

AI ARBORIST:

PROPER CULTIVATION AND CARE FOR YOUR BEHAVIOR TREES

Mika Vehkala — Lead AI Programmer — Remedy Entertainment

Bobby Anguelov — Lead Animation Programmer — WB Games Montreal

Ben Weber — Data Science Manager - Twitch



MIKA VEHKALA
REMEDY ENTERTAINMENT

Overview

1. Database / Blackboard
2. Decorators
3. Splitting trees
4. BTs for coordination

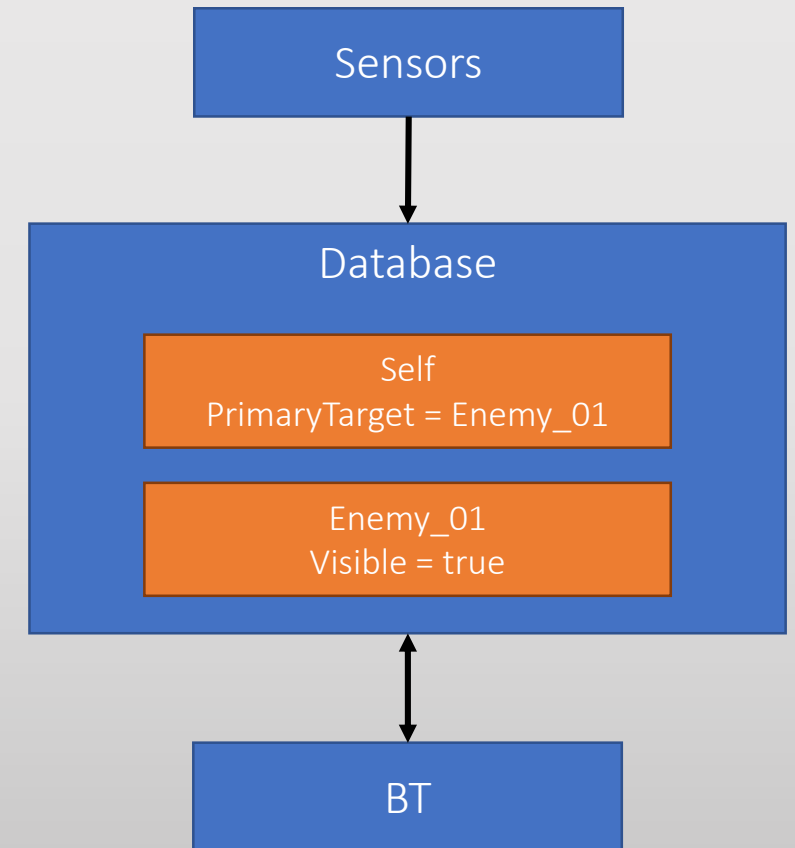
SOME CHALLENGE AREAS

- Generalization and granularity of tasks
 - How to create library of re-usable building blocks
- How about interruptions / transitions



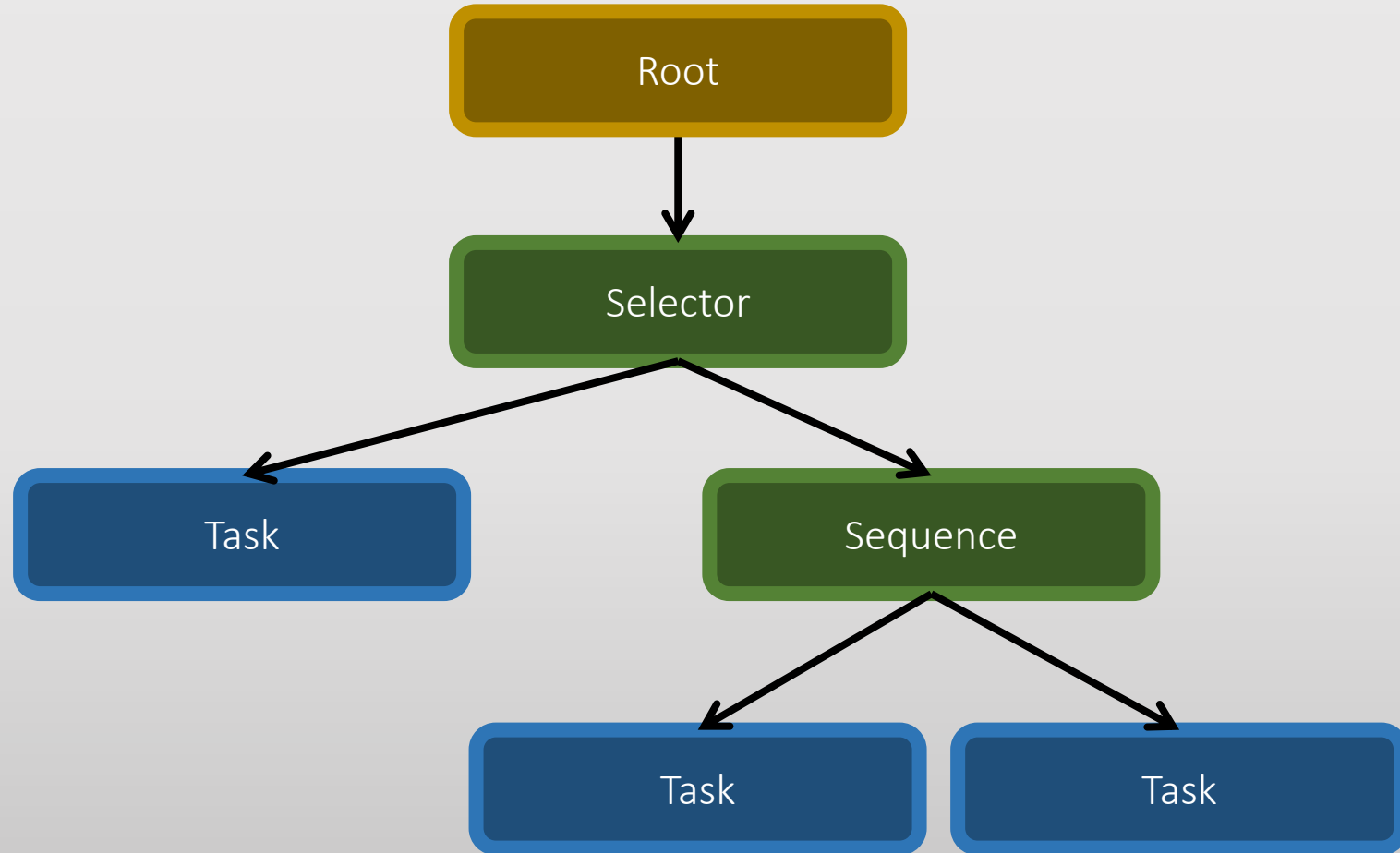
DATABASE FOR BT

- Have simple, uniform and compact database (blackboard)
 - Conditions are easier to implement
 - Debugging is easier
 - Performs better
 - For example, list of dictionaries
 - One dictionary per tracked entity
 - Reference with paths "Self.PrimaryTarget.Visible"



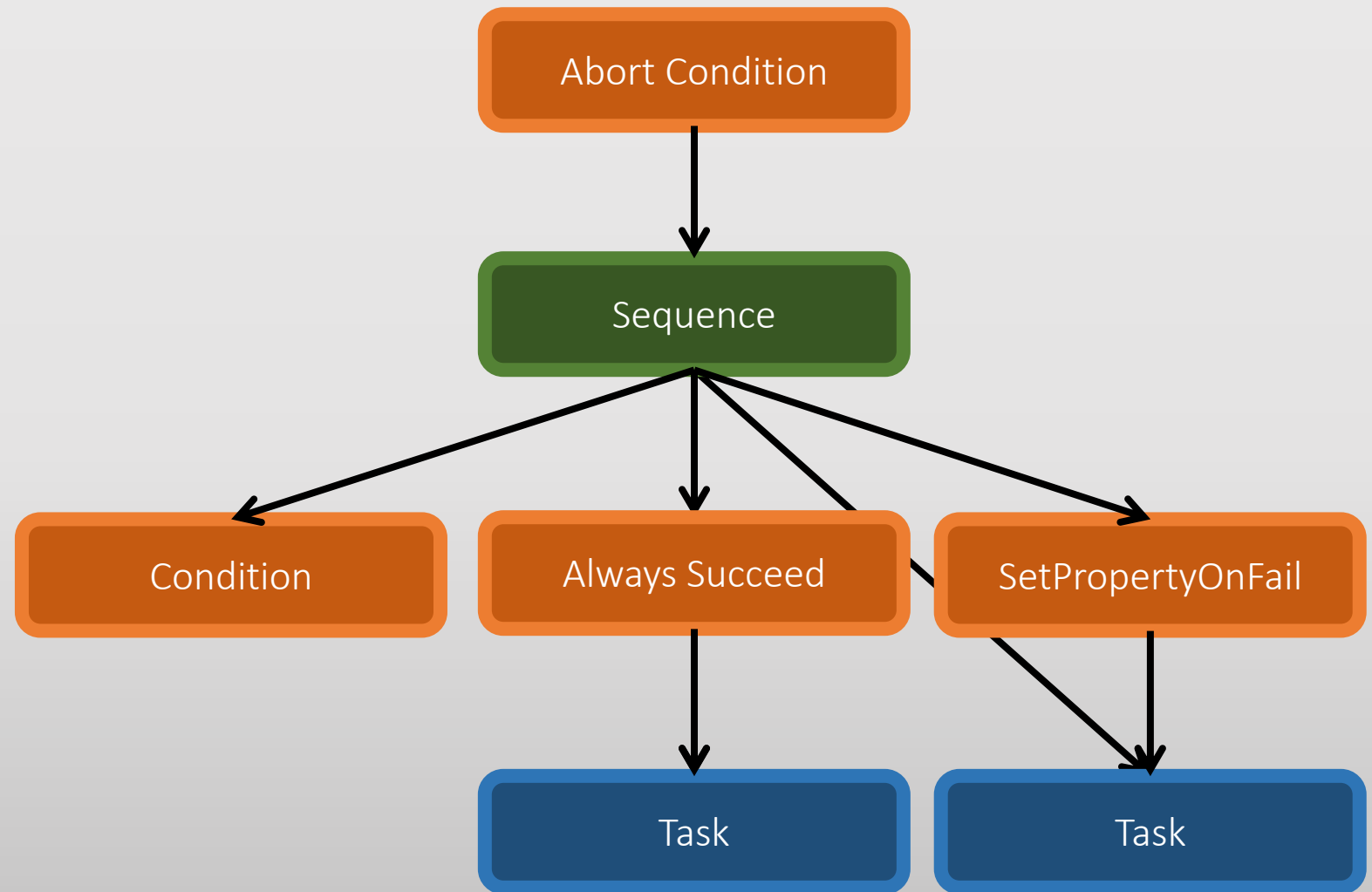
BEHAVIORTREE NODES AND DECORATORS

- Root
- Control flow nodes
 - Selector
 - Sequence
 - Parallel
- Execution nodes



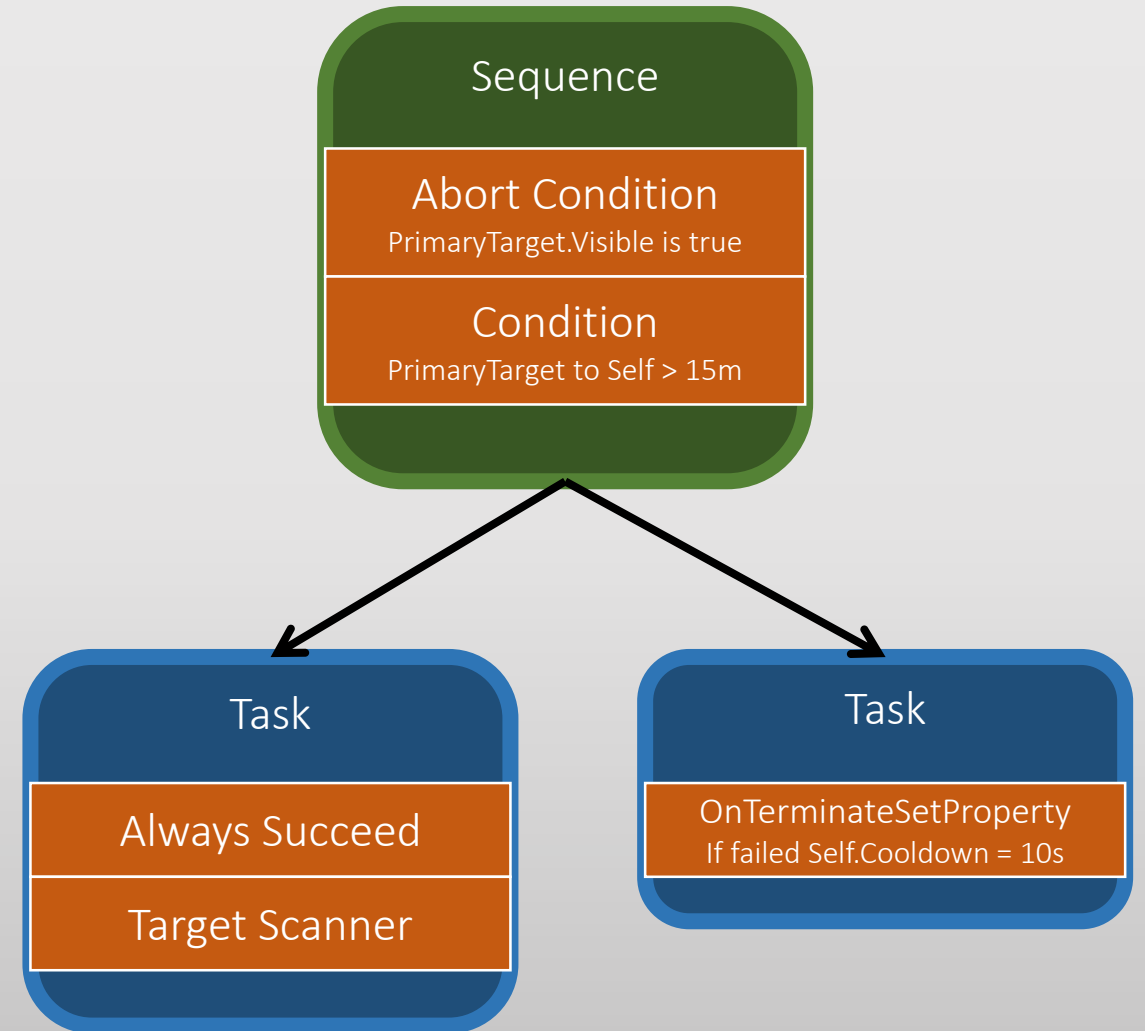
DECORATOR NODES

- Common types
 - Control activation / deactivation
 - Modify termination status
 - Add-on functionality

