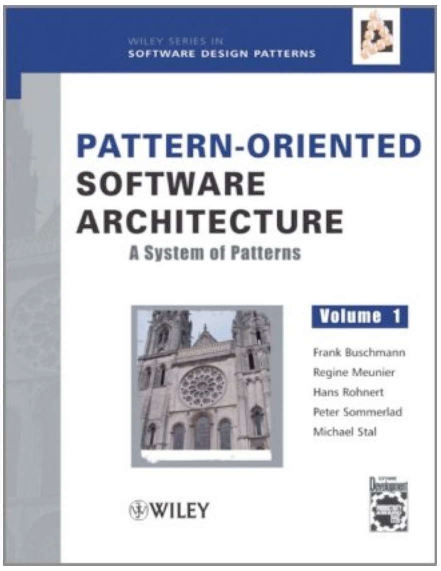
# Architectural Patterns

Baris Aktemur

CS 534 | Ozyegin University

Contents from *Pattern-Oriented Software Architecture*, by Buschmann, Meunier, Rohnert, Sommerlad, Stal. 1996, John Wiley Books.



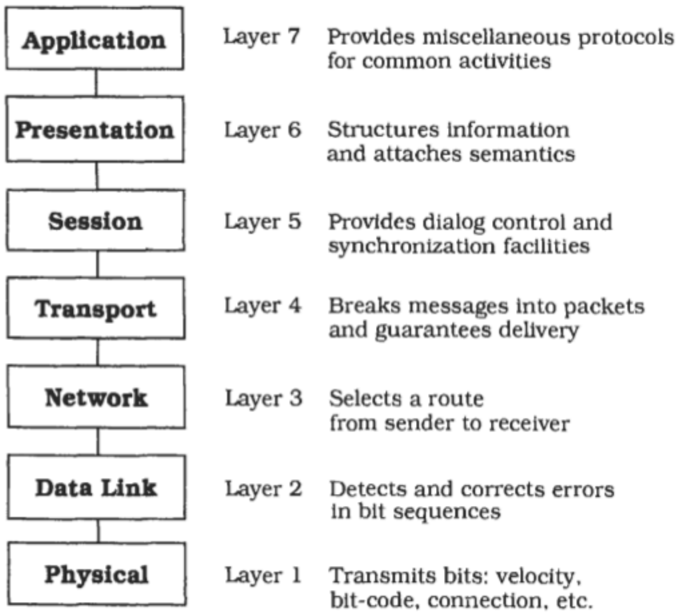
## Architectural Patterns

- Architectural patterns express fundamental structural organization schemas for software systems.
  - provide a set of predefined subsystems,
  - specify their responsibilities, and
  - include rules and guidelines for organizing the relationships between them.

## Layers

## Layers

- Helps to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.



| | | |
|---|---|---|
| **Application** | Layer 7 | Provides miscellaneous protocols for common activities |
| **Presentation** | Layer 6 | Structures information and attaches semantics |
| **Session** | Layer 5 | Provides dialog control and synchronization facilities |
| **Transport** | Layer 4 | Breaks messages into packets and guarantees delivery |
| **Network** | Layer 3 | Selects a route from sender to receiver |
| **Data Link** | Layer 2 | Detects and corrects errors in bit sequences |
| **Physical** | Layer 1 | Transmits bits: velocity, bit-code, connection, etc. |

## Layers

- Each layer deals with a specific aspect of communication and uses the services of the next lower layer.
- System is a mix of low- and high-level issues, where high-level operations rely on the lower-level ones.

