Introduction to Intelligent, Cognitive, and Knowledge-Based Systems

Robot Control: Unified or Hybrid?

Gal A. Kaminka
galk@cs.biu.ac.il

This week, on Robots….

- Hybrid control:
  - Using both reactive and predictive systems together
  - Sometimes referred to as reactive and deliberative systems
- Key points:
  - How to use systems in a complementary manner?
  - Guidelines for use and design?
- Three-Tier, RCS architectures as case studies

The success of behavior-based control

- Planning (ala STRIPS) proven insufficient
- Brooks, behavior-based crowd claimed unnecessary
  - No (uniform) representation of world
  - Fast reaction to sensors
- Indeed, behavior based control successful
  - Changed paradigm
  - Showed everyone that robots can actually move…

The limits of (simple) behavior-based control

- Lots of hard work by designer
  - Building behaviors is sometimes not easy
  - Coordinating behaviors can be difficult
  - Fine-tuning can be hell!
- Overly dependent on sensors
- Not that good at managing complexity
  - e.g., the need for a temporal-coordinating FSA

Needed: Prediction

- Behavior-based control showed planning insufficient
- But planning (prediction) is still necessary
  - Cases where sequence of behaviors is unknown in advance

How do we get the best of reactivity and planning?

The Scientific Ideal: Unified Representation and Algorithms

One mechanism to rule them all…

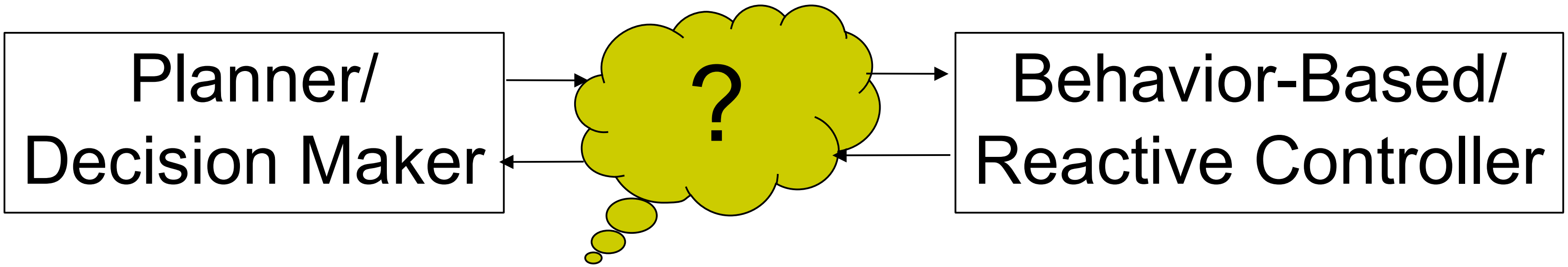
- Soar, ACT/R, EPIC, …
- Maes action-selection mechanism
- Online planning(?)

When ideal is too far? Engineer!

Hybrid Control

Have planning and reactive subsystems

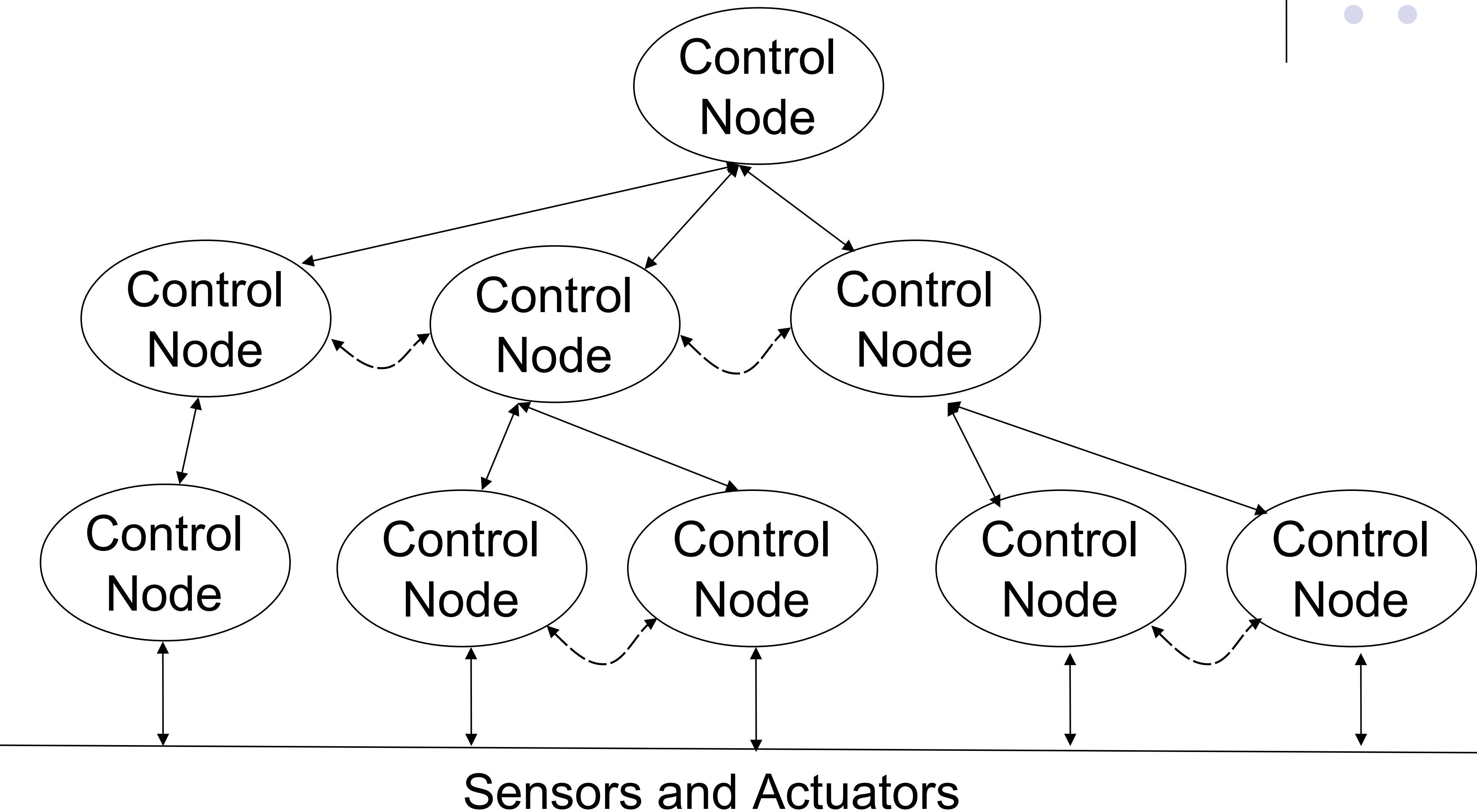
Somehow make them control robot together



