Mobile Robots: A case study on architectural styles

Typical software functions.

- Acquiring and interpreting input provided by sensors.
- Controlling the motion of wheels and other movable parts.
- Planning future paths.

Examples of complications.

- Obstacles may block path.
- Sensor input may be imperfect.
- Robot may run out of power.
- Mechanical limitations may restrict accuracy of movement.
- Robot may manipulate hazardous materials.
- Unpredictable events may demand a rapid (autonomous) response.

Evaluation criteria for a given architecture

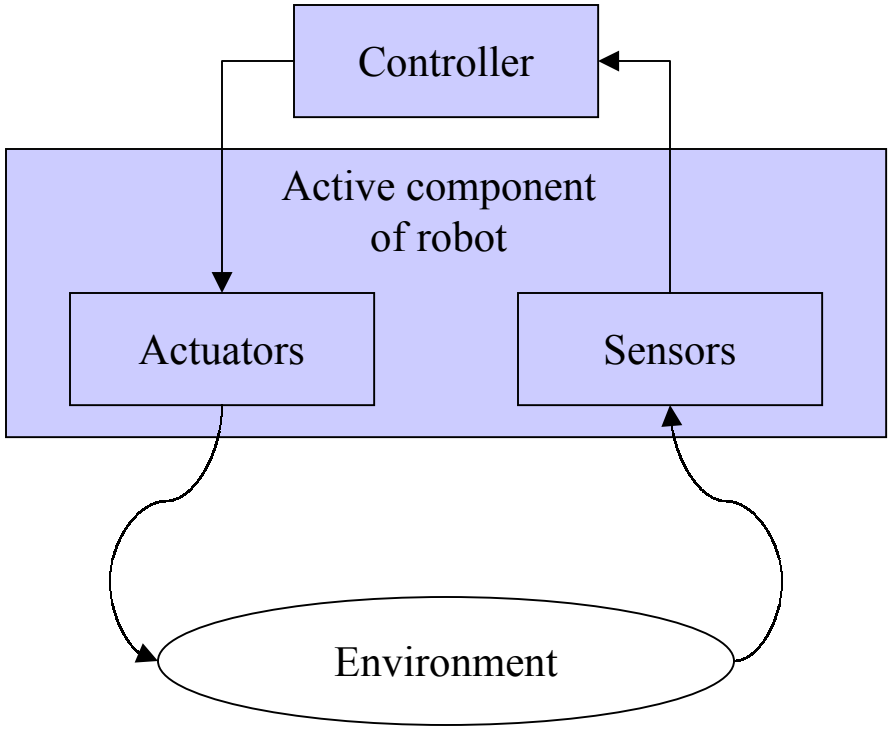
1. **Accommodation of deliberate and reactive behavior.** Robot must coordinate actions to achieve assigned objectives with the reactions imposed by the environment.

2. **Allowance for uncertainty.** Robot must function in the context of incomplete, unreliable and contradictory information.

3. **Accounting of dangers in the robot's operations and its environment.** Relating to fault tolerance, safety and performance, problems like reduced power supply, unexpectedly open doors, etc., should not lead to disaster.

4. **Flexibility.** Support for experimentation and reconfiguration.

Solution 1: Control loop



Control loop: evaluation

1. **Accommodation of deliberate and reactive behavior.** Robot must coordinate actions to achieve assigned objectives with the reactions imposed by the environment.

Simplicity. Problem in more unpredictable environments since there is basic assumption that changes in environment are continuous and require continuous reactions. Basic changes in behavior may be needed when confronted with disparate discrete events.

Control loop: evaluation (cont'd)

2. **Allowance for uncertainty.** Robot must function in the context of incomplete, unreliable and contradictory information.

Uncertainty is resolved by reducing unknowns through iteration: a problem if more subtle steps are needed.

Control loop: evaluation (cont'd)

3. **Accounting of dangers in the robot's operations and its environment.** Relating to fault tolerance, safety and performance, problems like reduced power supply, unexpectedly open doors, etc., should not lead to disaster.

Fault tolerance and safety are enhanced by the simplicity of the architecture.

Control loop: evaluation (cont'd)

4. **Flexibility.** Support for experimentation and reconfiguration.

Major components (supervisor, sensors, motors) can be easily replaced; more refined tuning must take place inside the modules.

Control loop: overall evaluation

Appropriate for simple robotic systems that must handle only a small number of external events and whose tasks do not require complex decompositions.

Solution 2: Layered architecture

Defined by following layers:

1. Supervisor
2. Global planning
3. Control
4. Navigation
5. Real-world modeling
6. Sensor integration
7. Sensor interpretation
8. Robot control

Layered architecture: evaluation

1. **Accommodation of deliberate and reactive behavior.** Robot must coordinate actions to achieve assigned objectives with the reactions imposed by the environment.

