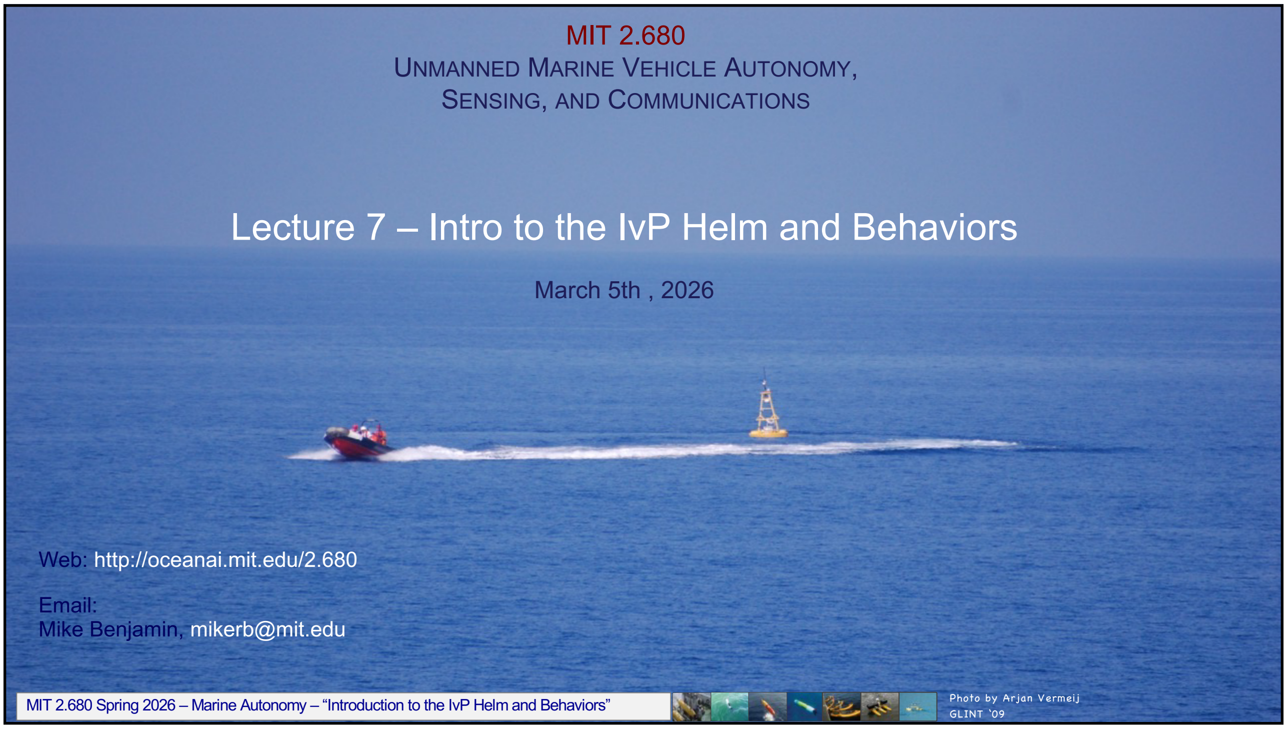


MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 7 – Intro to the IvP Helm and Behaviors

March 5th , 2026

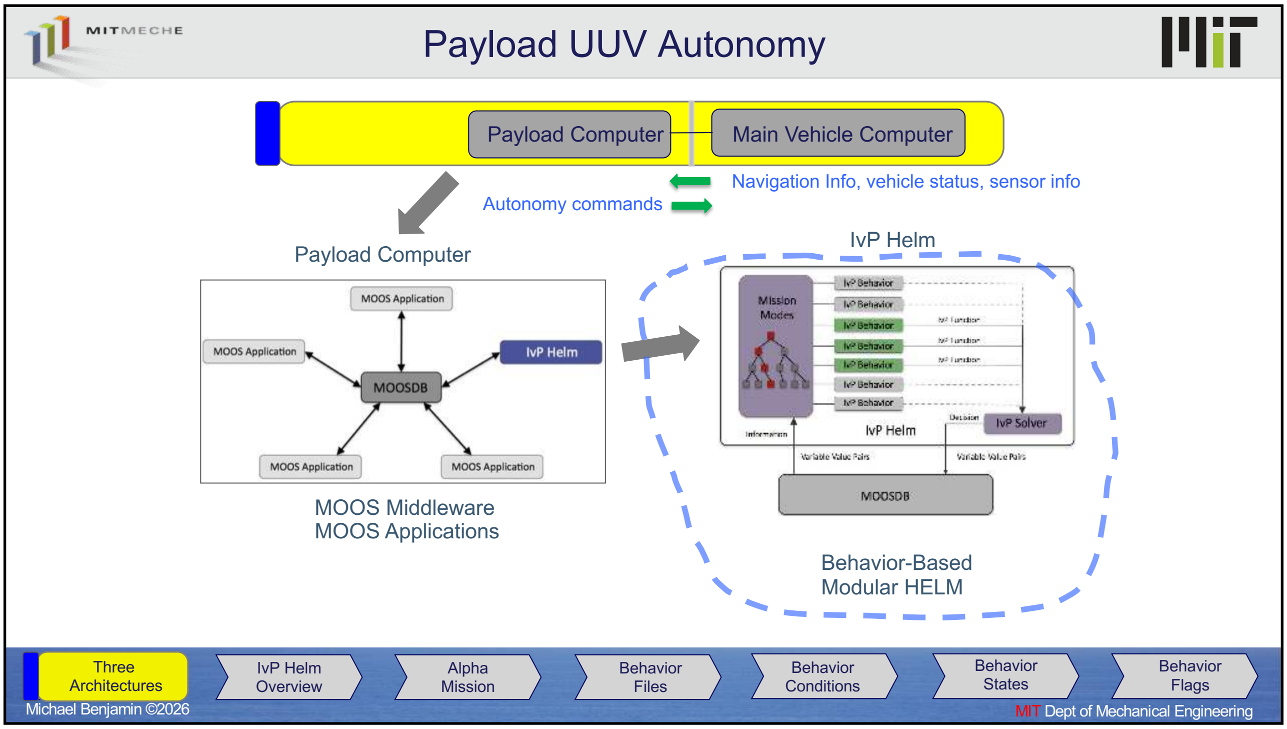
Web: http://oceanai.mit.edu/2.680

Email:
Mike Benjamin, mikerb@mit.edu

MIT 2.680 Spring 2026 – Marine Autonomy – "Introduction to the IvP Helm and Behaviors"

Photo by Arjan Vermeij
GLINT '09

1



MITMECHE

Payload UUV Autonomy

Payload Computer

Main Vehicle Computer

Navigation Info, vehicle status, sensor info

Autonomy commands

Payload Computer

MOOS Middleware
MOOS Applications

IvP Helm

Behavior-Based
Modular HELM

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

2

MITMECHE

IvP Helm Overview

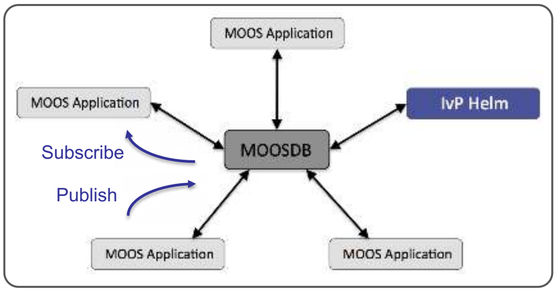
- Behavior Files
- Behavior Conditions
- Behavior States
- Behavior Flags

3

MITMECHE

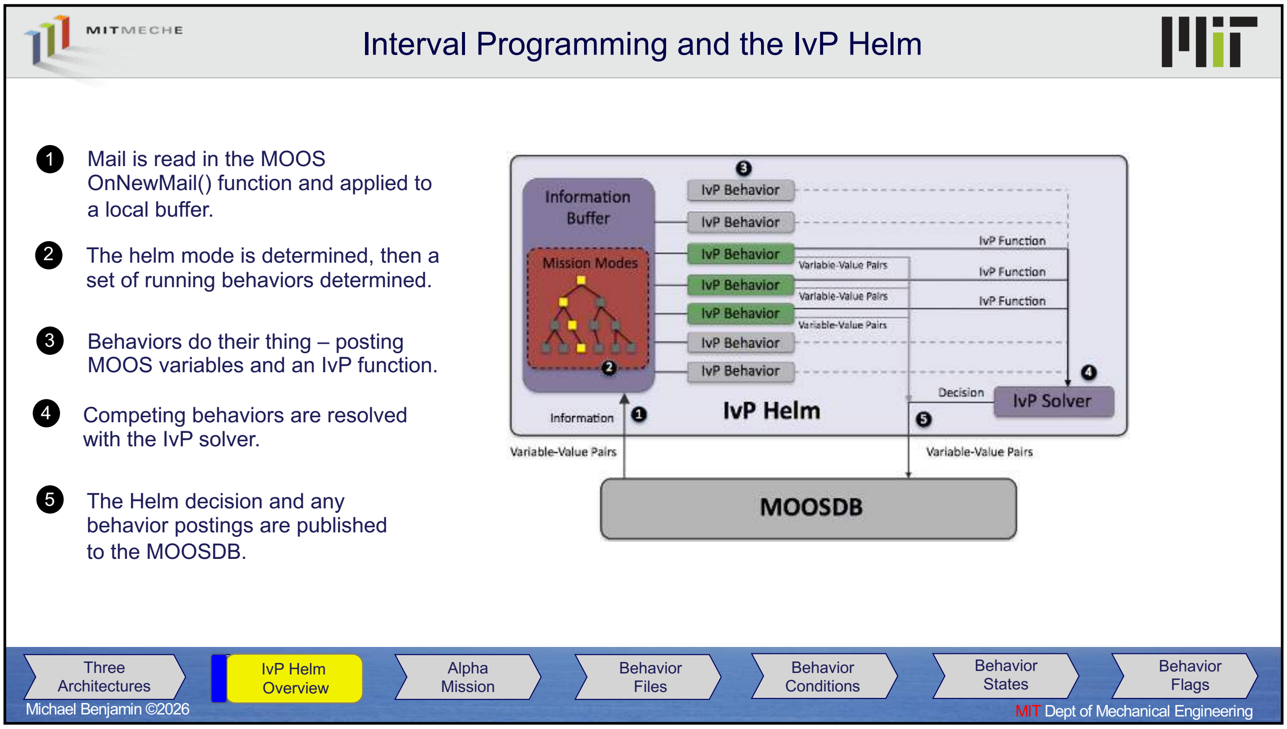
The IvP Helm

- The IvP Helm is a MOOS App, known as pHelmIvP
- The IvP Helm works with other MOOS Apps, performing sensor-processing, planning, communications.



A MOOS Community

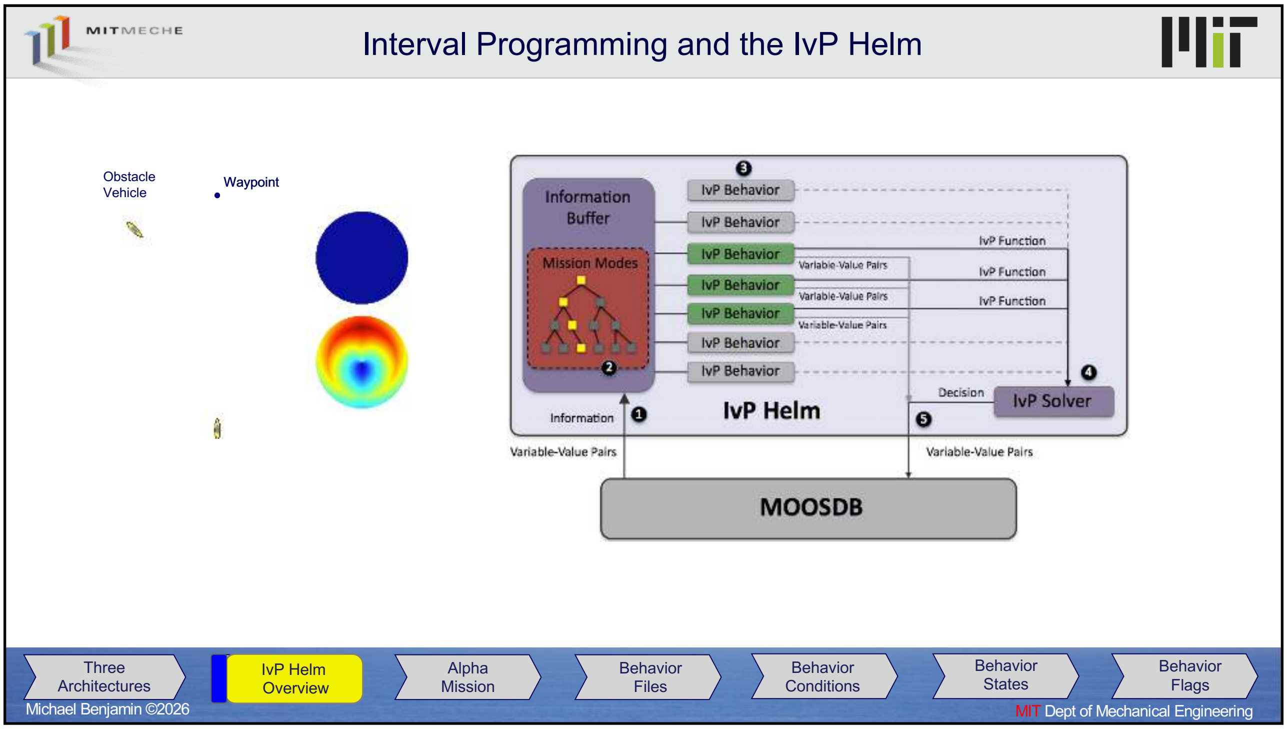
4



Interval Programming and the IvP Helm

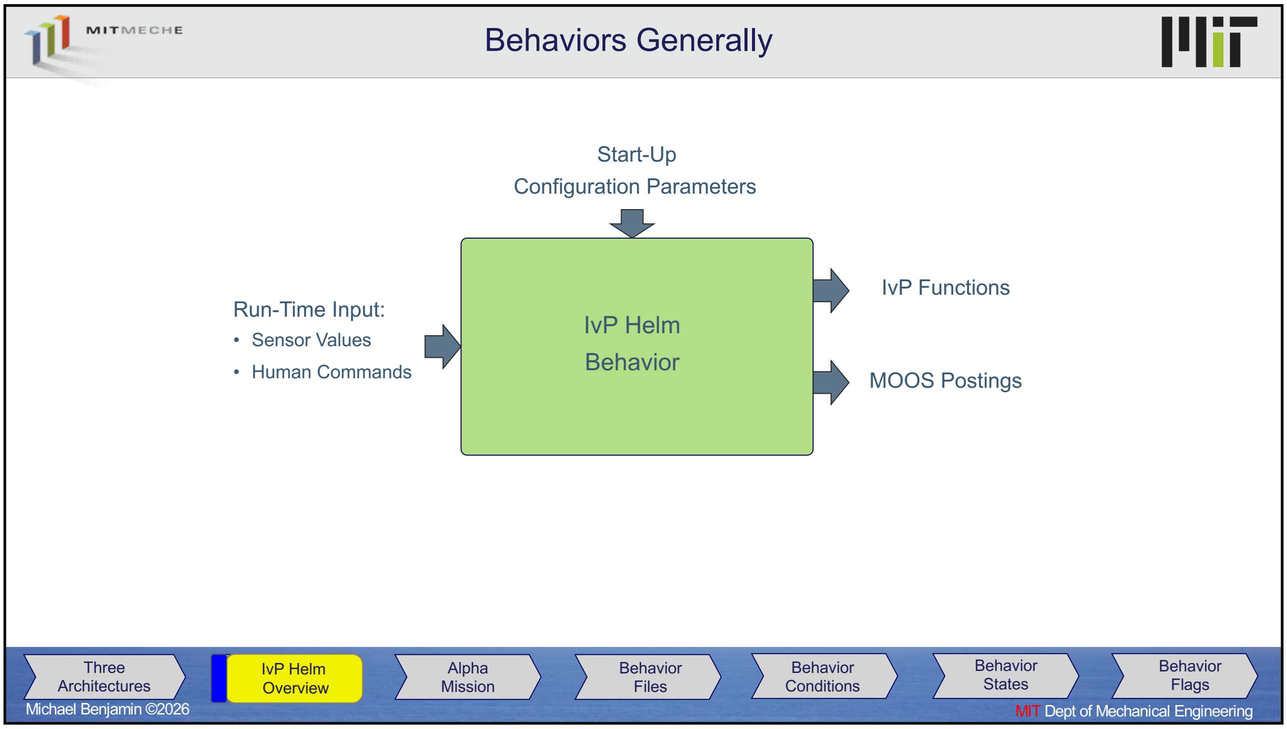
1. Mail is read in the MOOS OnNewMail() function and applied to a local buffer.
2. The helm mode is determined, then a set of running behaviors determined.
3. Behaviors do their thing – posting MOOS variables and an IvP function.
4. Competing behaviors are resolved with the IvP solver.
5. The Helm decision and any behavior postings are published to the MOOSDB.