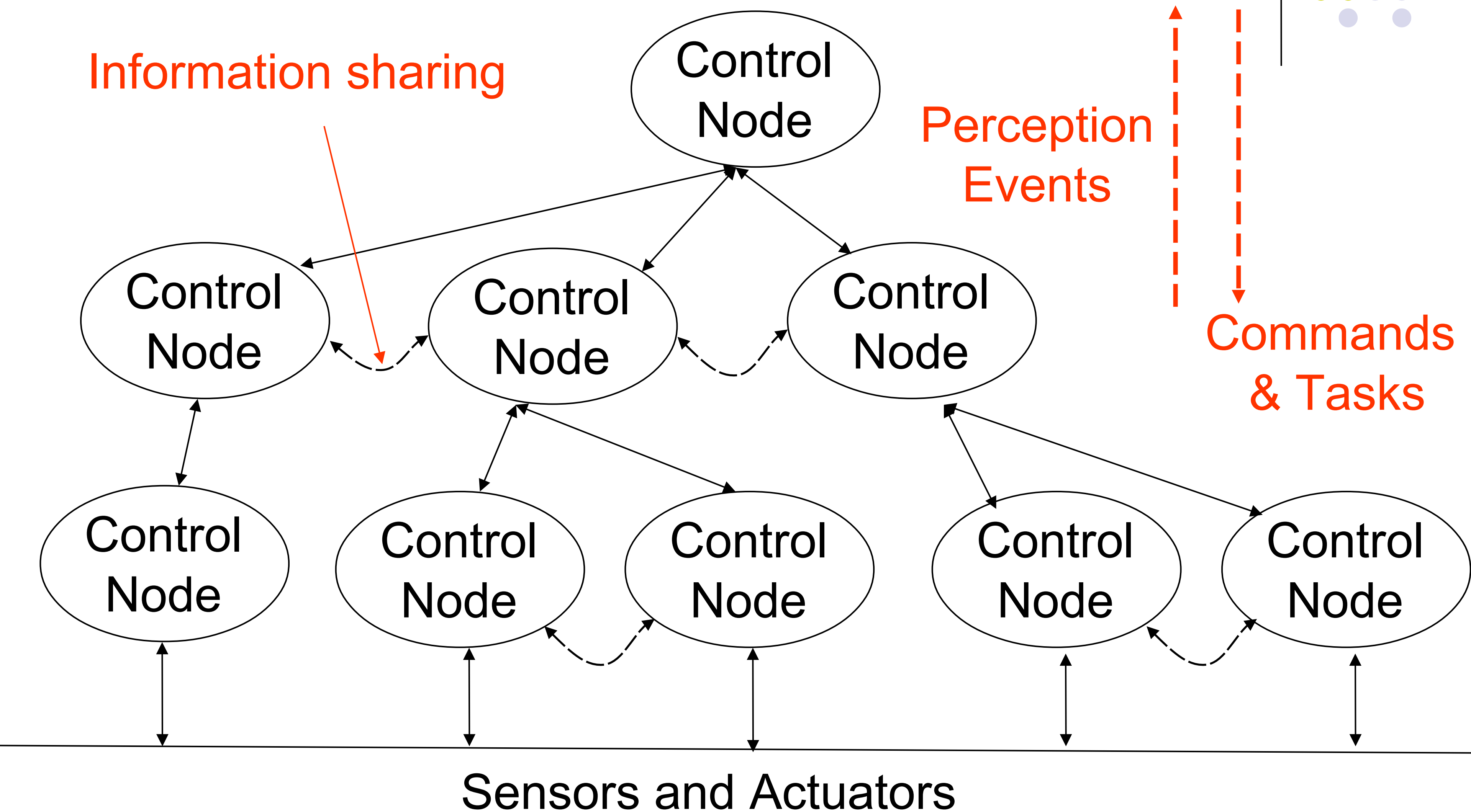
Two Case Studies

- Atlantis, a 3-Tier hierarchical architecture
  - A planner
  - An executive/scheduler/command sequencer
  - A set of (reactive) controllers
  - Actually one of 2-3 architectures with similar configuration
- RCS (Realtime Control System)
  - Hierarchical, many layers (as many as necessary)
  - Each layer can have several concurrent controllers
  - Controller: Sensor readings, world modeling, action, etc.

