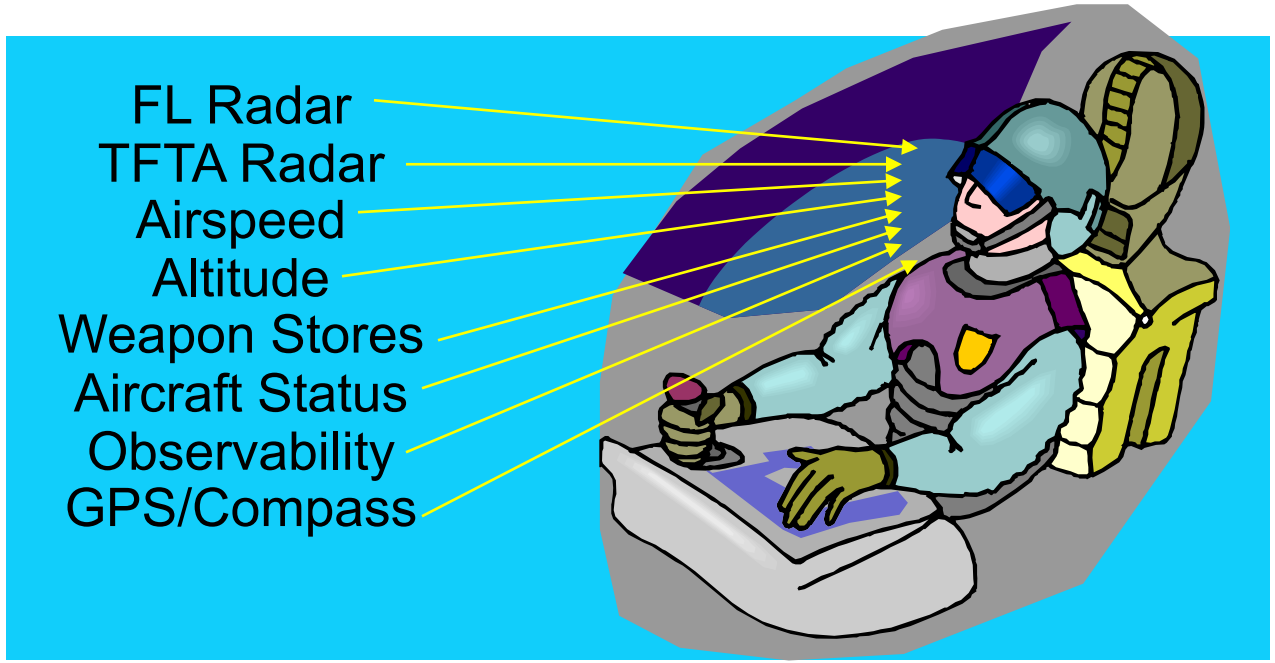
- 설계 순서
  1. 문제의 도메인을 정의하고 해법을 찾기 위해 일반적인 지식 분야를 상세히 살펴본다. 시스템의 입력, 출력, 상호작용을 자세히 정의한다.
  2. 문제에 대한 해법 공간**(Solution space)**을 정의한다. 해법에 대한 추상화 수준을 상위 수준에서 하위 수준까지 구별하며, 또한 부분 해법을 구별한다.
  3. 해법 수준에 맞게 **Knowledge source**를 정의하고 각 수준으로 분할한다.
  4. 모든 **Knowledge source**가 **blackboard**와 상호작용하는 표현 방식을 찾아서 정의한다. **(blackboard** 어휘를 정의한다.**)**
  5. **Control**을 정의한다. 또한 **Control**이 변경을 수행할 **Knowledge source**를 선택하는 전략을 수립한다.

Blackboard Pattern: Case Studies

- F-22 Sensor Fusion System
  - 비행기 조종사는 넘쳐나는 데이터로 인해 많은 스트레스를 받는다.
    - avoid being detected
    - engage enemy fighters
    - engage ground targets
    - avoid terrain obstacles
    - avoid surface-to-air ordinance
    - navigate and find the targets
    - oh yeah,... and fly the airplane too!