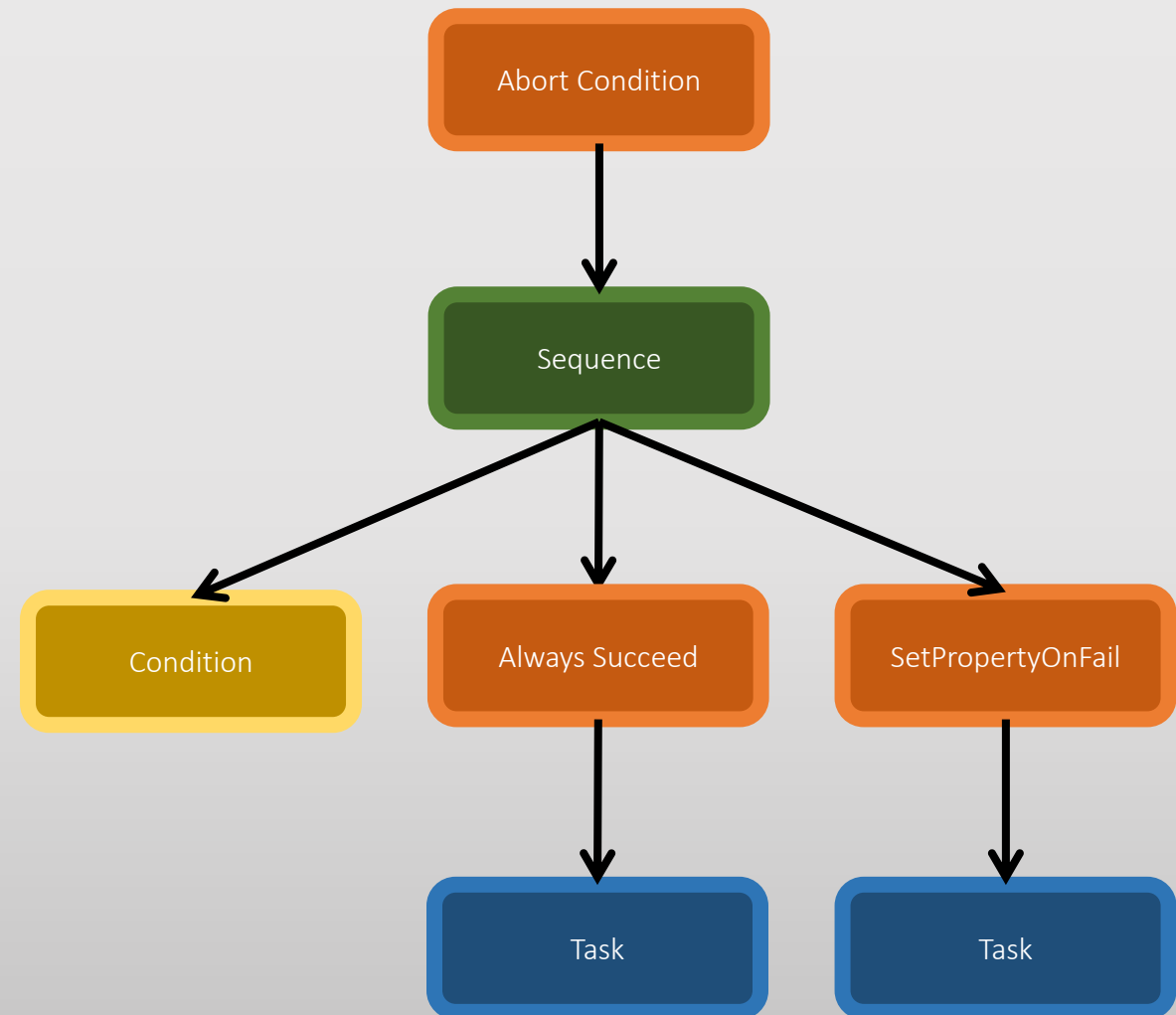
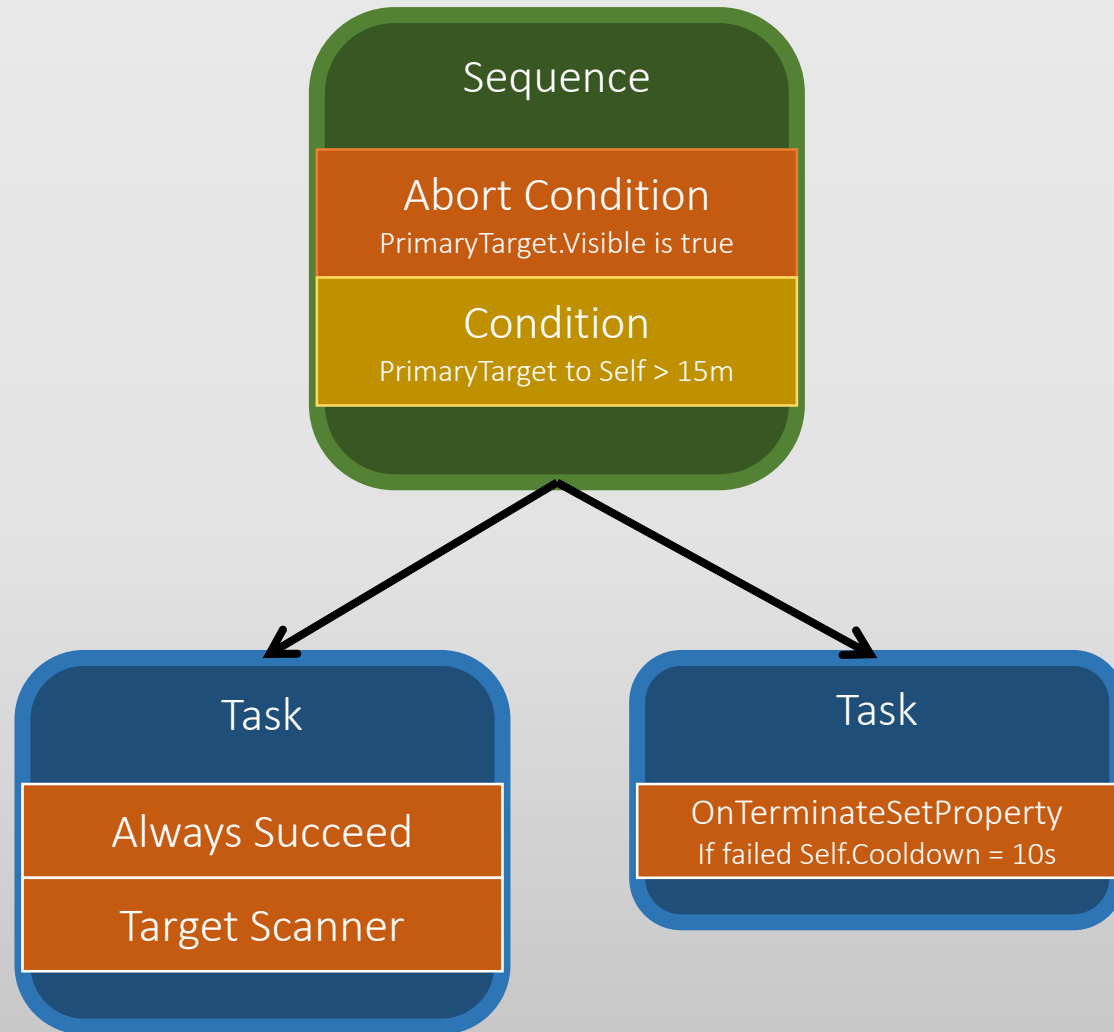


NODE DECORATORS

- Node decorators vs decorator nodes
 - Unreal 4 BT's (Mieszko Zielinski) does this
 - Compact
 - More options to simplify nodes
- Common types
 - Control activation / deactivation
 - Modify termination status
 - Parallel execution
 - Add-on functionality

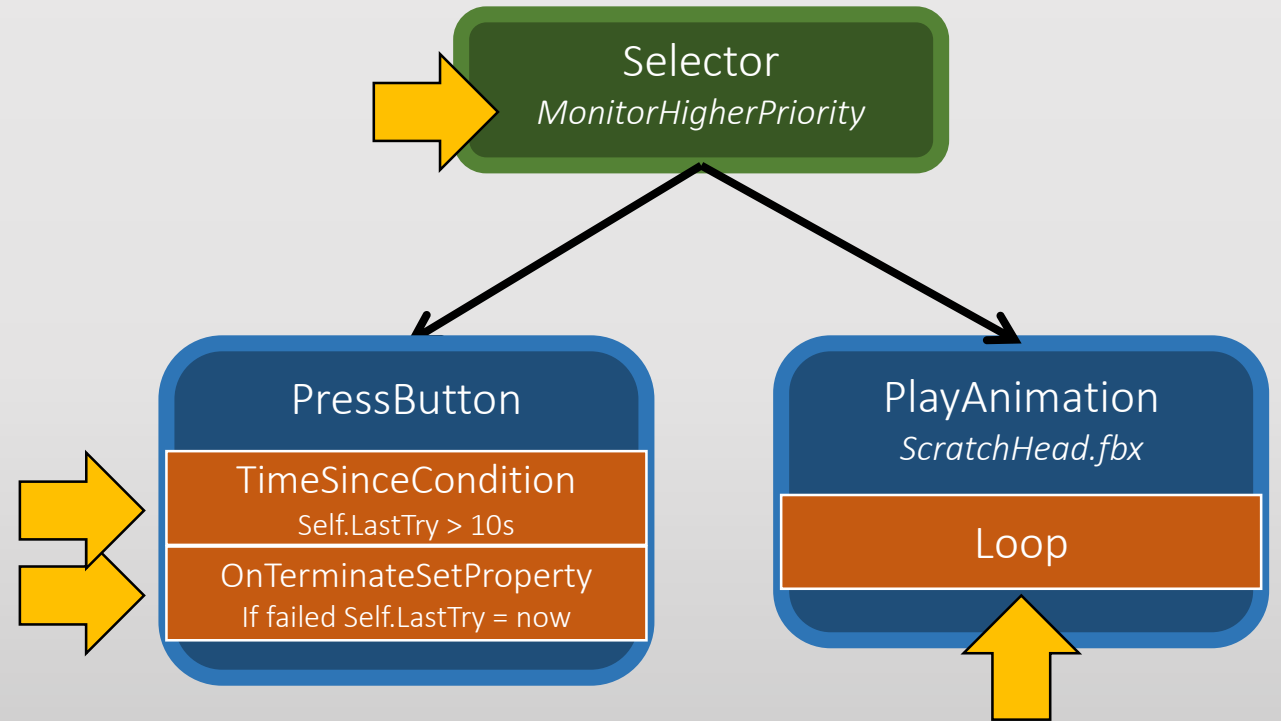


NODE DECORATORS VS DECORATOR NODES



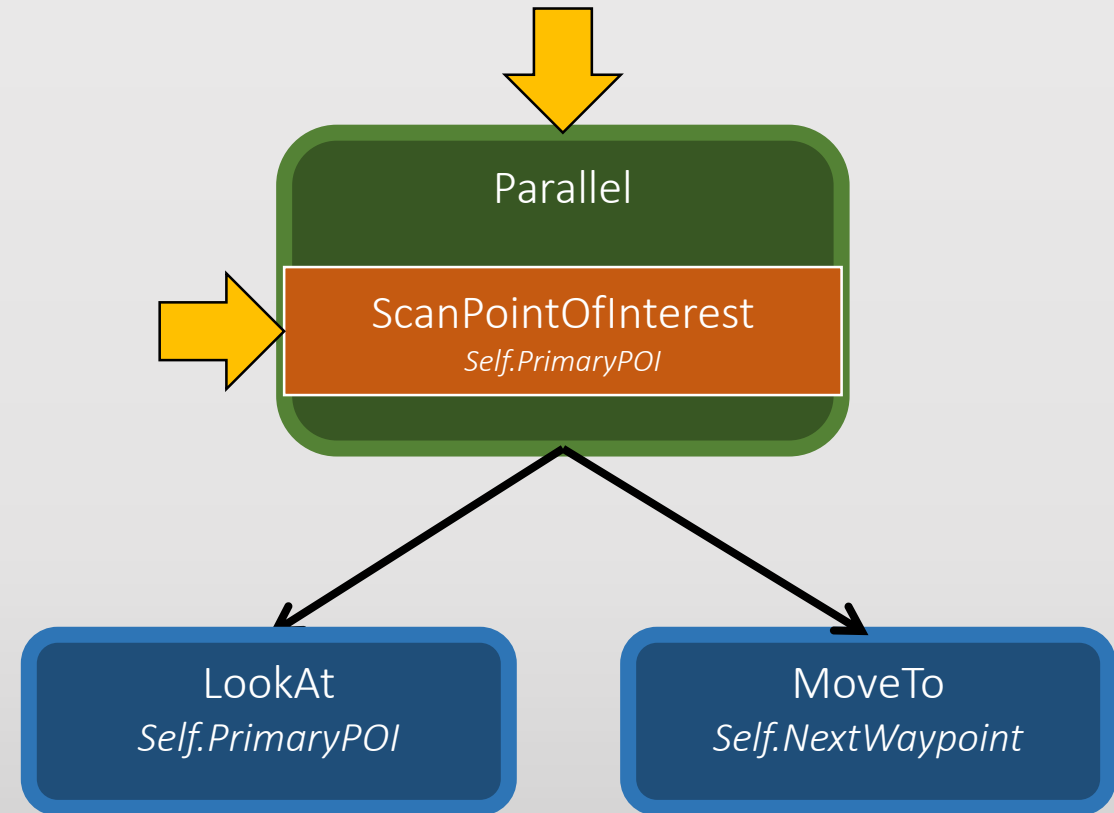
EXAMPLE: RETRY

- OnTerminationSetProperty
 - When node completes with specified result set property
- TimeSinceCondition
 - Check delta to timestamp property value
- Loop
- Selector to monitor higher priority branches



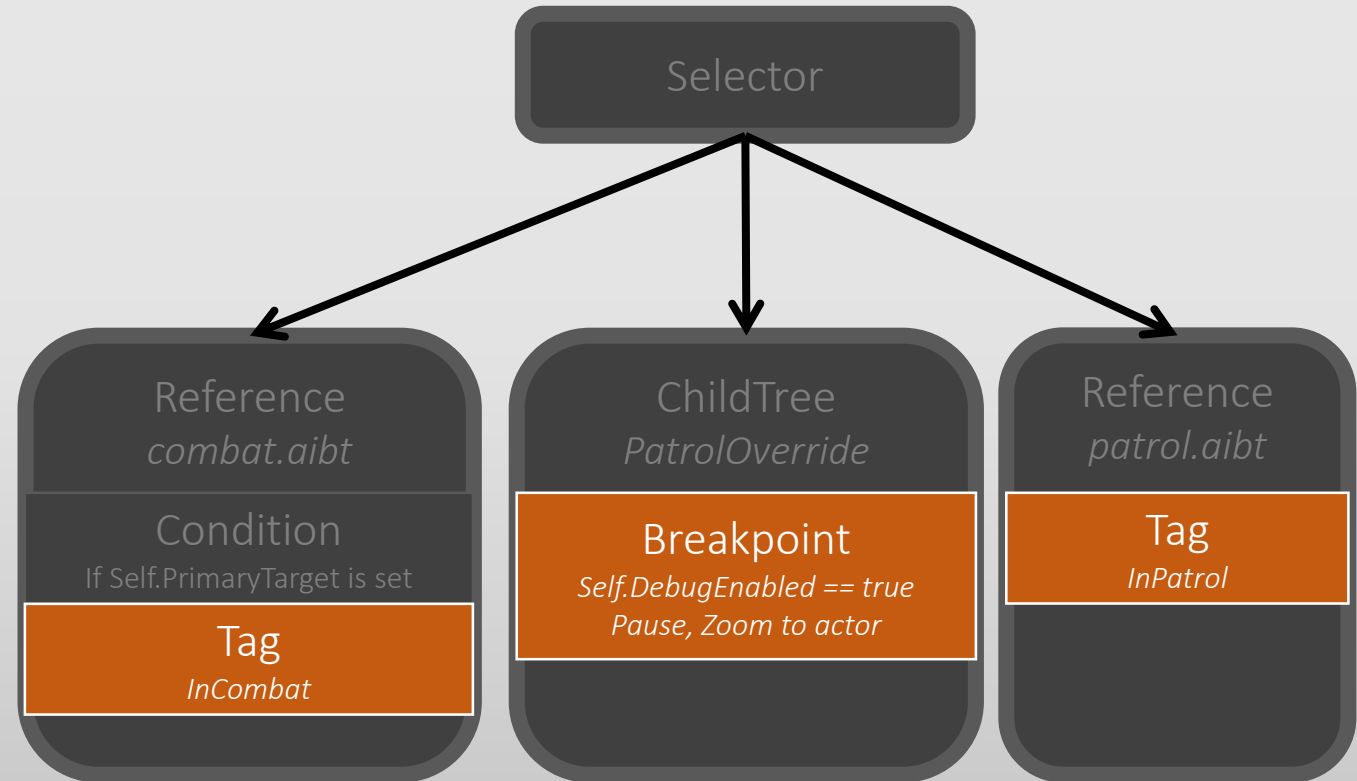
EXAMPLE: LOOKAT

- Two types of parallel execution
 - Updateable decorator
 - Parallel node with two children