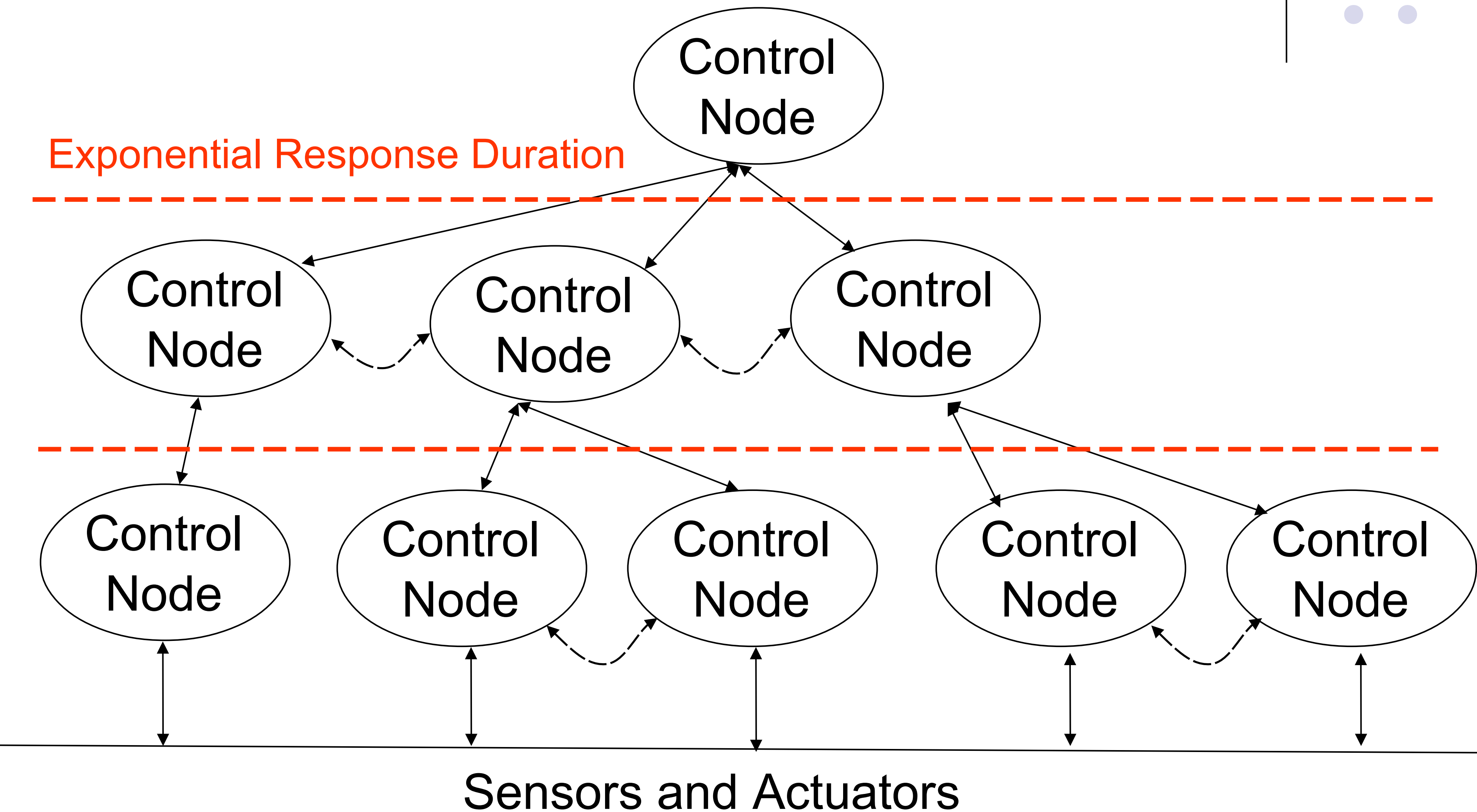
Real-time Control System (RCS)