## Benefits

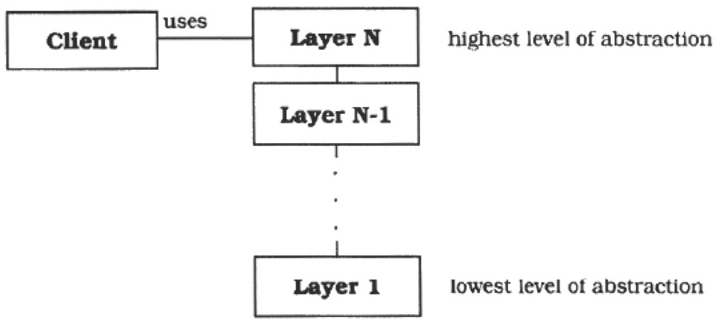
- aiding development by teams
- supporting incremental coding and testing.
- easier exchange of individual parts at a later date.
- increasing chances of reuse

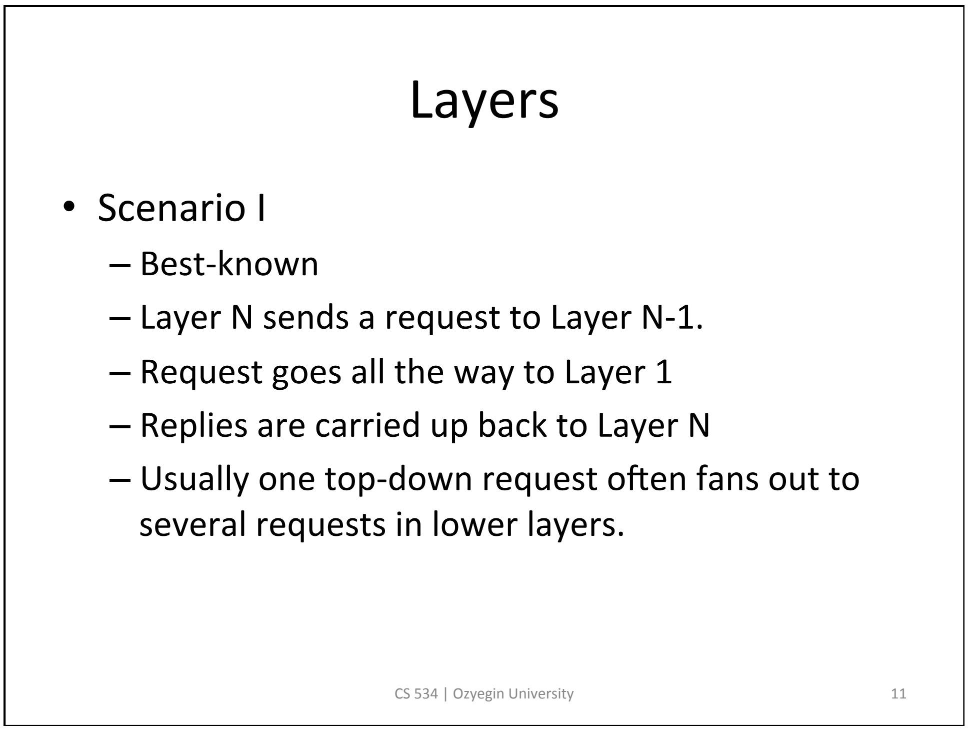
## Forces

- Interfaces should be stable. and may even be prescribed by a standards body
  - Façade?
- Similar responsibilities should be grouped to help understandability and maintainability.
  - Each component should be coherent.
  - Grouping and coherence are conflicting at times.
- Crossing component boundaries may impede performance
  - when a substantial amount of data must be transferred over several boundaries