Nicely organizes components needed to coordinate operation. However, does not fit the actual data and control-flow patterns. Information exchange is less straightforward: exceptional events may force direct communication between levels 2 and 8, for example. Also, there are really two abstraction hierarchies that actually exist: a data hierarchy and a control hierarchy.

Layered architecture: evaluation (cont'd)

2. **Allowance for uncertainty.** Robot must function in the context of incomplete, unreliable and contradictory information.

Existence of abstraction layers nicely addresses need for managing uncertainty: things get more certain the higher one gets.

Layered architecture: evaluation (cont'd)

3. **Accounting of dangers in the robot's operations and its environment.** Relating to fault tolerance, safety and performance, problems like reduced power supply, unexpectedly open doors, etc., should not lead to disaster.

Fault tolerance and passive safety are also served by abstraction mechanism. Performance and active safety issues may force the communication pattern to be short-circuited.

Layered architecture: evaluation (cont'd)

4. **Flexibility.** Support for experimentation and reconfiguration.

Interlayer dependencies are an obstacle to easy replacement and addition of components.

Layered architecture: overall evaluation

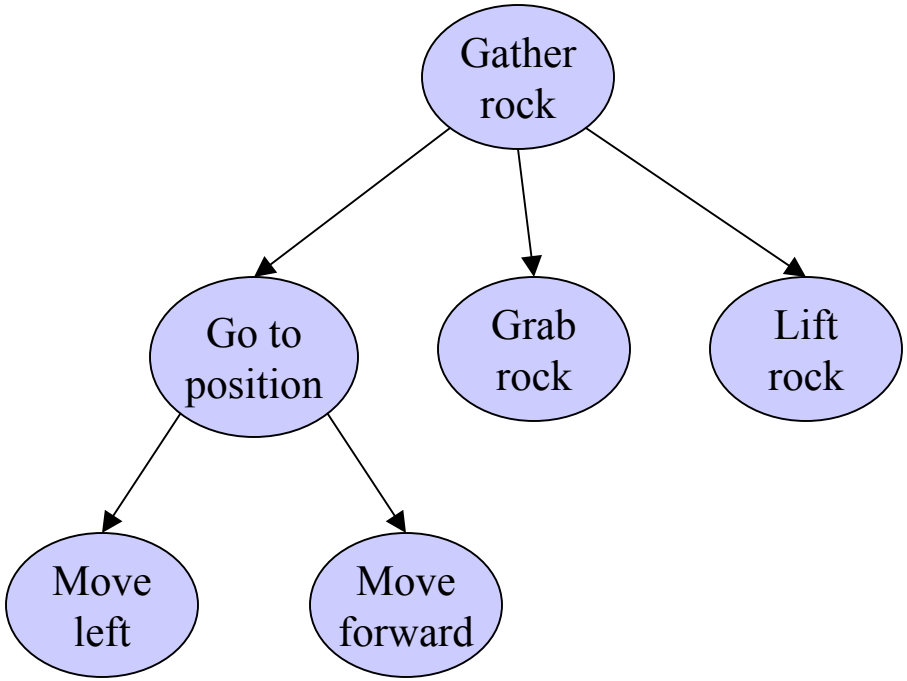
Nice, high-level view of robot control, but breaks down as an implementation view since the communication patterns are not likely to follow the orderly scheme implied by the architecture.

Solution 3: Implicit invocation with TCA

TCA is short for Task-Control Architecture and its use represents another approach based on an implicit invocation style of architecture. TCA organizes components as hierarchies of task called task trees. Temporal constraints can also be associated with tasks occurring in a task tree. Tasks communicate by multicasting messages via a message server, which redirects those messages to (one or more) other tasks registered to handle them. Three specific roles of tasks have special support.

- Exception handlers: can change task trees, for example.
- Wiretappers: can see messages going to other tasks.
- Monitors: are invoked when message data satisfies predefined conditions.

Implicit invocation with TCA: Example task tree



Implicit invocation with TCA: evaluation

1. **Accommodation of deliberate and reactive behavior.** Robot must coordinate actions to achieve assigned objectives with the reactions imposed by the environment.

Task trees and task roles permit a clear-cut separation of action and reaction. Explicit support for concurrent agents.

Implicit invocation with TCA: evaluation (cont'd)

2. **Allowance for uncertainty.** Robot must function in the context of incomplete, unreliable and contradictory information.

How TCA addresses uncertainty is less clear. Often tentative task trees are created when an exceptional event happens.

Implicit invocation with TCA: evaluation (cont'd)

3. **Accounting of dangers in the robot's operations and its environment.** Relating to fault tolerance, safety and performance, problems like reduced power supply, unexpectedly open doors, etc., should not lead to disaster.

TCA exception, wiretapping and monitoring features address needs for performance, safety and fault tolerance.