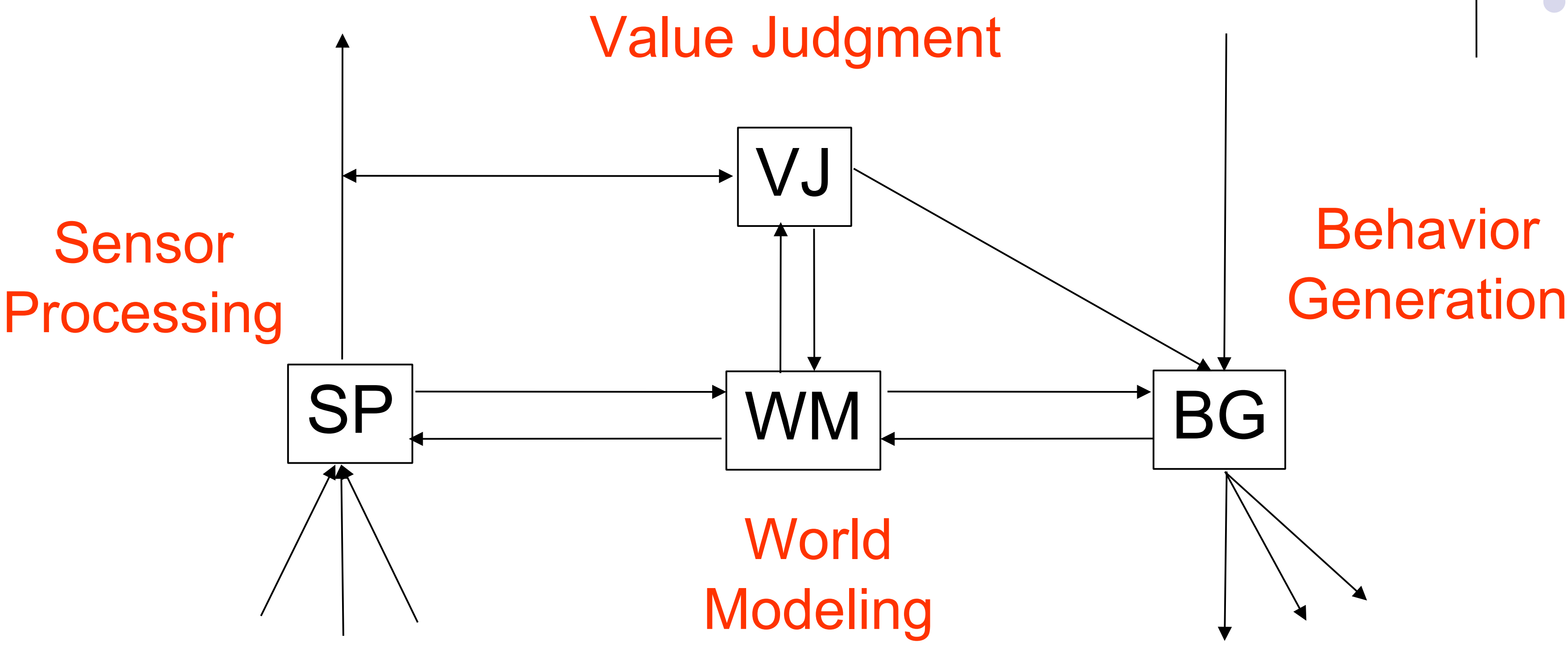
Real-time Control System (RCS)



Real-time Control System (RCS)

