Robotic Paradigms

Dr. Divya Udayan J, Ph.D.(Konkuk University, S.Korea)

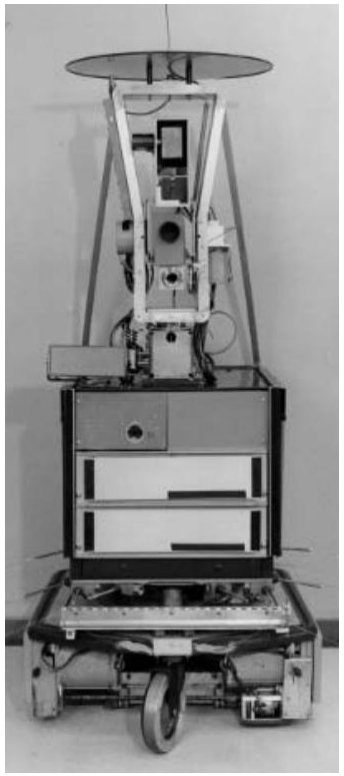
Department of CSE
Amrita School of Engineering, Amritapuri Campus,
Amrita Vishwa Vidyapeetham
Email: divyaudayanj@am.amrita.edu
Mobile: 9550797705

Hierarchical Paradigm

- The Hierarchical Paradigm is historically the oldest method of organizing intelligence in mainstream robotics. It dominated robot implementations from 1967, with the inception of the first AI robot, Shakey.

- SENSE, PLAN, ACT primitives and by its sensing representation.

- Robotic paradigm is defined by the relationship between the three primitives (SENSE, PLAN, ACT) and by the way sensory data is processed and distributed through the system.
- The Hierarchical Paradigm is sequential and orderly .



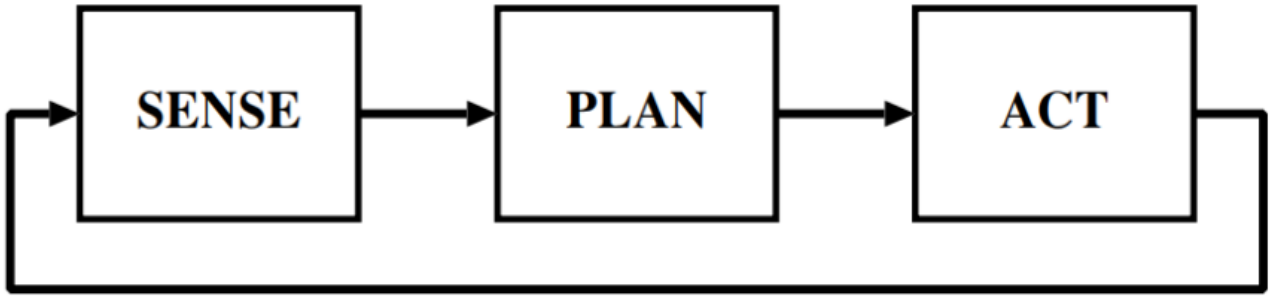


Figure S,P,A organization of Hierarchical Paradigm.

ROBOT PRIMITIVES	INPUT	OUTPUT
SENSE	Sensor data →	Sensed information
PLAN	Information (sensed and/or cognitive) →	Directives
ACT	directives →	Actuator commands

Figure Alternative description of how the 3 primitives interact in the Hierarchical Paradigm.

- Videos

Hierarchical Paradigm

- Hierarchical Paradigm is monolithic:
  all the sensor observations are fused into one global data structure, which the planner accesses. The global data structure is generally referred to as a *world model*.

- In the Hierarchical Paradigm, the world model typically contains

1. an *a priori* (previously acquired) representation of the environment the robot is operating in (e.g., a map of the building),
2. sensing information (e.g., "I am in a hallway, based on where I've traveled, I must be in the northwest hallway"), plus
3. any additional cognitive knowledge that might be needed to accomplish a task (e.g., all packages received in the mail need to be delivered to Room 118).