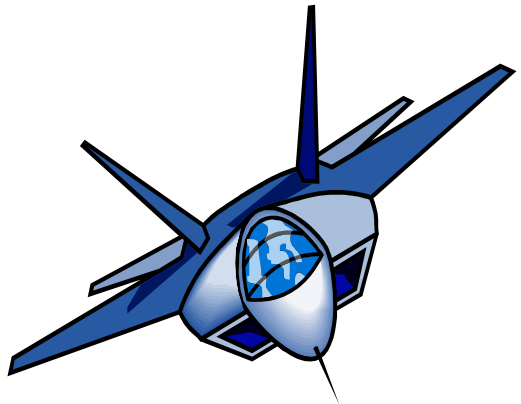
Blackboard Pattern: Case Studies

- 기존 시스템은 센서, 표시판, 조절 기능이 분리되었음
- 조종사가 데이터를 조합하고 판단하여야 함



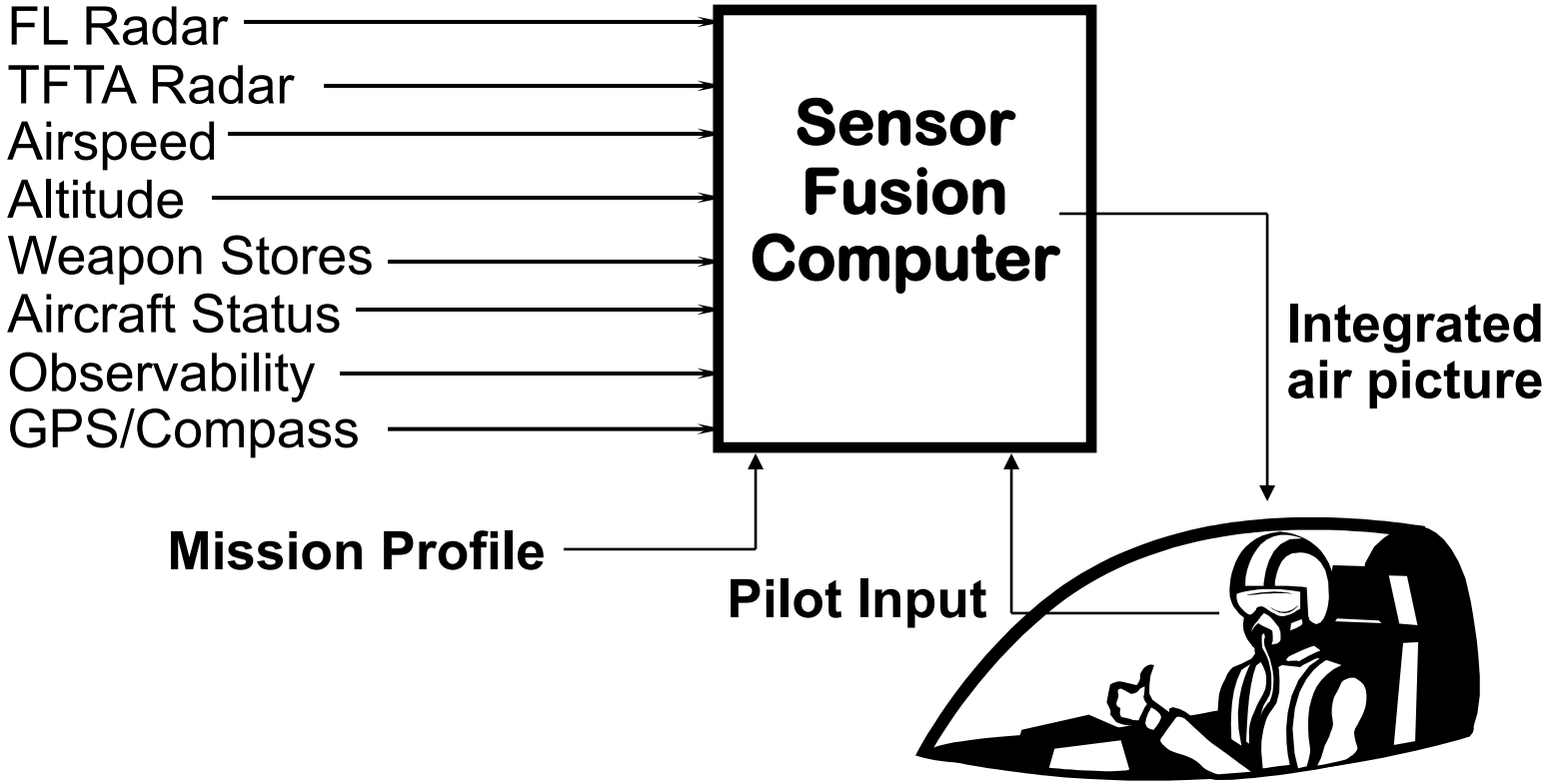
Blackboard Pattern: Case Studies

- **F-22 Advanced Tactical Fighter's Sensor Fusion System**
  - 목적
    - 데이터 통합
    - 필요 데이터만 선별적으로 보드에 표시
    - 보기 좋은 형태로 데이터를 표시
- **F-22** 비행기에서 가장 복잡한 기계 중에 하나로서 **blackboard** 패턴을 사용