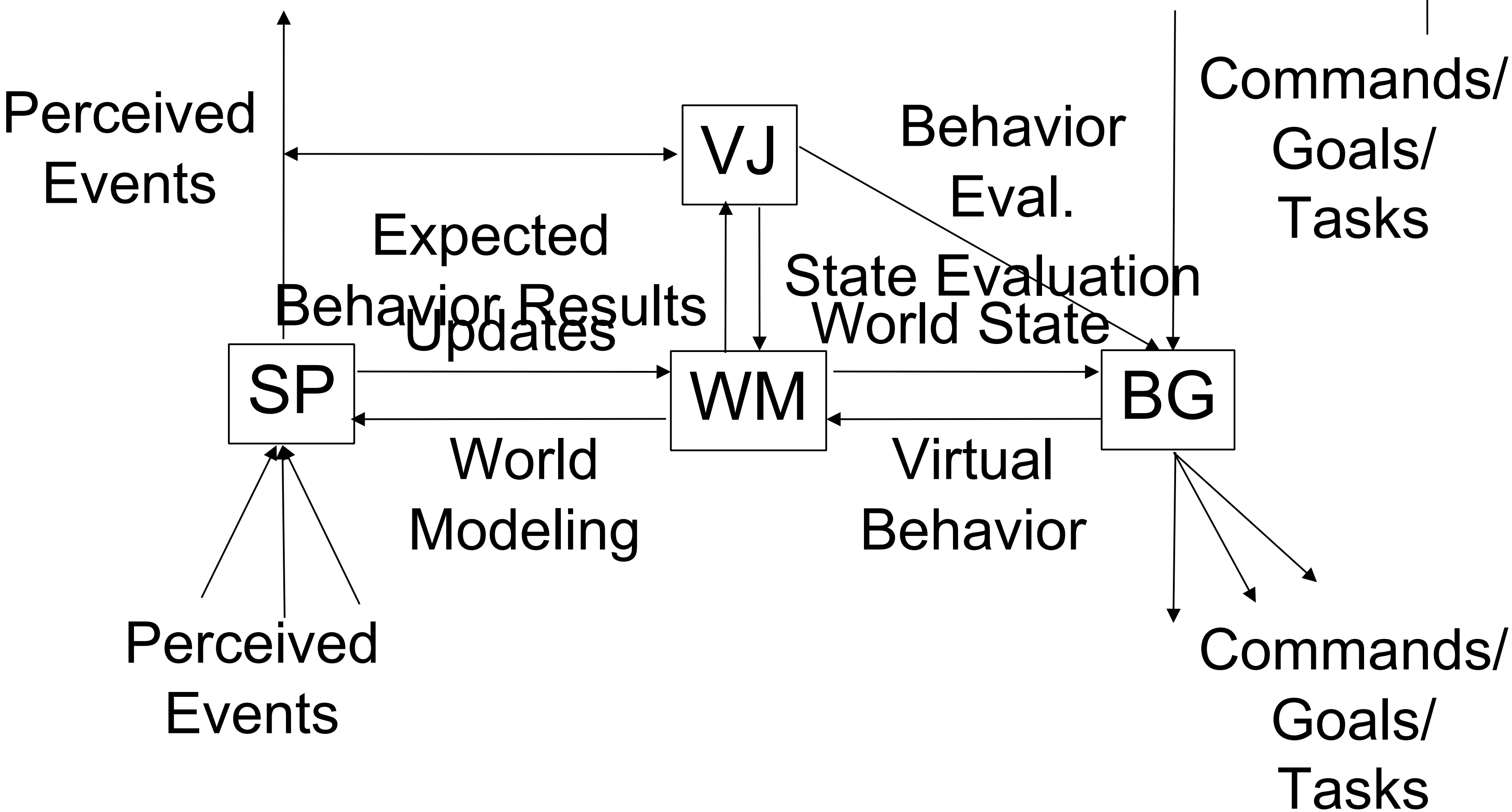


RCS Control Node: Modules



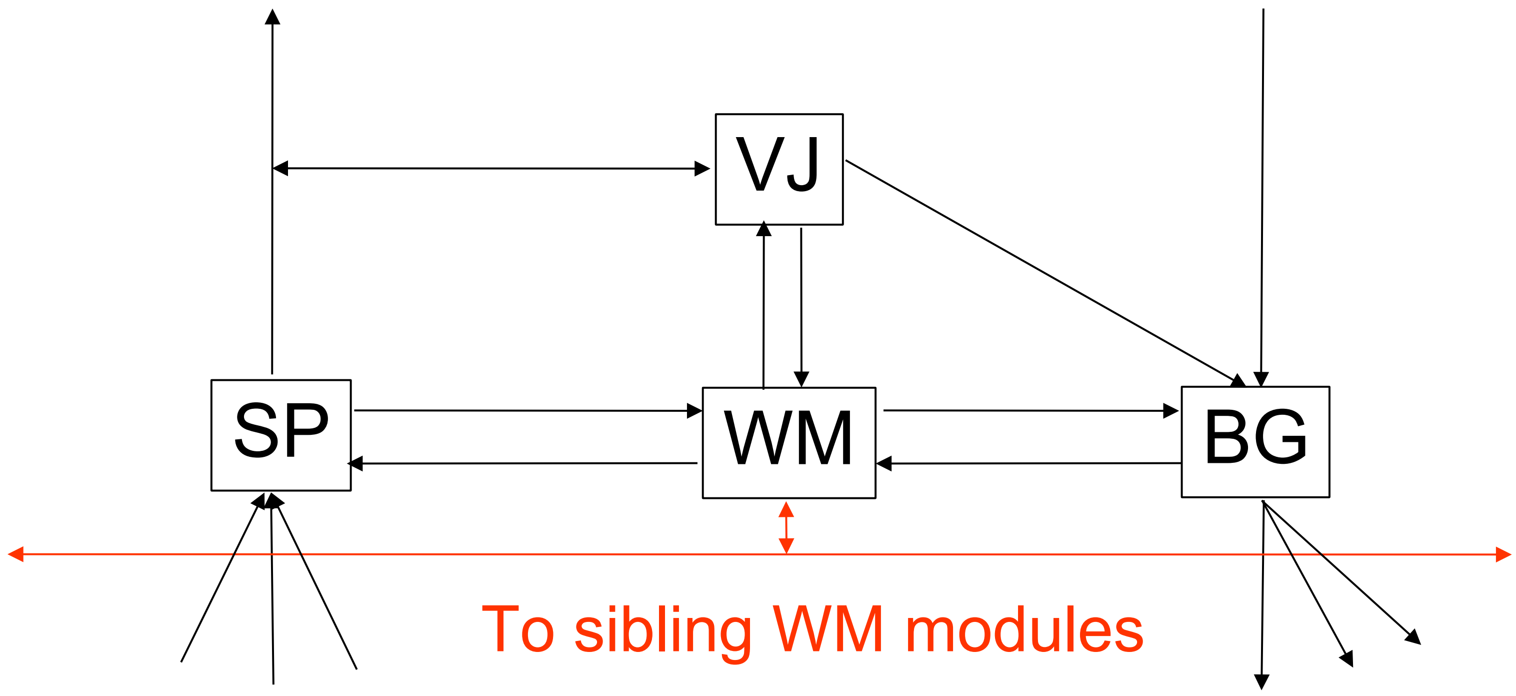
- Standard module components, designer-implemented
- Designer can merge components

