

---

# Repository Style

---

Jing Dong

CS6362 Software Architecture and Design

1

---

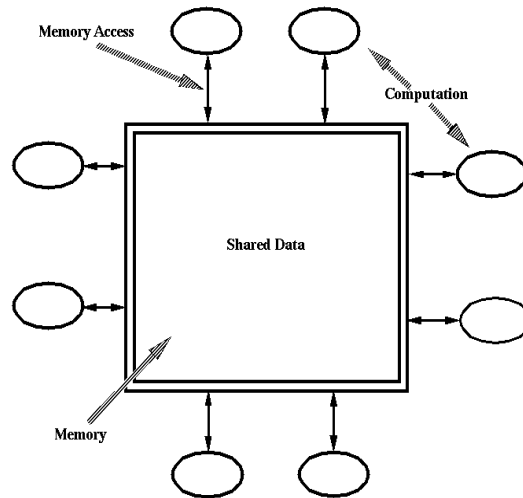
# Repository Style

- Suitable for applications in which the central issue is establishing, augmenting, and maintaining a complex central body of information.
- Typically the information must be manipulated in a variety of ways. Often long-term persistence is required.
- Components:
  - A central data structure representing the correct state of the system.
  - A collection of independent components that operate on the central data structure.
- Connectors:
  - Typically procedure calls or direct memory accesses.

---

2

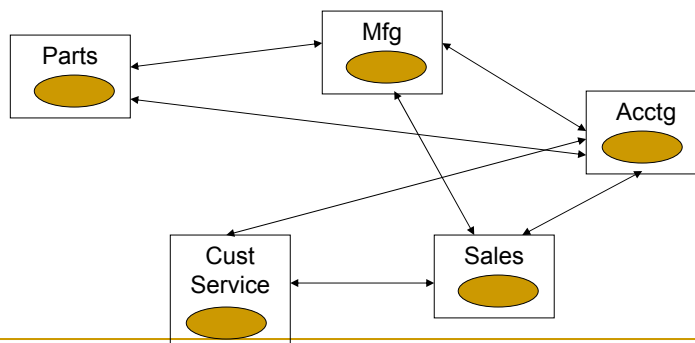
## Repository Style



3

## What Is Not Repository?

- A collection of independent components, each with its own data store, communicate with each other.



4

## Issues of Repository

- Synchronization
- Configuration and schema management
- Atomicity
- Consistency
- Persistence
- Performance
- Security
- Availability

5

## Repository Lineage

- Earliest repositories appear in batch sequential systems
  - mainframes, drums, magnetic tapes, disc drives
  - resources manually managed
- Pressure for on-line access to data
  - requirement to make access to data easy and instant
  - provided the need for the mass uses computer technology
  - help to drive the shift from batch-sequential to interactive processing
- Today
  - shared information systems appear everywhere from the smallest business, to the most advanced scientific applications
  - many applications provide access mechanisms to shared data
  - the Web has become a giant distributed repository

6

## Main Categories of Repositories

- According to the type of interactions between the repository and its external components
  - (traditional) repository database
    - Input transactions activate processes to execute
  - Blackboard
    - Its current state is the main trigger for selecting processes to execute

## Repository Style Specializations

- Changes to the data structure trigger computations.
- Data structure in memory (persistent option).
- Data structure on disk.
- Concurrent computations and data accesses.