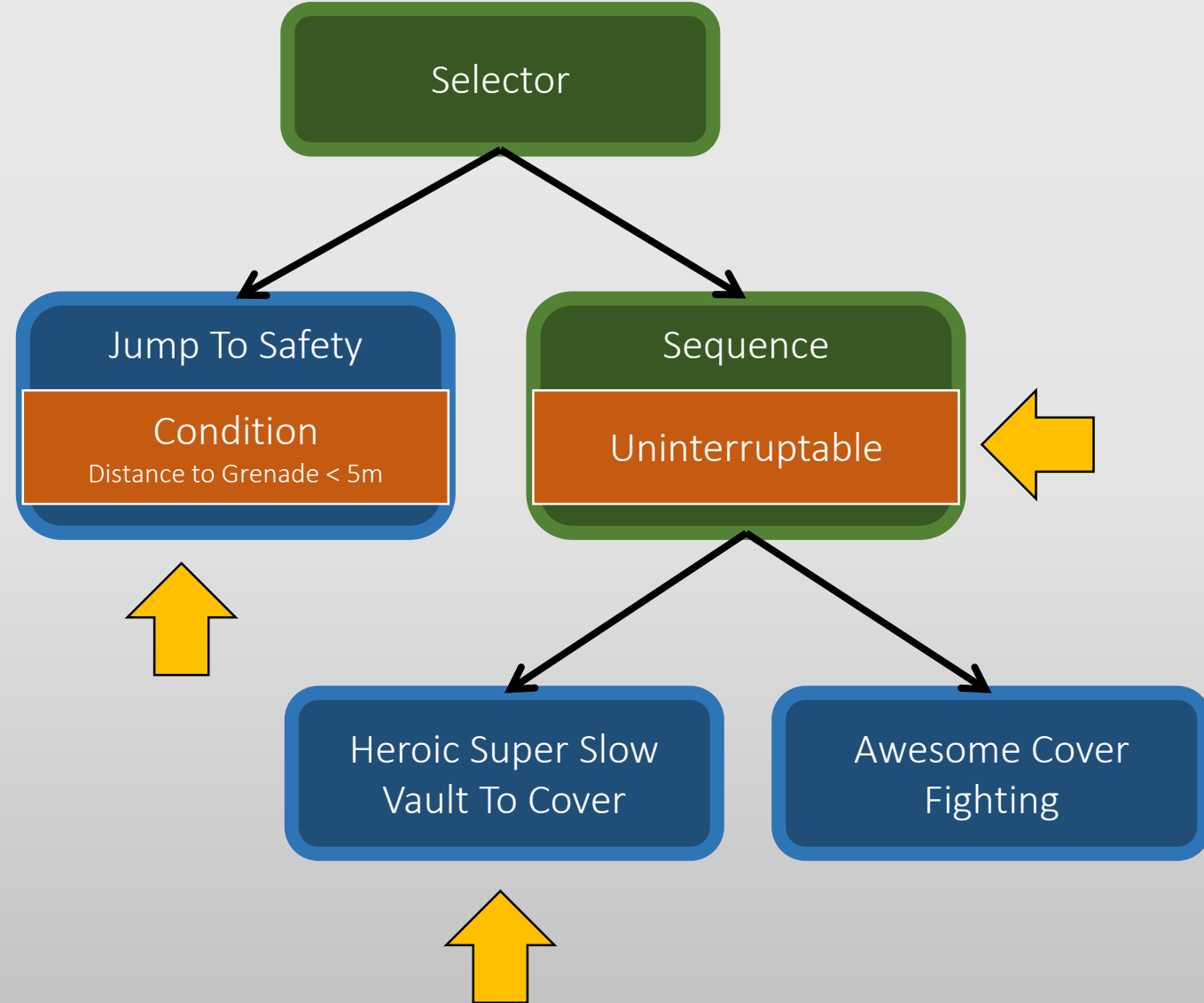
EXAMPLE: DEBUGGING

- Tag decorators
 - Can query for active tags
- Message decorators
 - Conditional
- Breakpoint decorators
 - Conditional
 - Actions
 - Pause
 - Break in code
 - Zoom to NPC



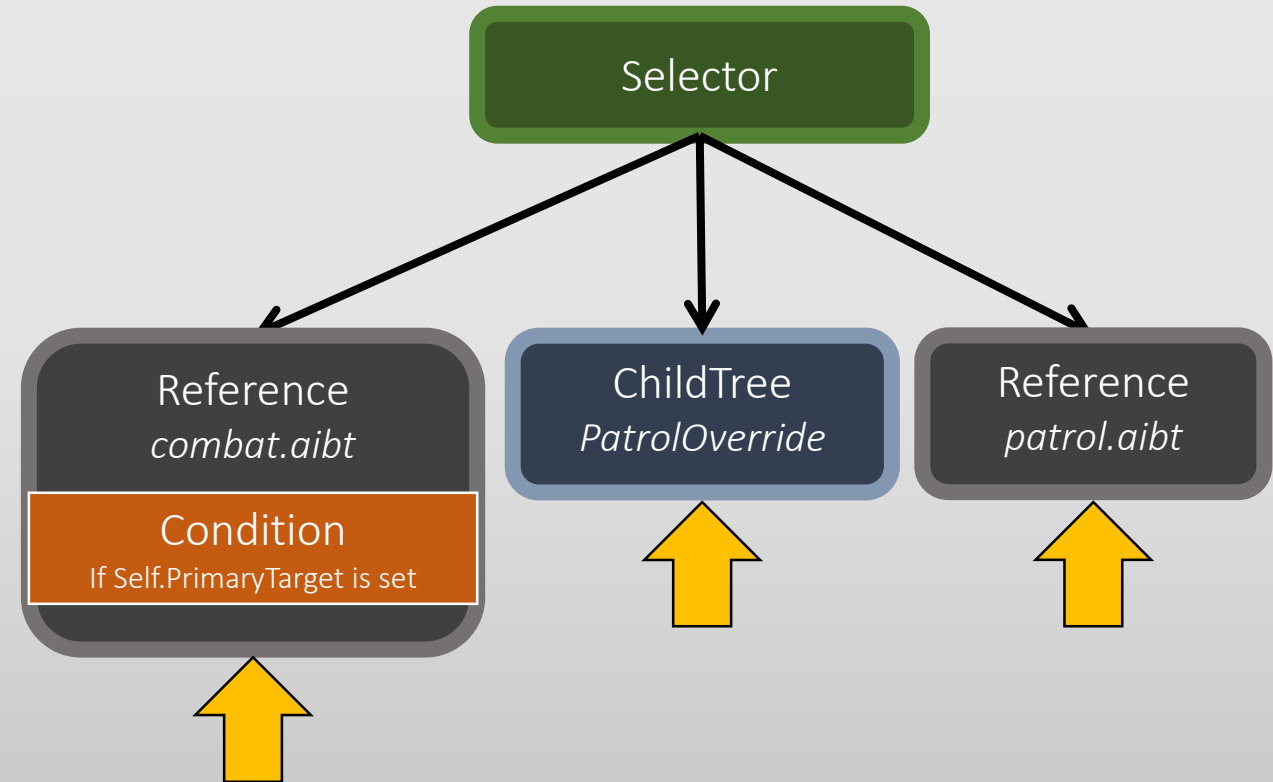
INTERRUPTIONS

- Task being interrupted may do cleanup, but terminates immediately
- New task starting may have to wait until it can perform all of its logic
- Can also use decorator to restrict tree update but this should be used carefully



SPLITTING TREES

- Reference nodes
 - Design time
 - Merged during export or load
- Child-tree nodes
 - Runtime
 - Allows for level specific changes

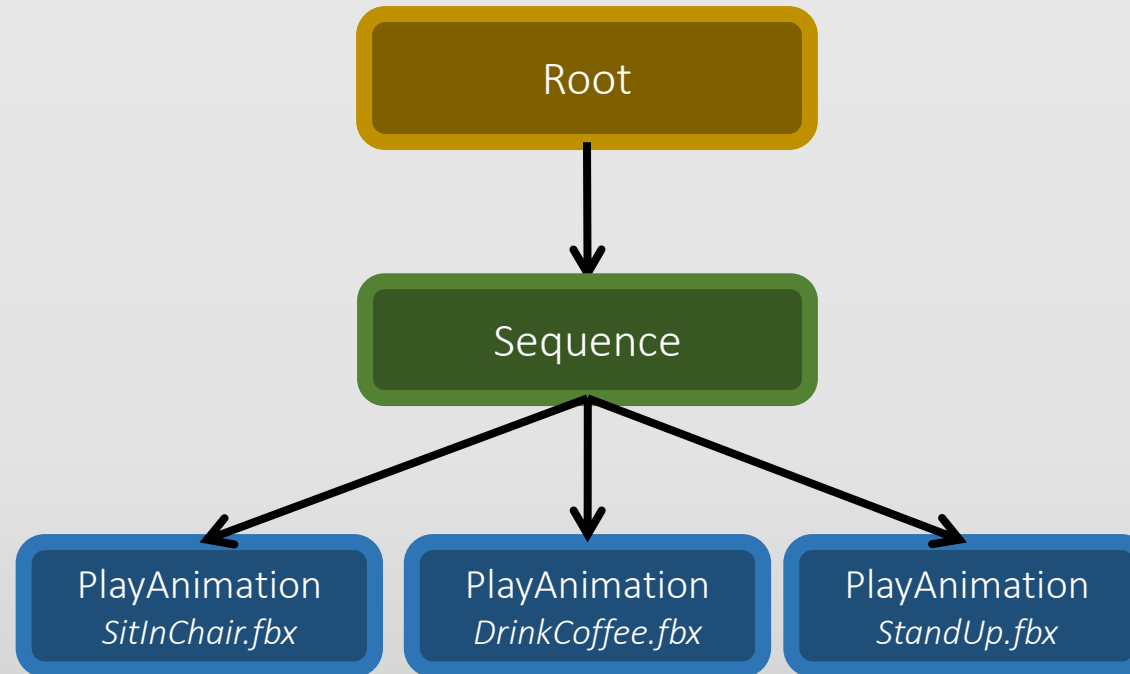


CHILD TREE USE-CASES

- Smart objects
 - Embed tree, instructions how to use
 - Can embed also data
- Items and Weapons
 - Combat tree in weapons
- Level scripting
 - Create custom tree and override at highest level
 - Spawning, do cool entrance, can have different options based on situation

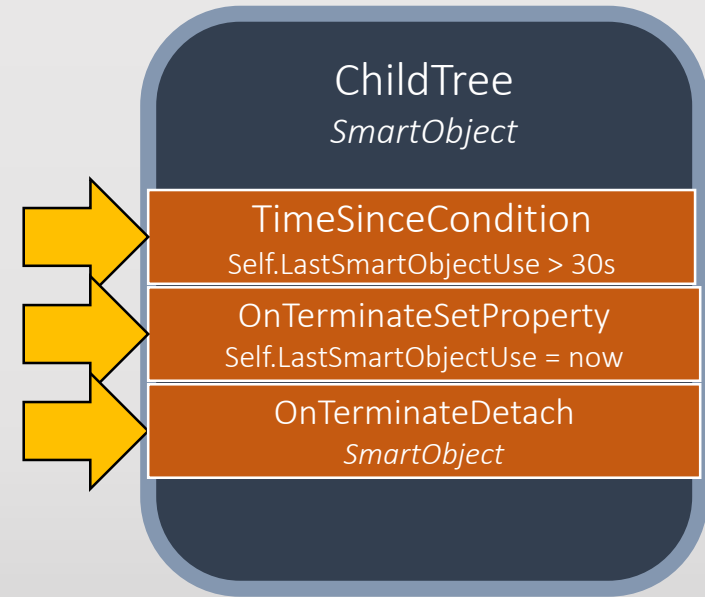


EXAMPLE: SMART OBJECTS



EXAMPLE: SMART OBJECTS

- Cooldown used to block re-use immediately
- Exposing detach as decorator



COORDINATOR

- BT assigned into a volume
- Custom BT nodes and decorators
 - Update goals / objectives
 - Update influence maps
 - Assign roles
 - Write to individual NPC memory
 - Share data accessible by NPCs



WRAPUP

- Node decorators are great
- Don't be limited to a single static tree
- BTs are not character specific
 - Also, not your only mechanism to control characters

Read more

- AIGameDev.com
<https://aigamedev.com/insider/presentations/behavior-trees/>
- Unreal 4 Behavior Trees
<https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/HowUE4BehaviorTreesDiffer/index.html>
- AngryAnt's Behave
<http://eej.dk/community/documentation/behave/>





BOBBY ANGUELOV

WB GAMES, MONTREAL

Overview

1. Common Pitfalls
2. Best Practices
3. Authoring Guidelines

BEST PRACTICES



- This is a best practices talk on Behavior Trees
- We're gonna discuss some problems
- We're also going to discuss some solutions



REINVENTING THE WHEEL



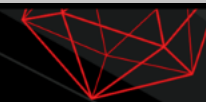
- Nothing I'm gonna mention is [Revolutionary](#)
- Going to provide some observations and suggestions
 - [Both Dave Mark and Alex Champandard have touched on these in the past](#)
- Hoping to prevent some pain and kick off some interesting discussions