RCS Control Node: Structure



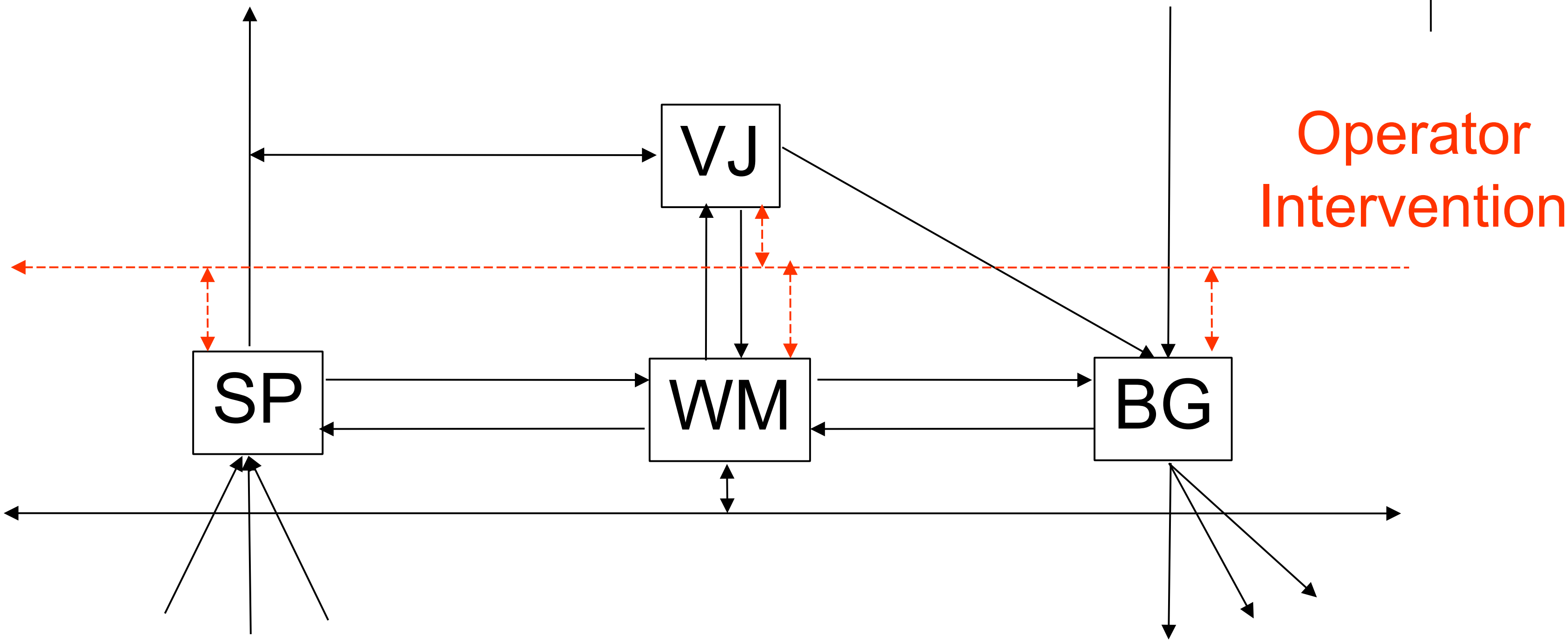
Standardized module interfaces

Connecting nodes at same level



- All WM are connected to each other, share information
- In principle, a distributed shared knowledge base