Hierarchical Paradigm

Strips

- Shakey, the first AI mobile robot, needed a generalized algorithm for planning how to accomplish goals.

(An *algorithm* is a procedure that is correct and terminates.

The method finally selected was a variant of the *General Problem Solver* method, called *Strips*.

- Strips uses an approach called *means-ends analysis*,

where if the robot can't accomplish the task or reach the goal in one "movement", it picks an action which will reduce the difference between what state it was in now (e.g., where it was) versus the goal state (e.g., where it wanted to be).

- This is inspired by cognitive behavior in humans; if you can't see how to solve a problem, you try to solve a portion of the problem to see if it gets you closer to the complete solution.

initial state:	Tampa, Florida (0,0)
goal state:	Stanford, California (1000,2828)
difference:	3,000

- Difference Table

difference	operator
d≥200	fly
100<d < 200	ride_train
d ≤ 100	drive
d< 1	walk

difference	operator	preconditions
d≤200	fly	
100<d<200	ride_train	
d≤100	drive_rental	at airport
	drive_personal	at home
d<1	walk	

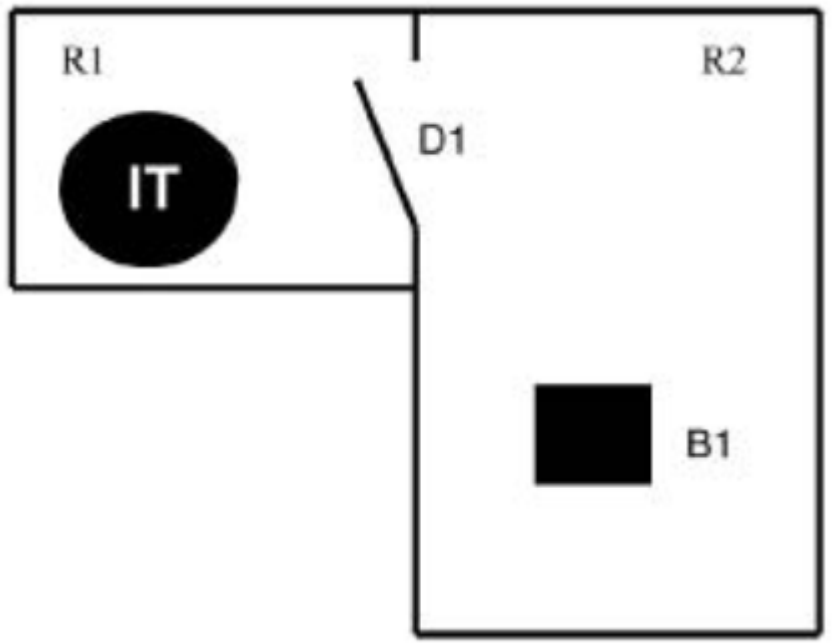
difference	operator	pre-conditions	add-list	delete-list
$d \leq 200$	fly		at Y at airport	at X
$100 < d < 200$	train		at Y at station	at X
$d \leq 100$	drive_rental	at airport		
	drive_personal	at home		
$d < 1$	walk			

`INROOM(x, r)`	where `x` is an object of type `movable_object`, `r` is type `room`
`NEXTTO(x, t)`	where `x` is a `movable_object`, `t` is type `door` or `movable_object`
`STATUS(d, s)`	where `d` is type `door`, `s` is an enumerated type: `OPEN` or `CLOSED`
`CONNECTS(d, rx, ry)`	where `d` is type `door`, `rx, ry` are the `room`

The robot's knowledge represented as predicates

```
initial state:
INROOM(IT, R1)
INROOM(B1,R2)
CONNECTS(D1, R1, R2)
CONNECTS(D1, R2, R1)
STATUS(D1,OPEN)
```

Representation of world model for the initial state of the world



An example for Strips of two rooms joined by an open door

Strips summary

Strips implementations requires the designer to set up a:

- ❖ world model representation
- ❖ difference table with operators, preconditions, add, and delete lists
- ❖ difference evaluator

The steps in executing Strips are:

1. Compute the difference between the goal state and the initial state using the difference evaluation function. If there is no difference, terminate.