## Layers

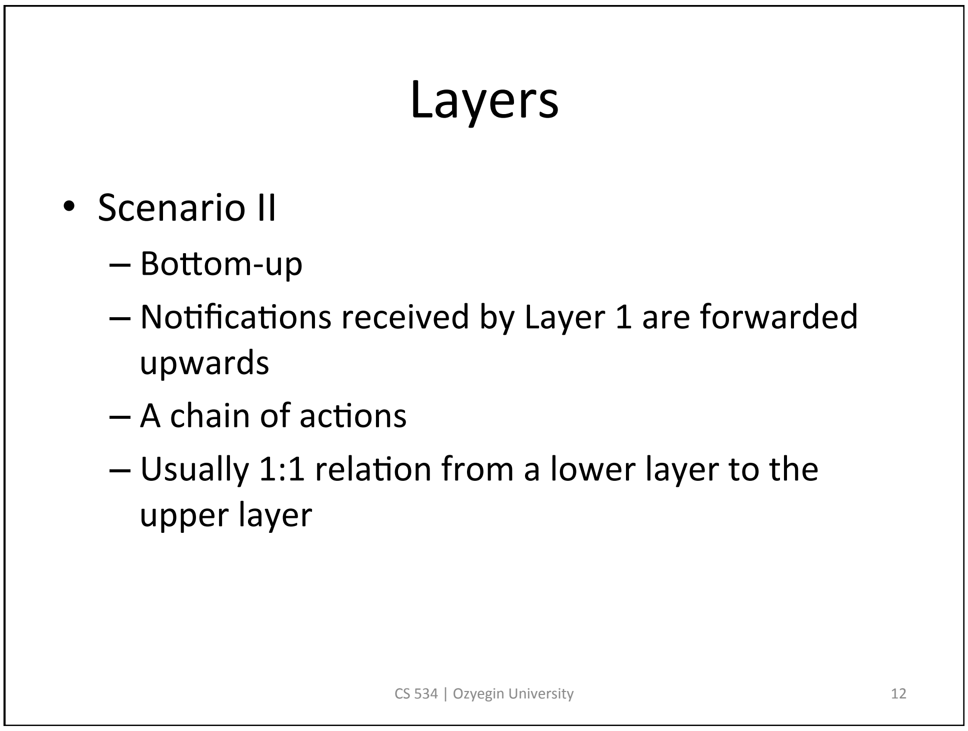
- Each individual layer shields all lower layers from direct access by higher layers





## Layers

- Scenario I
  - Best-known
  - Layer N sends a request to Layer N-1.
  - Request goes all the way to Layer 1
  - Replies are carried up back to Layer N
  - Usually one top-down request often fans out to several requests in lower layers.



## Layers

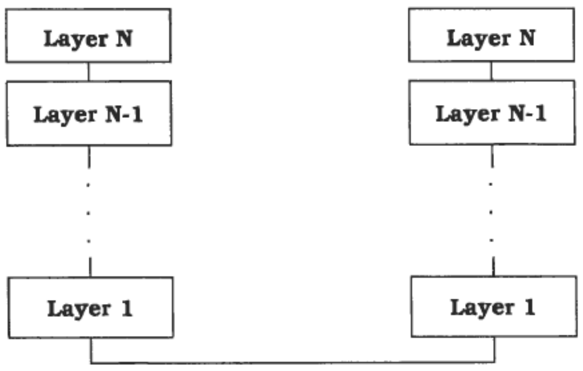
- Scenario II
  - Bottom-up
  - Notifications received by Layer 1 are forwarded upwards
  - A chain of actions
  - Usually 1:1 relation from a lower layer to the upper layer

## Layers

- Scenario III
  - Request from the top layer does not go all the way down
  - Request satisfied by an intermediate layer
  - E.g: caching
- Scenario IV
  - Notification from Layer 1 does not go all the way up
  - E.g: Dropping a data packet

## Layers

- Scenario V
  - Communicating Layers



## Layers

- When an individual layer is complex it should be broken into separate components.
  - Bridge pattern can support multiple implementations of services provided by a layer.
  - Strategy can support the dynamic exchange of algorithms used by a layer.
- Push and pull model used in exchange of data between two layers

## Layers

- Often an upper layer is aware of the next lower layer, but the lower layer is unaware of the identity of its users.
- This implies a one-way coupling only: changes in Layer J can ignore the presence and identity of Layer J+ 1
- Dependency Inversion Principle?

## Error Handling

- Can be rather expensive for layered architectures with respect to processing time and programming effort.
- An error can either be handled in the layer where it occurred or be passed to the next higher layer.
- Try to handle errors at the lowest layer possible.