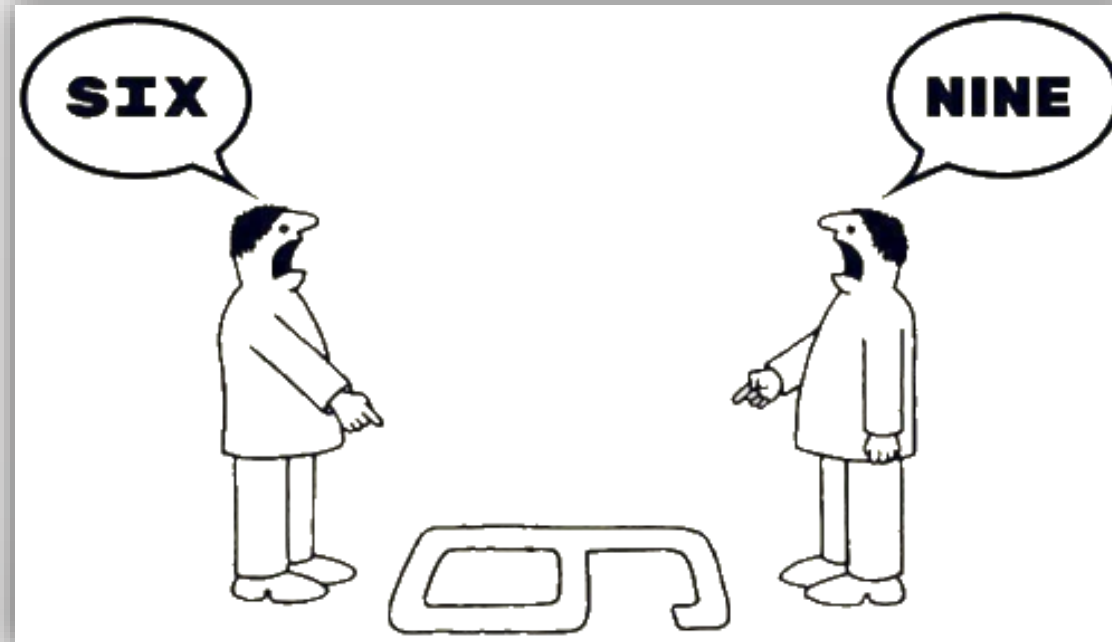
MY PERSPECTIVE



- My approach to engineering is that there are no Silver Bullets
- Every technique/approach/methodology has pros and cons

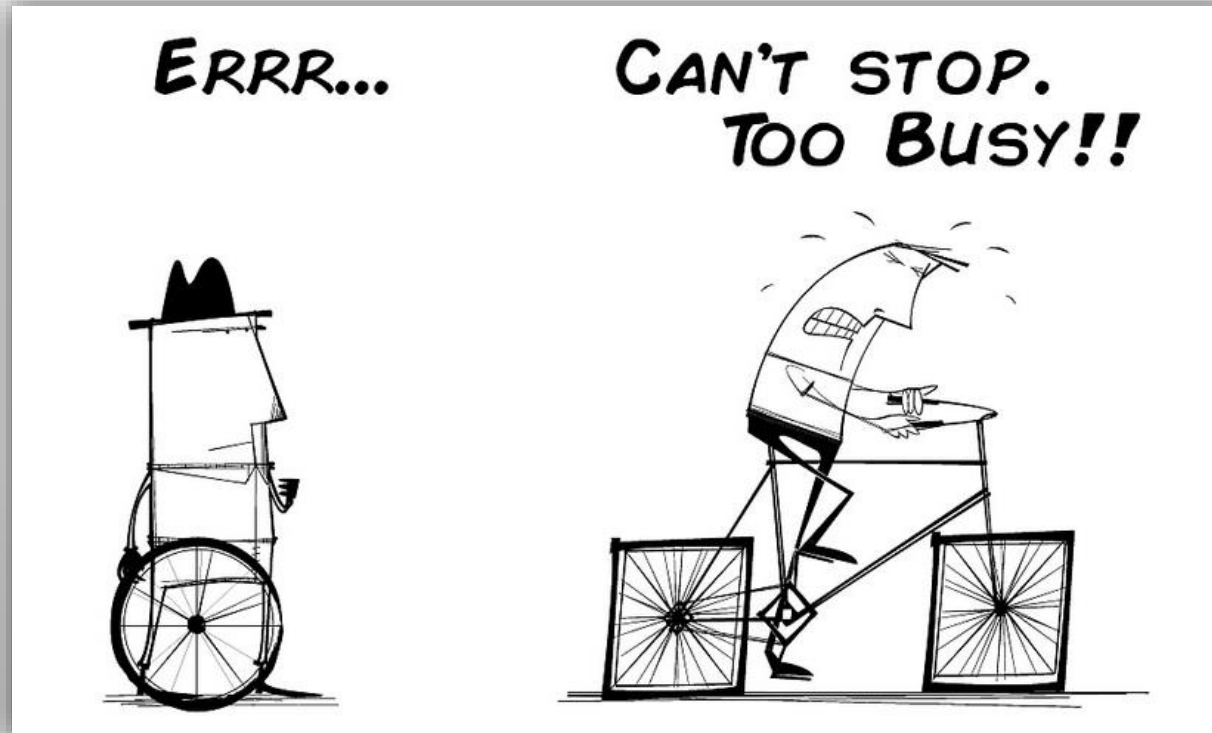


MY PERSPECTIVE



- We often lose perspective and brute force our solutions
- It's our job as developers to understand our tools
 - Leverage their strengths
 - Avoid their pitfalls

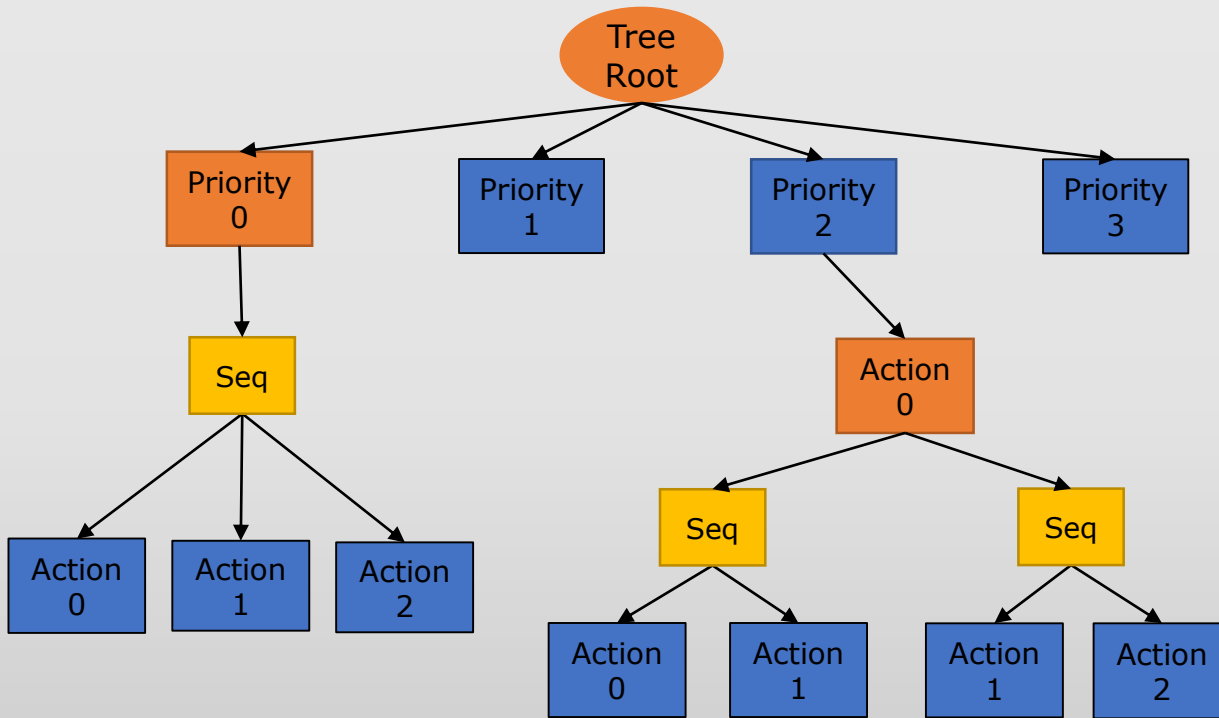
MY PERSPECTIVE



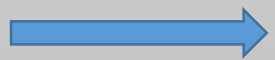
- We're often so busy that we miss the obvious
- Good tools = Better workflows = More iterations = Better result

BEHAVIOR TREE VISUALIZATION

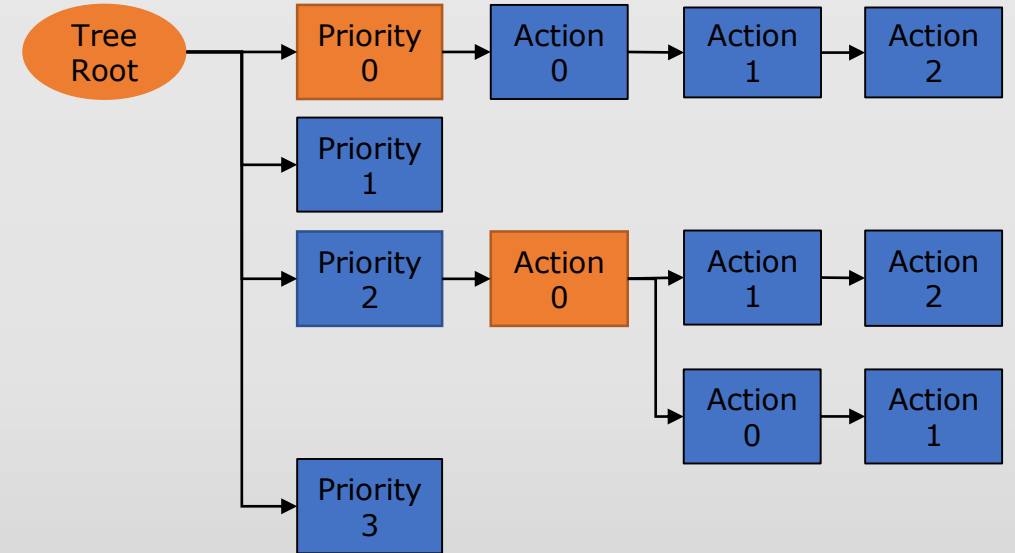
- I tend to think of, and visualize, behavior trees in a different way



Priority



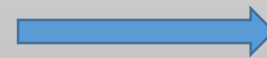
Sequence Order



Note: Sequences are built-in/implied



Priority



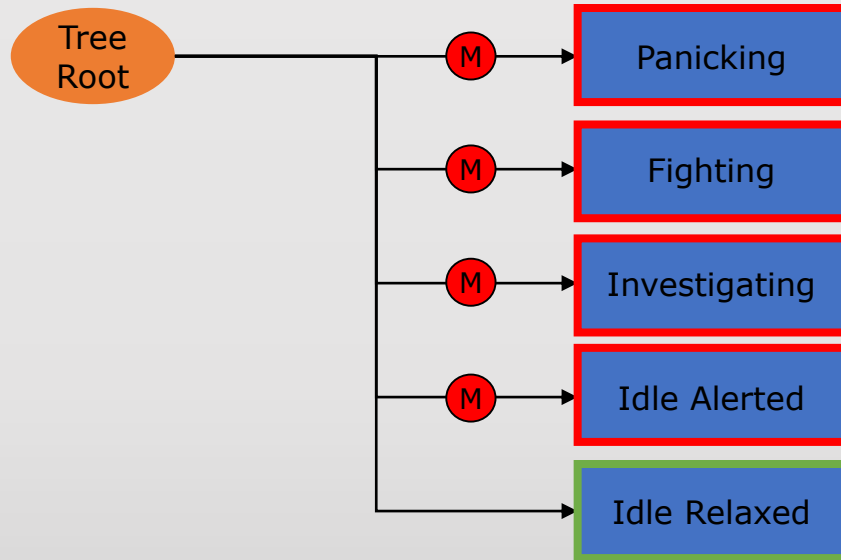
Sequence Order

BEHAVIOR REACTIVITY AND PRIORITY

- Behavior Trees are inherently bad at two things
 - Transitions/Interruptions
 - Behavior Prioritization



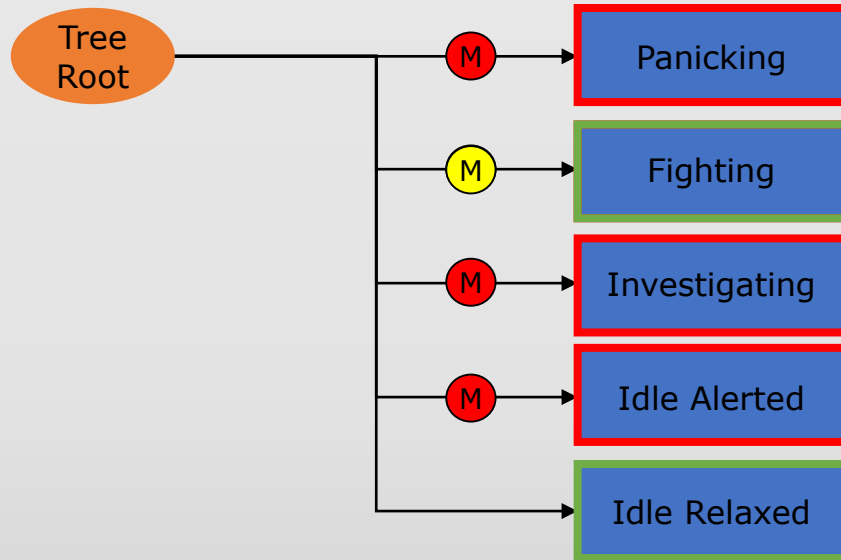
HOW TRANSITIONS OCCUR



Event Driven Behavior Trees

- Event driven behavior trees feature event monitor nodes which are evaluated with each BT update.
- The deeper that we are in the tree, the more monitors we need to update on subsequent updates.

