Planning, Execution & Learning: Execution Architectures

Reid Simmons
Manuela Veloso

What is a Robot Architecture?

• Conceptual Framework for Designing Robot Systems
• Implementation “Glue” for Integrating and Coordinating Robot Systems

• Important Constraints on Architecture Design
  – Situated / embedded / interacting
  – High perceptual bandwidth
  – Dynamic / unpredictable environment
  – Uncertainty in sensing and action
**Capabilities**

- **Support Goal Achievement**
  - Complex tasks involving multiple steps
  - Conditional execution depending on environment

- **Support Acting in a Dynamic Environment**
  - Monitor for (relevant) changes
    - Contingencies / failures
    - Unexpected opportunities

**Architectural Design Principles**

- **Modularity**
  - Reduces complexity
  - Algorithms and representations tuned to particular roles

- **Concurrency**
  - Monitor environment while carrying out plans
  - Concurrent planning and execution

- **Hierarchy**
  - Layers of increasingly complex behaviors
  - Promotes reactivity
  - Disagreements on how to create hierarchy
Approaches to Hierarchy

- Temporal
- Behavioral
- Functional

Temporal Architectures

- RCS (Albus, 1991)
  - Layers operate at different temporal scales (order of magnitude)
  - Fixed, rigid communication patterns
  - Feedback through global database / memory

![Diagram of Temporal Architectures]

Planning, Execution & Learning: Execution 5 Simmons, Veloso: Fall 2014

Planning, Execution & Learning: Execution 6 Simmons, Veloso: Fall 2014
**Behavioral Architectures**

- Subsumption (Brooks, 1986)
  - Collection of concurrent FSAs
    - Direct, fixed connections between sensors and effectors
  - Minimal internal state
    - “World is its own best model”
  - Higher layers subsume output of lower layers

---

**Functional Architectures**

- Upper layers utilize functionality of lower layers to implement more complex tasks
- Upper layers typically operate at lower *temporal* and *spatial* resolutions

---

### Xavier Architecture (Simmons et al. 1995)

- Task Planning (Prodigy)
- Path Planning (Decision-Theoretic)
- Map-Based Navigation (POMDPs)
- Local Obstacle Avoidance (Curvature Velocity Method)
- Servo-Control (Commercial)
Anatomy of a Layer

- Each layer provides “guidance” to next lower level
- Each layer has *relative autonomy* to achieve tasks
  - Robustly, in face of uncertainty
- Each layer *abstracts data* for higher levels
  - Each layer must monitor progress of lower level

Three-Tiered Architectures

- **Planning**: Deals with goals and resource interactions
  - Task decomposition; Task synchronization; Monitoring;
  - Exception handling;
  - Resource management

- **Executive**: Deals with sensors and actuators

Planning, Execution & Learning: Execution Simmons, Veloso : Fall 2014
3T (Bonasso & Kortenkamp, 1996)

• Explicit Separation of Planning, Sequencing, and Control
  – Upper layers provide control flow for lower layers
  – Lower layers provide status (state change) and synchronization (success/failure) for upper layers

• Heterogeneous Architecture
  – Each layer utilizes algorithms tuned for its particular role
  – Each layer has a representation to support its reasoning

Remote Agent (NASA, 1999)

• First Truly Autonomous System in Space
  – Controlled DS1 spacecraft for several days in 1999
  – Closed-loop, goal-based commanding
  – Model-based programming
  – Real-time inference
  – Integrated declarative/procedural paradigms
**Managing Sets of Behaviors**

- 3T “Skill Manager”
  - *Skills* are concurrent behaviors, including perceptual behaviors
  - Dynamic creation of real-time feedback loops
    - Higher tier (“Sequencer”) connects sensing and action modules and *enables* subsets of skills
    - Skills indicate status by passing signals back to Sequencer

**Executive / Sequencer**

- Forms a Bridge Between Planning and Behaviors
  - Discrete vs. continuous control
  - Symbolic vs. numeric representations
  - Real-time considerations

- Basic Roles
  - Decompose task into subtasks and dispatch tasks
  - Monitor execution for contingencies and opportunities
  - Reschedule tasks (or schedule new tasks) upon failure

- Differences Between Approaches
  - Methods for distributing functionality
  - Representation of domain and control knowledge
  - PRS (Georgeoff); TDL (Simmons); ESL (Gat)
**TDL (Simmons & Apfelbaum, 1998)**

- High-Level Language Tailored to Task-Level Control
  - Extension of C++ with explicit syntax for task-level control constructs
  - Syntax for task decomposition, sequencing, monitoring, error handling
  - Compiles into pure C++ with calls to task management library

- Requirements
  - *Simple concepts* should be expressible in *simple terms*
  - Do not *preclude* expression of complex control constructs
  - *Natural integration* with existing code

**Task Trees**

- TDL Maintains and Coordinates *Task Trees*
  - *Execution trace* of hierarchical plans
    - Created dynamically at run time
    - Can be conditional and recursive
  - Temporal constraints (partially) order task execution
  - Planning and sensing treated as schedulable activities; Concurrent planning, sensing, and execution

- Task Trees Augmented with *Reactive Elements*
  - Task-specific execution monitors
  - Context-dependent, hierarchical exception handlers

- Replan by Analyzing and Manipulating Task Trees
  - Terminate subtrees
  - Add new nodes and/or temporal constraints
**CLARAty (Volpe & Nessnas, 2000)**

- **Two-Tiered Architecture**
  - Functional layer: Object oriented, reusable
  - Decision layer: Tightly integrates planner (ASPEN) and executive (TDL)
- **Developed at NASA for Next-Generation Mars Rovers**
  - Generic interfaces for sensing, control, and planning

**ROS (Robot Operating System)**

- **Communicating, distributed “nodes”**
  - Nodes are blocks of functional software
  - Nodes are meant to be general and reusable
    - Controllers, sensors, algorithms
    - Communication is anonymous publish/subscribe
- **“Stacks” of related functionality (e.g., navigation)**
- **Seamless logging and playback for development, debugging and visualization**
- **Has become the standard for robot software development and code sharing**
**Syndicate (Sellner et al. 2006)**

Synchronization / Coordination

- Planning
  - Executive
  - Behavioral Control

**Distributed Servo Loops**

- Roving eye tracks fiducials
  - Moves to provide best views
  - Provides 6-DOF transform between fiducials
- Manipulators use information to plan how to achieve goal
  - Use data base describing positions of fiducials on objects
- Behavioral layer enables dynamic, transparent inter-agent connections

- Roving Eye
  - Tracking
  - Images (via cameras)
  - The World

- Mobile Manipulator
  - Visual Servo
  - End effector delta
  - Arm Control
  - Manipulate environment (via arm)
Coordinated Synchronization

- Tasks can easily be synchronized between robots
  - Robots spawn tasks and add temporal constraints
  - Tasks are executed when constraints are met
  - No robot maintains the whole plan
- Executive layer enables dynamic, transparent inter-agent connections