## Relaxed Layered Model

- Each layer may use the services of all layers below it, not only of the next lower layer.

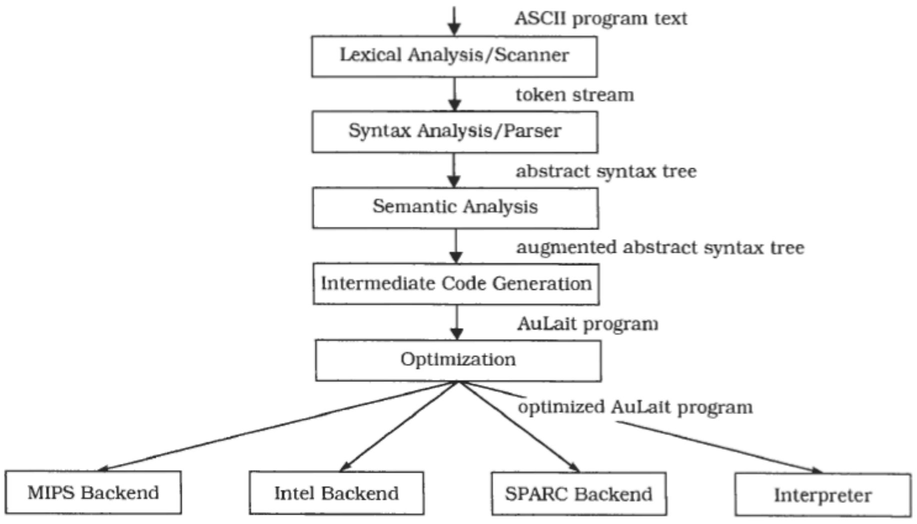
## Layering Through Inheritance

- Lower layers are implemented as base classes.
- A higher layer requesting services from a lower layer inherits from the lower layer's implementation
- Advantage: higher layers can modify lower-layer services according to their needs.
- Disadvantage: higher layer closely tied to the lower layer.

## Pipes and Filters

## Pipes and Filters

- Provides a structure for systems that process a **stream of data**.
- Each processing step is encapsulated in a filter component.
- Data is passed through pipes between adjacent filters.
- Recombining filters allows you to build families of related systems.



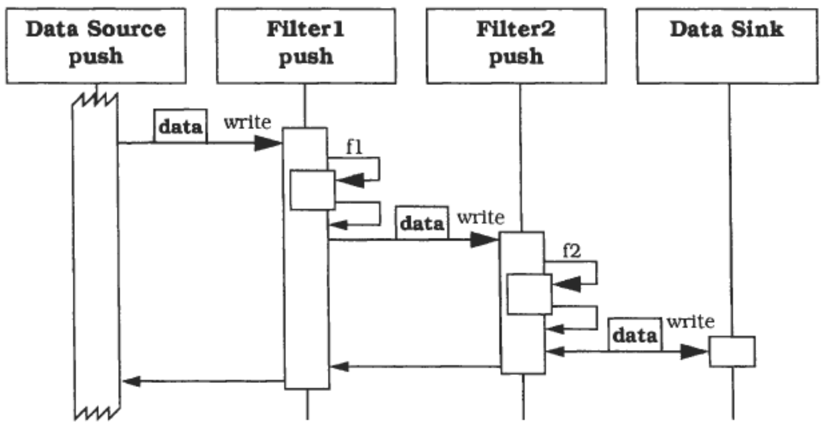
## Forces

- Be able to exchange processing steps
- Small processing steps are easier to reuse in different contexts than large components.
- Non-adjacent processing steps do not share information.
- Different sources of input data exist
  - network connection or a hardware sensor
- You may not want to rule out multi-processing the steps
  - E.g. pipelined processors