5



Interval Programming and the IvP Helm

6



MITMECHE

Behaviors Generally

Start-Up
Configuration Parameters

Run-Time Input:
- Sensor Values
- Human Commands

IvP Helm
Behavior

IvP Functions

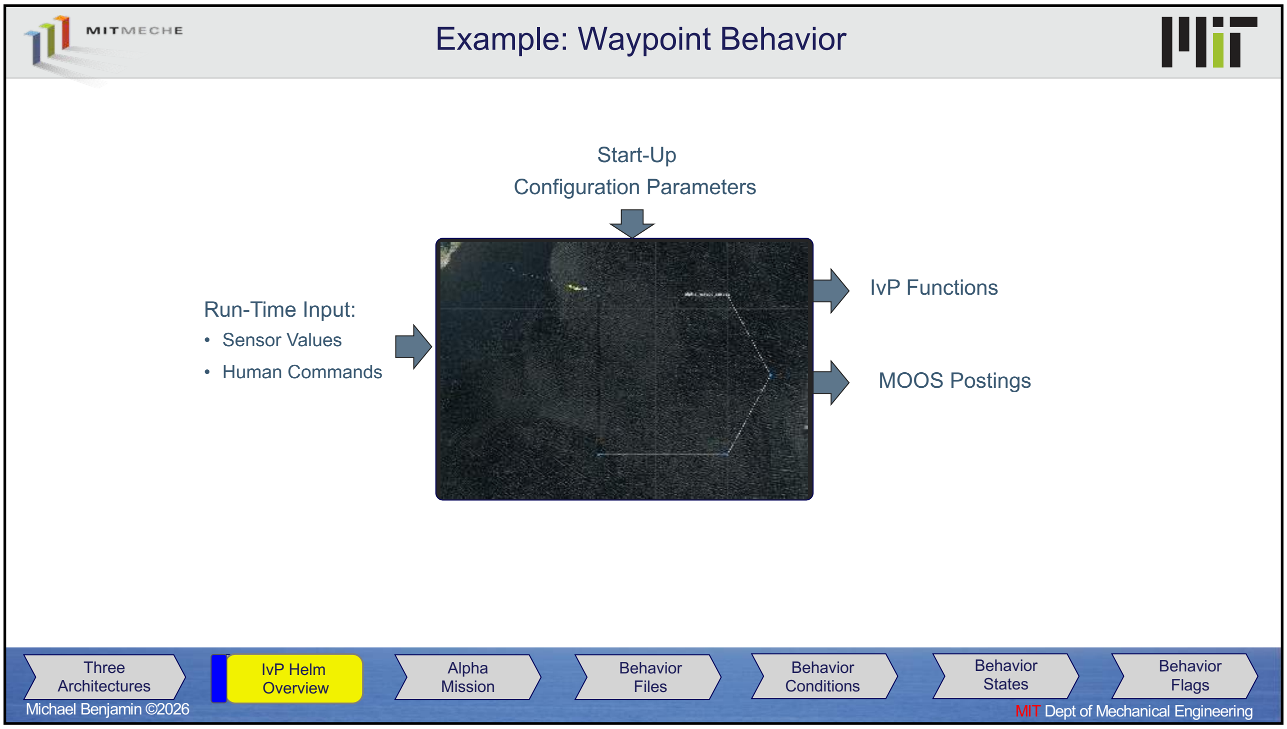
MOOS Postings

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

7



MITMECHE

Example: Waypoint Behavior

Start-Up
Configuration Parameters

Run-Time Input:
- Sensor Values
- Human Commands

IvP Functions

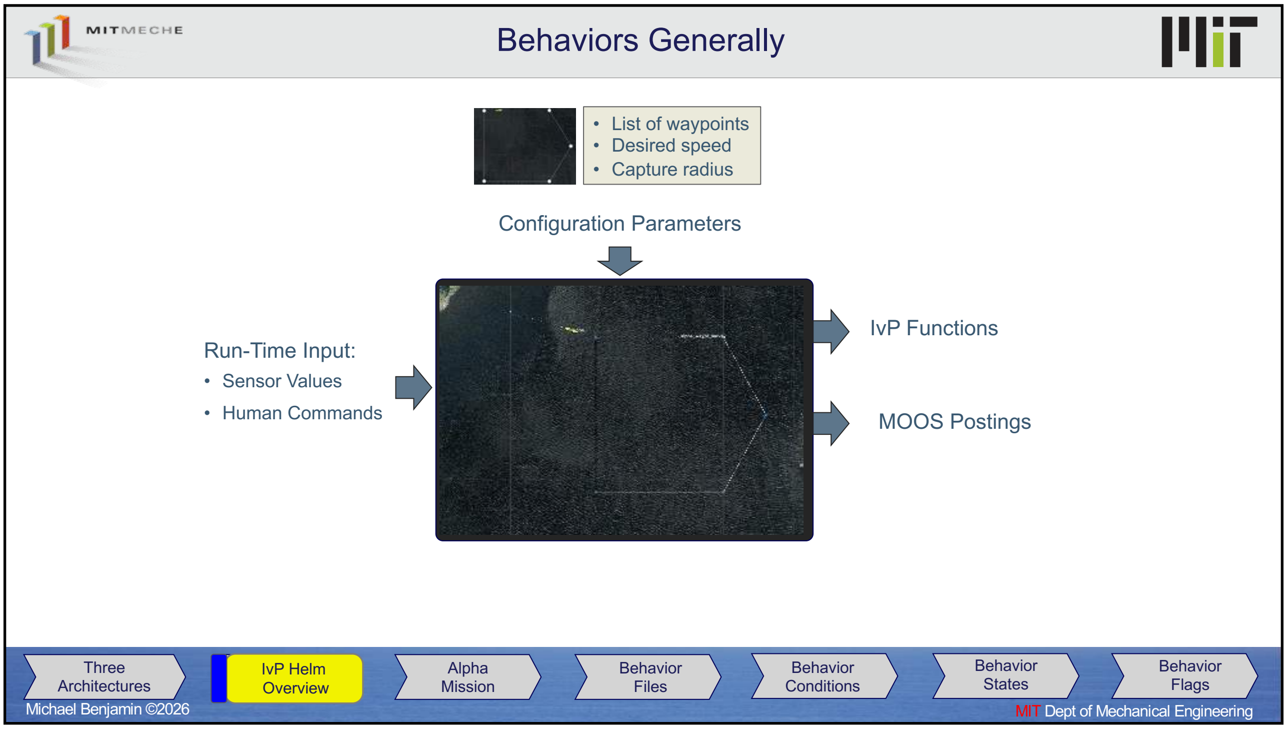
MOOS Postings

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

8



Behaviors Generally

- List of waypoints
- Desired speed
- Capture radius

Configuration Parameters

Run-Time Input:

- Sensor Values
- Human Commands

IvP Functions

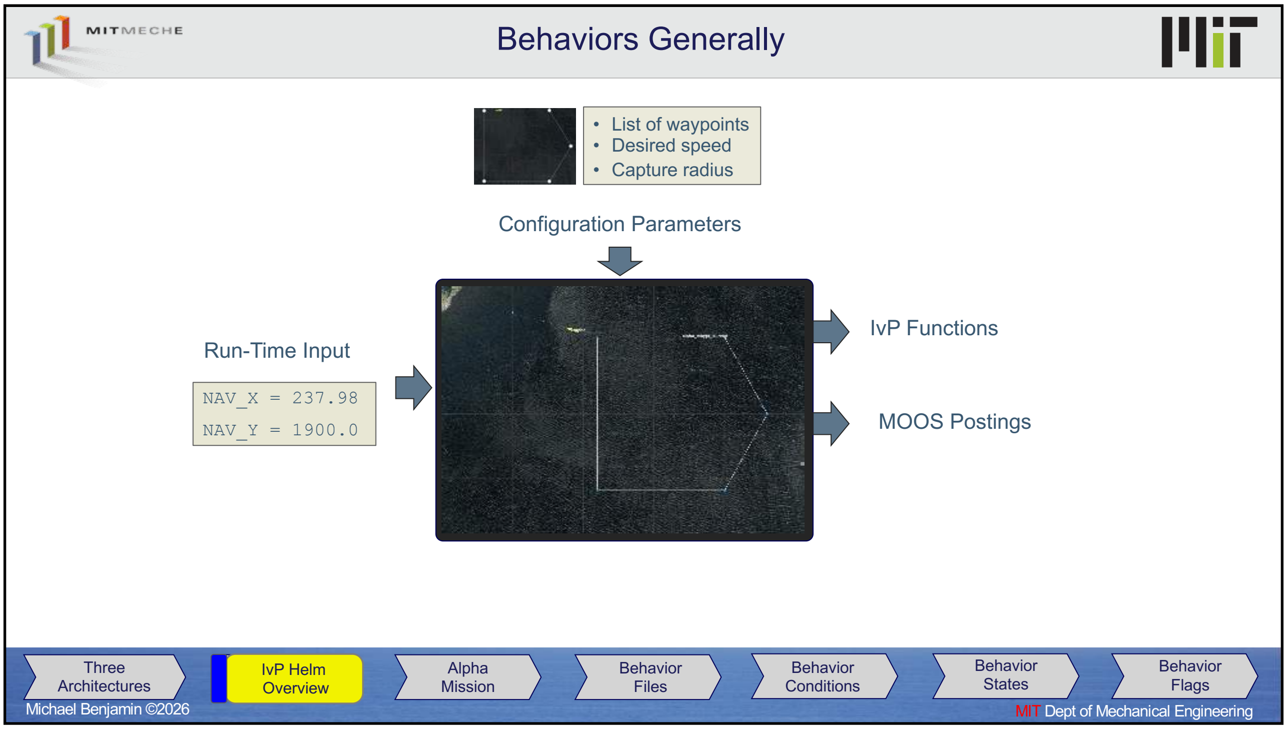
MOOS Postings

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

9



Behaviors Generally

- List of waypoints
- Desired speed
- Capture radius

Configuration Parameters

Run-Time Input

```
NAV_X = 237.98
NAV_Y = 1900.0
```

IvP Functions

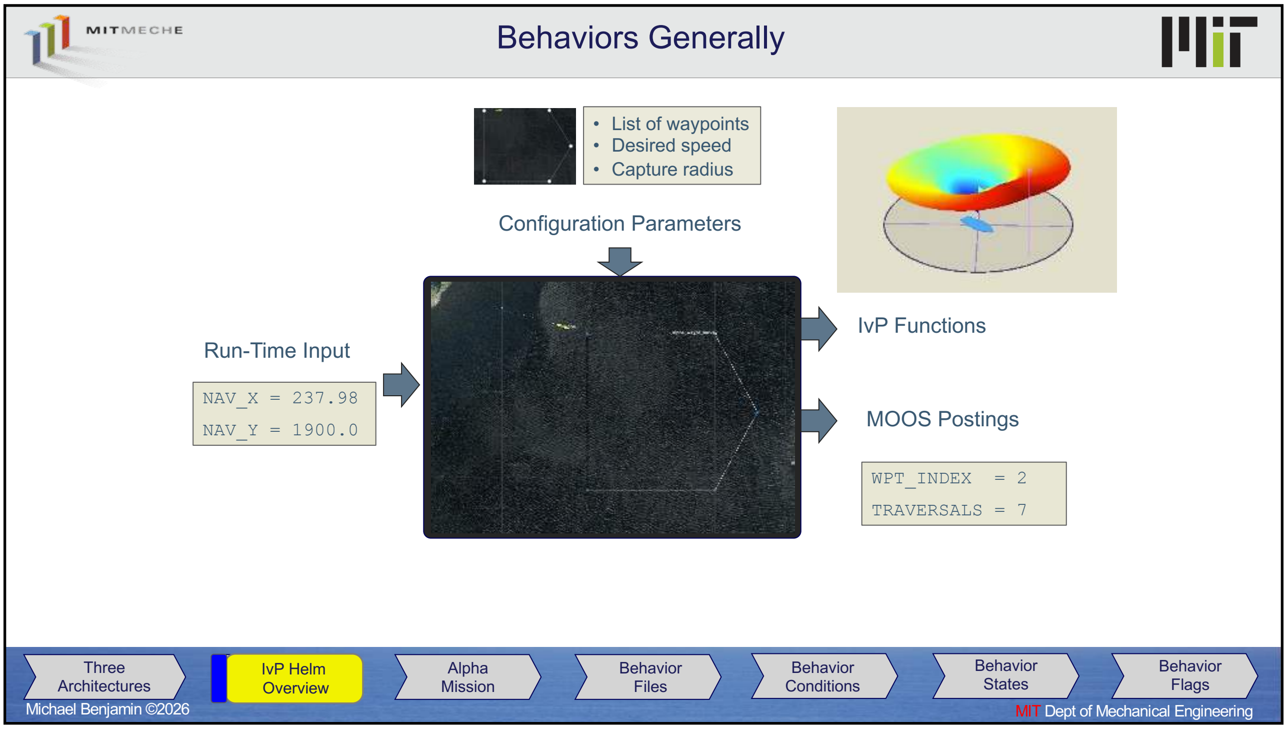
MOOS Postings

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

10



Behaviors Generally

- List of waypoints
- Desired speed
- Capture radius

Configuration Parameters

Run-Time Input

```
NAV_X = 237.98
NAV_Y = 1900.0
```

IvP Functions

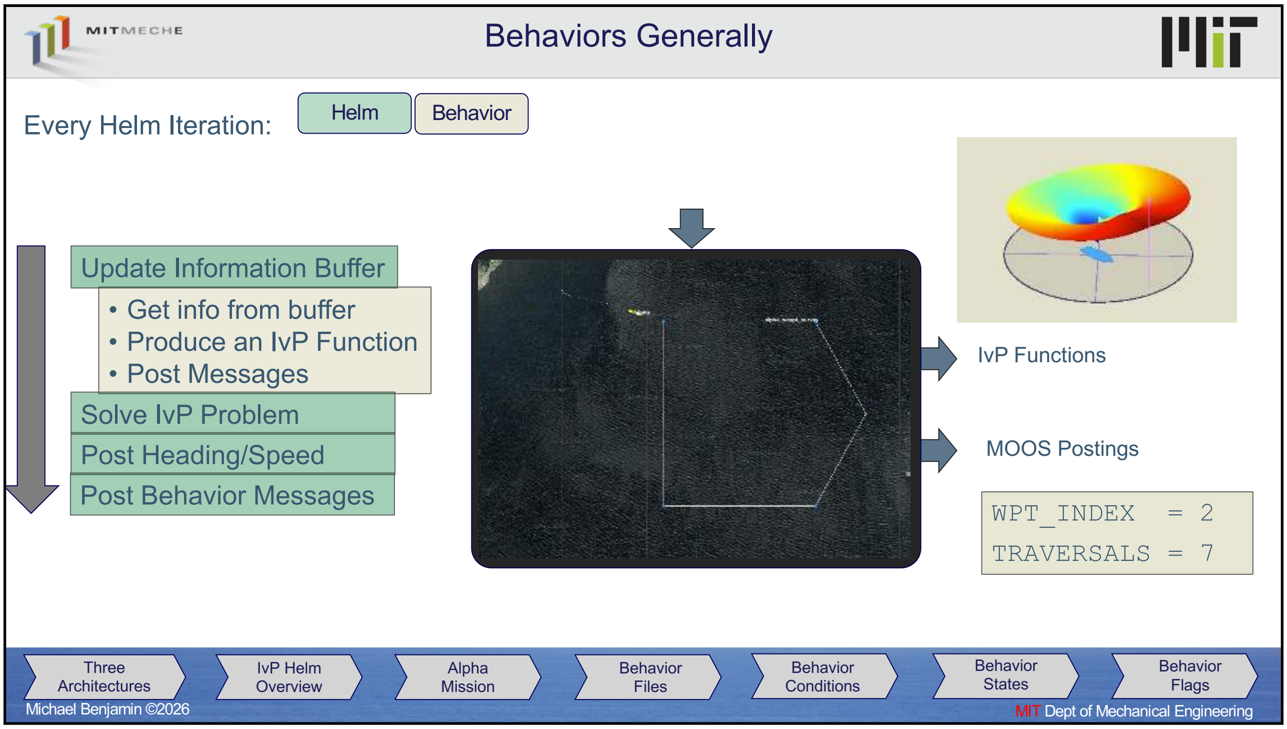
MOOS Postings

```
WPT_INDEX  = 2
TRAVERSALS = 7
```