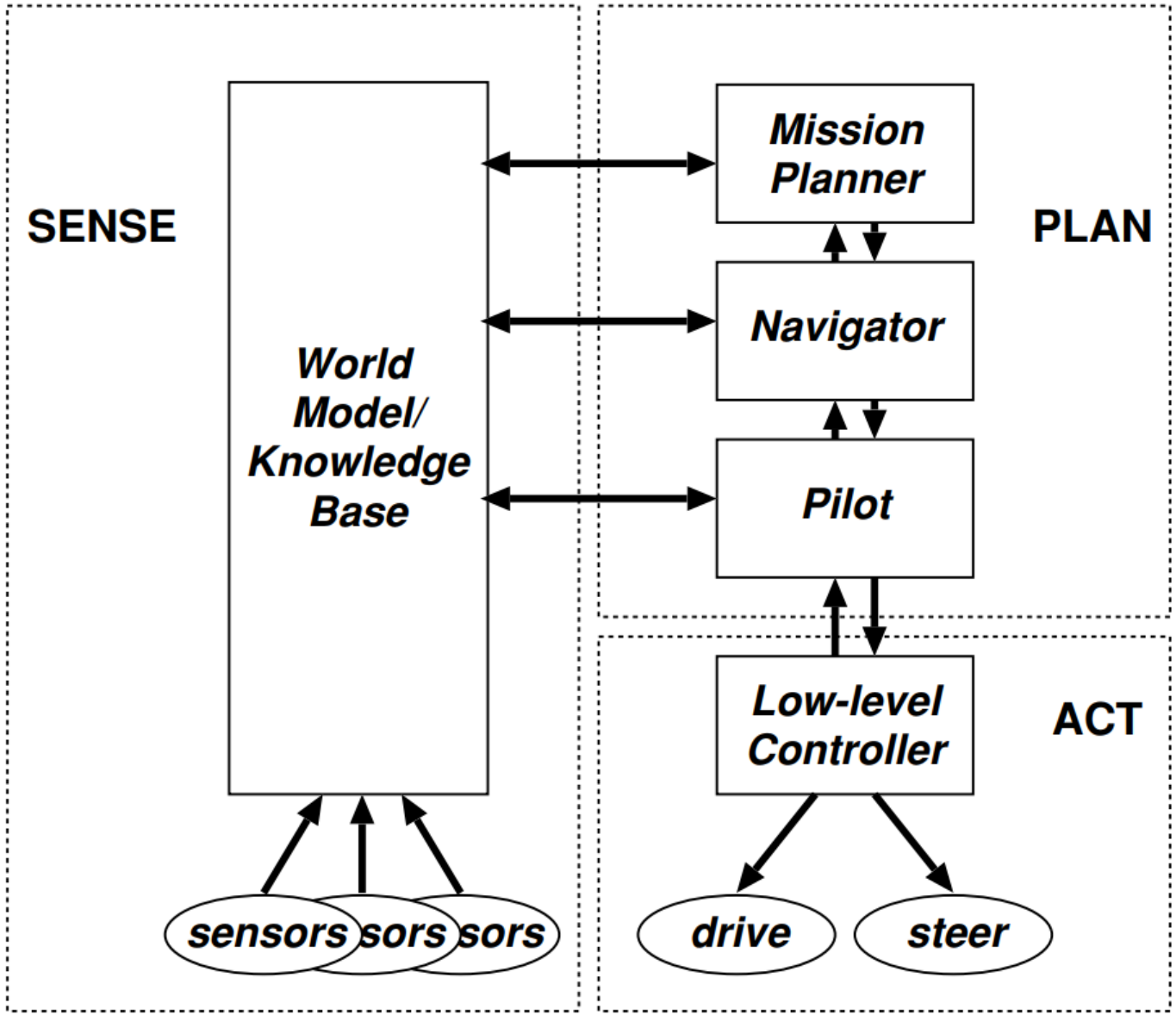
2. If there is a difference, reduce the difference by selecting the first operator from the difference table whose add-list has a predicate which negates the difference.