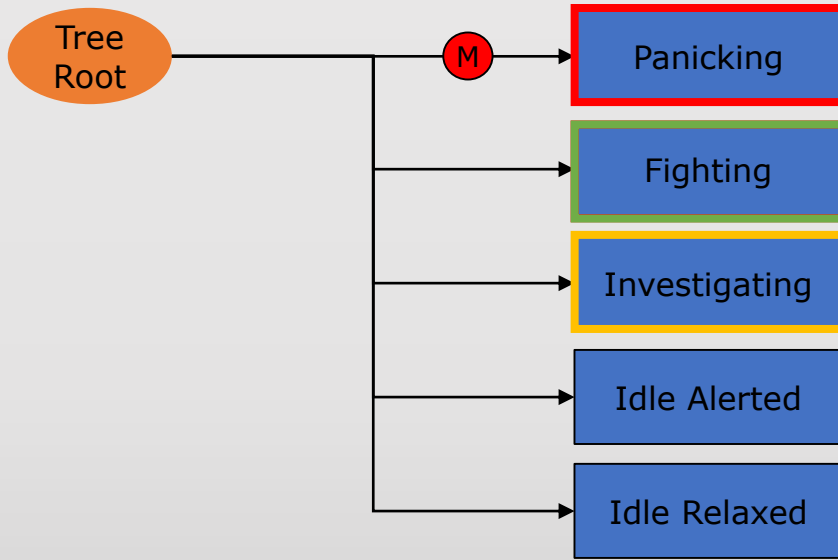
HOW TRANSITIONS OCCUR



Event Driven Behavior Trees

- Event driven behavior trees feature event monitor nodes which are evaluated with each BT update.
- The deeper that we are in the tree, the more monitors we need to update on subsequent updates.
- When an event monitor triggers, it will cause the tree to transition into a branch

TRANSITIONS PROBLEMS

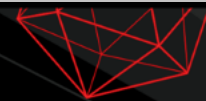


Problem: Static prioritization

- Only way to transitioning from higher to lower priorities is by aborting or completing current behavior
- To **abort** we usually need to sprinkle special case conditions for each lower behavior higher up in the tree
- We end up polluting higher priority behaviors with knowledge of lower priority ones
 - End up increasing complexity of the tree
- We often also need to change priorities based on the context of the current situation

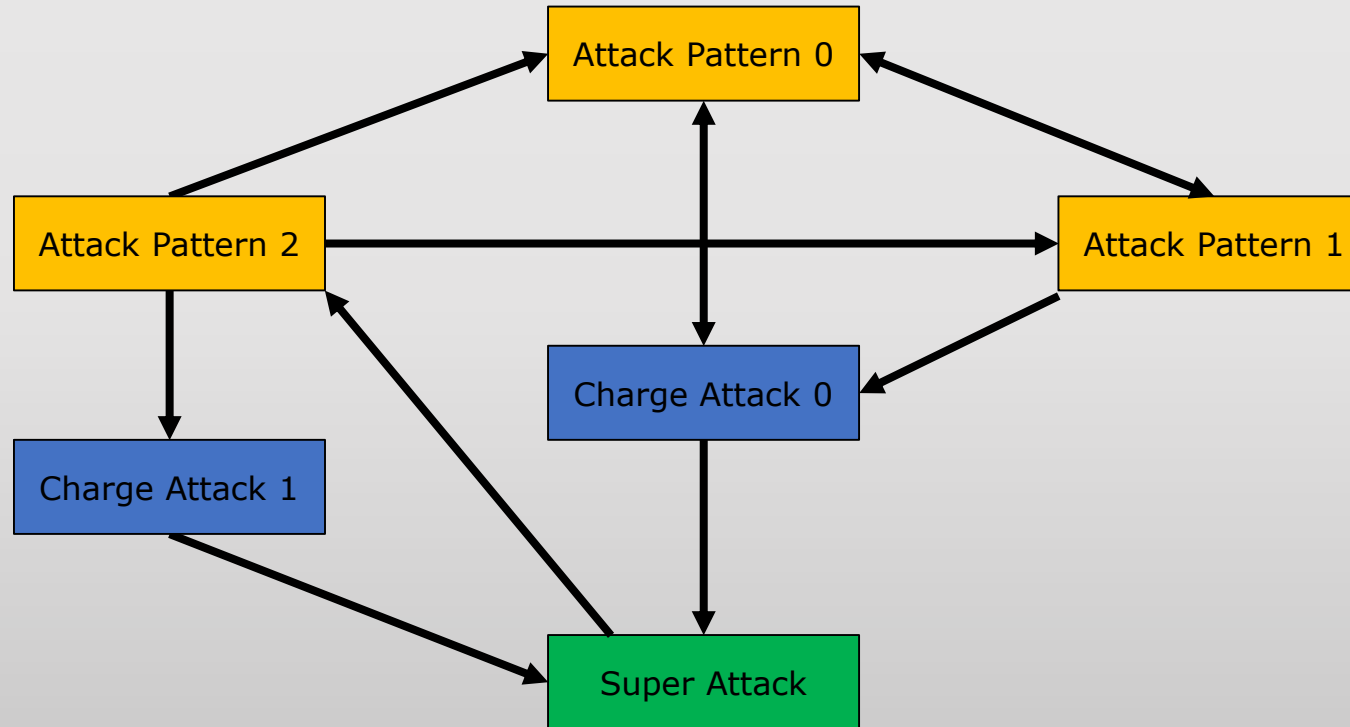
QUESTION TIME

Let me ask you a question!



QUESTION TIME

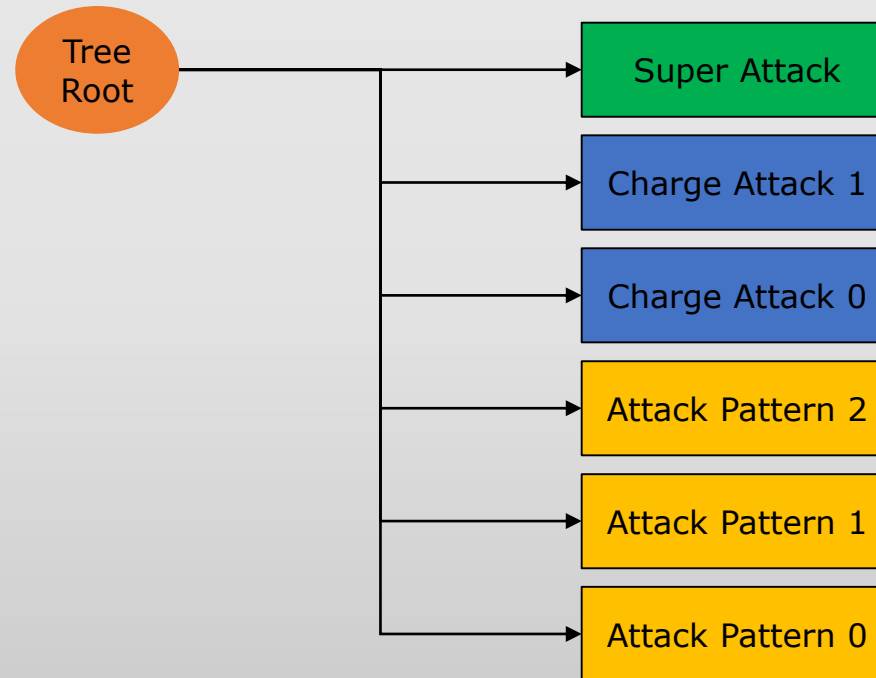
Do you understand this AI design?



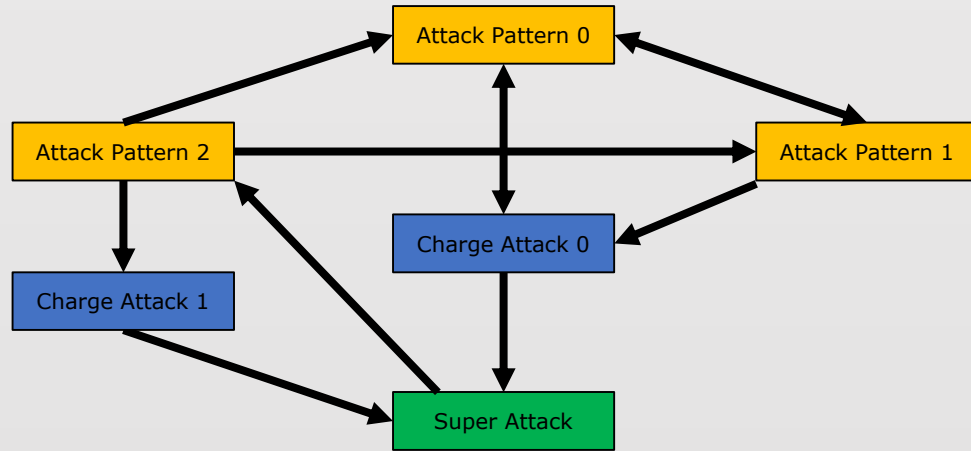
QUESTION TIME

Do you understand this AI design?

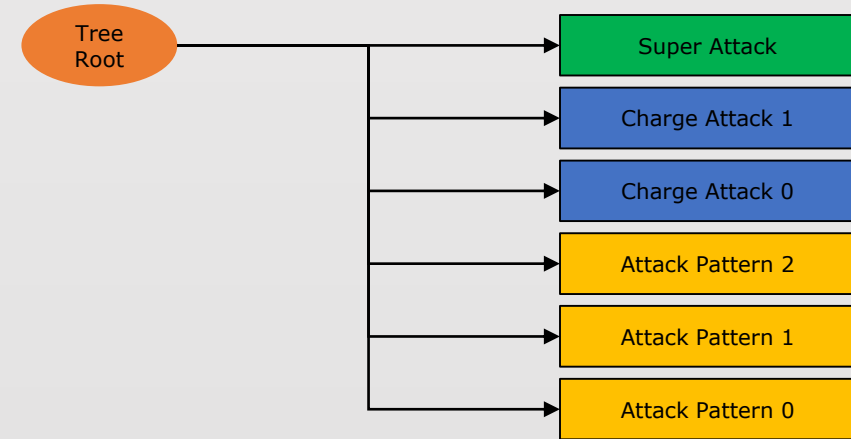
How about now?



BEHAVIOR PRIORITIZATION

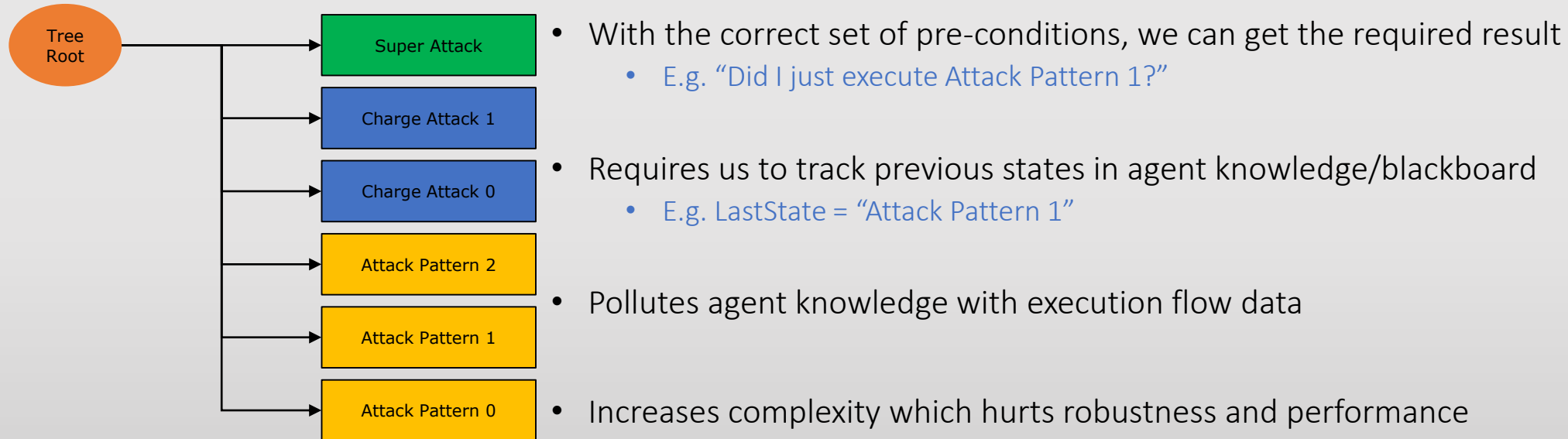


VS.

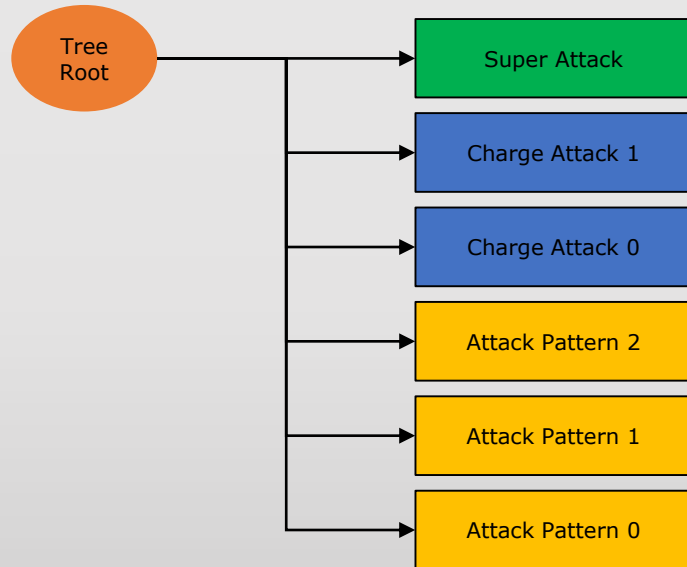


- Behavior Trees are directed **ACYCLIC** graphs
- High Level AI decision making is usually cyclic and contextual
- We are trying to model **cyclic** behavior with an **acyclic** data structure
 - It can be done!
 - Doesn't mean you should!

MODELLING CYCLIC BEHAVIOR



MODELLING CYCLIC BEHAVIOR



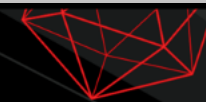
- We are creating **IMPLICIT transitions and dependencies** between the sub-trees and agent data

My new preferred metric for code/systems quality is:
"How easy is it to delete that code/data?"

- How easy is it to delete a behavior?
 - Probably gonna have to touch half the tree and potentially agent data as well

THE COMMON SOLUTIONS

In my experience most solutions end up like this:

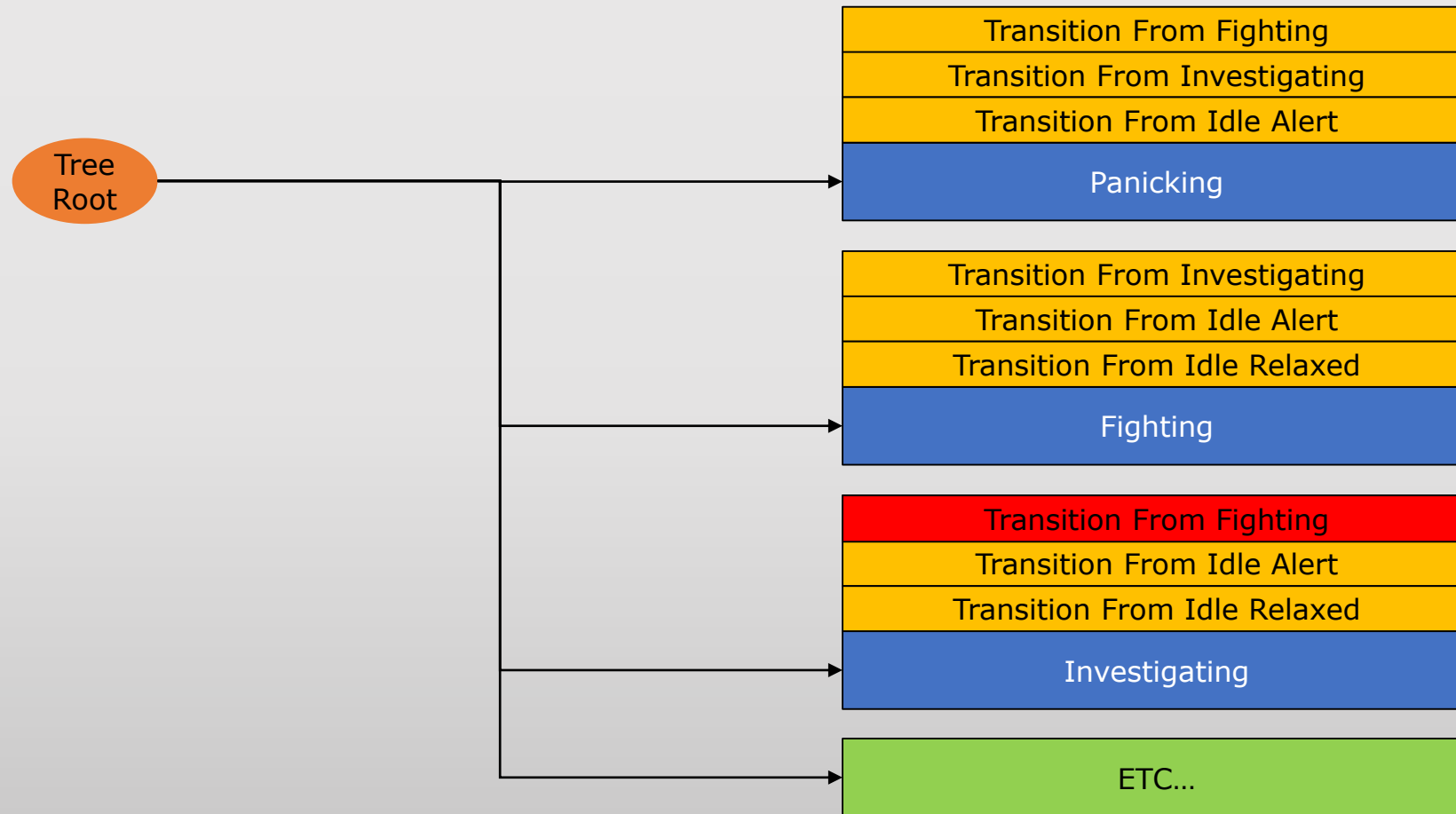


TRANSITION BEHAVIORS

- Another issue is that we often need to visually signal large behavior state changes.
 - E.g. Startled reaction to take us from idle to investigating
- Simplest case, we can just play an animation + a sound
- Sometimes we need to execute a whole transition behavior
 - E.g. Turn to face event + change stance + draw weapon + play specific anim, etc...
- These reactions and behaviors are placed into the tree at various points



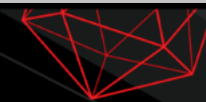
TRANSITION BEHAVIORS



WAIT, WHAT?