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

11



Behaviors Generally

Every Helm Iteration: Helm Behavior

Update Information Buffer

- Get info from buffer
- Produce an IvP Function
- Post Messages

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

IvP Functions

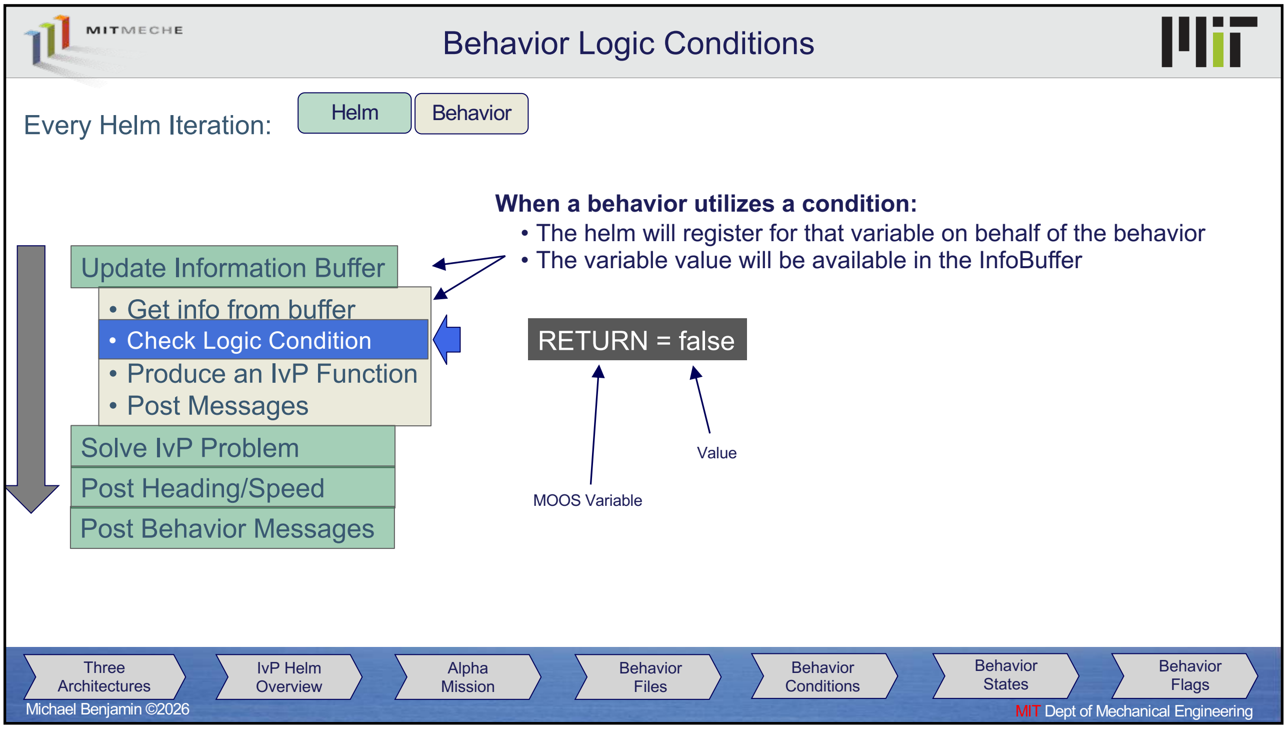
MOOS Postings

```
WPT_INDEX  = 2
TRAVERSALS = 7
```

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

12



Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

When a behavior utilizes a condition:
- The helm will register for that variable on behalf of the behavior
- The variable value will be available in the InfoBuffer

Update Information Buffer
- Get info from buffer
- Check Logic Condition
- Produce an IvP Function
- Post Messages

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

RETURN = false

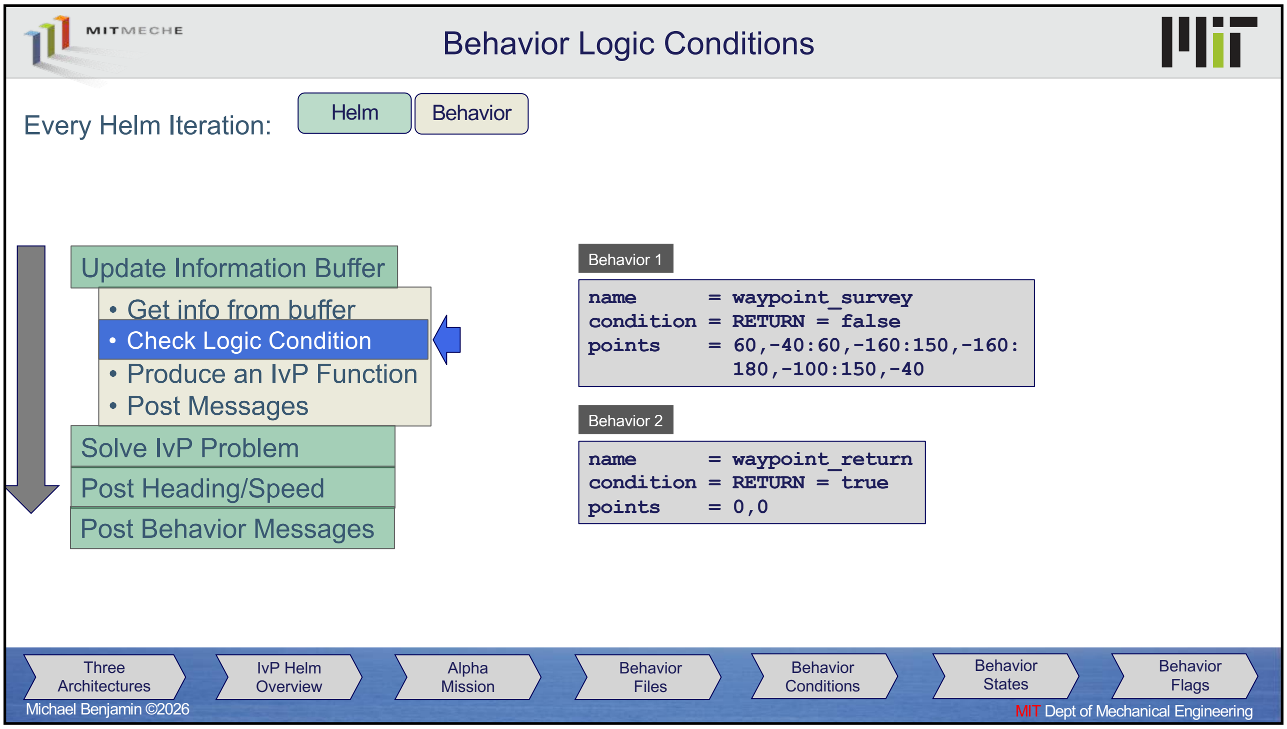
MOOS Variable

Value

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

13



Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

Update Information Buffer
- Get info from buffer
- Check Logic Condition
- Produce an IvP Function
- Post Messages

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

Behavior 1

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

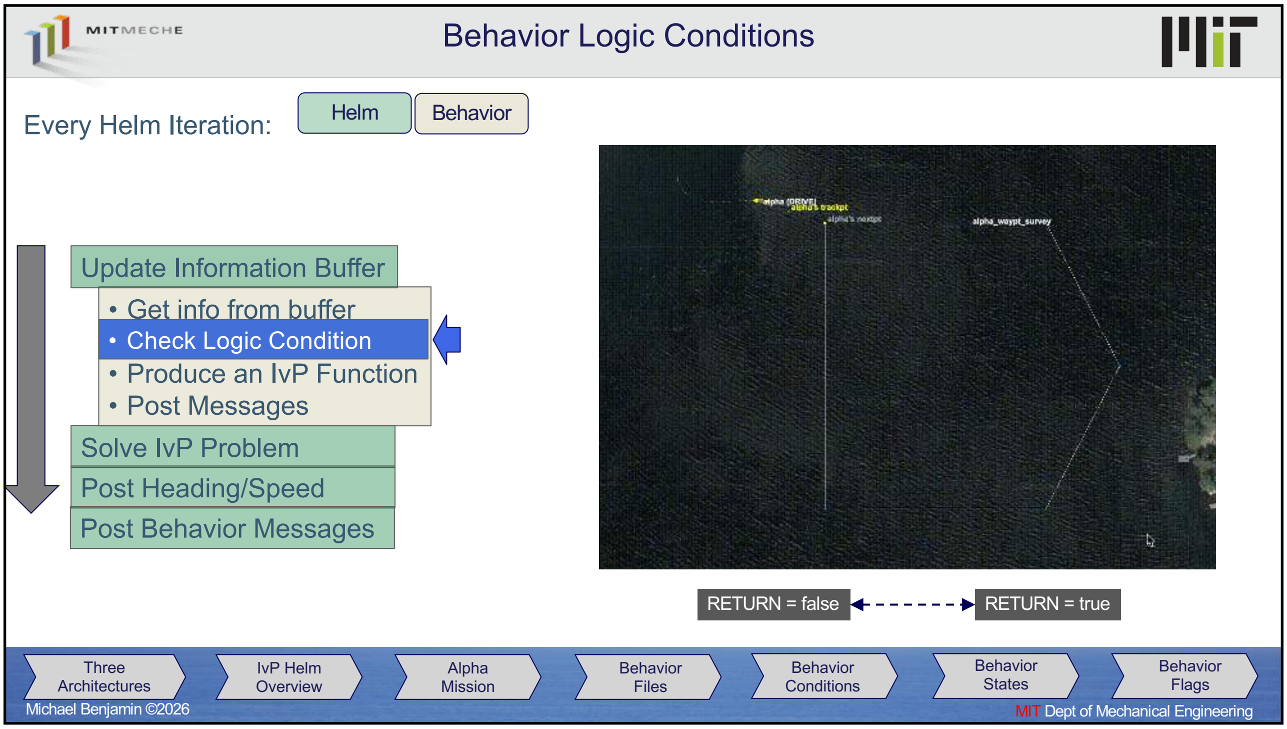
Behavior 2

```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

14



MITMECHE

Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

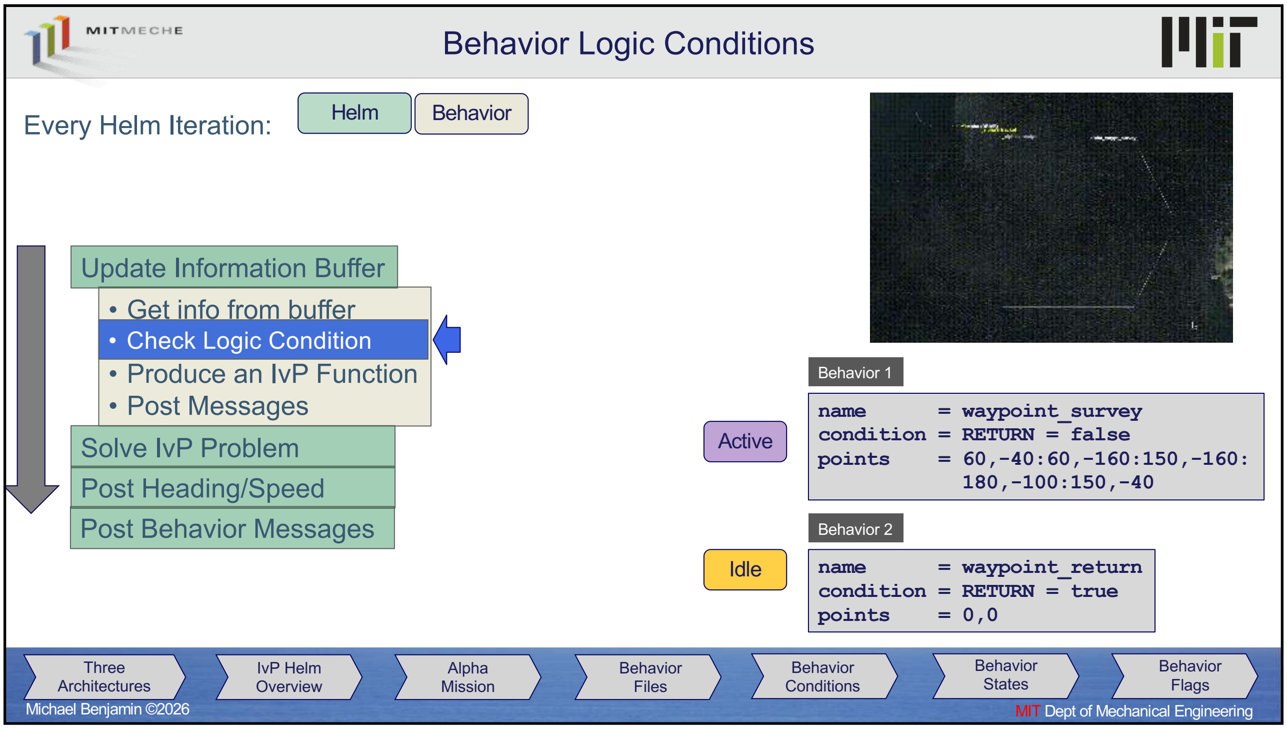
RETURN = false ◄- - - - - - ► RETURN = true

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

15



MITMECHE

Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

Behavior 1

Active

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

Behavior 2

Idle

```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```

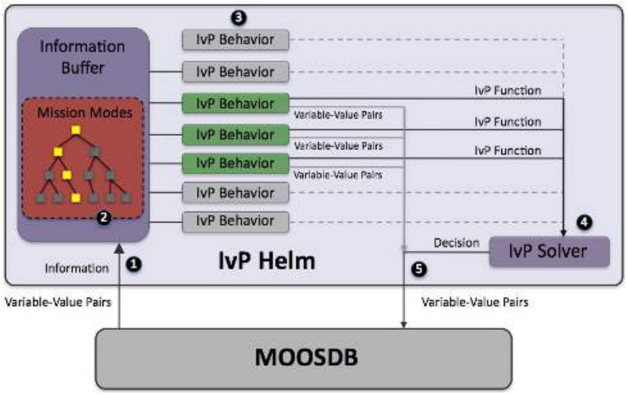
Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

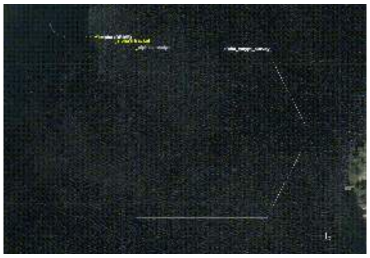
Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