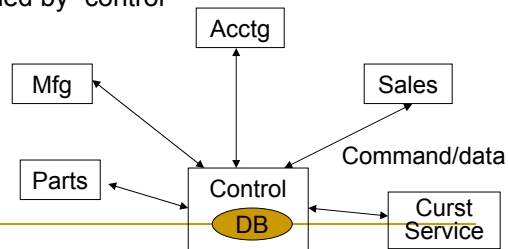
## Repository Style Examples

- Information Systems
- Software Repository, Programming Environments
- Virtual Repository

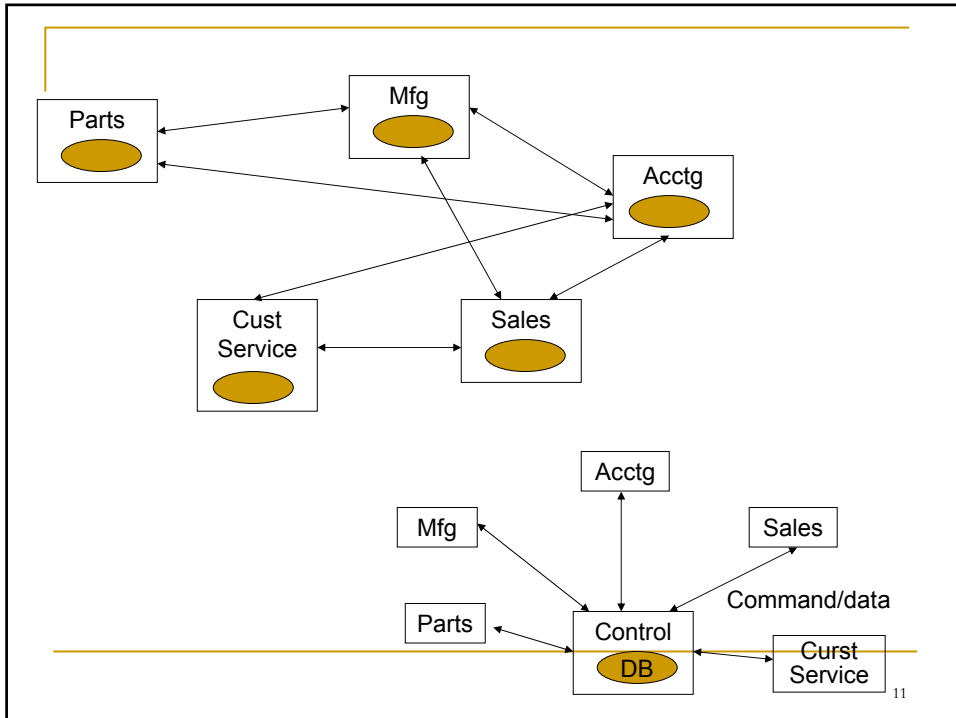
9

## Information Systems (Database)

- Two trends away from batch sequential processing
  - Interactive technology for on-line incremental updates and queries
  - Growth in the set of transactions and queries
- Architecture
  - Each transaction (in each component) does an update or a query
  - DB stores persistent data shared among different transactions
  - No fixed ordering among transactions (cf., batch sequential)
  - Concurrency control handled by “control”



10



## Software Repository

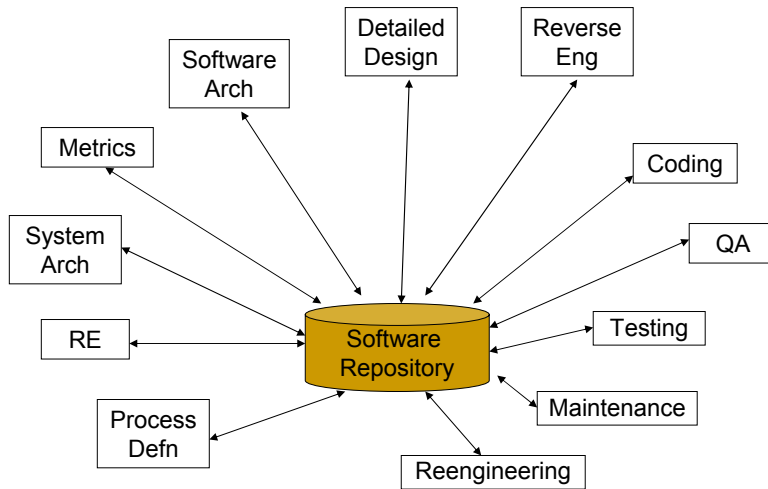
### ■ Purpose

- Allow the user to define, store, access and manage all the information about any software
- Tools access data thru open representation standard, CDIF(CASE data interchange format)