RCS operator interface

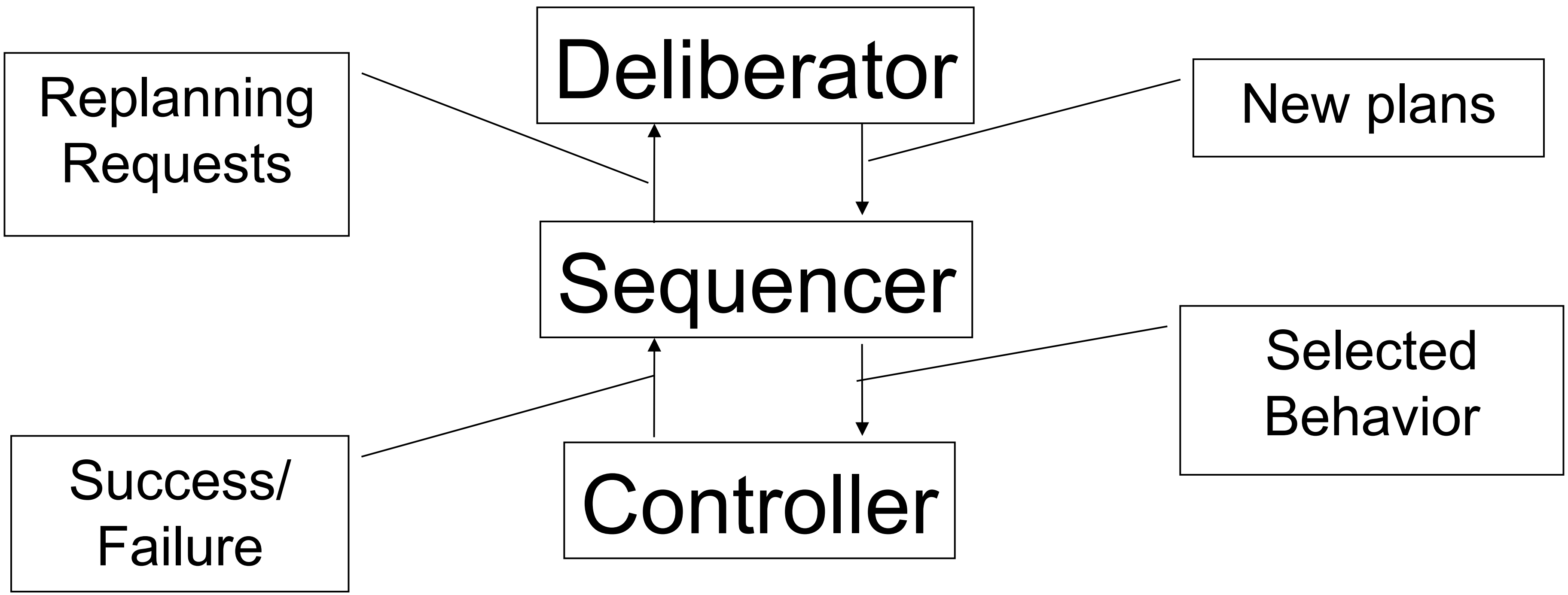


Operator interface not clearly specified

RCS Key points

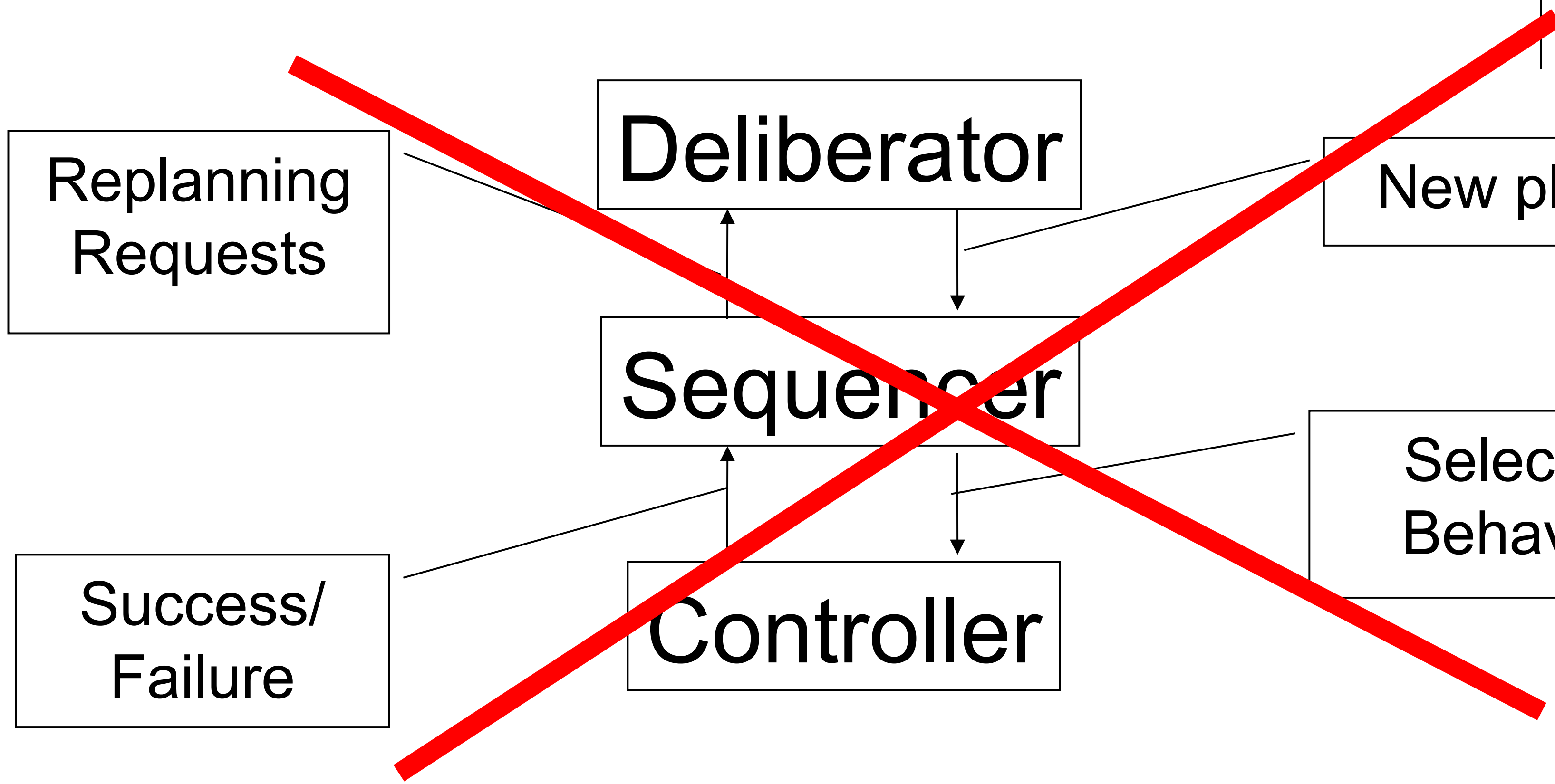
- Framework, not a system
  - Implementations exist
  - Implementations can deviate in instantiation of modules
- Differentiates levels along exponential time scales
  - Response duration
- Coupled sensing and acting at each level
  - Similar to subsumption architecture
  - But uses explicit world models, internal state, planning
- Strict hierarchy:
  - Commands passed down, top node drives execution