16

Behavior Logic Conditions





Behavior 1

Active

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

Behavior 2

Idle

```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```

17

Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

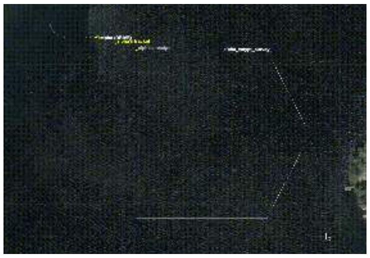
Update Information Buffer

- Get info from buffer
- Check Logic Condition
- Produce an IvP Function
- Post Messages

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages



Behavior 1

Active

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

Behavior 2

Idle

```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```

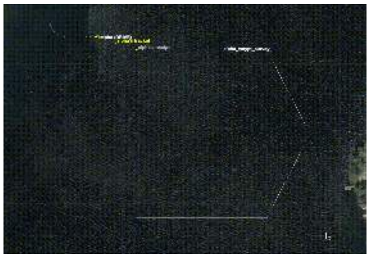
18

Behavior Logic Conditions

Every Helm Iteration: Helm | Behavior

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

RETURN	
true	false
Idle	Active
Active	Idle



Behavior 1

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

Behavior 2

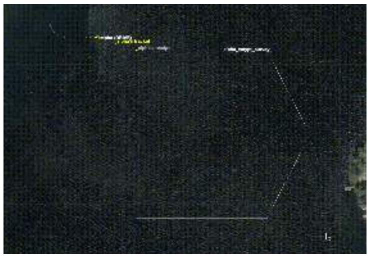
```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```

Behavior Logic Conditions

Every Helm Iteration: Helm | Behavior

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

MODE	
returning	surveying
Idle	Active
Active	Idle

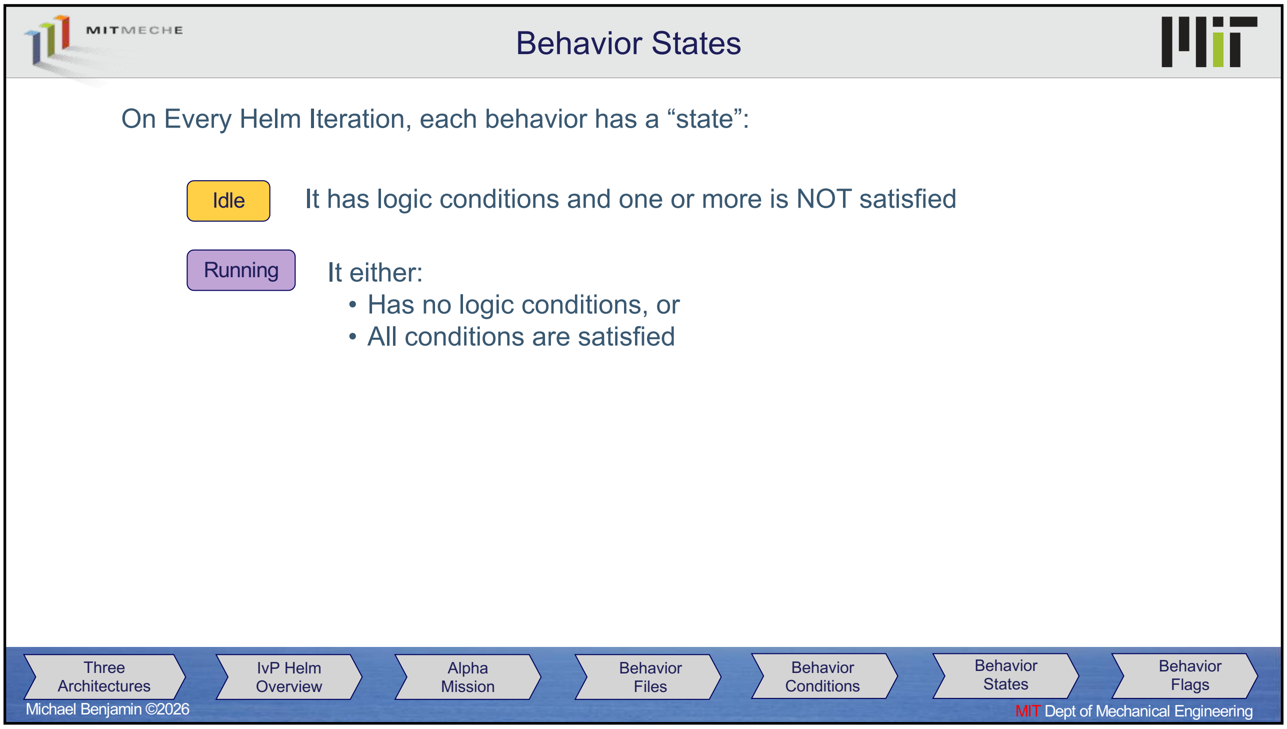


Behavior 1

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
```

Behavior 2

```
name      = waypoint_return
condition = RETURN = true
points    = 0,0
```



MITMECHE

Behavior States

On Every Helm Iteration, each behavior has a “state”:

Idle — It has logic conditions and one or more is NOT satisfied

Running — It either:

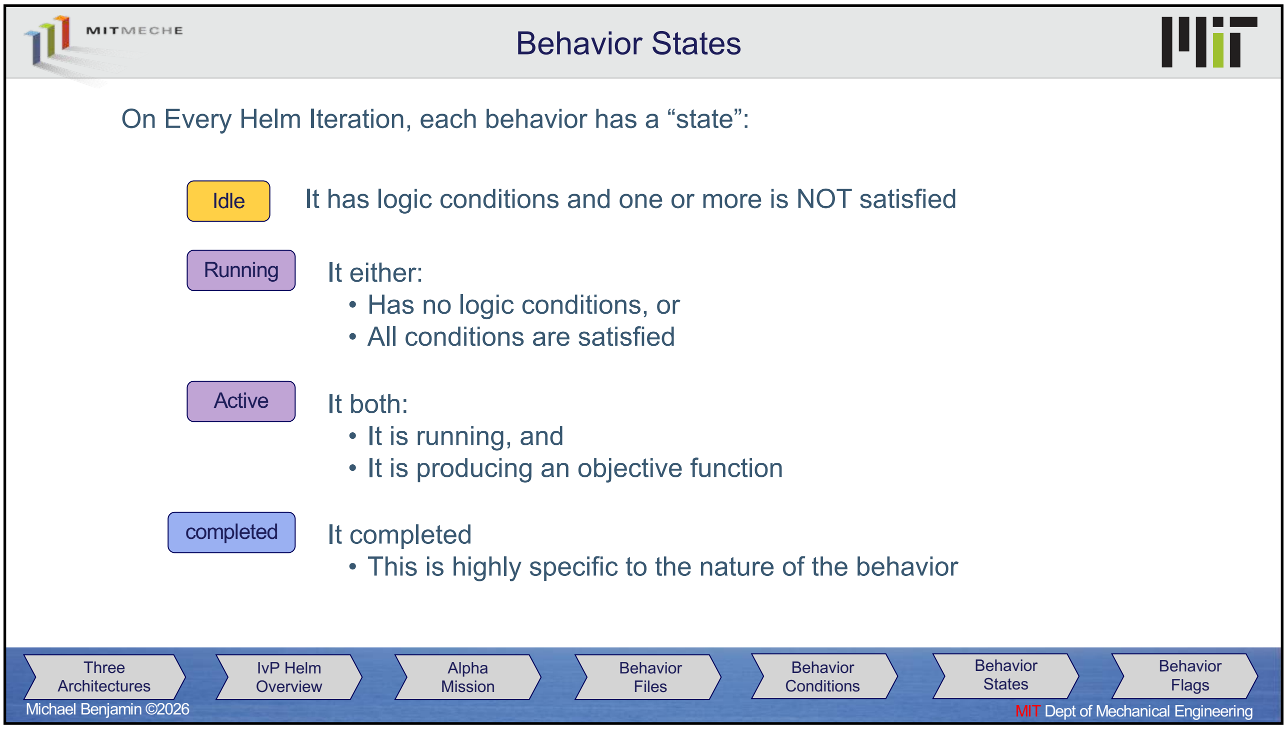
- Has no logic conditions, or
- All conditions are satisfied

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

21



MITMECHE

Behavior States

On Every Helm Iteration, each behavior has a “state”:

Idle — It has logic conditions and one or more is NOT satisfied

Running — It either:

- Has no logic conditions, or
- All conditions are satisfied

Active — It both:

- It is running, and
- It is producing an objective function

completed — It completed

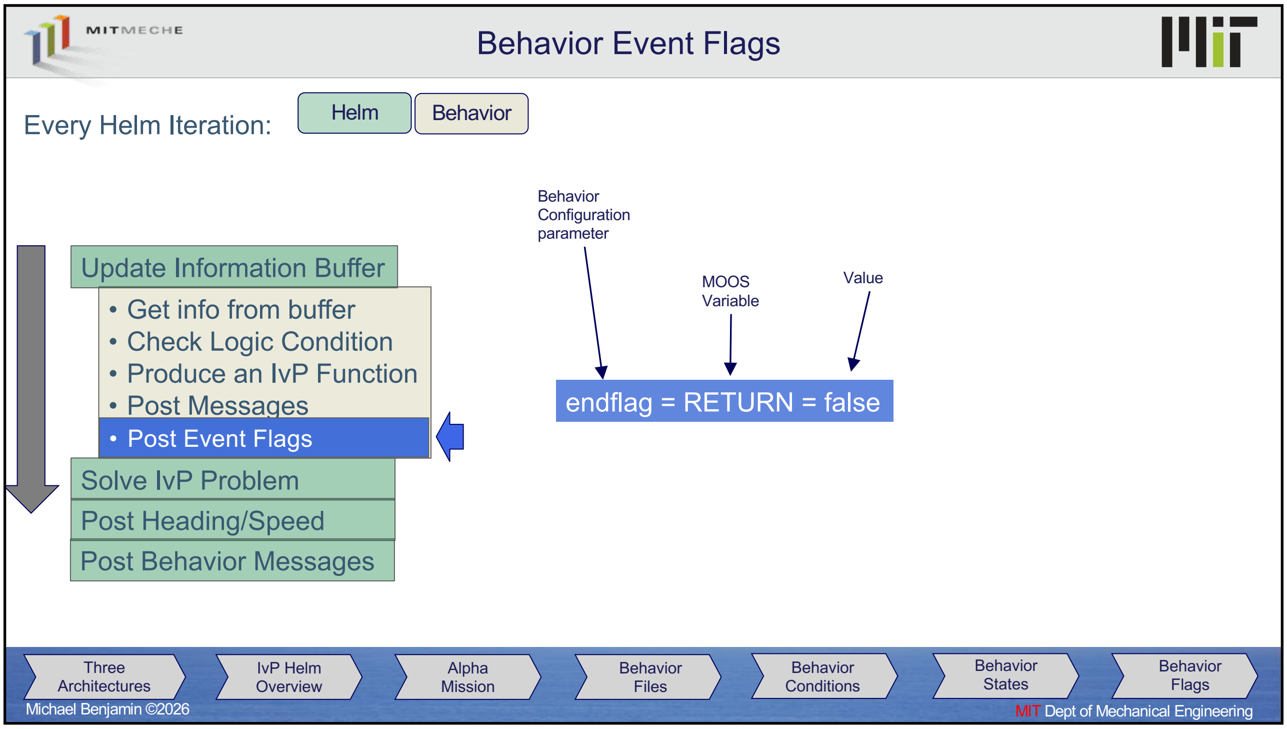
- This is highly specific to the nature of the behavior

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

22



Behavior Event Flags

Every Helm Iteration:

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
  - Post Event Flags
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

Behavior Configuration parameter

MOOS Variable

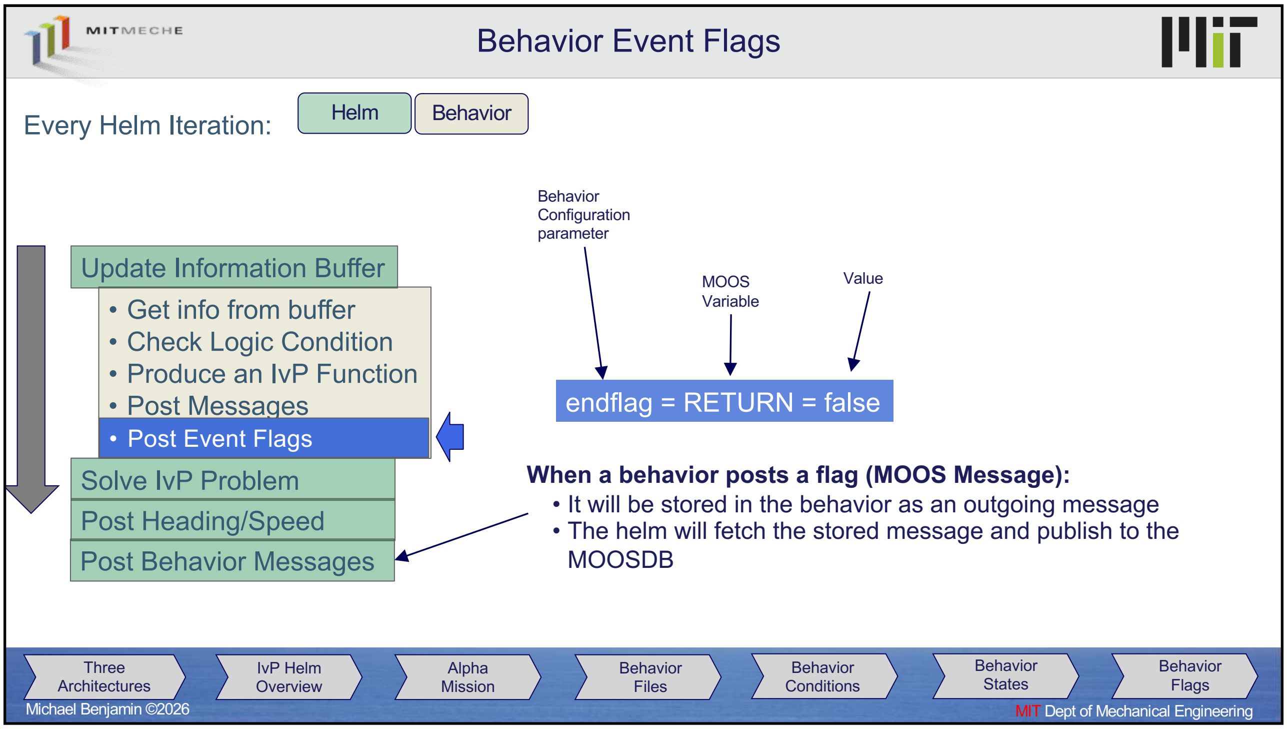
Value

endflag = RETURN = false

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

23



Behavior Event Flags

Every Helm Iteration:

- Update Information Buffer
  - Get info from buffer
  - Check Logic Condition
  - Produce an IvP Function
  - Post Messages
  - Post Event Flags
- Solve IvP Problem
- Post Heading/Speed
- Post Behavior Messages

Behavior Configuration parameter

MOOS Variable

Value

endflag = RETURN = false

When a behavior posts a flag (MOOS Message):

- It will be stored in the behavior as an outgoing message
- The helm will fetch the stored message and publish to the MOOSDB

Michael Benjamin ©2026

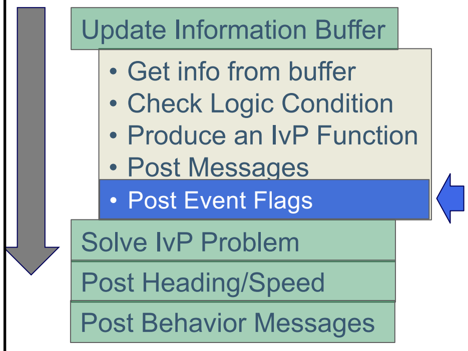
MIT Dept of Mechanical Engineering

24

MITMECHE

Behavior Event Flags

Every Helm Iteration: Helm Behavior


Update Information Buffer

- Get info from buffer
- Check Logic Condition
- Produce an IvP Function
- Post Messages
- Post Event Flags