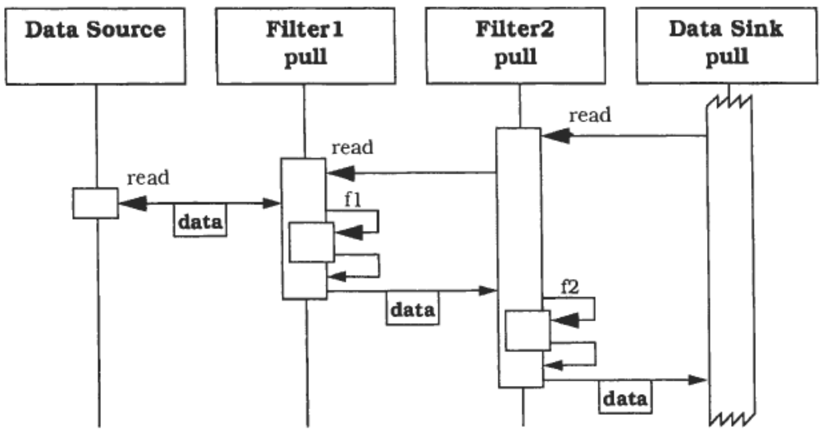
## Data Flow

- Pipes denote the connections between filters
  - a first-in- first-out buffer
- Most commonly, the filter is active in a loop, pulling its input from and pushing its output down the pipeline.

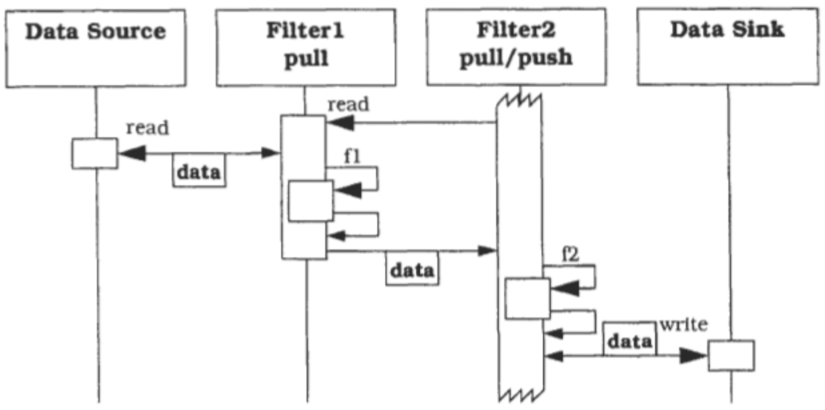
## Scenario I: Push Pipeline



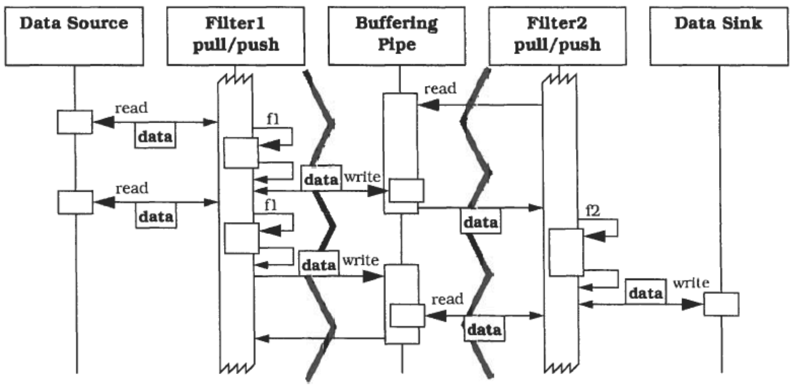
## Scenario II: Pull Pipeline



## Scenario III: Mixed



## Scenario III: Active Loop

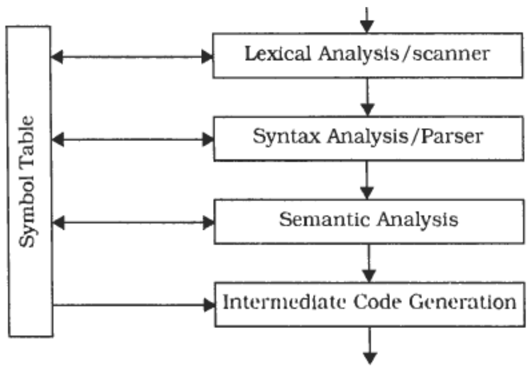


- Each filter therefore runs in its own thread of control

## Implementation

- What is the data format to be passed along each pipe?
  - ASCII: very flexible but inefficient
- Active pipe: converts a format to another
- Observer: Notify a filter if buffer has new data
- Error handling: Hard, because pipeline components do not (usually) share any global state