ATLANTIS: Structure Overview



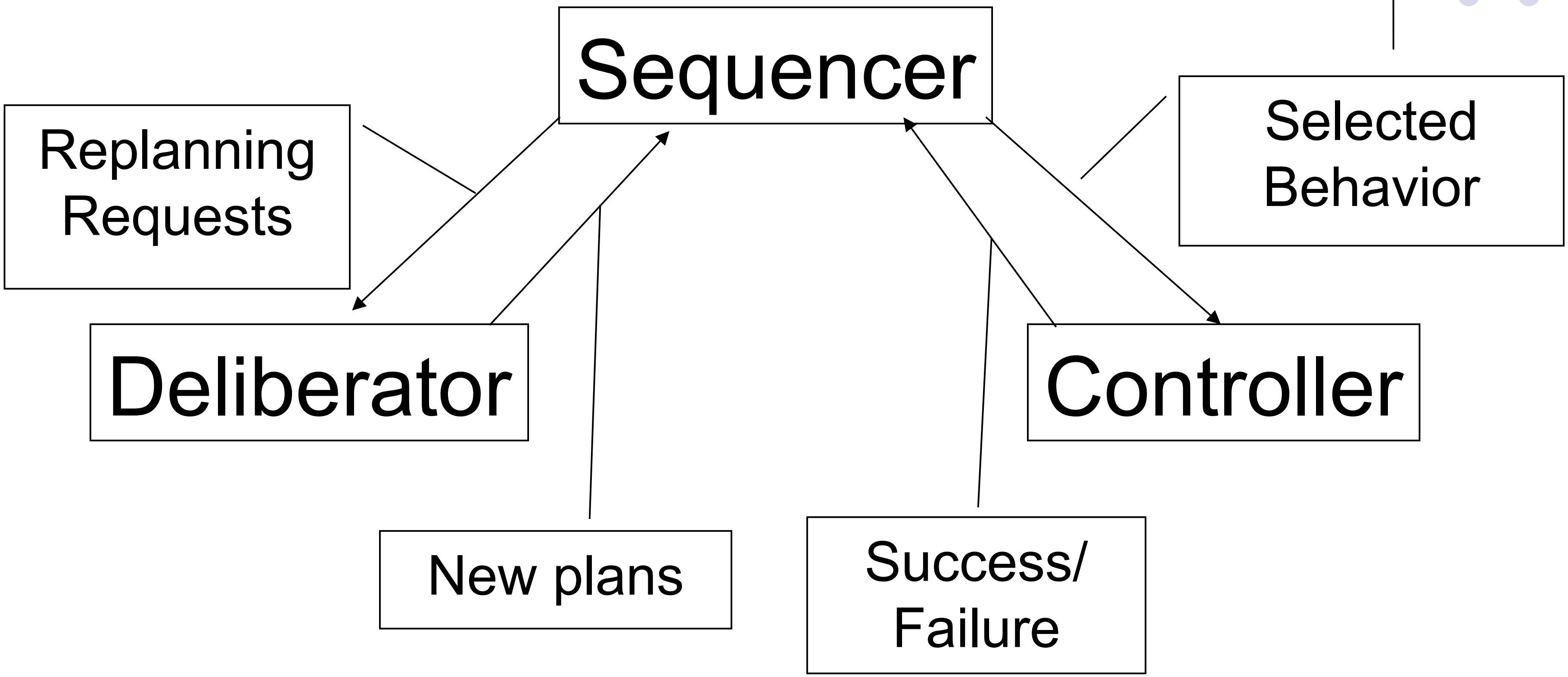
- Sequencer drives the control
- Makes queries to deliberator

ATLANTIS: Structure Overview



- Sequencer drives the control
- Makes queries to deliberator

ATLANTIS: Structure Overview



- Sequencer drives the controller
- Makes queries to deliberator

Controller

- Closed-loop controller (uses feedback)
  - Fuzzy, PID, predictive
- Must be fast enough (constant time/space)
- Must be able to detect failure
  - So sequencer can select another behavior, call replanner
- Avoid internal state (other than for state estimation)

- May have a library of controllers/behaviors
- Focus on one simple behavior at a time

Deliberator

- Operation initiated and terminated by sequencer
- No sensing---all internal state
- Time consuming tasks:
  - e.g., Planning routes using maps
- Can have any representation or implementation
  - As long as can pass useful information, on request

Sequencer (executive/scheduler)

- Interfaces controller to planner (deliberator)
- Selects which behavior to apply
  - Sequences, loops, conditionals, parallel threads
- Drives execution
  - Get success/failure status from controller
  - Examine state of world
  - Queries *deliberator* for heavy computations

Sequencer maintains internal state

- Key task is to select behaviors
- Makes decisions about selection
  - For instance, when several options available

- Maintain queue of behaviors pending execution
- Keeps track of previously selected behaviors
- Keeps track of successes and failures

Key points

- Non uniform representation, methods
- 3T is a framework, with very loose guidelines
  - e.g., Avoid internal state in controller, use more in deliberator
  - e.g., gray area when it comes to differentiating the layers
- Layers are distinguished by processing speed
  - Speed: With respect to response timing in the environment
- Layers function in parallel, asynchronously
- Controller must recognize successes and failures
- Planning is necessary, but only to guide execution
  - 3T not a strict hierarchy—middle layer drives execution

3T and RCS Hybrid Control

- Both differentiate time scales
  - 3T: response timing, RCS: response duration
- Heterogeneous representations
  - 3T almost no constraint, RCS structural constraints
- 3T not strict hierarchy—middle layer drives execution
- Differ in level of guidance to designer
  - 3T less structured, less guiding
  - RCS-based designs more guided,
- RCS more complex structure
  - Designer has to instantiate more modules