Solve IvP Problem

Post Heading/Speed

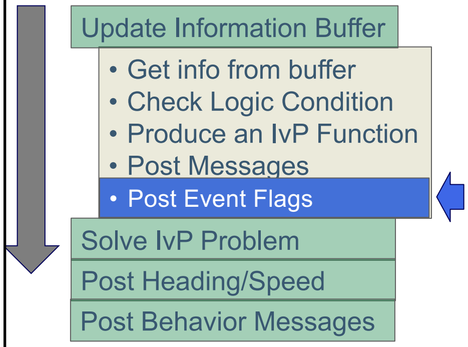
Post Behavior Messages

endflag = RETURN = true

MITMECHE

Types of Event Flags

Every Helm Iteration: Helm Behavior


Update Information Buffer

- Get info from buffer
- Check Logic Condition
- Produce an IvP Function
- Post Messages
- Post Event Flags

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

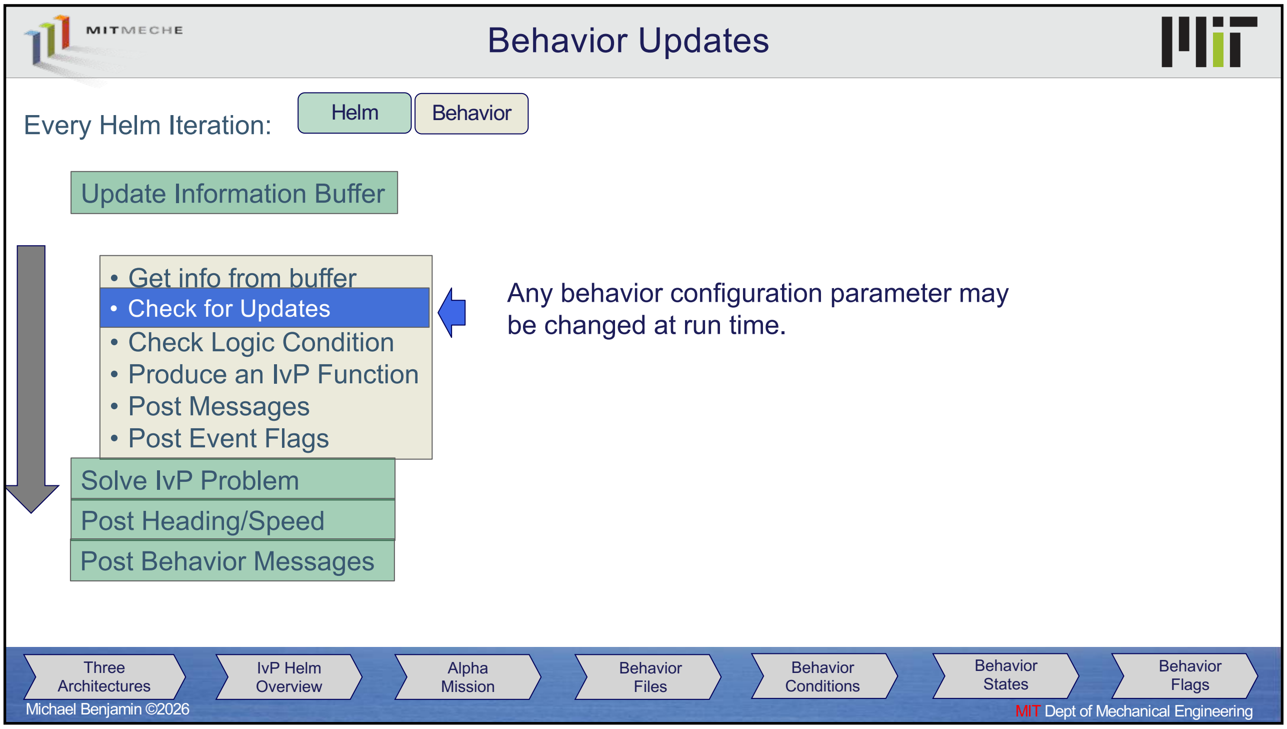
endflag Posted when a behavior completes

idleflag Posted when a behavior enters the idle state

runflag Posted when a behavior enters the running state

runxflag Posted when a behavior on ***every iteration*** when running

activeflag Posted when a behavior enters the active state



Behavior Updates

Every Helm Iteration: Helm Behavior

Update Information Buffer

- Get info from buffer
- Check for Updates
- Check Logic Condition
- Produce an IvP Function
- Post Messages
- Post Event Flags

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

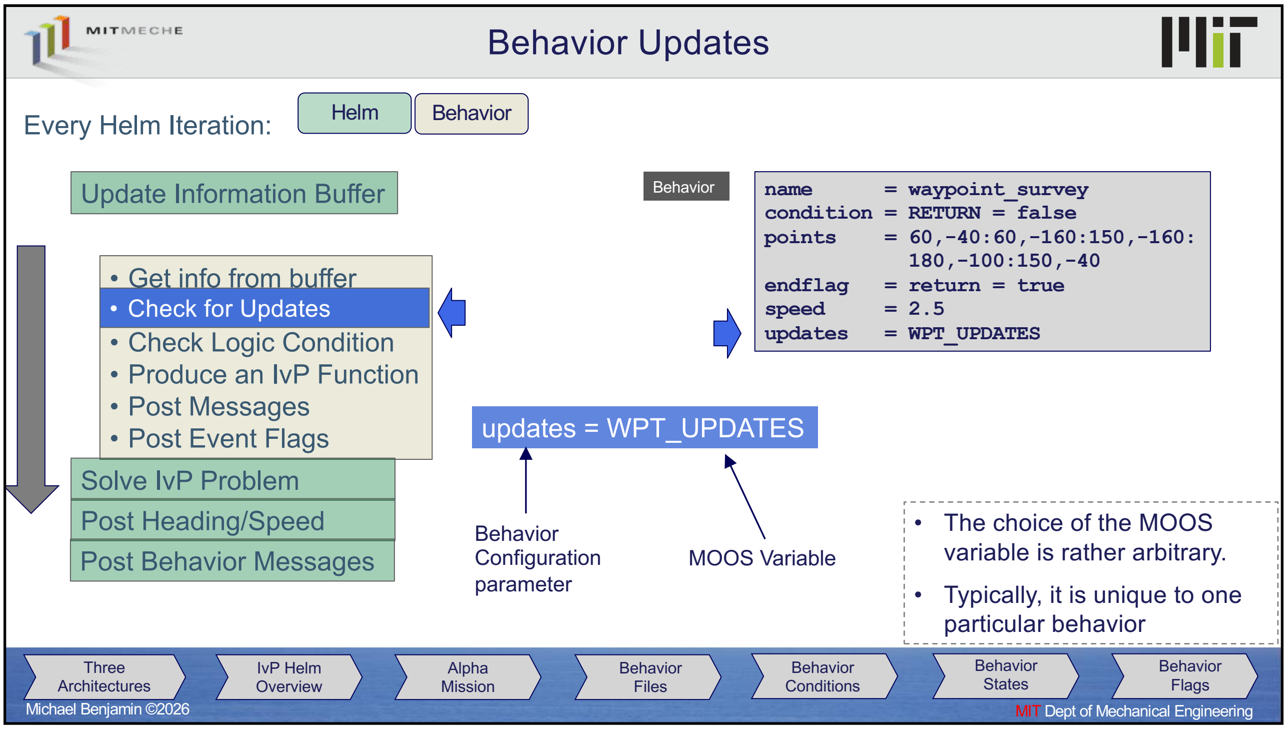
Any behavior configuration parameter may be changed at run time.

Three Architectures
IvP Helm Overview
Alpha Mission
Behavior Files
Behavior Conditions
Behavior States
Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

27



Behavior Updates

Every Helm Iteration: Helm Behavior

Update Information Buffer

- Get info from buffer
- Check for Updates
- Check Logic Condition
- Produce an IvP Function
- Post Messages
- Post Event Flags

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

Behavior

```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
endflag   = return = true
speed     = 2.5
updates   = WPT_UPDATES
```

updates = WPT_UPDATES

Behavior Configuration parameter

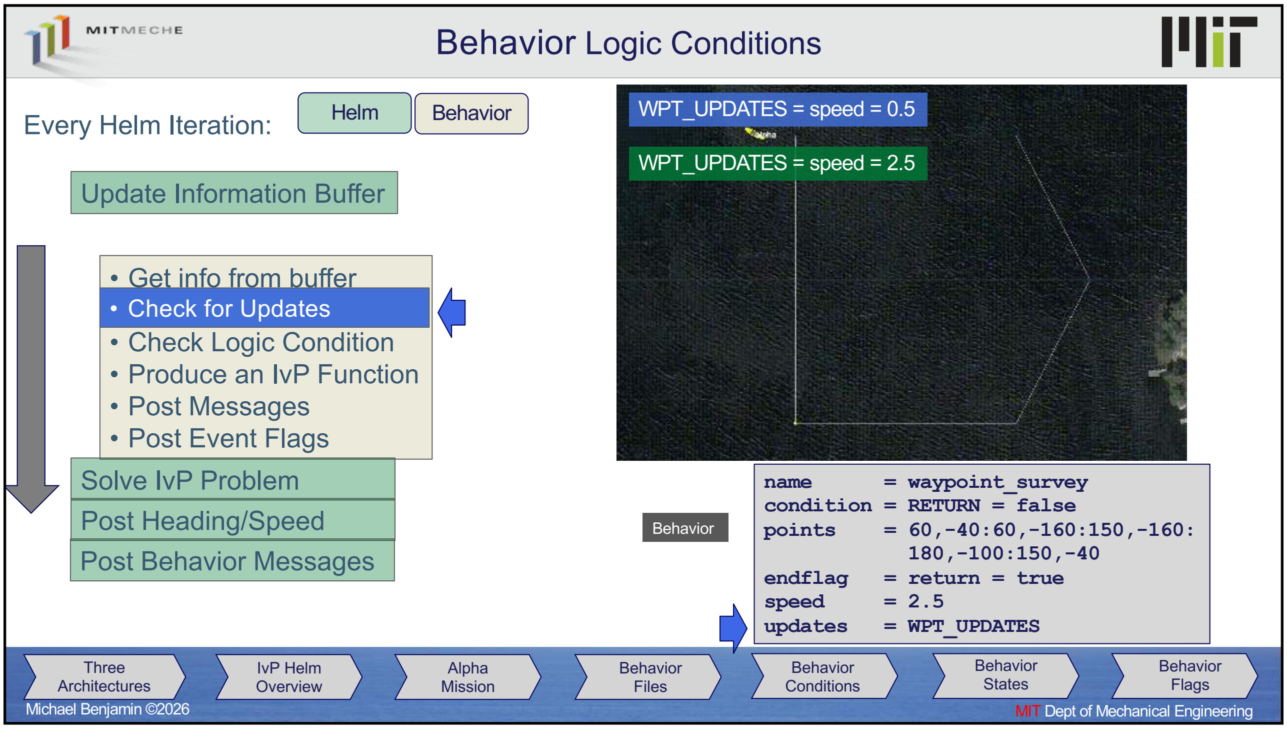
MOOS Variable

- The choice of the MOOS variable is rather arbitrary.
- Typically, it is unique to one particular behavior

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

28



Behavior Logic Conditions

Every Helm Iteration: Helm Behavior

Update Information Buffer

- Get info from buffer
- Check for Updates
- Check Logic Condition
- Produce an IvP Function
- Post Messages
- Post Event Flags