Implicit invocation with TCA: evaluation (cont'd)

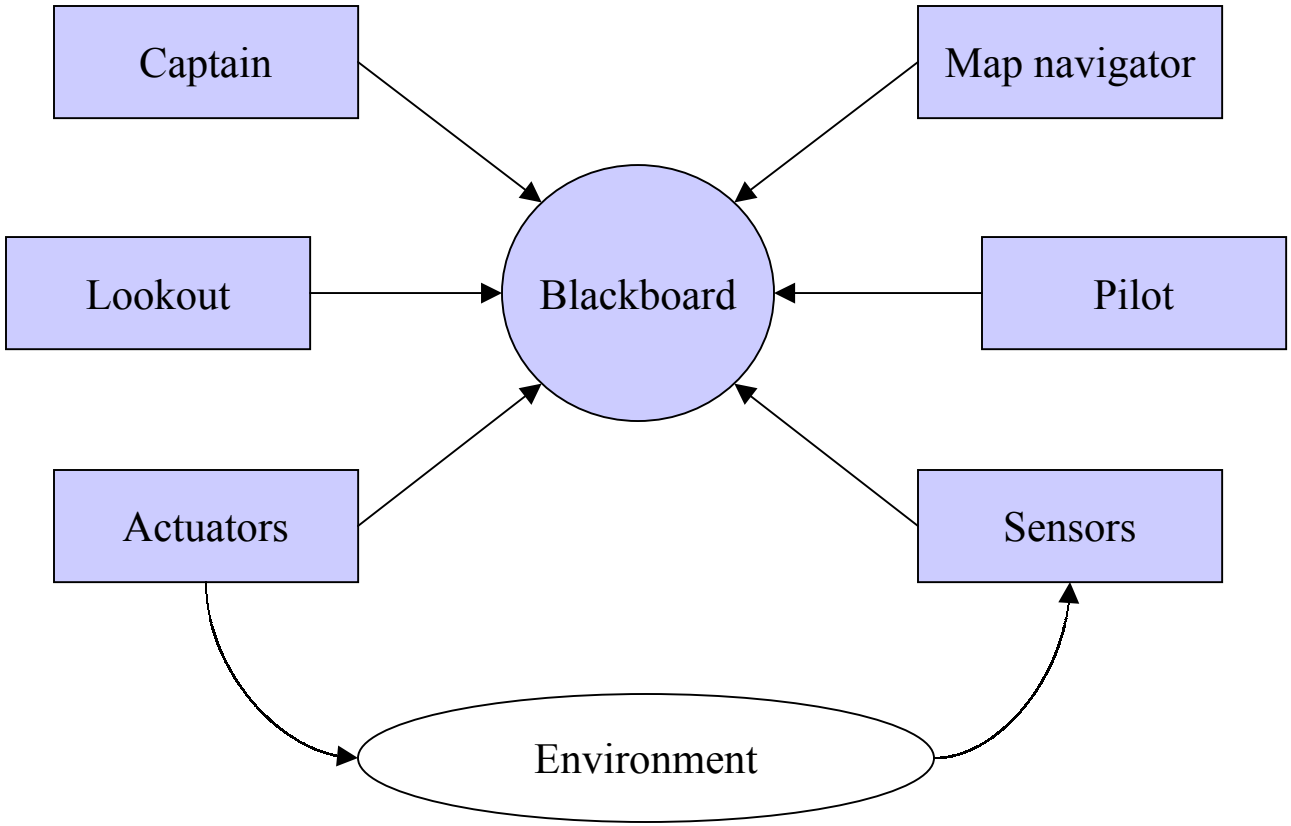
4. **Flexibility.** Support for experimentation and reconfiguration.

Implicit invocation style supports incremental development and replacement of components.

Implicit invocation with TCA: overall evaluation

TCA offers a comprehensive set of features for coordinating the tasks of a robot. Most appropriate for more complex robot projects.

Solution 4: Blackboard architecture



Blackboard architecture: evaluation

1. **Accommodation of deliberate and reactive behavior.** Robot must coordinate actions to achieve assigned objectives with the reactions imposed by the environment.

Components communicate via the shared repository: modules indicate their interest in certain types of information; the database returns relevant data either immediately or when some other module inserts the relevant data into the database.

Blackboard architecture: evaluation (cont'd)

2. **Allowance for uncertainty.** Robot must function in the context of incomplete, unreliable and contradictory information.

The blackboard helps to resolve conflicts or uncertainty in the robot's world view.

Blackboard architecture: evaluation (cont'd)

3. **Accounting of dangers in the robot's operations and its environment.** Relating to fault tolerance, safety and performance, problems like reduced power supply, unexpectedly open doors, etc., should not lead to disaster.

The TCA exception mechanisms, wiretapping and monitoring roles can be implemented by defining separate modules that watch the database for exceptional circumstances.

Blackboard architecture: evaluation (cont'd)

4. **Flexibility.** Support for experimentation and reconfiguration.

Supports concurrency and maintenance by decoupling senders from receivers.

Blackboard architecture: overall evaluation

Capable of modeling the cooperation of tasks in a flexible manner thanks to an implicit invocation mechanism based on the contents of the database.