3T Architecture
Experiences with an Architecture for Intelligent Reactive Agent

Combining Different Types of Control

• We have been studying local control laws.
  - Each law makes simple local assumptions
  - Tightly-coupled closed-loop control

• How do we combine several of these?
• How do we do high-level planning?
Brooks “Subsumption Architecture”

• Changed the course of AI Robotics.
  - The Sense-Plan-Act loop is too slow, and can never have an accurate enough model.

• Use a hierarchy of fast reactive loops.
  - Each loop capable of complete behavior.
  - Higher loops modify the behavior of lower ones. (James Albus had proposed this earlier.)
“Our architecture separates the general robot intelligence problem into three interacting layers or tiers and is thus known as 3T. The particular Implementation of 3T consists of

- A dynamically reprogrammable set of reactive skills coordinated by a skill manager [Yu et al., 1994].

- A sequencer that activates and deactivates sets of skills to create networks that change the state of the world and accomplish specific tasks. For this we use the Reactive Action Packages (RAPs) system [Firby, 1989].

- A deliberative planner that reasons in depth about goals resources and timing constraints. For this we use a system known as the Adversarial Planner AP [Elsaesser and Slack, 1994].”
Three-Level Architectures

- Deliberation
  - Partial Task Ordering
- Sequencing
  - Instantiated Tasks
- Reactive Skills
  - Sensor Readings
  - Actuator Commands

World / Environment
Three-Level Architectures

• Reactive skills:
  – Control laws tightly coupled with environment
  – Generalized to observers and sensory transforms

• Sequencing:
  – Select task network of currently active skills
  – Accomplish specified task using skill hierarchy

• Planning:
  – Reason about goals, resources, and timing.
  – Standard AI planning methods.
Skills

• A skill can be a local control law,
  - or it can estimate features from sensory input,
  - or it can compute signals for the sequencer.

• It makes a “simple-world” assumption.

• A skill has several components:
  - Input and output specifications
  - Computational transform from input to output
  - Initialization on system setup
  - Enable function on skill startup
  - Disable function on skill shutdown.
Skill Network for Finding and Recognizing People

skill blocks

get-face
search-color
track-color
vh-map
sonar-approach
turn-robot

recognize-face
vh-free-dir
vh-move

skill events
crop-done
id-ready
found-color
no-color
at-color
lost-color
at-goal
at-person
at-angle

to RAPs
Skill Network (partial) for Navigation task
Sequencing

To accomplish tasks that the robot must routinely perform the 3T architecture has a sequencing system

Assemble an appropriate network of skills.
- Enable and link skills needed for current task
- Disable skills no longer needed
Reactive Action Package (RAPs)

(define-rap (attach-at-site ?thesite)
  (succeed (docked ?thesite)
    (method
      (context (ferrous-hull ?thesite))
      (task-net
        (sequence
          (t1 (approach-site ?thesite))
          (t2 (magnetically-attach ?thesite))
          (wait-for (docked ?thesite)))))))

(method
  (context (not (ferrous-hull ?thesite))
    (task-net
      (sequence
        (sequence
          (t1 (approach-site ?thesite))
          (t2 (grip-attach ?thesite))
          (wait-for (docked ?thesite))))))))
Reactive Action Plackage (RAPs)

RAP: A Set of Methods for Accomplishing a Task

• Task name and arguments
• Success (termination) condition
• Multiple methods (OR)
  – Context (applicability) condition
  – Network of subtasks (AND)

The sequencer contains a library of such RAPs, each keyed to specific situations and each activating a different set of skills in order to accomplish its particular task.
Planning

“The role of reaction is to control real time behavior.

The role of sequencing is to generate well known series of real time behaviors. The sequencing tier will raise the level of abstraction of the activities with which the planner will concern itself. This simplifies the planning problem because it lets a few operators stand for large families of similar execution time actions.

The planner operates at the highest level of abstraction possible so as to make its problem space as small as possible.”
Planning Operator

(Operator grasp-bulky-object
 :purpose
 (holding ?planner ?large-thing)
 :arguments
 ((?size-of-thing
   (get-value ?large-thing 'size)))
 :preconditions
 ((top ?large-thing clear)
   (on ?large-thing ?something))
 :constraints
 ((can-lift ?arm-or-robot1

   (* 0.5 ?size-of-thing))
 (can-lift ?arm-or-robot2

   (* 0.5 ?size-of-thing))
:plot
  (simultaneous
    (grip ?arm-or-robot ?large-thing)
    (grip ?arm-or-robot ?large thing))
:effects
  ((holding ?planner ?large-thing)
    (top ?something clear)
    (on ?large-thing nothing)))
Prototype space station maintenance robot

Task
A list of sites to be inspected or repaired is presented to the planning tier by the ground control supervisor and a maintenance plan consisting of repairs and inspections is generated for the day. The planner then installs each plan step on the RAPs agenda along with the recommendation of the agent to use in the step (arms, cameras, etc.)
Malfunctions and Error recovery

- **Simple failures** such as failure of an arm or a grapple fixture are handled at the RAP level. Should an arm fail, the RAP level is able to substitute an alternative agent, e.g., left arm instead of the recommended right arm, and this will cause the planner to adjust agent assignments for future tasks.

- **More drastic failures** will cause the planner to abandon all tasks at a given site.

- **With enough malfunctions** the planner abandons the entire plan and directs the robot back to its docking station.

“3T deals with failure at three levels: environmental variation in the skills, variation in routine activity in the RAPs, and variation in time and resources in the planner.”
Malfunctions and Error recovery

- malfunction (subtask cannot be accomplished)
  OR
- new situation

Subtask cannot be accomplished

Skill Manager

- skill-internal adaptation compliance
- switching skills on and off
- skill-internal adaptation compliance

Skills #1
Skills #2
...
Skills #n
Allocating knowledge across the architecture

“One important research issue is how to decide whether a certain aspect of a task belongs at the skill level, the sequencer level or the planning level.”
Allocating knowledge across the architecture

**Time**

skill level has a cycle time on the order of milliseconds, sequencer level of seconds and planning level seconds to tens of seconds

- if something must run in a **tight loop** (i.e. obstacle avoidance) then it should be a **skill**

- if something runs **slowly** (i.e. path planning) then it should **not** be a **skill** as other skills depending on its answer will be slowed down unduly

Similar constraints hold when deciding whether something should be at the sequencer level or the planner level
Allocating knowledge across the architecture

Bandwidth

- data connection between different skills in the skill manager is very fast and often carries a lot of data like images, sonar values, and real-time tracked target position updates.

- Interface between the skill system and the RAP system consists primarily of commands to enable and disable skills and signals that certain skill-based events have occurred.

- This restricted bandwidth interface allows a very modular connection and easy distribution of the various tiers of the architecture across different machines (i.e., the connection between skills and RAPs can be implemented via TCP/IP).

- Thus, skills are generally written to abstract perceptual information so that only small amounts of data are passed to RAPs.
Allocating knowledge across the architecture

Task requirement

Each level of the architecture has built in functionality that makes certain operations easier

Example:
- RAPs has mechanisms for skill selection
- If skill contains many methods for handling different contingencies then it might be useful to break skill into several smaller skills for each contingency and let a RAP choose among them
Allocating knowledge across the architecture

**Modifiability**

skills must run very quickly and typically must be compiled in run time networks

RAP system and planner are both based on interpreters and their behavior can be easily changed by adding or modifying RAP descriptions and planning operators

✦ When a certain routine will require on line modification by a human operator then it should be put at the sequencer or planner level not at the skill level