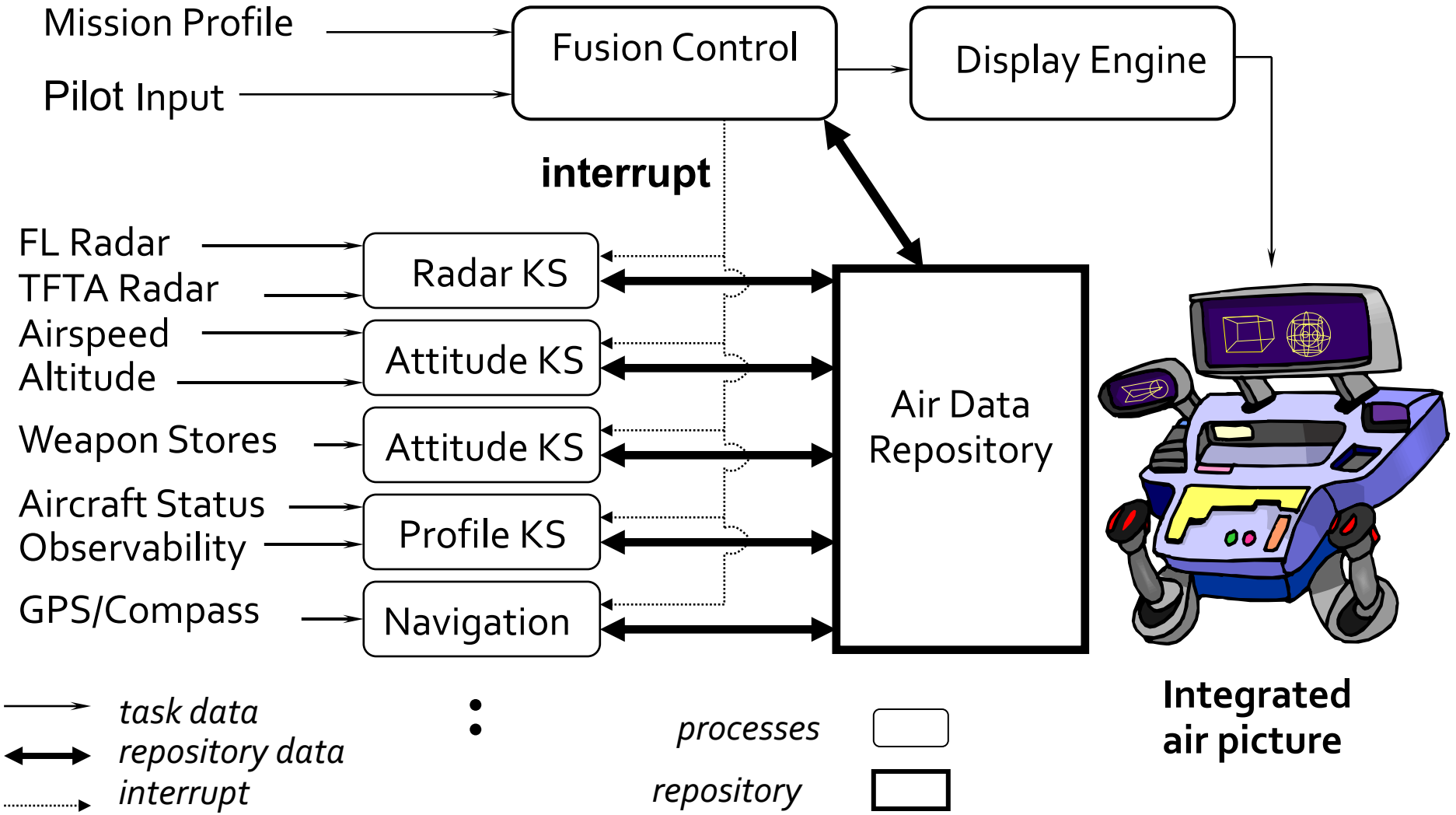


Blackboard Pattern: Case Studies

- **The F-22 Advanced Tactical Fighter's Sensor Fusion System** 은 조종사가 필요로 하는 정보를 하나의 계기판에 표시해 준다."



Blackboard Pattern: Case Studies



Blackboard Pattern: Benefits

- 완벽한 해법을 찾기 어려운 경우에 사용할 수 있음; 다양한 실험 가능
- **KS, Control, Blackboard** 가 독립적으로 동작하여 가변성이나 유지보수성이 좋음
- **KS**는 타 문제 도메인에 재사용될 수 있음
- 장애 허용**(Fault tolerance)**과 저항성**(robustness)**을 지원함

Blackboard Pattern: Liabilities

- 계산 결과가 항상 동일하지 않아 테스트가 어려움
- 어떤 좋은 해법도 완벽하다고 장담할 수 없음;
- 좋은 전략 수립 어려움; 효율이 낮음
- 많은 시간에 걸쳐 수정되어야 하므로 개발에 많은 노력이 필요

Question?

Seonah Lee

saleese@gmail.com