3. Next, examine the preconditions to see if a set of bindings for the variables can be obtained which are all true. If not, take the first false precondition, make it the new goal and store the original goal by pushing it on a stack. Recursively reduce that difference by repeating step 2 and 3.

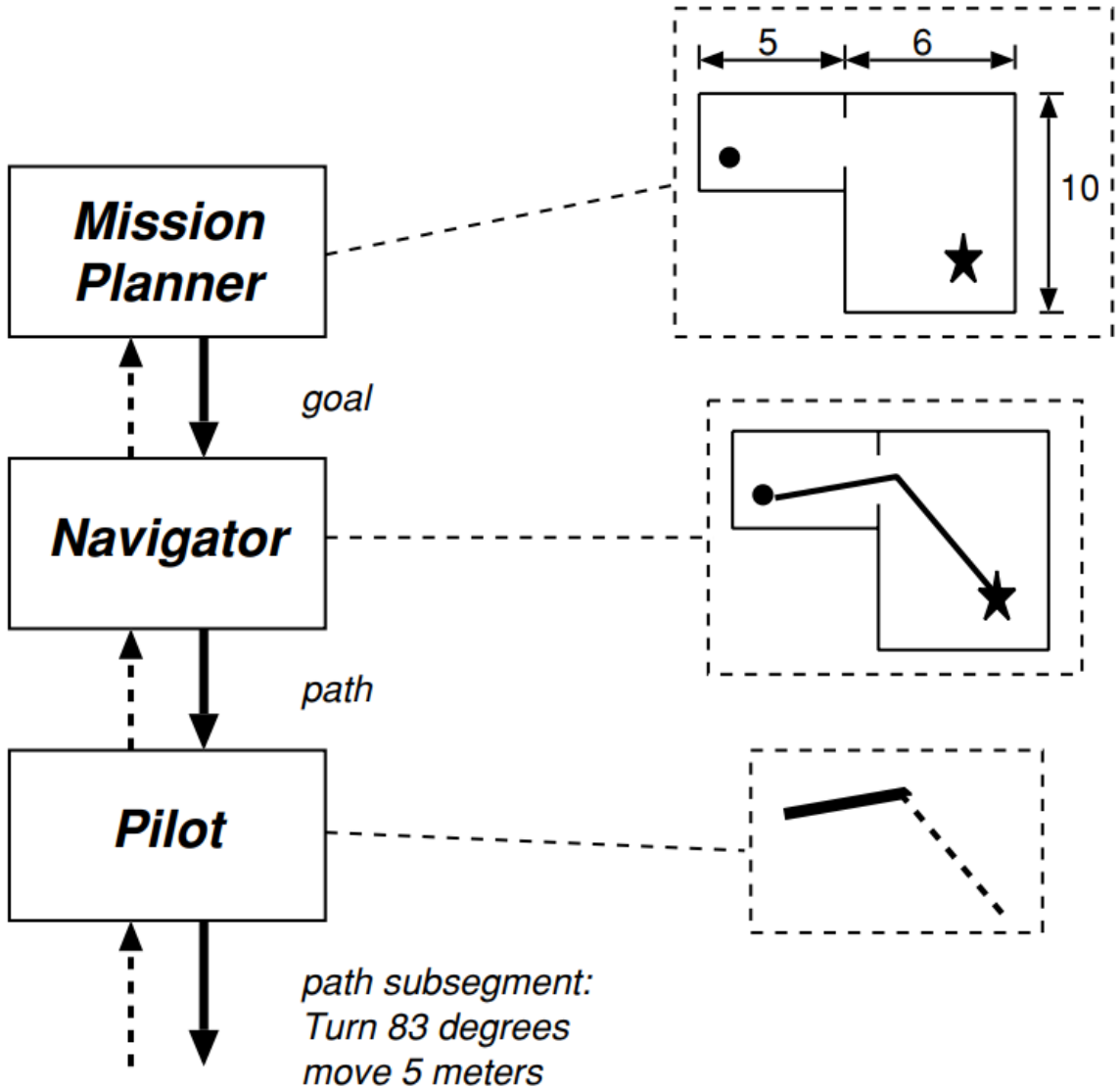
4. When all preconditions for an operator match, push the operator onto the plan stack and update a copy of the world model. Then return to the operator with the failed precondition so it can apply its operator or recurse on another failed precondition.