Solve IvP Problem

Post Heading/Speed

Post Behavior Messages

WPT_UPDATES = speed = 0.5

WPT_UPDATES = speed = 2.5

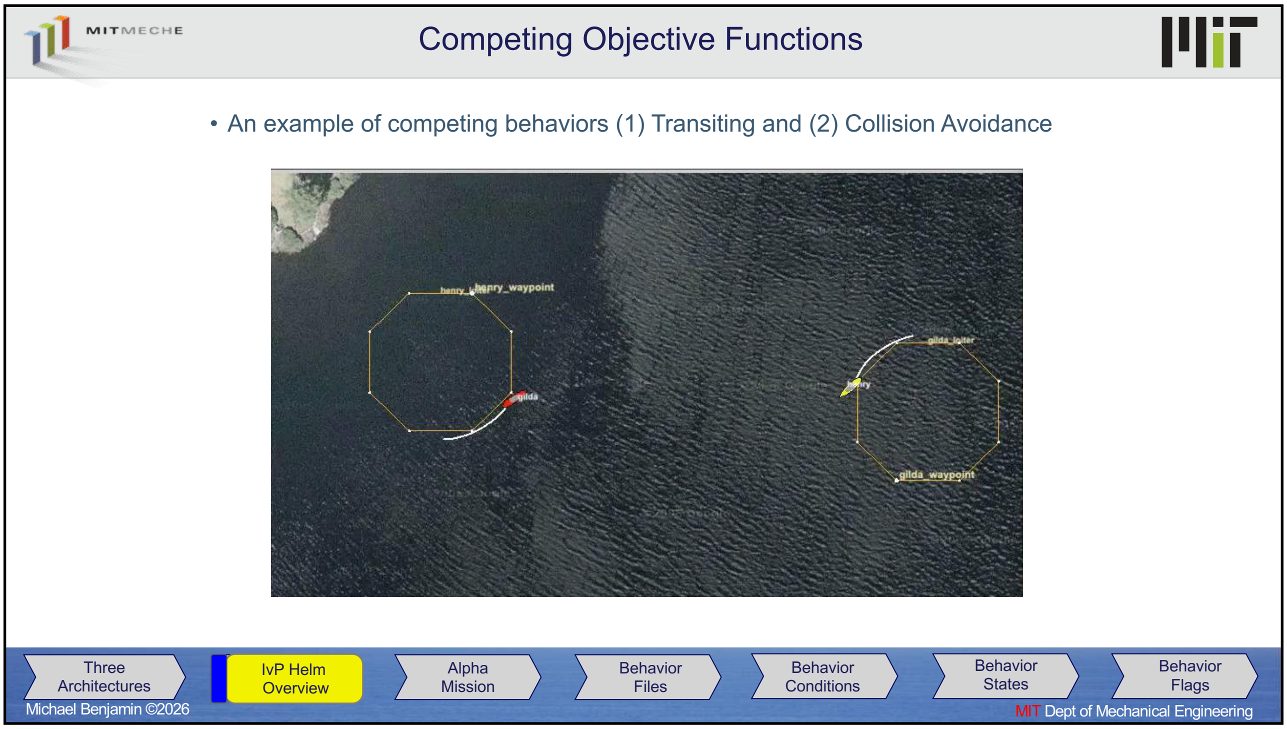
```
name      = waypoint_survey
condition = RETURN = false
points    = 60,-40:60,-160:150,-160:
            180,-100:150,-40
endflag   = return = true
speed     = 2.5
updates   = WPT_UPDATES
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

29



Competing Objective Functions

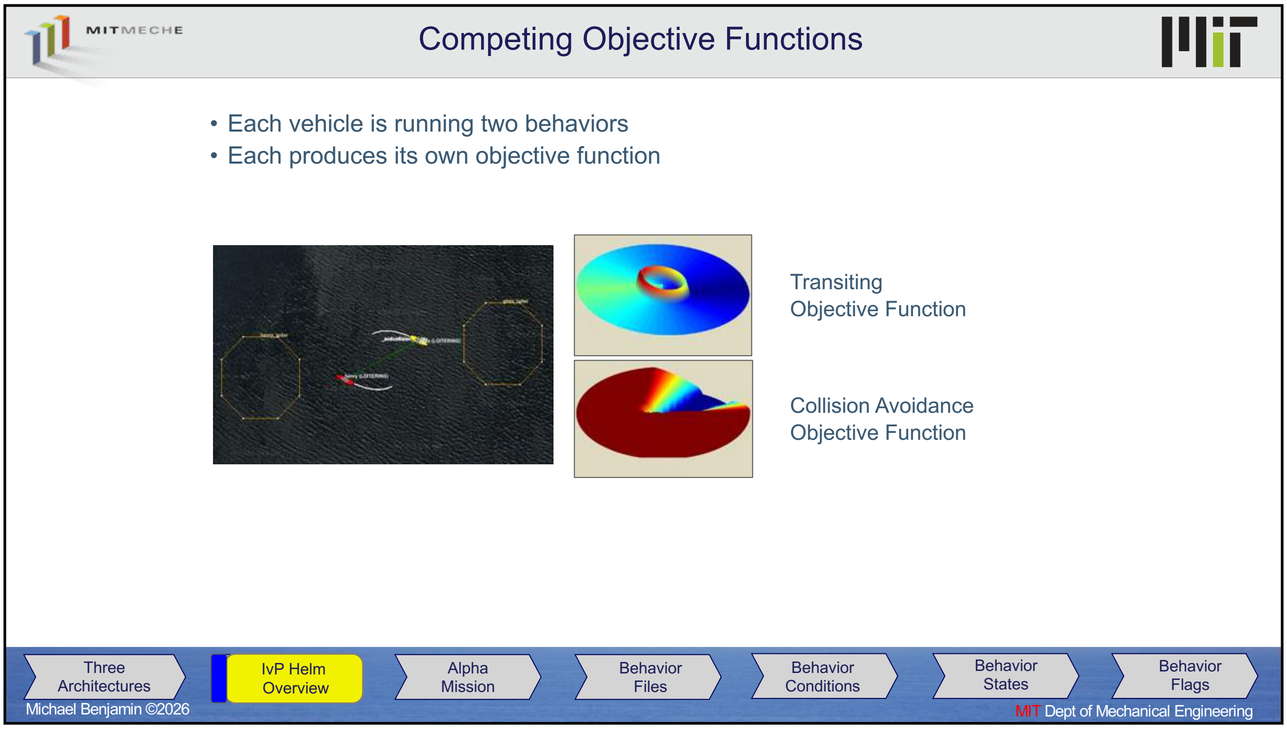
- An example of competing behaviors (1) Transiting and (2) Collision Avoidance

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

30



Competing Objective Functions

- Each vehicle is running two behaviors
- Each produces its own objective function

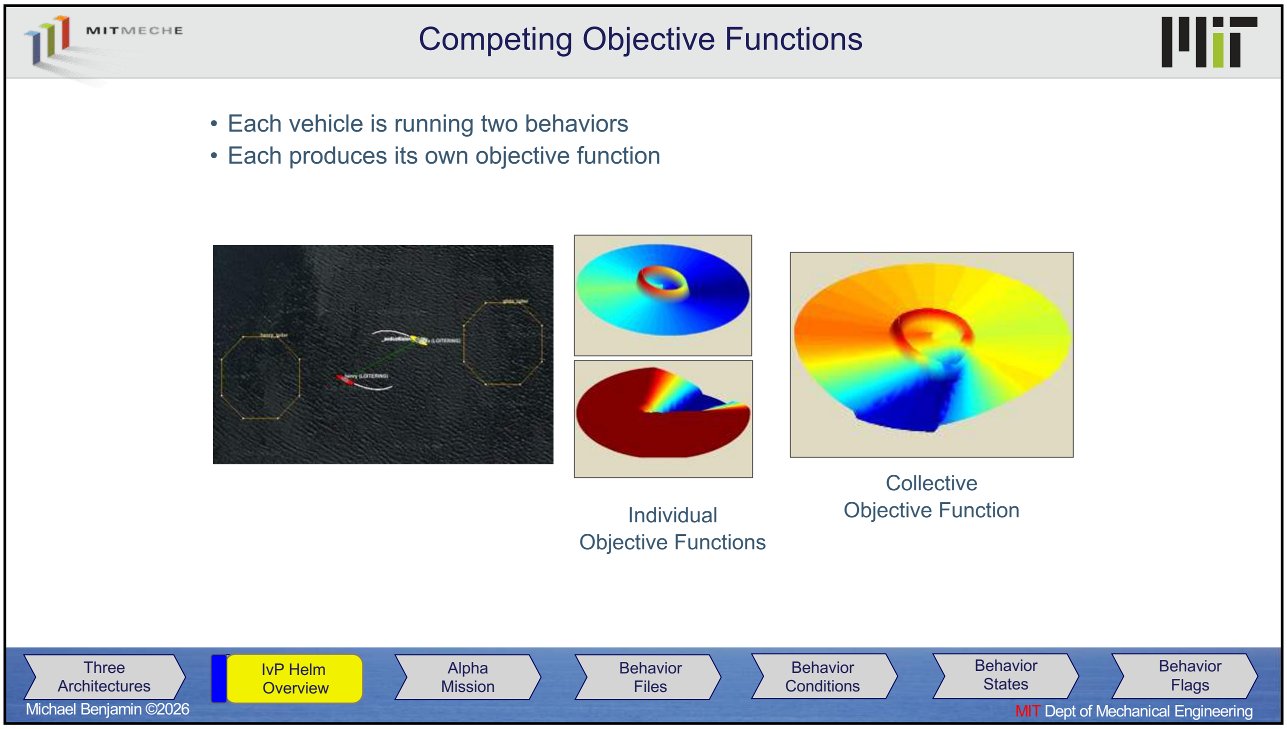
Transiting Objective Function

Collision Avoidance Objective Function

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

31



Competing Objective Functions

- Each vehicle is running two behaviors
- Each produces its own objective function

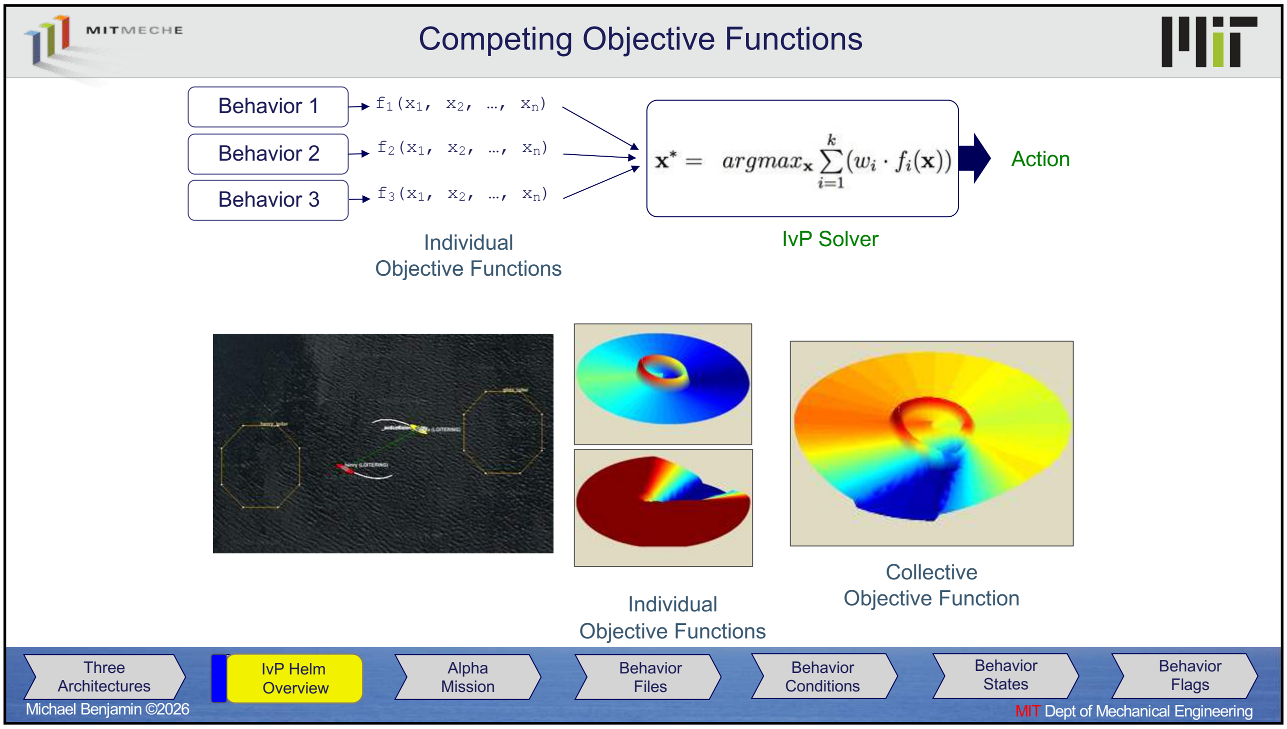
Individual Objective Functions

Collective Objective Function

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

32



Competing Objective Functions

Behavior 1 → $f_1(x_1, x_2, \ldots, x_n)$

Behavior 2 → $f_2(x_1, x_2, \ldots, x_n)$

Behavior 3 → $f_3(x_1, x_2, \ldots, x_n)$

Individual Objective Functions

$$\mathbf{x}^* = argmax_{\mathbf{x}} \sum_{i=1}^{k}(w_i \cdot f_i(\mathbf{x}))$$

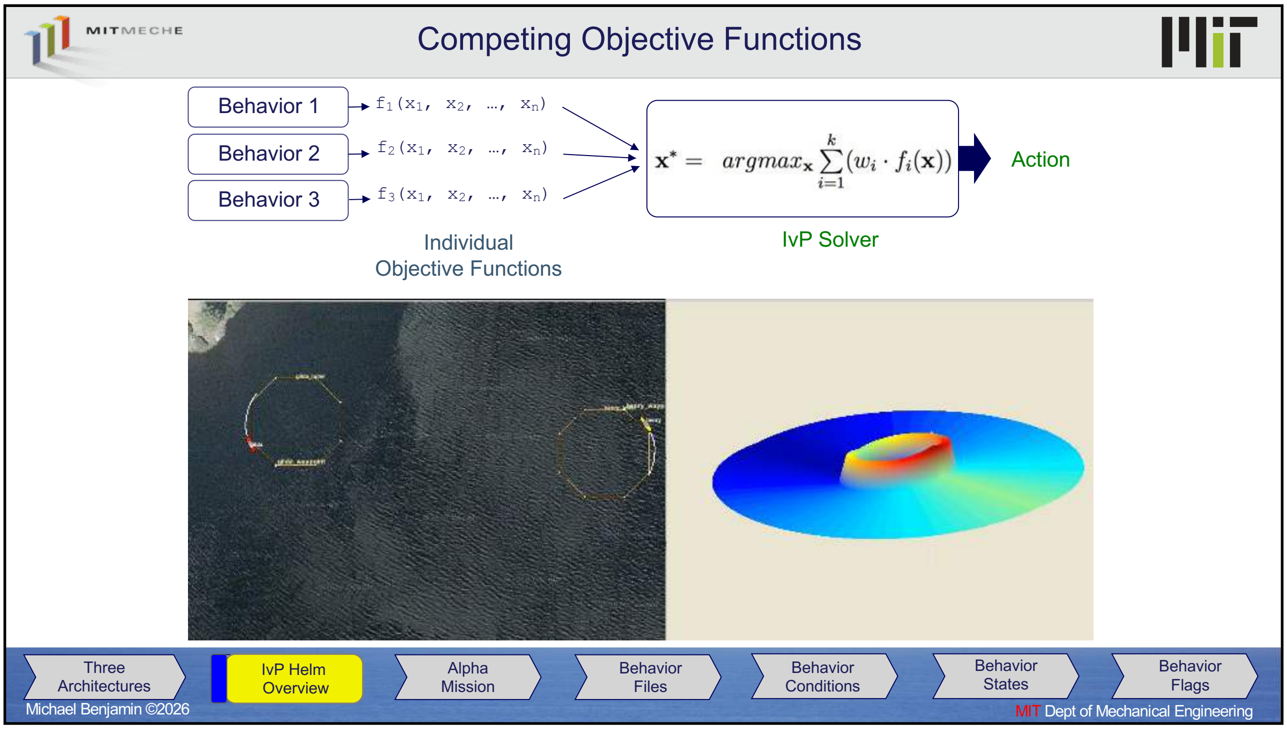
IvP Solver

Action

Individual Objective Functions

Collective Objective Function

33



Competing Objective Functions

Behavior 1 → $f_1(x_1, x_2, \ldots, x_n)$

Behavior 2 → $f_2(x_1, x_2, \ldots, x_n)$

Behavior 3 → $f_3(x_1, x_2, \ldots, x_n)$

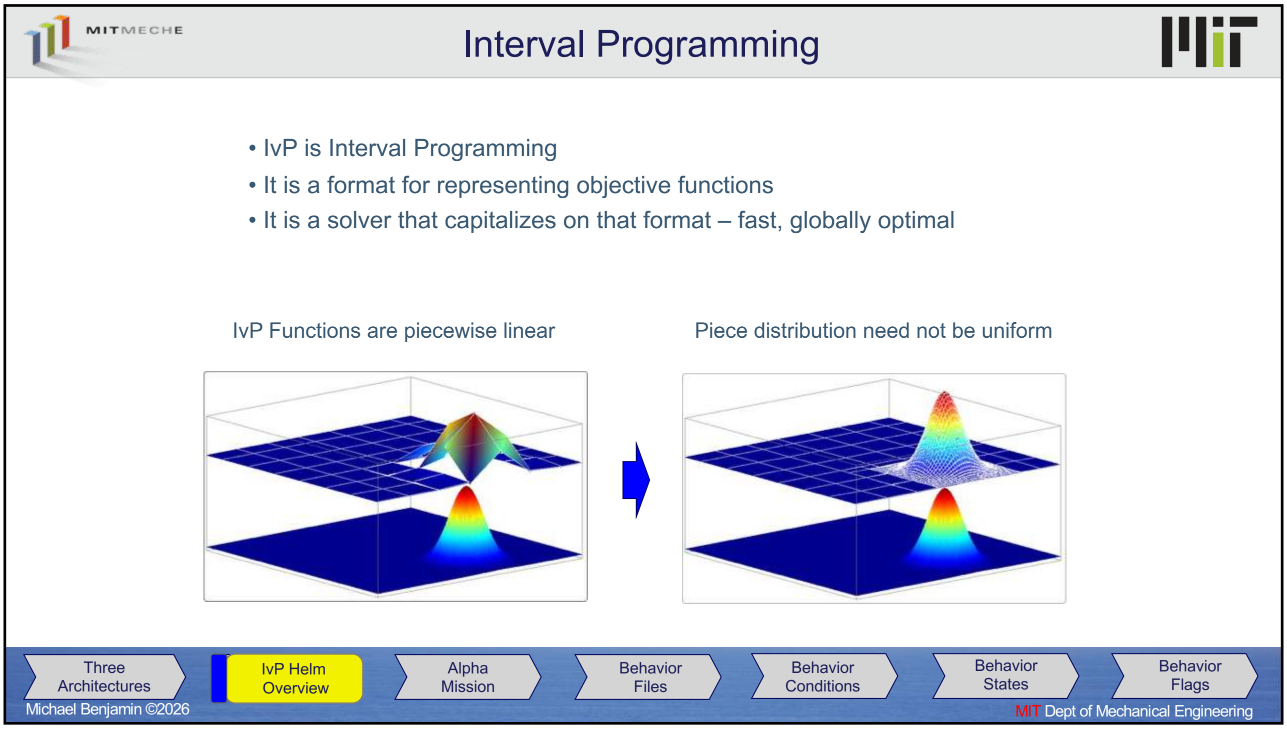
Individual Objective Functions

$$\mathbf{x}^* = argmax_{\mathbf{x}} \sum_{i=1}^{k}(w_i \cdot f_i(\mathbf{x}))$$

IvP Solver

Action

34



Interval Programming

- IvP is Interval Programming
- It is a format for representing objective functions
- It is a solver that capitalizes on that format – fast, globally optimal

IvP Functions are piecewise linear

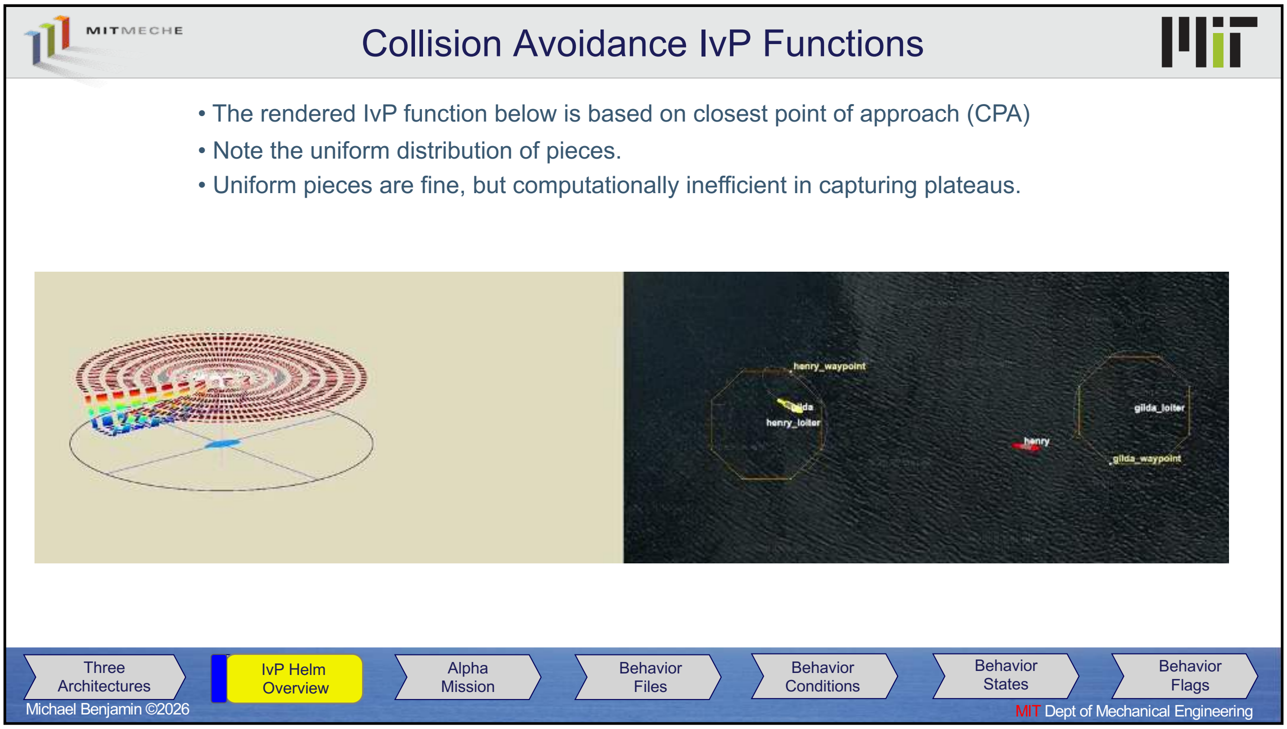
Piece distribution need not be uniform

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

35



Collision Avoidance IvP Functions

- The rendered IvP function below is based on closest point of approach (CPA)
- Note the uniform distribution of pieces.
- Uniform pieces are fine, but computationally inefficient in capturing plateaus.

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

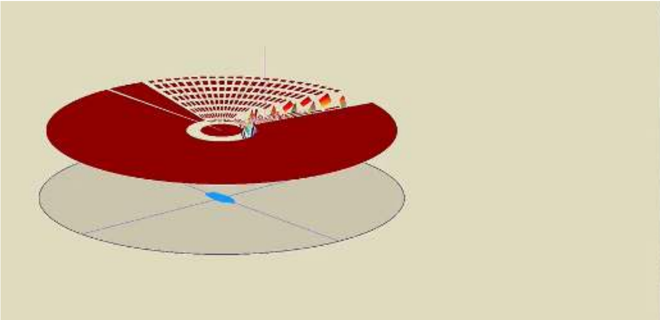
Michael Benjamin ©2026

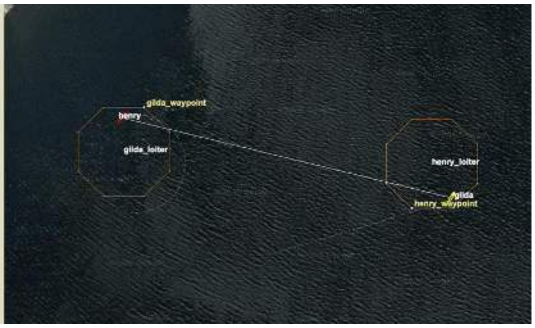
MIT Dept of Mechanical Engineering

36

Collision Avoidance IvP Functions

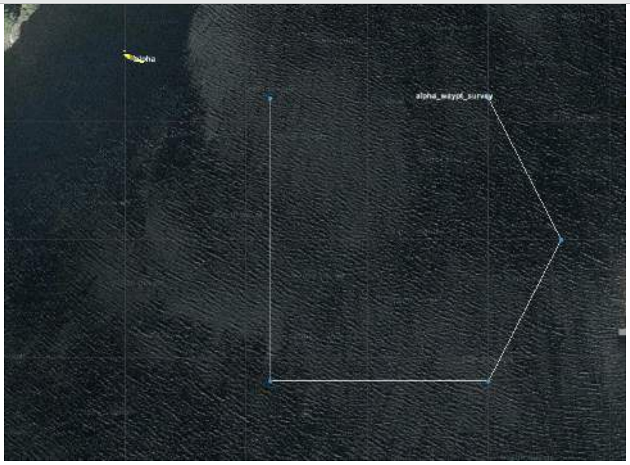
- The IvP functions need not be uniform
- The plateau regions can be identified in real time and replace many smaller pieces
- Note the large plateaus in the IvP function below.





The Alpha Mission

The Alpha Mission



To launch yourself:

