Blackboard Architecture

Greg Butler

Computer Science and Software Engineering
Concordia University, Montreal, Canada

Email: gregb@cs.concordia.ca

Blackboard Architectural Pattern

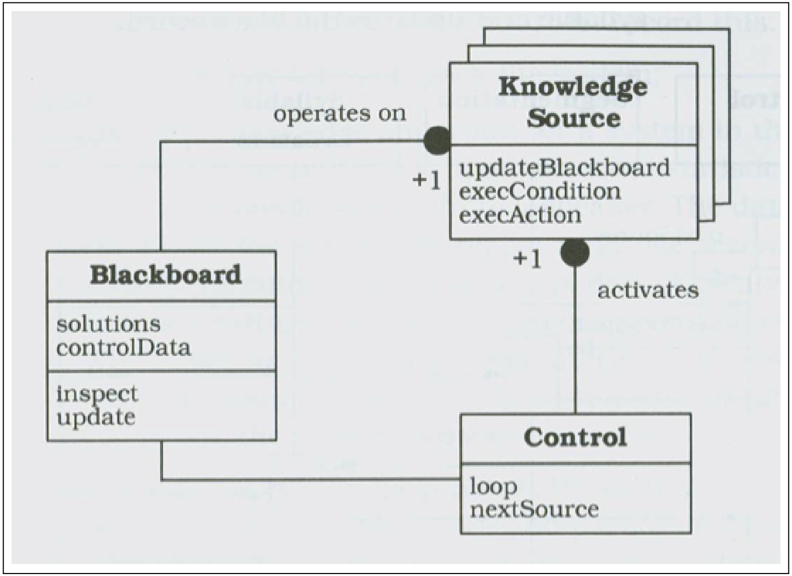
Blackboard

The Blackboard arrchitectural pattern is useful for problems for which no deterministic soluion strategies are known. In Blackboard several specialized subsystems assemble their knowledge to build a possibly partial or approximate solution.

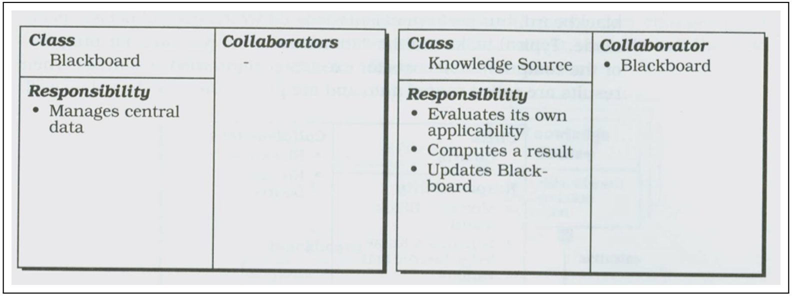
Reference

page 71–98 in Buschmann F., Meunier R., Rohnert H., Sommerlad P. & Stal M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons.

Classes in Blackboard Architecture

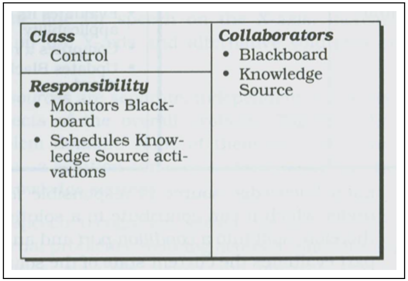


CRC Cards for Blackboard Architecture



Class Blackboard	**Collaborators**
Responsibility • Manages central data	-

Class Knowledge Source	**Collaborator**
Responsibility • Evaluates its own applicability • Computes a result • Updates Black-board	• Blackboard



Class Control	**Collaborators**
Responsibility • Monitors Black-board • Schedules Know-ledge Source acti-vations	• Blackboard • Knowledge Source

Pattern: Blackboard

Context
Open problem domain with various partial solutions

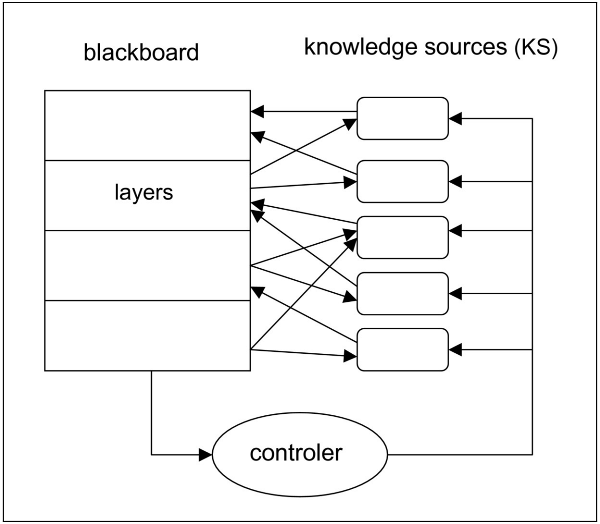
Problem
Flexible integration of partial solutions

Solution
Decompose system: 1 blackboard, many knowledge sources, 1 control

- Blackboard is common data structure
- Knowledge sources independently fill/modify the blackboard contents
- Control monitors changes and launches next knowledge sources

Tradeoffs
How well can you specify the common data structure?
How many partial solutions exist? How will this evolve?
How well can you compose an overall solution from the partial solutions?
Can you afford partial solutions that do not contribute the current task?