Representative architectures

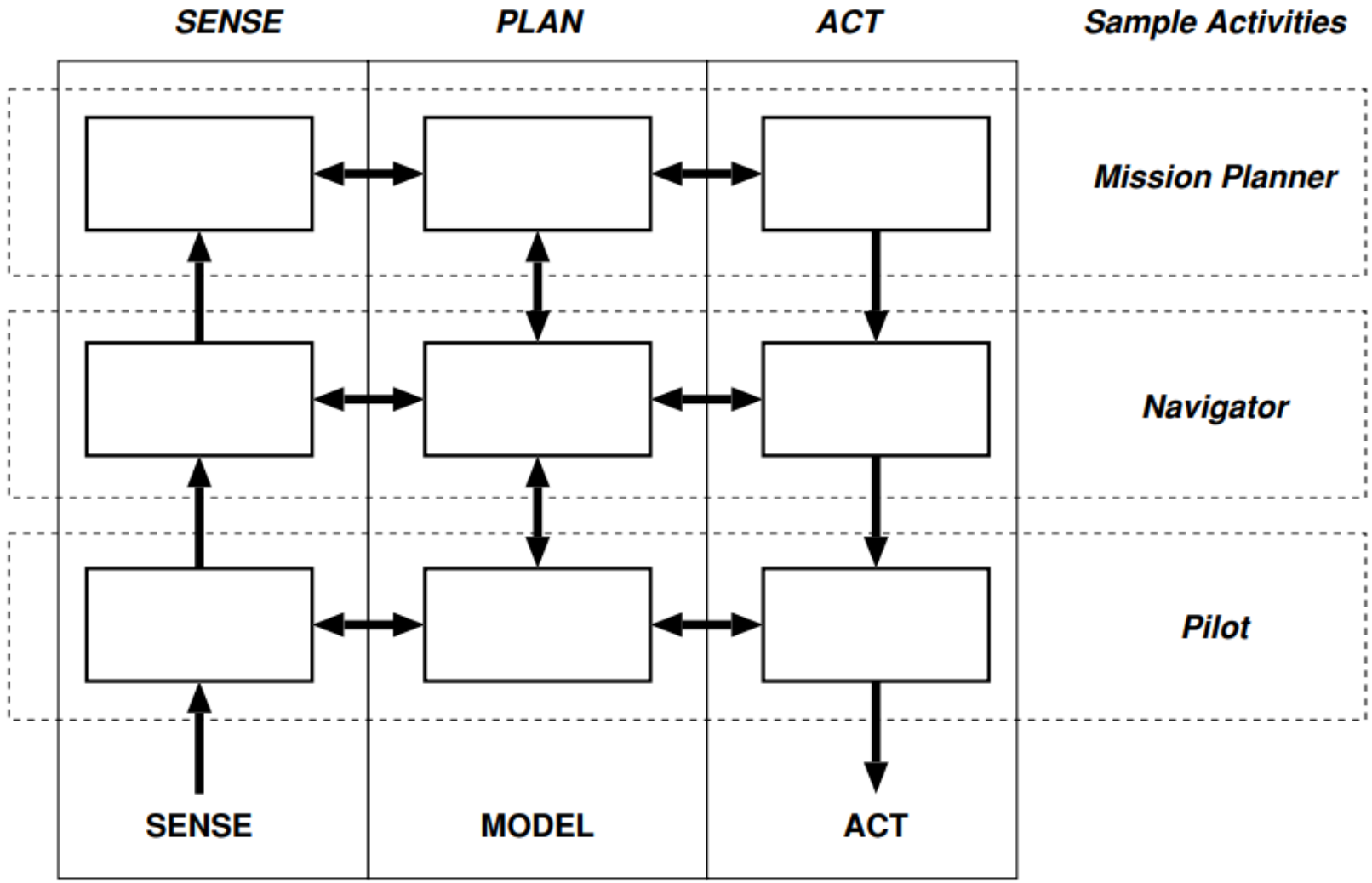
- An architecture is a method of implementing a paradigm, of embodying the principles in some concrete way.
- Ideally, an architecture is generic; like a good object-oriented program design, it should have many reusable pieces for other robot platforms and tasks.
- Possibly the two best known architectures of the Hierarchical period are the **Nested Hierarchical Controller (NHC)** developed by Meystel and the **NIST Realtime Control System** (RCS) originally developed by Albus, with its teleoperation version for JPL called NASREM.