## Pipes and Filters with Global State



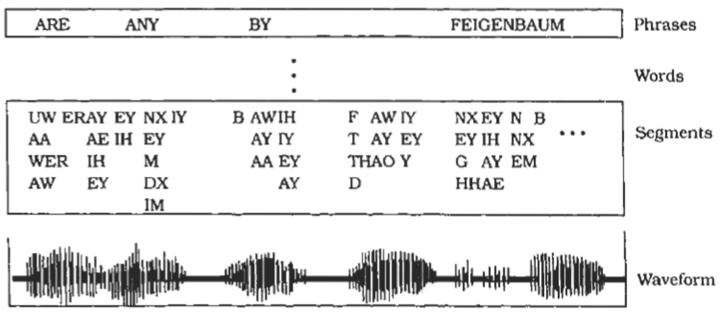
## Blackboard

## Blackboard

- Useful for problems for which no deterministic solution strategies are known.
- Several specialized subsystems assemble their knowledge to build a possibly partial or approximate solution.

## Motivation

- Speech recognition



Divide a waveform into phones, check syntax, combine segments, "guess" correct words based on context and statistical information, etc.

## Blackboard

- No consistent algorithm that combines all the necessary procedures for recognizing speech
- Ambiguities of spoken language, noisy data, and the individual peculiarities of speakers such as vocabulary, pronunciation, and syntax.

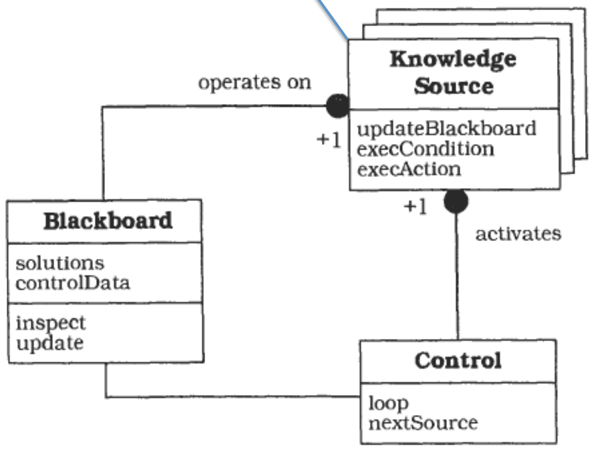
## Blackboard

- Context: *An immature domain in which no closed approach to a solution is known or feasible.*
- The problem spans several fields of expertise when decomposed into subproblems
- The solutions to the partial problems require different representations and paradigms.
- Uncertain or approximate knowledge

## Blackboard

- Similar to human experts sitting in front of a real blackboard and working together to solve a problem.
- Each expert separately evaluates the current state of the solution, and may go up to the blackboard at any time and add, change or delete information.
- Humans usually decide themselves who has the next access to the blackboard.
  - moderator

Knowledge sources are separate, independent subsystems that solve specific aspects of the overall problem