```
$ cd moos-ivp/ivp/missions/s1_alpha
$ ./launch.sh 10
```

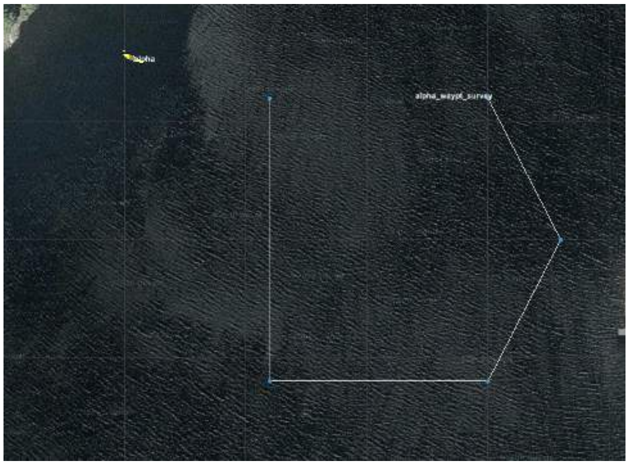
Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

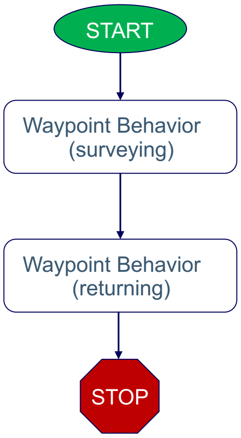
Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

39

Alpha Mission Has Two Behaviors





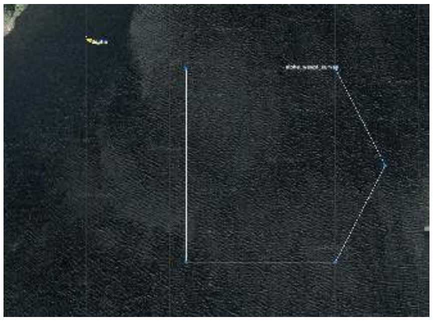
Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

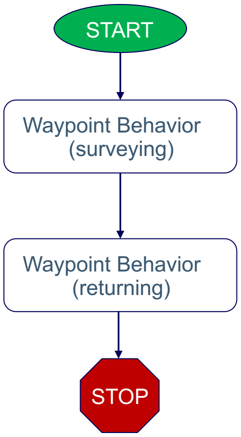
Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

40

Alpha Mission Has Two Behaviors



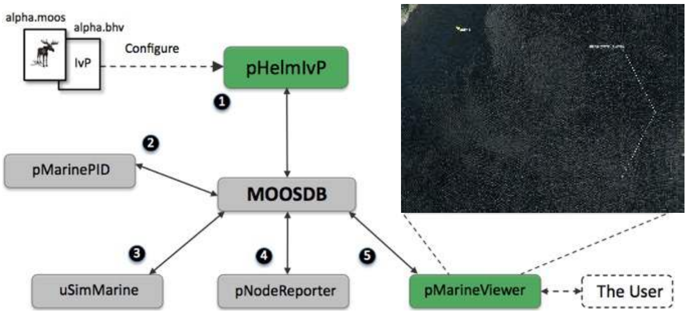


Three questions discussed next:

- How is this mission configured?
- What initiates this mission?
- How does the helm transition to return?

Configuring the Helm for a Mission

A mission is configured with two files:
`alpha.moos` – configures all MOOS apps, including general helm parameters
`alpha.bhv` – configures all Helm behaviors

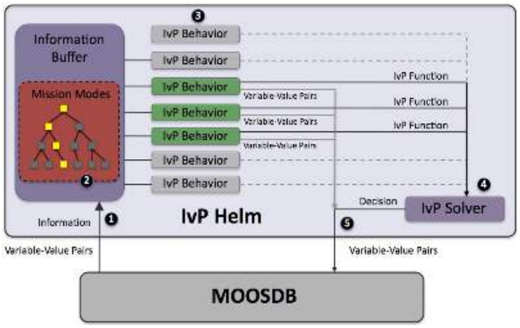


To launch it yourself:

```
$ cd moos-ivp/ivp/missions/s1_alpha
$ ./launch.sh 10
```

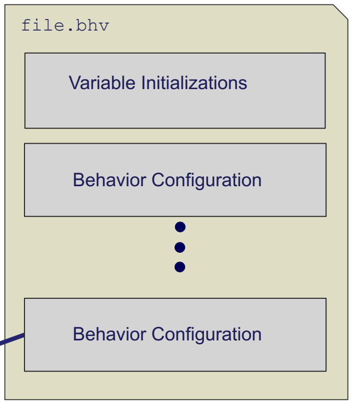
MITMECHE

Behavior Files



```
Behavior = <behavior_name>
{
   parameter = value
   . . .
   parameter = value
}
```

Helm configuration file structure:



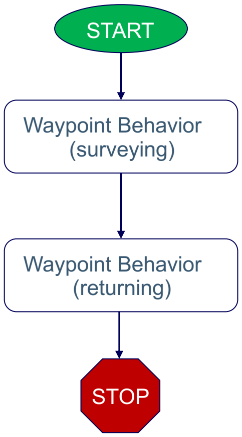
MITMECHE

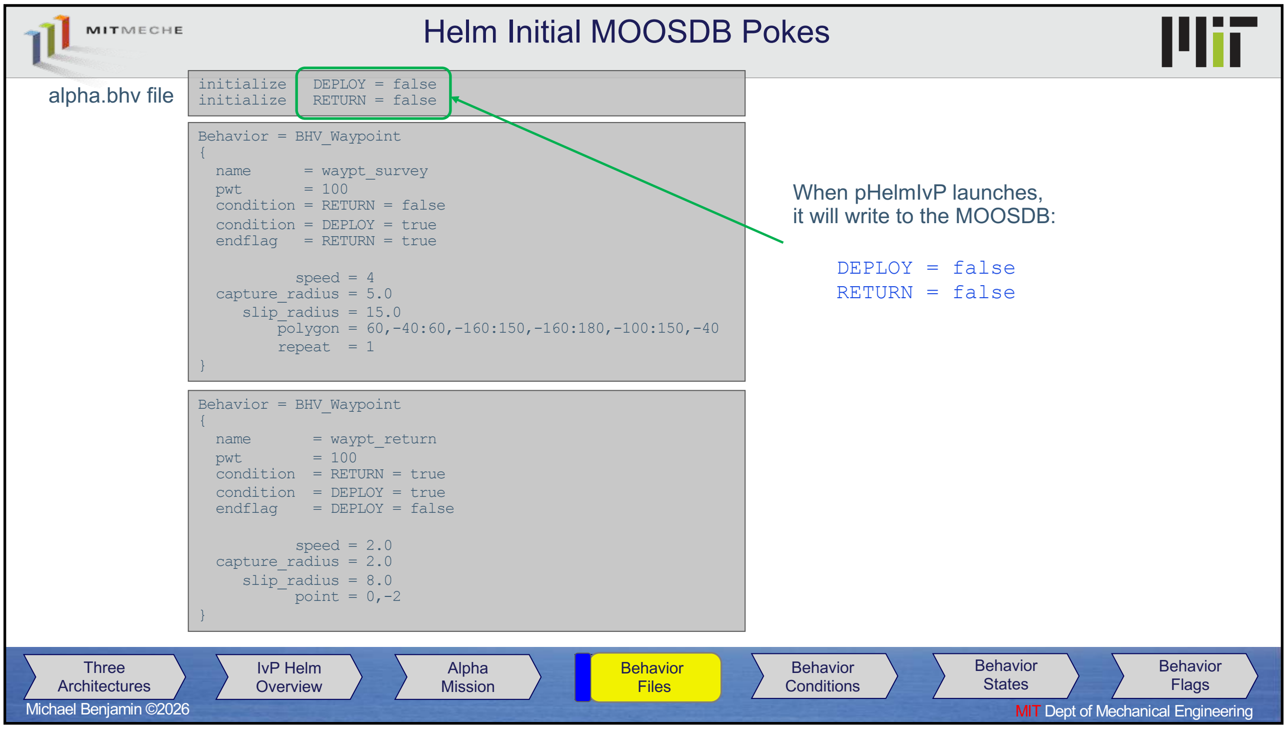
Alpha Mission Behavior File

```
initialize   DEPLOY = false
initialize   RETURN = false
```

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

```
Behavior = BHV_Waypoint
{
  name       = waypt_return
  pwt        = 100
  condition  = RETURN = true
  condition  = DEPLOY = true
  endflag    = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
          points = 0,-2
}
```





Helm Initial MOOSDB Pokes

alpha.bhv file

