Blackboard Architecture

Team

Elgin Lee Wei Sheng | 20753953
Lee Kian Lon | 20755364
Harold Lim Jie Yu | 20755365
Yeo Guo Kuan Norman | 20755357

In general, what is Blackboard architecture?

- The architecture is a task solving strategy using independent modules called knowledge sources to solve a common problem.
- The architecture takes its name from how people collaborate around a blackboard, every individual would sit around the board and a problem would be written on the board. When the sub-problem can be solved by a person, the person would go to the blackboard and add the partial solution which he knows. This is repeated till a collective solution is found.
- The whole process is opportunistic, the order of which knowledge sources are activated are not pre-determined. It is purely determined at every control cycle depending on the current state of the Blackboard.
- Knowledge sources do not communicate with one another and exclusively collaborate through the Blackboard.

Its own vocabulary for its components and connectors?

Components:

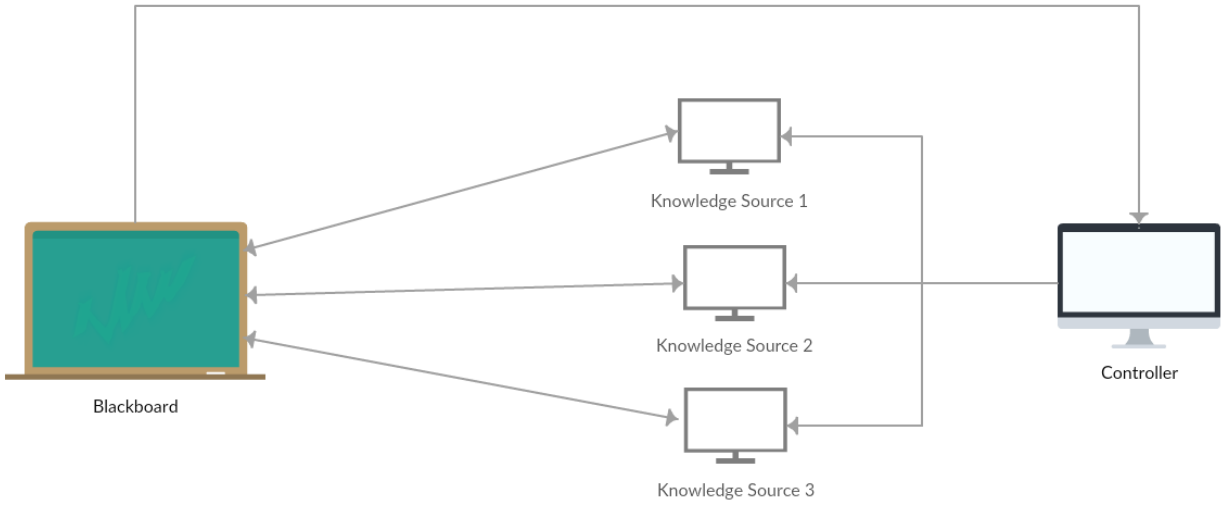
- **Blackboard**
  A Blackboard is a global accessible database which stores inputs, partial solutions and partial results. The Blackboard represents the current system state.
- **Knowledge Source**
  A knowledge source is an independent module which contain specialized knowledge needed for solving a problem. Each knowledge source has a set of triggering conditions, when satisfied, the knowledge source is enabled and executed to help progress the state of the problem on the Blackboard by solving a sub-problem which they can solve.
- **Controller**
  There might be contention and conflicting knowledge sources trying to access and update the Blackboard. This is where the controller is involved, it must choose the most suitable knowledge source for immediate execution. As the state of the Blackboard changes and new partial solutions are reached, the controller will schedule new knowledge sources to modify the Blackboard.

Connectors:

- Varies widely based on implementation. The connector to the Blackboard could be simple database queries if the Blackboard was just a simple database. The connector could also be a memory reference if all the knowledge sources are on the same machine. The connector could be as complex as a full-blown API which requires specific calls to retrieve values and update the Blackboard.

Impose specific topological constraints? (diagram)

- Three components, multiple knowledge sources with one blackboard and controller
- Knowledge sources cannot communicate with each other directly. All communication goes through the blackboard.
- Only knowledges sources can modify blackboard.
- Controller can only read from blackboard and choose the next best knowledge source to execute depending on current state of blackboard.
- Blackboard is accessible to both controllers and knowledge sources.



Most applicable to specific kinds of problems?