### ■ Architecture

- External (logical) schema: individual users' view
- Conceptual model: comprehensive view of entire contents
- Physical model: physical Implementation for data storage

## Software Repository



13

## Virtual Repository

- Multiple databases
  - Distributed, heterogeneous but (often) transparent
  - Due to corporate reorganization, mergers, consolidations, etc.
- Heterogeneity
  - Different schemas, names (tables, attributes), data representations

Item#	title	author
1234		
3434		
7676		

Item#	title	author
sfdfd		
fbgy		
bnbn		

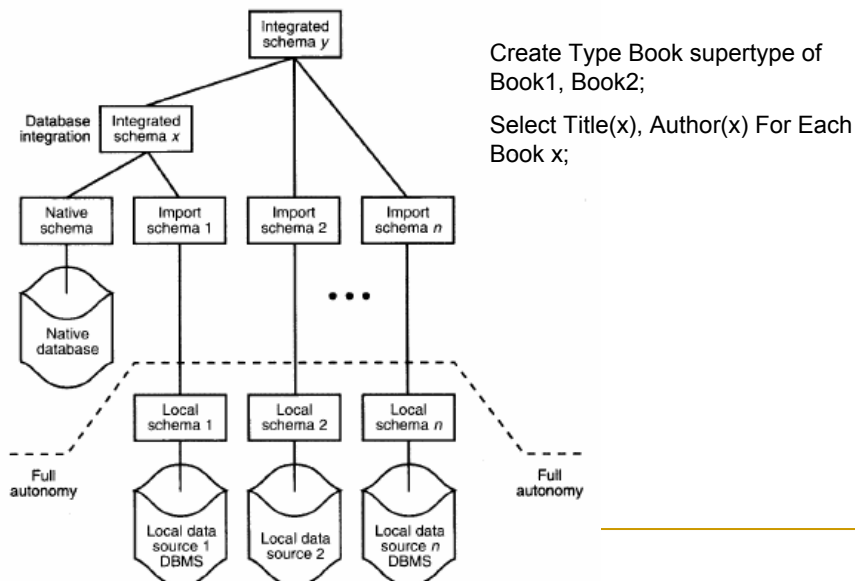
14

## Virtual Database

- Two problems
  - Representation differences
  - Communicate results across distributed systems with different data (and database schema) representations
- Federated approach
  - Combine multiple distributed schemas
  - Reconcile representational differences
  - Communicate results across distributed systems
- Virtual integrated schema
  - If imported schemas are consistent, simple merging, otherwise create a superset schema including every definition in each imported schema (metadata)
  - To users, the integrated schema acts as the virtual database (i.e., local schemas are transparent)