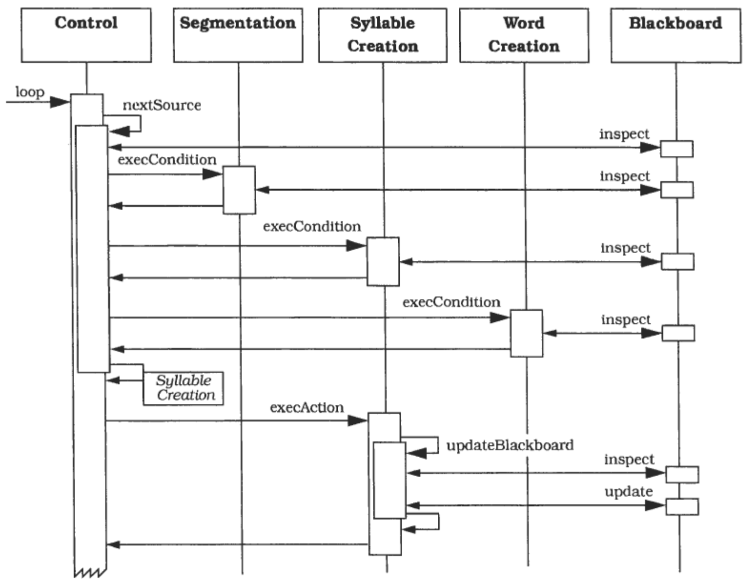


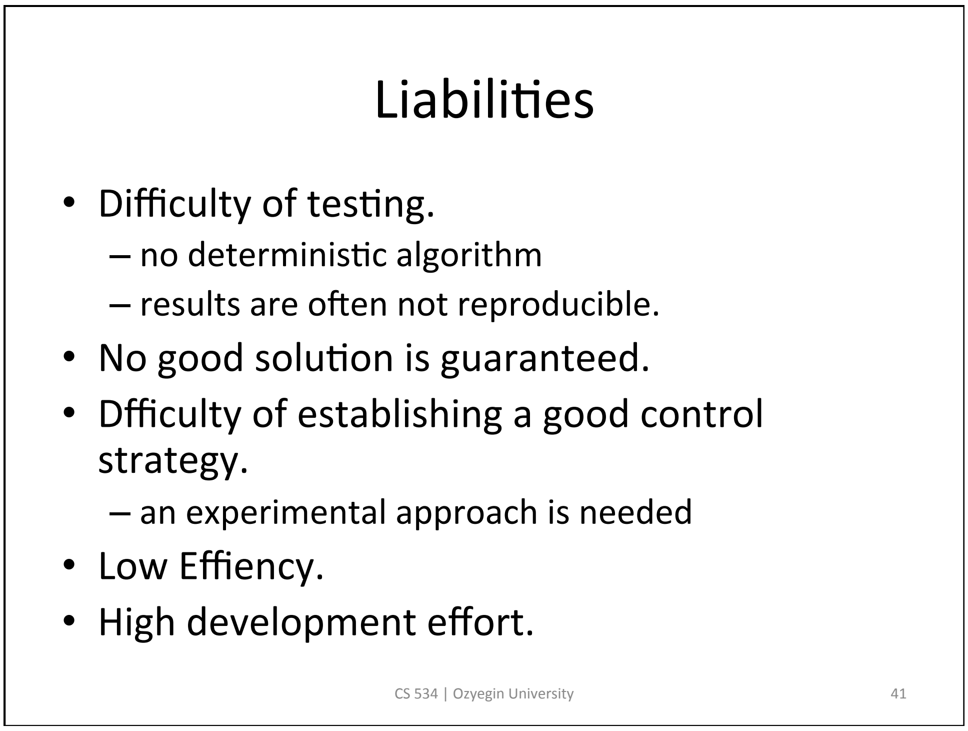
## Knowledge Source

- Each knowledge source is responsible for knowing the conditions under which it can contribute to a solution.
  - condition part: evaluates the current state of the solution process, as written on the blackboard, to determine if it can make a contribution.
  - action-part: produces a result that may cause a change to the blackboard's contents.

## Control

- The control component runs a loop that monitors the changes on the blackboard and decides what action to take next.
- It schedules knowledge source evaluations and activations according to a knowledge application strategy.
- The basis for this strategy is the data on the blackboard.





## Liabilities

- Difficulty of testing.
  - no deterministic algorithm
  - results are often not reproducible.
- No good solution is guaranteed.
- Dfficulty of establishing a good control strategy.
  - an experimental approach is needed
- Low Effiency.
- High development effort.