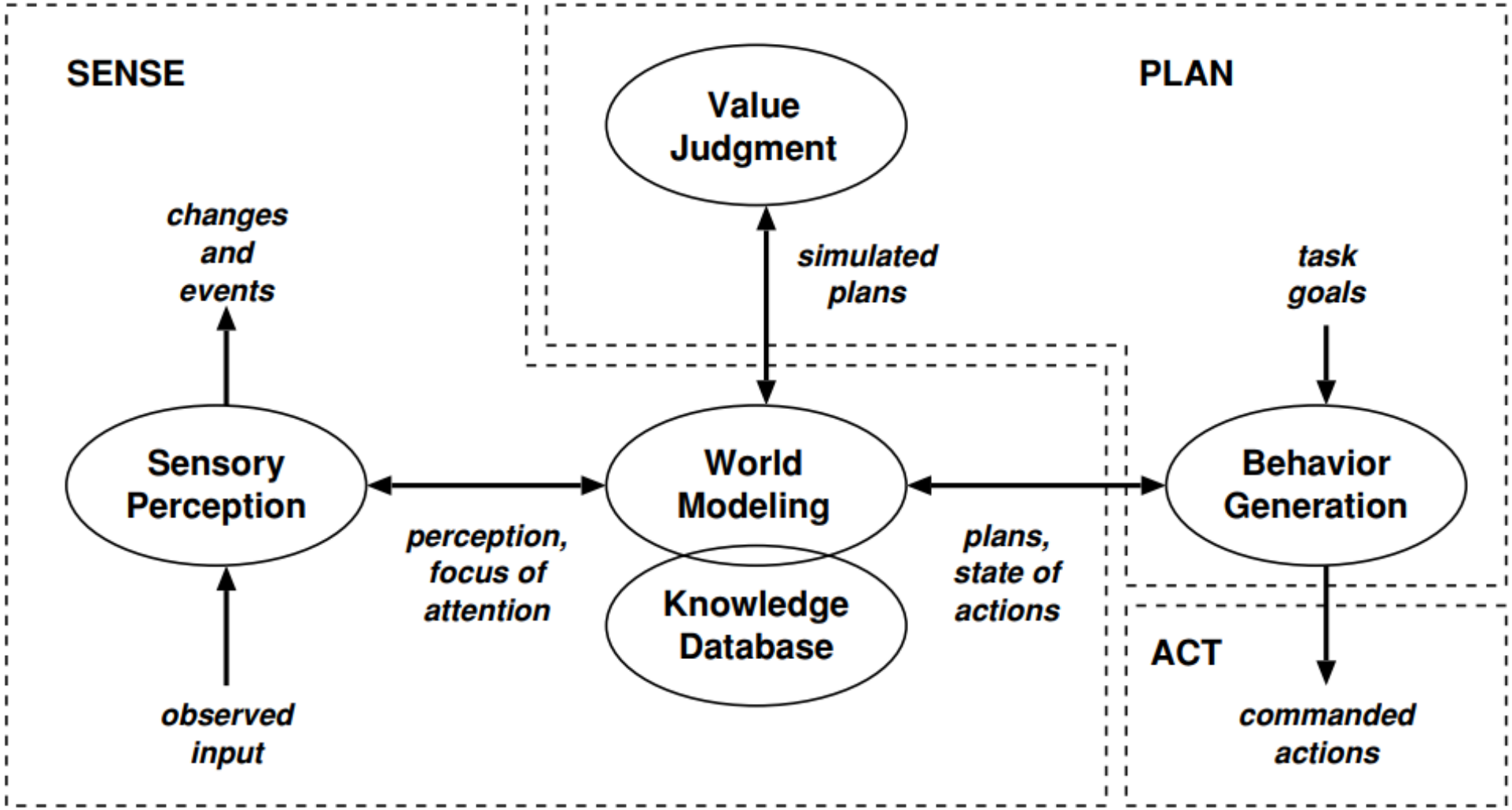
Nested Hierarchical Controller



Examination of planning components in the NHC architecture.



Layout of RCS: HIERARCHICAL layering of sense-model-act



Layout of RCS: Functional decomposition

Evaluation of hierarchical architectures

Four criteria for evaluating an architecture:

- Support for modularity
- Niche targetability
- Ease of portability to other domains
- Robustness

a.

b.

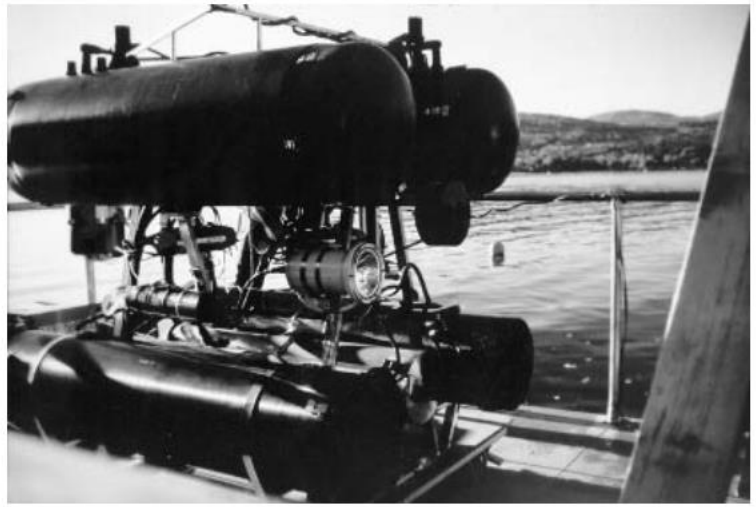


Figure Three of the diverse mobile robots that have used RCS: a.) a commercial floor cleaning robot, b.) a mining robot, and c.) a submersible or underwater robot. (Photographs courtesy of the National Institute of Standards and Technology.)

Advantages and Disadvantages

- The primary **advantage** of the Hierarchical Paradigm was that it provides an ordering of the relationship between sensing, planning, and acting.
- The primary **disadvantage** was planning. Every update cycle, the robot had to update a global world model and then do some type of planning. The sensing and planning algorithms of the day were extremely slow (and many still are), so this introduced a significant bottleneck.
- The dependence on a global world model is related to the frame problem.
- Uncertainity – comes in many forms