“Okay... so what you’re saying is behaviors trees are **AWFUL** and we shouldn’t use them?!”



BEHAVIOR TREES AT THEIR BEST

- A **behavior tree** is a mathematical **model** of **plan execution** (Wikipedia)

Plan

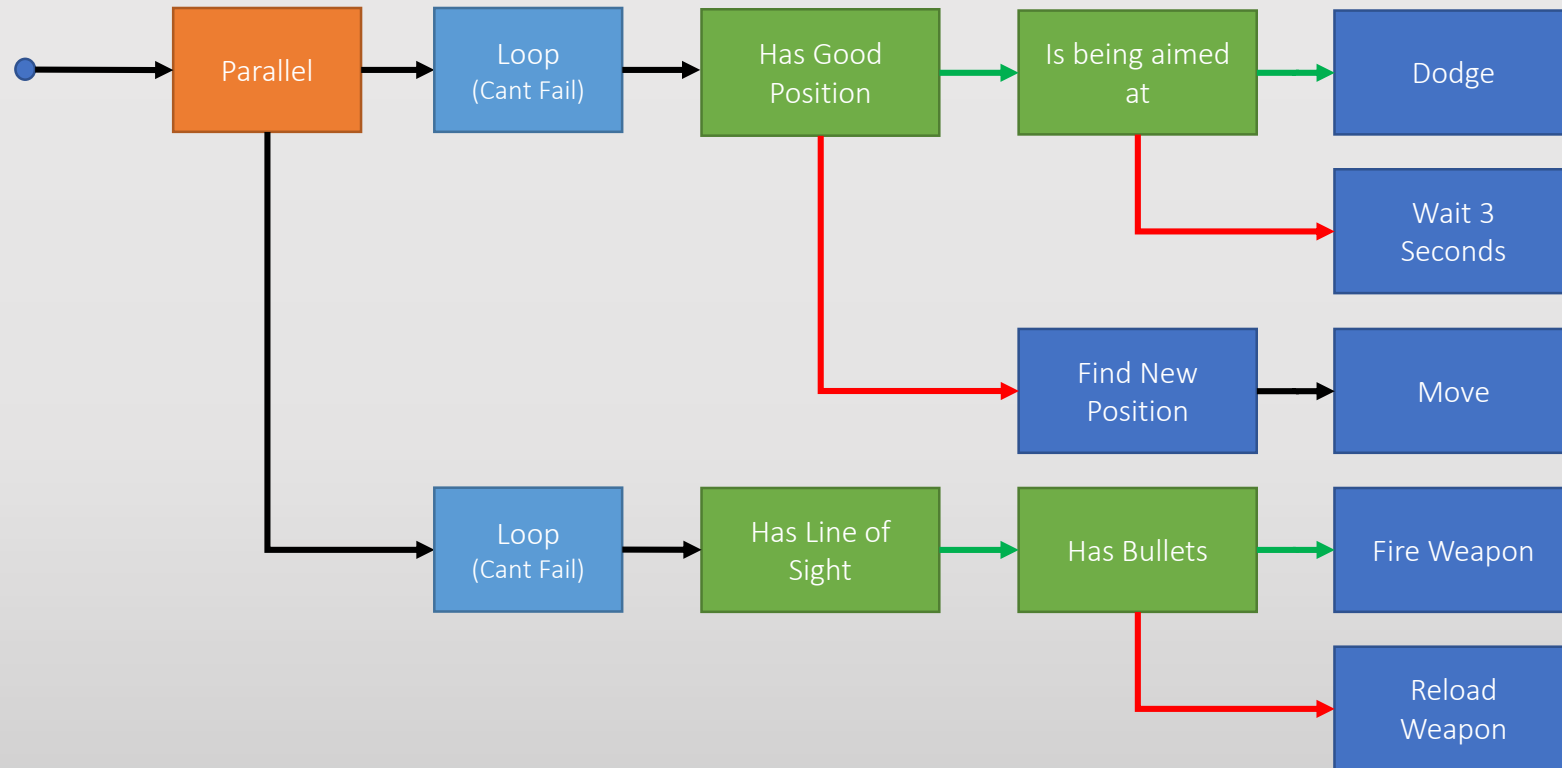
Execution

Model

- Implies that a plan already exists!
- BTs are **excellent** for describing a complex sequence of actions to perform

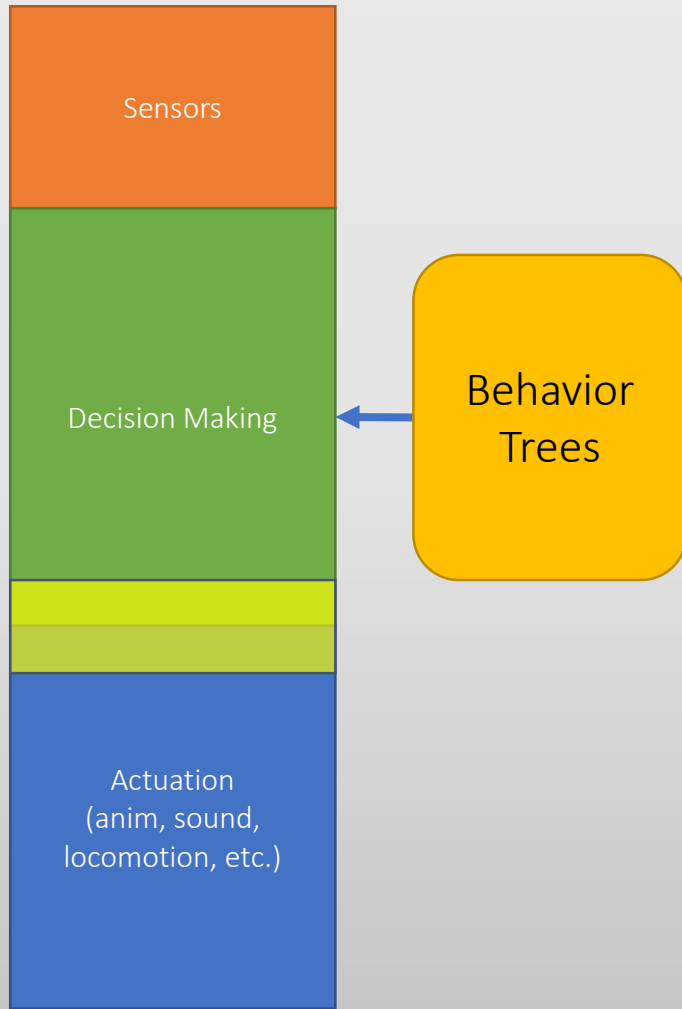


BEHAVIOR TREES AT THEIR BEST



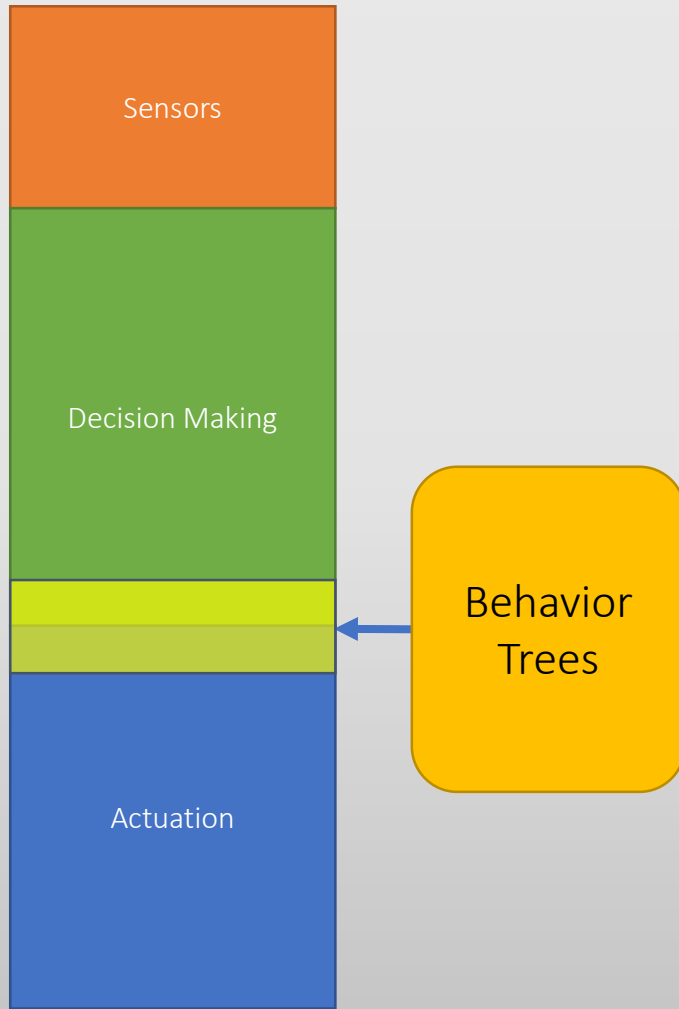
Question: What does this behavior do?

BEHAVIOR TREES AT THEIR BEST



- I like to think of behaviors tree as in between decision making and actuation
 - For me, a BT is not a decision making tool
- It's acts as control layer between high level decision making and low level actuation/execution.
- We can still make some decisions in the trees but those decisions don't change the overall goal or agent intentions
 - E.g. In prev example: if I don't have LOS, find a position with LOS

BEHAVIOR TREES AT THEIR BEST



- Requires a higher granularity of actions in behavior trees
 - E.g. Move, look-at, reload, fire, etc.
- Simpler and smaller trees, each executing a single clear goal
 - E.g. Fight from Cover, Investigate Sound
- We still make some decisions in the trees but only decisions needed for executing that specific behavior
 - **Previous Example:** “Check whether we have a good position”

BEHAVIOR TREES AT THEIR BEST

Moving to a simpler execution-oriented approach has several benefits:

- Easy to build
- Easy to understand and debug
- Easy to identify the working data set for the tree
- Easy to test independently
- Perform better

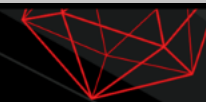


BEHAVIOR TREES AT THEIR BEST



My Authoring Rule of Thumb

“I’ve made a decision, what are the steps I need to execute it”



WAIT, WHAT?



“Okay... So don’t use behavior trees for the decision making... Then what should I do?”



LEVERAGE OTHER TECHNIQUES

- In game AI, we are spoiled for choice when it comes to techniques:
 - State Machines
 - Planners
 - Utility systems
 - Etc...

THAT DOESN'T MEAN YOU ONLY HAVE TO CHOOSE ONE!

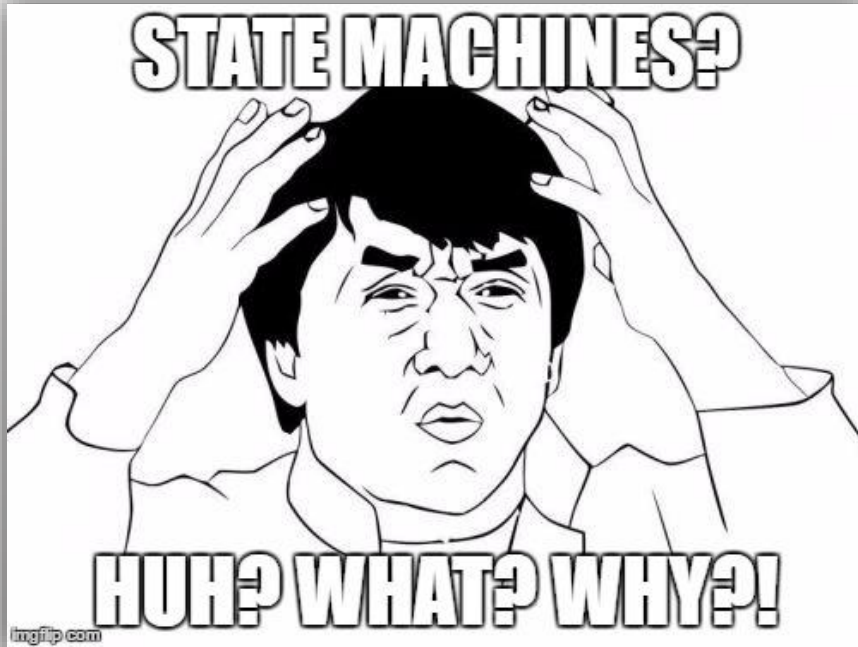


EXAMPLE TIME

What I've found to be an elegant approach to agent AI is the following:

A Hierarchical Finite State Machine Behavior Tree Hybrid

(HFSMBTH for short)



STATE MACHINE / BT HYBRID

- State machines are excellent for describing states and transitions
 - Easy to visualize, author and understand
 - Bad for defining sequence of actions
- Behavior trees are excellent for describing sequences of actions
 - Allow for parallel execution of actions
 - Bad for defining transitions and reactions
- Both techniques are proven and well understood
 - Low risk / high reward