Layered Blackboard Architecture



Blackboard Essentials

Blackboard allows multiple processes (or agents) to communicate by reading and writing requests and information to a global data store.

Each participant *agent* has expertise in its own field, and has a kind of problem solving knowledge (*knowledge source*) that is applicable to a part of the problem, i.e., the problem cannot be solved by an individual agent only.

Agents communicate strictly through a common *blackboard* whose contents is visible to all *agents*.

When a partial problem is to be solved, candidate *agents* that can possibly handle it are listed.

A *control unit* is responsible for selecting among the candidate *agents* to assign the task to one of them.

Advantages and Disadvantages

Advantages

Suitable when there are diverse sources of input data
Suitable for physically distributed environments
Suitable for scheduling and postponement of tasks and decisions
Suitable for team problem-solving approaches as it can be used to post problem subsomponents and partial results

Disadvantages

Expensive
Difficult to determine partitioning of knowledge
Control unit can be very complex

Guidelines to Design a Blackboard

(1) Define the problem

(2) Define the solution space for the problem

(3) Divide the solution process into steps

(4) Divide the knowledge into
specialized knowledge sources with certain tasks

(5) Define the vocabulary of the blackboard

(6) Specify the control of the system

(7) Implement the knowledge sources

(1) Define the problem

Specify the domain of the problem and the general fields of knowledge necessary to find a solution

Scrutinize the input to the system. Determine any special properties of the input such as noise content or variations on a theme — that is, does the input contain regular patterns that change slowly over time?

Define the output of the system. Specify the requirements for correctness and fail-safe behaviour.

Do you need to determine *credibility* of a solution?

Detail how the user interacts with the system.

(2) Define the solution space for the problem

Distinguish intermediate solutions from top-level solutions

A top-level solution is at the highest abstraction level.

Distinguish partial solutions from complete solutions

A complete solution solves the whole problem.

Note

A complete solution can belong to an intermediate level.

A partial solution can belong to the top-level.

(2) Define the solution space for the problem

Specify exactly what constitutes a top-level solution.

List the different abstraction levels of solutions.

Organize solutions into one or more abstraction hierarchies.

Find subdivisions of complete solutions that can be worked on independently.

(3) Divide the solution process into steps

Define how solutions are transformed into higher-level abstractions.

Describe how to predict hypotheses at the same abstraction level.

Detail how to verify predicted hypotheses by finding support for them in other levels.

Specify the kind of knowledge that can be used to exclude parts of the solution space.

(4) Divide the knowledge into specialized knowledge sources with certain tasks

Subtasks often correspond to areas of specialization

Knowledge sources must be complete

that is,

for most of the inputs, there must exist at least one possible sequence of KS activations that leads to an acceptable solution.

(5) Define the vocabulary of the blackboard

Elaborate your first definition of the solution space and its abstraction levels.

Find a representation for solutions that allows all knowledge sources to read from and contribute to the blackboard.

If necessary, provide components that translate between blackboard entry representations and the internal representation within the knowledge source.

Aim to Plug-and-Play for KS

(6) Specify the control of the system

Often most complex part of blackboard system

Often rule-based

Use Strategy design pattern

(6) Specify the control of the system

Classify changes to the blackboard

Does the change affect which KS is applicable, or not?

Relate classes of changes to applicable KS

Focus of control

"*Focus*" could be a set of partial results to work on next

"*Focus*" could be a set of KS preferred for next execution

Use a queue of KS awaiting execution

(6) Some control strategies

Prioritize applicable KS based on ...
    condition-part of rules
    probability of making progress
    cost of applying KS

Forward-chaining or backward chaining ...
where preference is always given to low-level or high-level hypotheses, respectively.

Prefer hypotheses that cover a large part of the problem.

"Island driving" where an *"island of certainty"* is built into a complete top-level solution.

(7) Implement the knowledge sources

Identify condition-part and action-part of KS to match with Control.

Maintain independence and exchangeability of KS
by not making any assumptions about other KS or about Control