- A problem is too complex and needs to be split into simpler sub-problems
- A non-deterministic problem
- A problem that has many different solutions
- Internet-of-thing, where each sensor is a knowledge source
- Distributed computing where each node has its own function in a larger system
- Heuristic problem solving

Advantages

- Efficient sharing of copious amounts of data among all the knowledge sources as all information would be stored in the central blackboard instead of being duplicated across the multiple knowledge sources.
- Strategies to complex problems do not need to be pre-planned. Each knowledge source would contribute what it can, when it can to advance the solution
- Low coupling as each knowledge sources is independent, modular and highly scalable. This increases the flexibility of applications built on the Blackboard architecture, where knowledge sources can be added or removed from the system easily.
- Knowledge sources do not only apply to a specific problem. They can be easily applied to other Blackboards to tackle other problems where knowledge source is simply reused.
- Efficiency as knowledge sources can work concurrently when they are not dependent on each other
- When a function can be performed by more than one knowledge source, the controller can choose the most suitable knowledge source for execution. If a knowledge source is unable to provide the full solution to a problem, it can post a partial solution hoping that some other knowledge source will be able to use the data to come up with a full solution.

Disadvantages

- High Complexity as controllers are extremely complex and difficult to implement. They must manage the knowledge sources (decide on which knowledge source to use, scheduling, etc.). There is no best way to utilize the knowledge sources as it differs from system to system. Hence, it takes a lot of effort to fine tune the controller.
- It is difficult to test due to the non-deterministic nature of how Blackboard architecture works, every execution may not necessarily yield the same results
- The quality of the solution is not guaranteed due to its non-deterministic nature. Problems might only be partially solved.
- Not optimal if regulation of data is needed or the data frequently changes and must be updated on all clients as it causes a cascade of changes.

NFPs supported

- **Reusability**
  As knowledge sources are independent from blackboards, they are easily reusable by different blackboards and are not affected by the blackboards themselves.
- **Scalability**
  Because of the low coupling afforded by the blackboard architecture, projects are easily scalable and do well even at a very large scale.
- **Changeability**
  The blackboards do not know how or where the data is generated, so knowledge sources are highly interchangeable. Should a knowledge source be deemed inadequate or of too low quality, a better alternative can be swapped in without having to change the blackboard.
- **Robustness**
  Robustness is also provided by the low coupling of the blackboard architecture. If a knowledge source is down, a replacement source can simply be transparently swapped in. This means that the odds that a knowledge source is unavailable is extremely low.
- **Evolvability**
  Individual knowledge sources can be updated independently without affecting the overall system. This allows patches and updates to components to integrate into the system easily.

NFPs not supported

- **Dependability**
  The Blackboard is a single point of failure and if it goes down, all knowledge sources are unable to continue communication.
- **Portability**
  Because of the system's high complexity, it is hard to achieve portability. Many components will have to be reworked for use in different environments, and different system specifications mean that the developers might have to find workarounds for core features in certain environments.
- **Reliability**
  Due to the nondeterministic nature of the blackboard architecture, there is a possibility of the system only generating different solutions every run. This means that the results generated are not always reliable and not always of the same quality.

Scenario 1:
The physical blackboard would be used as the “Blackboard” and 2 people would be knowledge sources. Each knowledge source would have specialized knowledge on how to perform addition and multiplication respectively. The “complex” problem would be written on the board “ (1 + 1) * 3 ”. The knowledge sources would then solve parts that they know. This shows how the blackboard architecture works in general.

Scenario 2:
In this scenario, the blackboard would be blocked by a person to represent that the blackboard is down. The knowledge sources would be unable to communicate and access shared data. This shows how the blackboard is a single point of failure and has poor dependability.

Scenario 3:
In this scenario, two different blackboards with different mathematic problems are presented. Both with an addition function. The addition knowledge source can be applied to both. This shows how knowledge sources are easily reusable across a variety of problems.

Scenario 4:
In this scenario, the mathematics solution has a subtraction function which no knowledge source knows how to solve. Another knowledge source that has specialized knowledge on how to perform subtraction will be added. This shows scalability and ease of adding new knowledge sources. Alternatively, a knowledge source can be taught how to do the subtraction showing ease of evolvability.

Scenario 5:
In this scenario, variables would be introduced, all variables would be initialized to 2. The complex problem would be
y = 1 + a
x = 3 * a
z = a / 2
The knowledge sources would update each variable according to what they know, and this shows that copious amounts of data can be shared efficiently as the Blackboard would store all variables. This also shows that knowledge sources can update shared variables on the Blackboard.

Scenario 6:
As a side note to the previous scenario, the presenter would point out that there is no need to preplan steps. That execution order is determined as the partial solution is reached and knowledge sources would dynamically react based on the current state of the Blackboard.