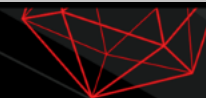


STATE MACHINE / BT HYBRID

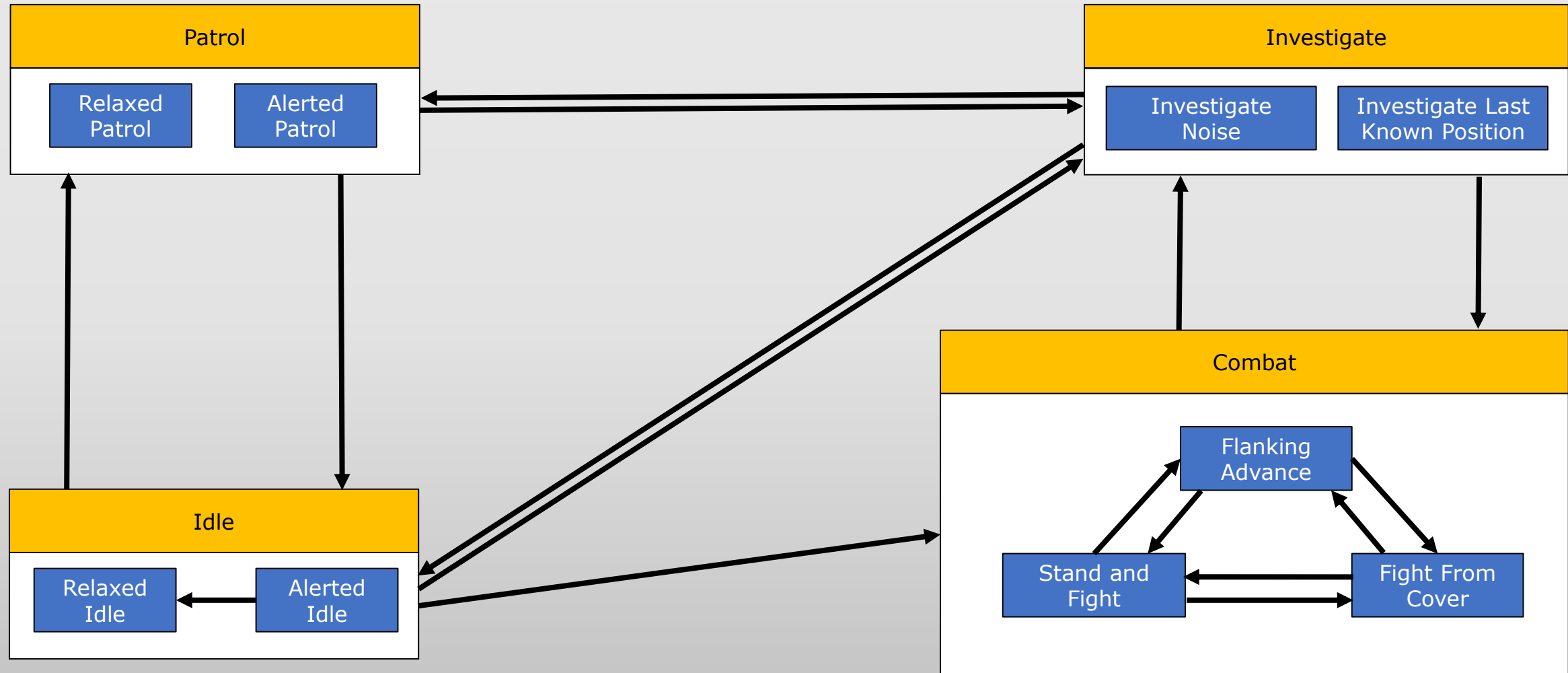
Both techniques have **problems...**



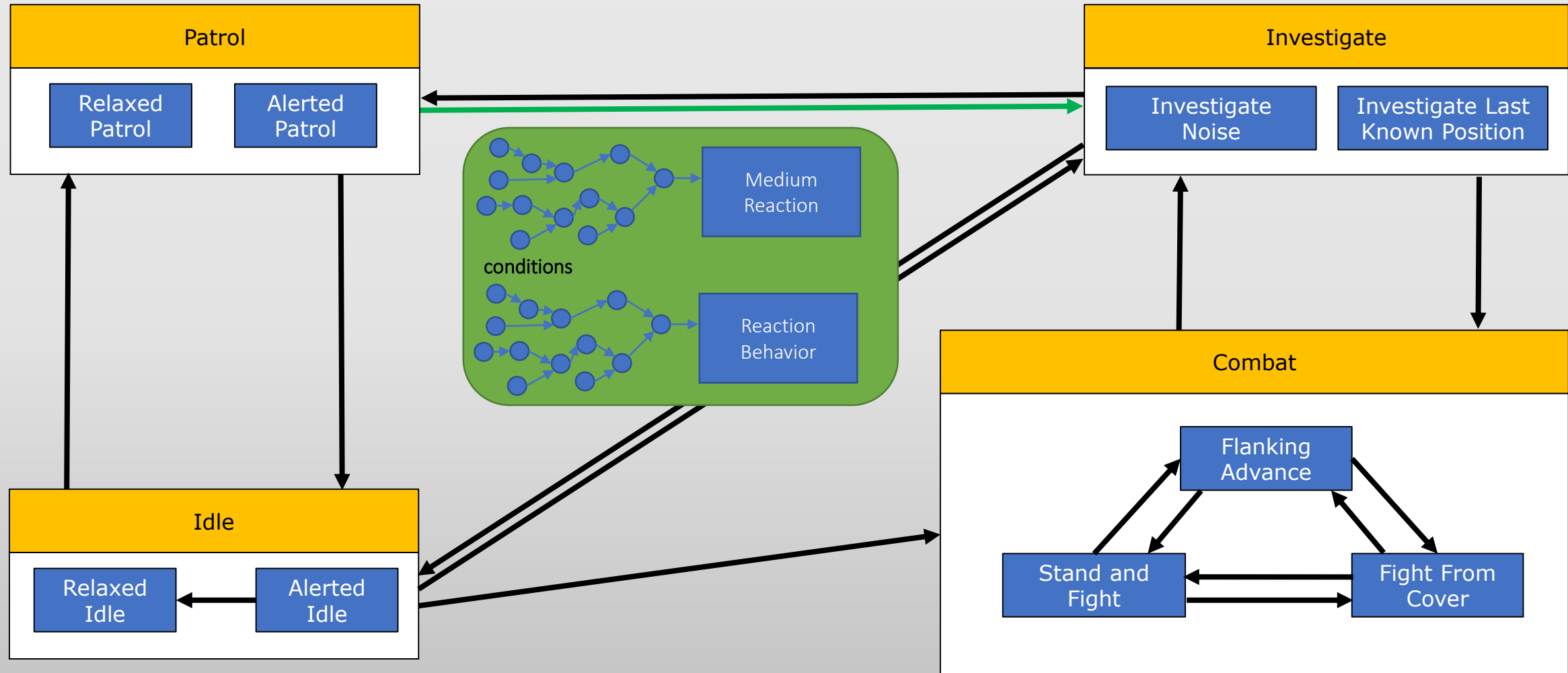
Together, they are more than the sum of their parts



EXAMPLE OF THE MYTHICAL HFSMBTH



EXAMPLE OF THE MYTHICAL HFSMBTH



DON'T MISS THE POINT



- Now don't get me wrong!
 - HFSMBTH isn't perfect
 - There are some tricks regarding hierarchies and entry states that are needed
 - [If you want more details on this approach chat to me afterwards](#)



The point isn't about the state machines

- We could have just as easily use a planner or some other technique
 - *E.g. BTs could be a good fit to describe tasks in an HTN planner*
- My point is that BTs are not an ideal technique for AI decision making
 - *We are abusing them and scaring new people away*
 - *We are struggling to implement simple concepts and drowning in our own complexity*



CONCLUSION



- Behavior Trees are excellent at describing individual behaviors
- Use them to define execution of decisions
- Keep them as small and as simple as possible
- Combine them with other techniques to get the most out of them



- Behavior Trees are pretty bad with interruptions/transitions
- Stop building cyclic behavior with an acyclic data structure
- Stop building large monolithic trees with multiple responsibilities
- Stop using them for building your AI's decision making logic





BEN WEBER
TWITCH

Overview

1. Additional Node Types
2. Design Patterns
3. Integrating other Architectures

EXTENDING BEHAVIOR TREES

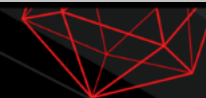
- Additional node types
 - Spawn goal
 - Working memory modifiers
 - Success test
- Issues
 - Dynamic behavior lookup
 - Task scheduling
 - Resource contention



BT DESIGN PATTERNS

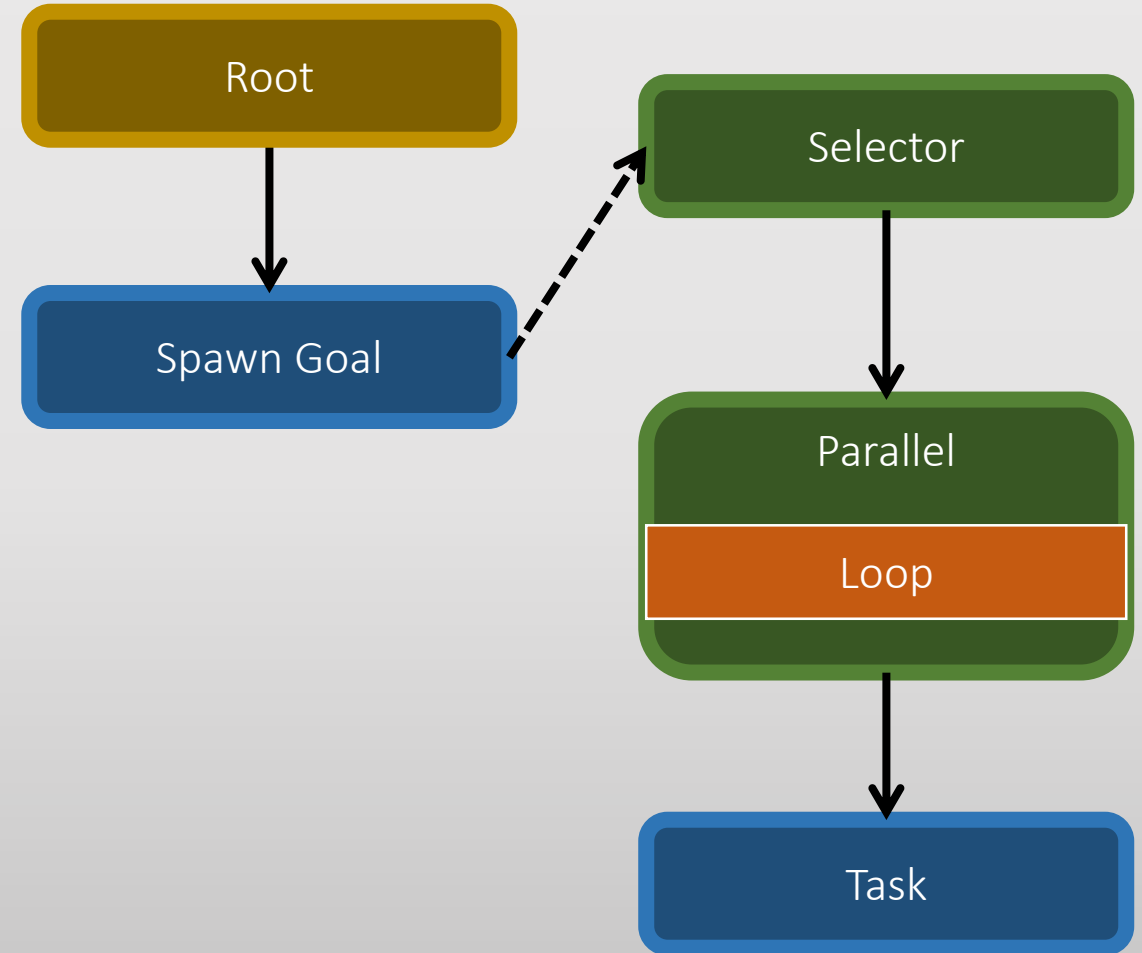
- Patterns in EISBot

- Daemon behaviors
- Managers
- Message passing
- Behavior locking
- Unit subtasks



DAEMON BEHAVIORS

- Enable concurrent goal pursuit
- Spawn a new goal which is persistently pursued by the agent

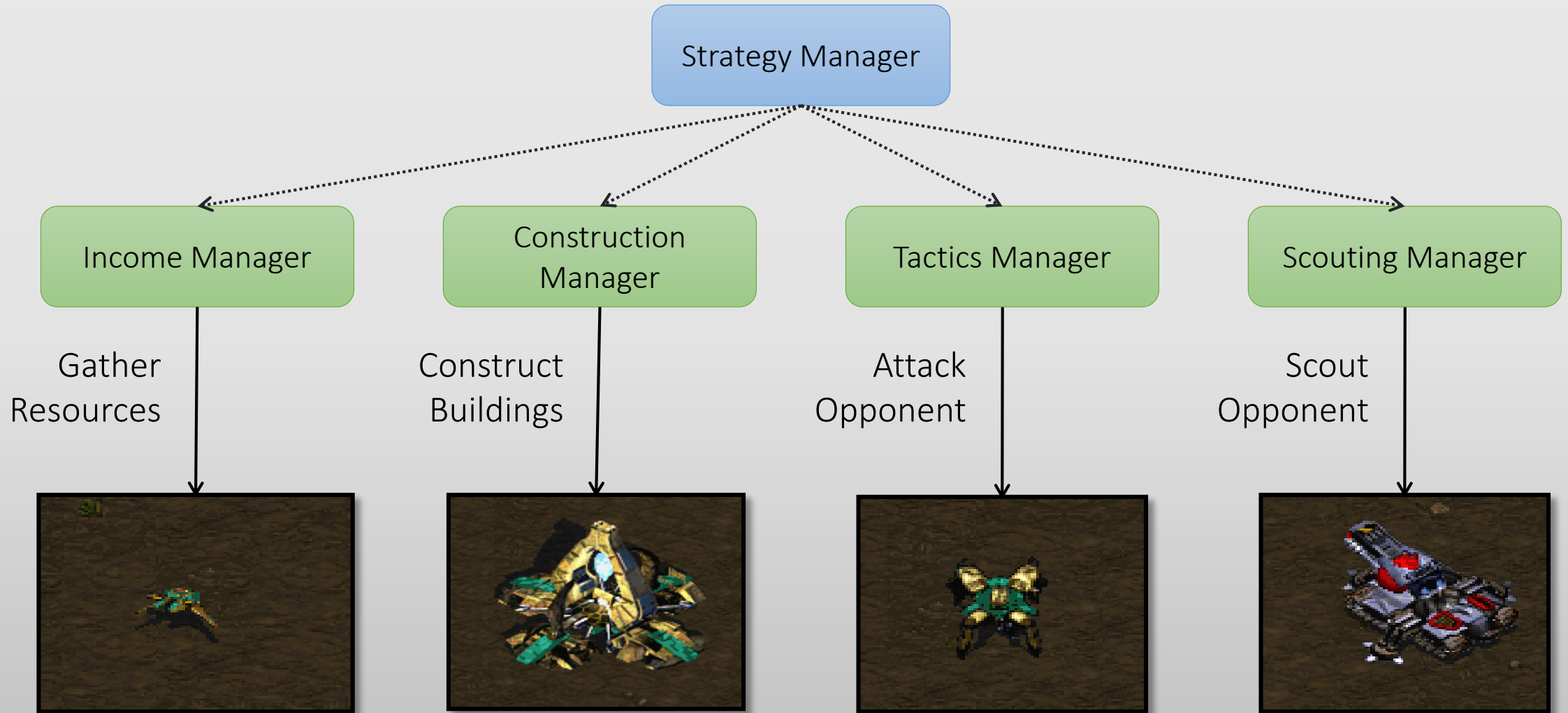


EXAMPLE DEAMON BEHAVIOR

```
initial_tree {  
    spawngoal restockUnits();  
  
    subgoal createManagers()  
}  
  
parallel behavior restockUnits() {  
    with (persistent) subgoal trainInterceptors();  
    with (persistent) subgoal trainScarabs();  
}
```