15

## Virtual Database



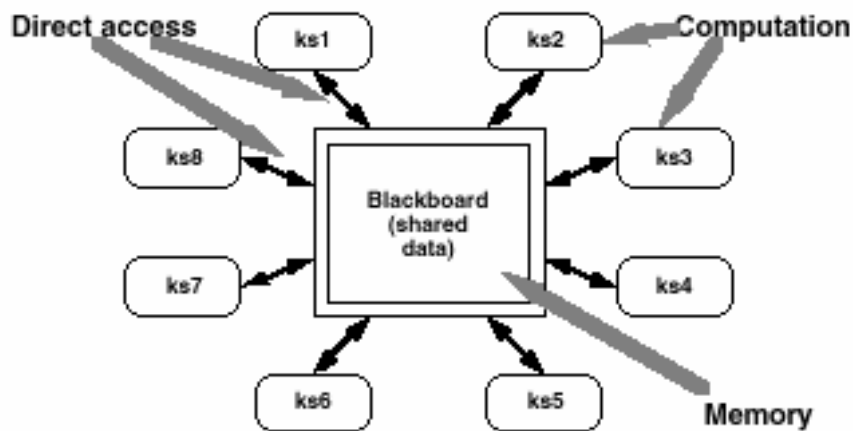
16

## Blackboard Systems

- Basic concepts
  - Variant of repository
  - The abstract model for access is “direct visibility”
  - Many human experts watch each other solve a problem at a real blackboard
- Three main components
  - The knowledge sources: separate, independent parcels of application-dependent knowledge; interaction takes place solely through the blackboard
  - The blackboard: problem-solving state data, organized into an application dependent hierarch. Knowledge sources make changes to the blackboard that incrementally lead to a solution to the problem
  - Control: driven entirely by state of blackboard. Knowledge sources respond opportunistically when changes in the blackboard make them applicable.

17

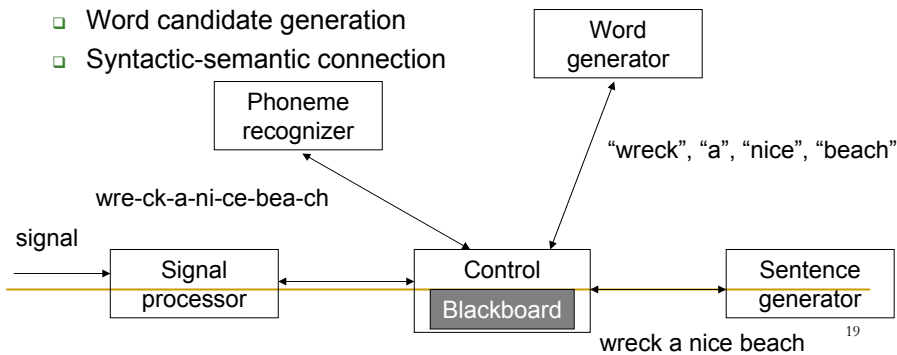
## Blackboard Systems



18

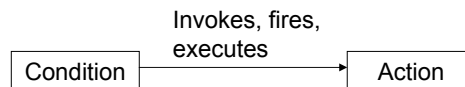
## Traditional Applications

- AI systems
  - Signal processing (speech and pattern recognition)
  - Shared data with loosely coupled agents
- Example “wreck a nice beach”
  - Signal segmentation for speech understanding
  - Phoneme recognition
  - Word candidate generation
  - Syntactic-semantic connection



## Knowledge Sources

- Knowledge can be represented in a number of ways
  - simple function
  - collection of complex and changing logical relationships (rules)
  - whatever representation is used, knowledge reflects an action (i.e. change to the blackboard) under appropriate circumstances
- Knowledge sources have two major sub-components



✓ specifies when KS has something to contribute  
 ✓ when condition is met, invokes appropriate action

✓ includes modification or placement of new facts on blackboard  
 ✓ new conditions may be embedded in actions (nested conditions)

## The Blackboard

- The blackboard is the source of all data on which a knowledge source will operate and is the destination for all conclusions from a knowledge source
- Two types of knowledge - static and dynamic
  - static knowledge - Unchanging long lived information. For example, factual data relating to initial conditions, parameter values, relationships.
  - dynamic knowledge - Knowledge that is generated during execution. For example, new facts, hypotheses, goals, requests for data, suggests. Dynamic knowledge will be updated, changed, and deleted often.