EISBOT MANAGERS



EXAMPLE MANAGER

```
parallel behavior incomeManager() {  
  
    // production  
    with (persistent) subgoal pumpProbes();  
    with (persistent) subgoal buildAssimilators();  
    with (persistent) subgoal processExpansionRequests();  
  
    // harvesting  
    with (persistent) subgoal mineMinerals();  
    with (persistent) subgoal putWorkersOnGas();  
    with (persistent) subgoal pullWorkersOffGas();  
}
```



MESSAGE PASSING

```
sequential behavior messageProducer() {  
  mental_act {  
    BehavingEntity.addWME(new MessageWME());  
  }  
}
```

```
sequential behavior messageConsumer() {  
  precondition { message = (MessageWME) }  
  
  subgoal processMessage(message);  
  mental_act {  
    BehavingEntity.deleteWME(message);  
  }  
}
```



BEHAVIOR LOCKING

- Using the blackboard to suspend the execution of behaviors

```
sequential behavior putWorkersOnGas() {  
    AssimilatorWME assimilator;  
  
    with (success_test {  
        !(GasHoldWME)  
        assimilator = (AssimilatorWME)  
    }) wait;  
  
    subgoal assignWorkers(assimilator);  
}
```

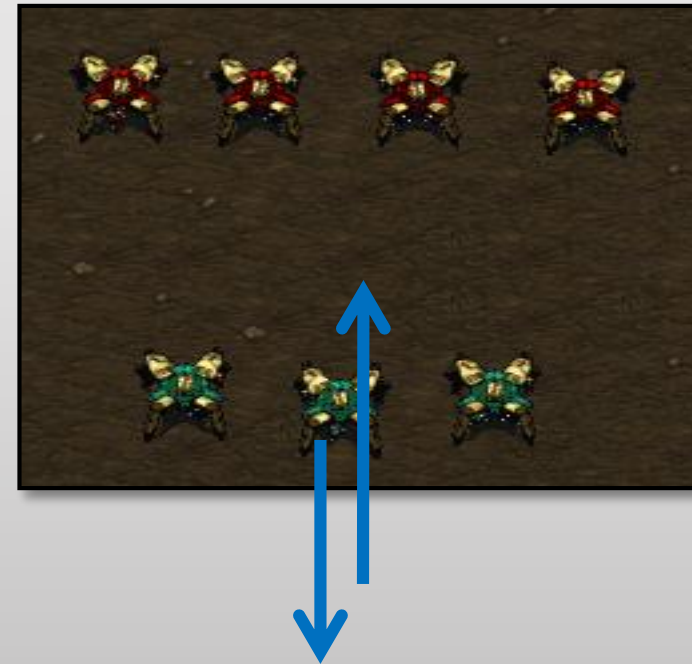


UNIT SUBTASKS

Behaviors that temporarily claim a unit to perform a task

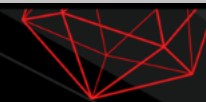
EISBot sub tasks

- Micromanagement
- Worker defense
- Building construction

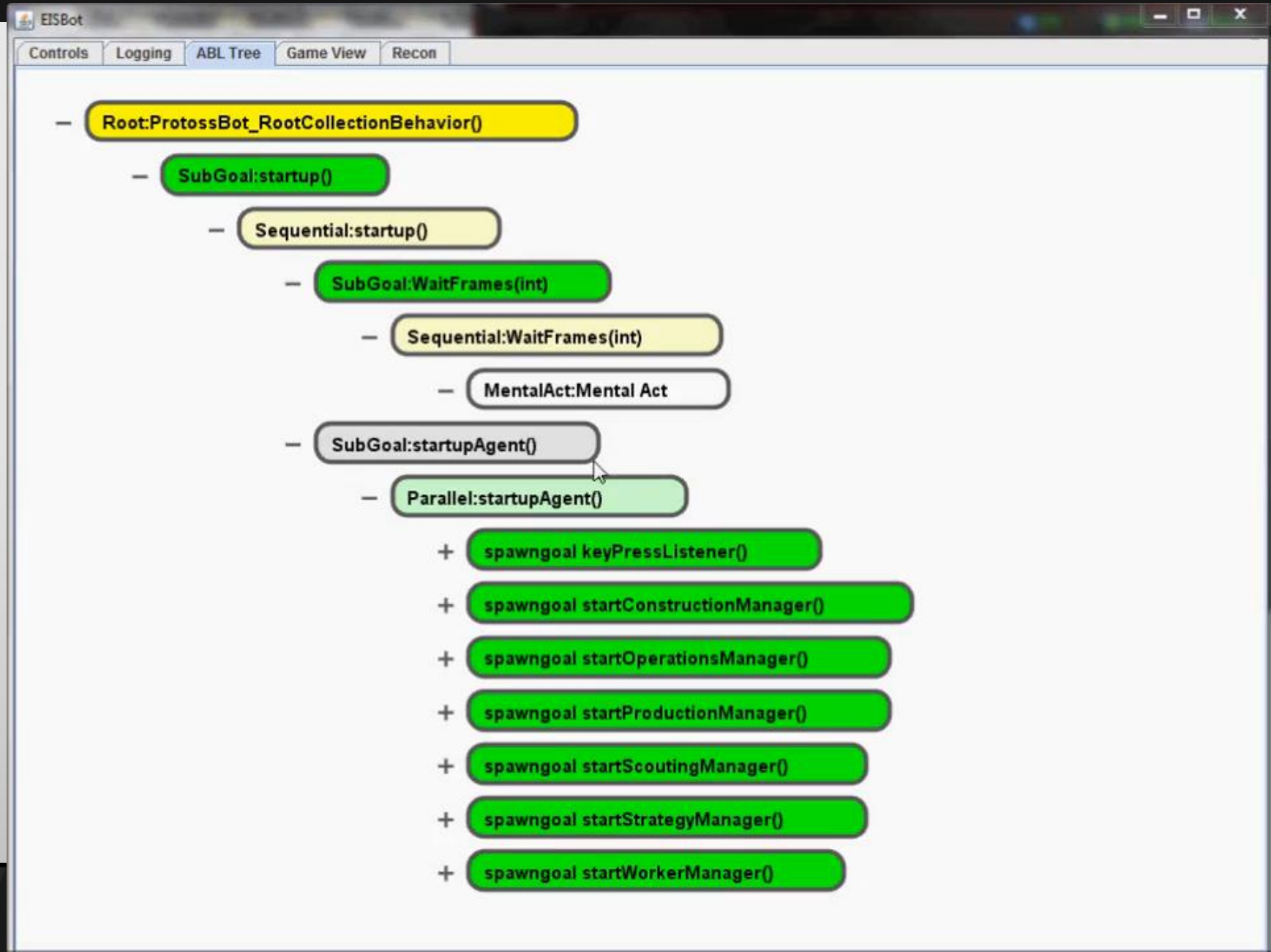


EXAMPLE UNIT SUBTASK

```
sequential behavior dragoonDance() {  
    DragoonWME unit;  
  
    with (success_test {  
        unit = (DragoonWME damaged==true task!=FIGHTER_FLEE)  
    }) wait;  
  
    mental_act {  
        unit.setTask(FIGHTER_FLEE);  
    }  
  
    spawngoal dragoonFlee(unit);  
}
```

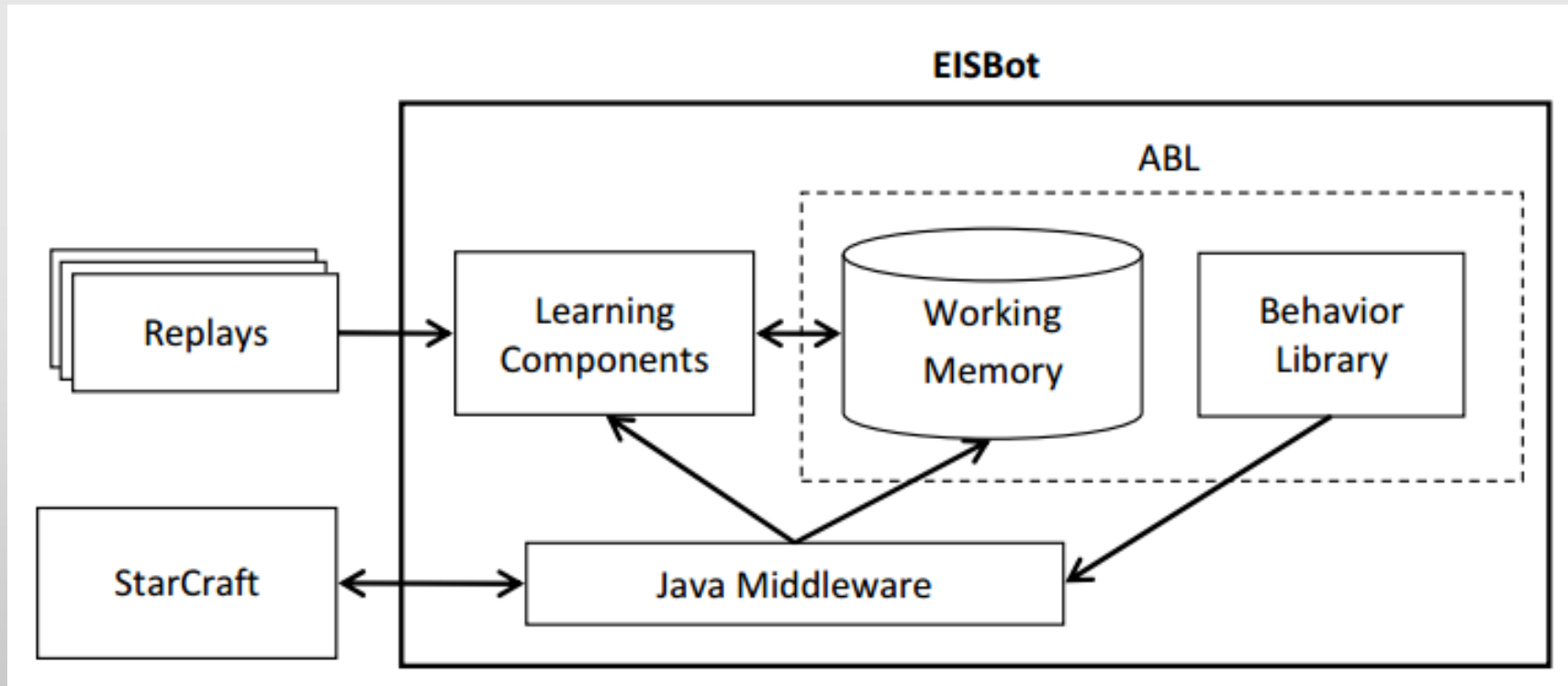


EISBOT VIDEO



AUGMENTING BEHAVIOR TREES

- How can we interface behavior trees with other architectures?



EXTERNAL PLAN GENERATION

- Creating sequential or parallel behaviors during runtime

Replay File

Warp Pylon

Train probe

Warp Gateway

Train Probe

Warp Assimilator

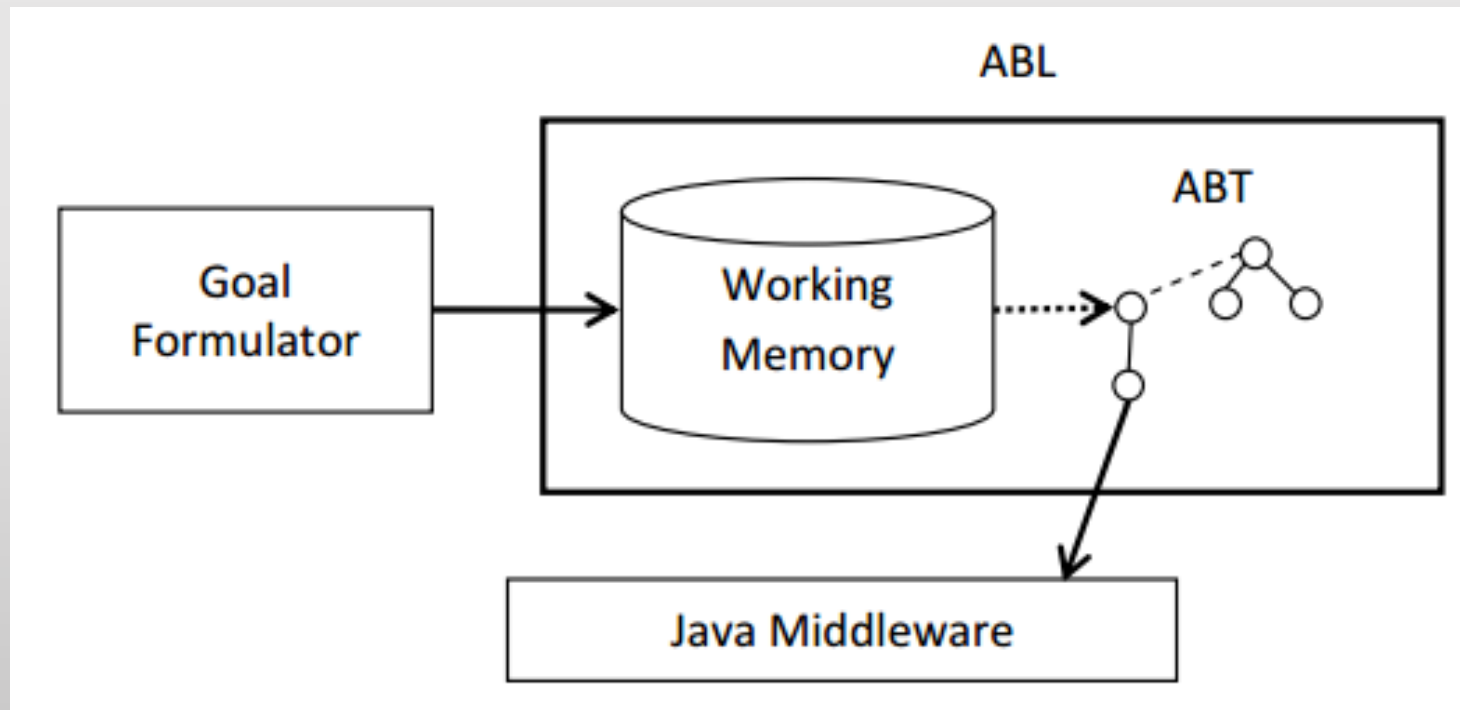


```
sequential behavior buildOrder() {  
    subgoal buildGateway()  
    subgoal buildAssimilator()  
    subgoal buildCyberneticsCore()  
}
```



EXTERNAL GOAL FORMULATION

- Creating new goals or behavior trees at runtime



BEYOND BEHAVIOR TREES

- Joint goals and behaviors
- Meta-behaviors
- Partial programming





WE'RE HIRING!!!



Twitch is hiring!

<http://www.twitch.tv/jobs>



Remedy is hiring!

<http://www.remedygames.com/careers/>

THANK YOU!



- Remedy is hiring!
<http://www.remedygames.com/careers/>



- Twitch is hiring!
<http://www.twitch.tv/jobs>