21

## Control

- The control problem - how to select the “best” action to execute next
  - how do you maximize progress on solving the problem and minimize computation
  - often difficult to quantify the value of a KS’s contribution
- Control components are also knowledge sources but focus on the problem solving process rather than (or in addition to) specific domain expertise
- Control components provide control in two ways:
  - by placing information on the blackboard that will influence the knowledge sources
  - by selecting the knowledge sources to invoke

22

## Control

- Mechanics of control components
  - event/interrupt driven
  - procedure call
  - RPC/RMI
- Control strategies
  - data driven
    - given loosely defined steps, control strategy is to call on appropriate KS to complete next step (ex. diagnostics)
    - tells a system what to do next what it can do next based on the available data
  - goal driven
    - control function calls on appropriate KS to contribute knowledge to reach goal
    - can be very difficult to codify “goals”
  - In practice a combination of both is used

23

## Control

- Problem solving models
  - backward reasoning
    - starts with goal and works backward to initial state
    - example: deterministic program verification
  - forward reasoning
    - starts with initial states and works toward goal
    - example: expression simplification by transformation
  - opportunistic reasoning
    - starts with some set of known facts
    - works whichever direction seems most productive
    - example: diagnostic system

24

## Control

- When do you stop looking for a solution?
- This is a difficult control issue, known as the “*Termination Problem*”
  - Codification of when a problem has been solved is called termination criteria
  - Often hard to codify, and as a result, many blackboard systems have under-constrained termination criteria.
    - often the ideal solution can be codified, but practical solutions can't
    - hard to codify “good enough for a given situation”
    - many goals and solutions are based on heuristics - again, often difficult to codify

25

## What Is Not Specified By Blackboard

- Structure of the knowledge within the framework
  - How is knowledge represented?
- Blackboard policies only partially specified
  - How do knowledge sources get data?
  - How do knowledge sources record the data in the blackboard?
  - How do knowledge sources use data from the blackboard?
  - How does control know what the fidelity of the solution is?
  - How does control actually control the behavior of the knowledge sources?

26

## Blackboard Systems

- Characteristics of problems that may benefit from a blackboard architecture
  - no direct algorithmic solution
    - multiple approaches to solving the problem
    - various domain expertise required to solve the problem
  - uncertainty
    - error and variability in data and solution
    - moderate to low “signal-to-noise-ratio” in data
  - best effort or approximation is good enough
    - no single discrete answer to problem, or “right” answer may vary
  - examples
    - signal processing
    - problem solving (planning, logistics, diagnostics)
    - compiler optimization

27

## Issues of Repository

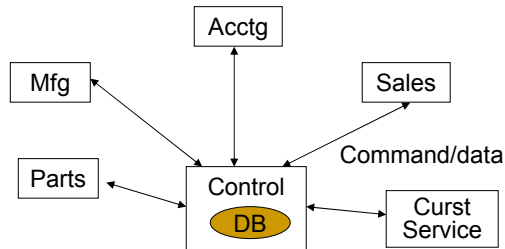
- Performance
  - Need to ensure repository can scale to load of components making requests of it
  - Ideally, do not want components waiting while repository operates on long queue of requests
  - For complex data structures, saves overhead of passing data structure from one component to another
  - If components on different machines, this overhead might be considerable
  - If components on different machines, cost of accessing repository might be high, e.g., each call to repository goes over network

28

## Issues of Repository

### ■ Performance

- Need to balance this cost vs cost of moving data between phases using pipes and filters, e.g.,
  - If "Parts" makes 1000 calls for information to repository, each call takes 1 ms, total time = 1 s. Compare to time to pass repository from "Parts" to "Mfg"



29

## Issues of Repository

### ■ Development Time

- Numerous infrastructures for repositories already exist -- e.g. commercial databases
- Need to balance savings in development time vs need to learn infrastructures

### ■ Security

- Repository potentially available to anyone who can connect to it
- Particularly dangerous in distributed setting where repository available via Internet protocol
  - E.g., don't want medical patient information available to anyone who knows IP address of medical database

### ■ Availability

- Repository becomes critical component in system
- Failure in repository causes entire system to fail

30

## Repository Style Advantages

- **Efficient** way to store large amounts of data.
- **Sharing** model is published as the repository schema.
- **Centralized management:**
  - backup
  - security
  - concurrency control

31

## Repository Style Disadvantages

- Must agree on a data model a priori.
- Difficult to distribute data.
- Data evolution is expensive.

32