```
initialize  DEPLOY = false
initialize  RETURN = false
```

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

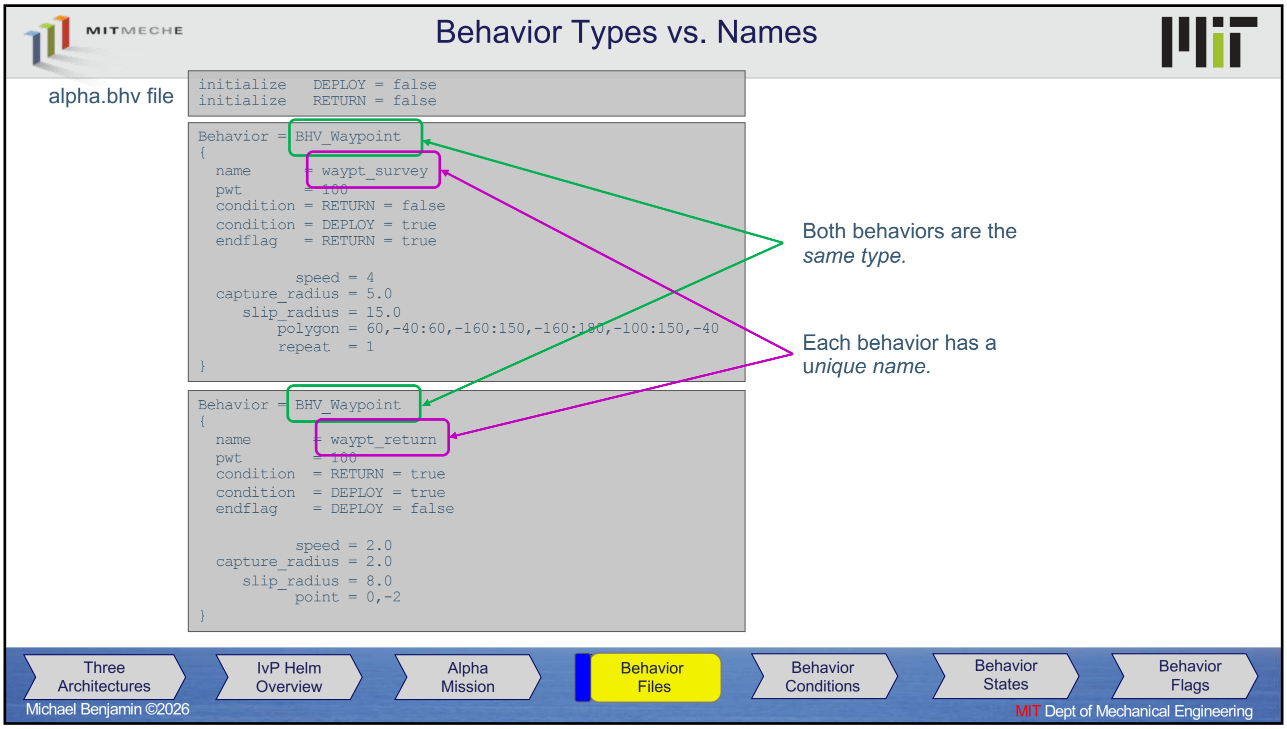
```
Behavior = BHV_Waypoint
{
  name        = waypt_return
  pwt         = 100
  condition   = RETURN = true
  condition   = DEPLOY = true
  endflag     = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
           point = 0,-2
}
```

When pHelmIvP launches,
it will write to the MOOSDB:

```
DEPLOY = false
RETURN = false
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

45



Behavior Types vs. Names

alpha.bhv file

```
initialize  DEPLOY = false
initialize  RETURN = false
```

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

```
Behavior = BHV_Waypoint
{
  name        = waypt_return
  pwt         = 100
  condition   = RETURN = true
  condition   = DEPLOY = true
  endflag     = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
           point = 0,-2
}
```

Both behaviors are the *same type.*

Each behavior has a *unique name.*

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

46

Waypoint Behavior Points

alpha.bhv file

```
initialize   DEPLOY = false
initialize   RETURN = false
```

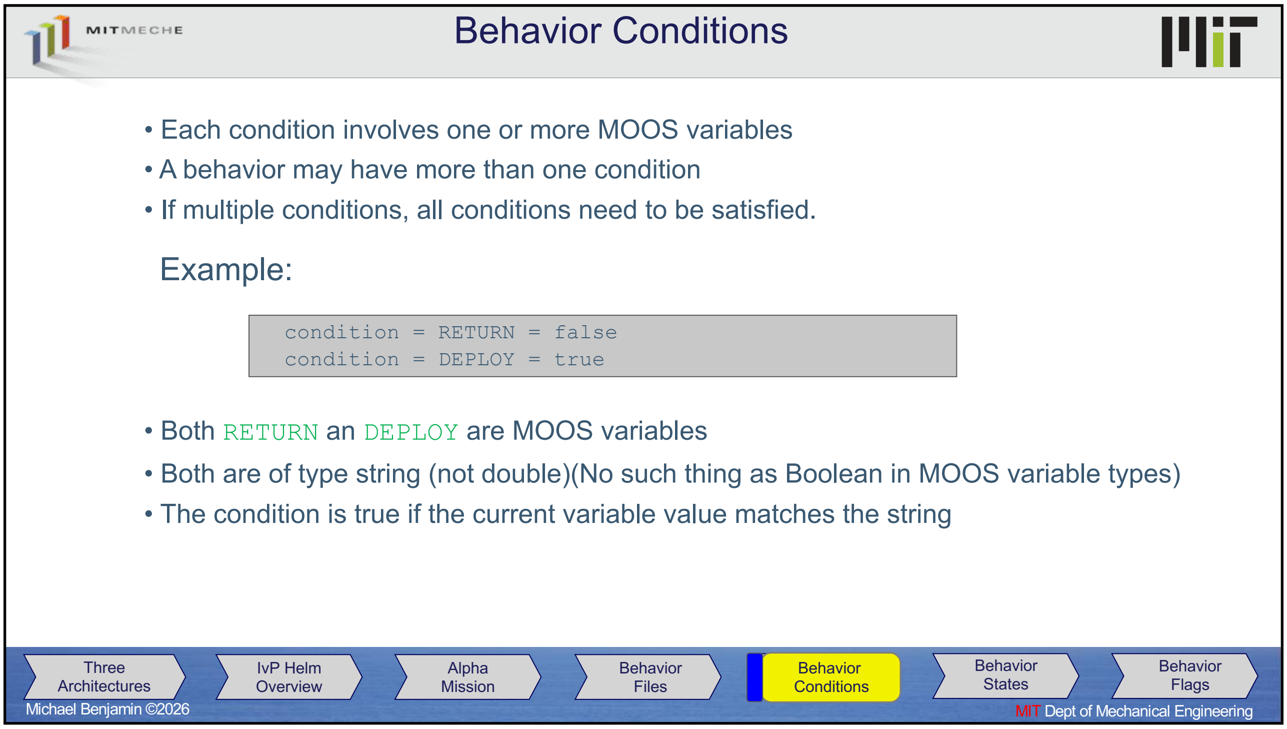
```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
          repeat  = 1
}
```

```
Behavior = BHV_Waypoint
{
  name        = waypt_return
  pwt         = 100
  condition   = RETURN = true
  condition   = DEPLOY = true
  endflag     = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
           point = 0,-2
}
```

The waypoint behavior accepts either:

- *a polygon*
- *a single point*

Behavior Conditions



MITMECHE

Behavior Conditions

- Each condition involves one or more MOOS variables
- A behavior may have more than one condition
- If multiple conditions, all conditions need to be satisfied.

Example:

```
condition = RETURN = false
condition = DEPLOY = true
```

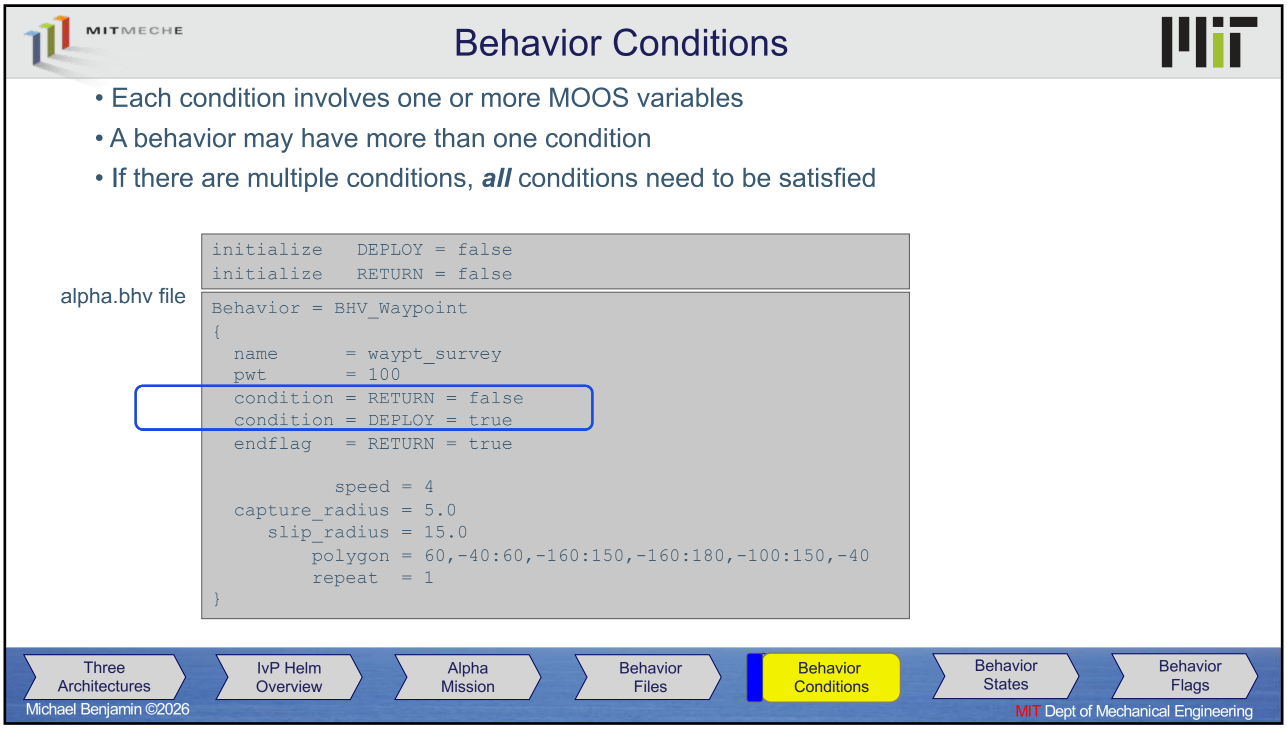
- Both `RETURN` an `DEPLOY` are MOOS variables
- Both are of type string (not double)(No such thing as Boolean in MOOS variable types)
- The condition is true if the current variable value matches the string

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

49



MITMECHE

Behavior Conditions

- Each condition involves one or more MOOS variables
- A behavior may have more than one condition
- If there are multiple conditions, ***all*** conditions need to be satisfied

alpha.bhv file

```
initialize   DEPLOY = false
initialize   RETURN = false
```

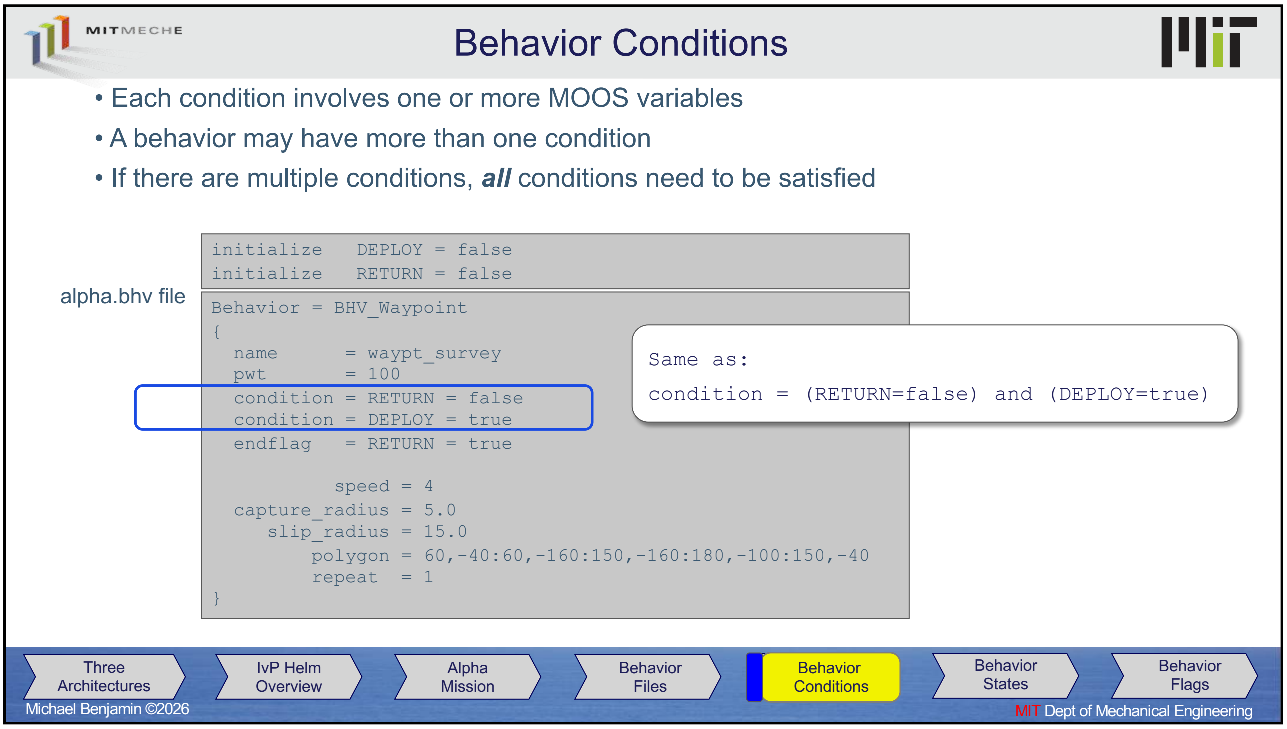
```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

50



Behavior Conditions

- Each condition involves one or more MOOS variables
- A behavior may have more than one condition
- If there are multiple conditions, ***all*** conditions need to be satisfied

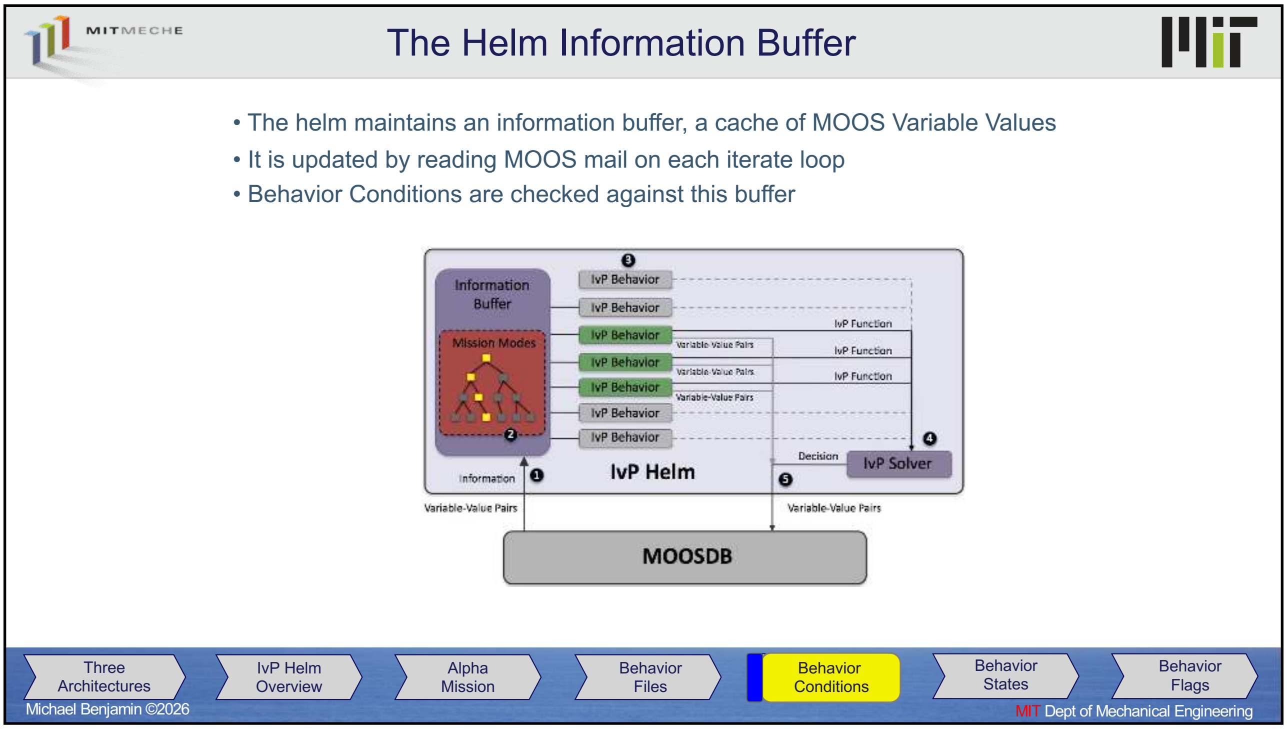
alpha.bhv file

```
initialize   DEPLOY = false
initialize   RETURN = false
```

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

```
Same as:
condition = (RETURN=false) and (DEPLOY=true)
```

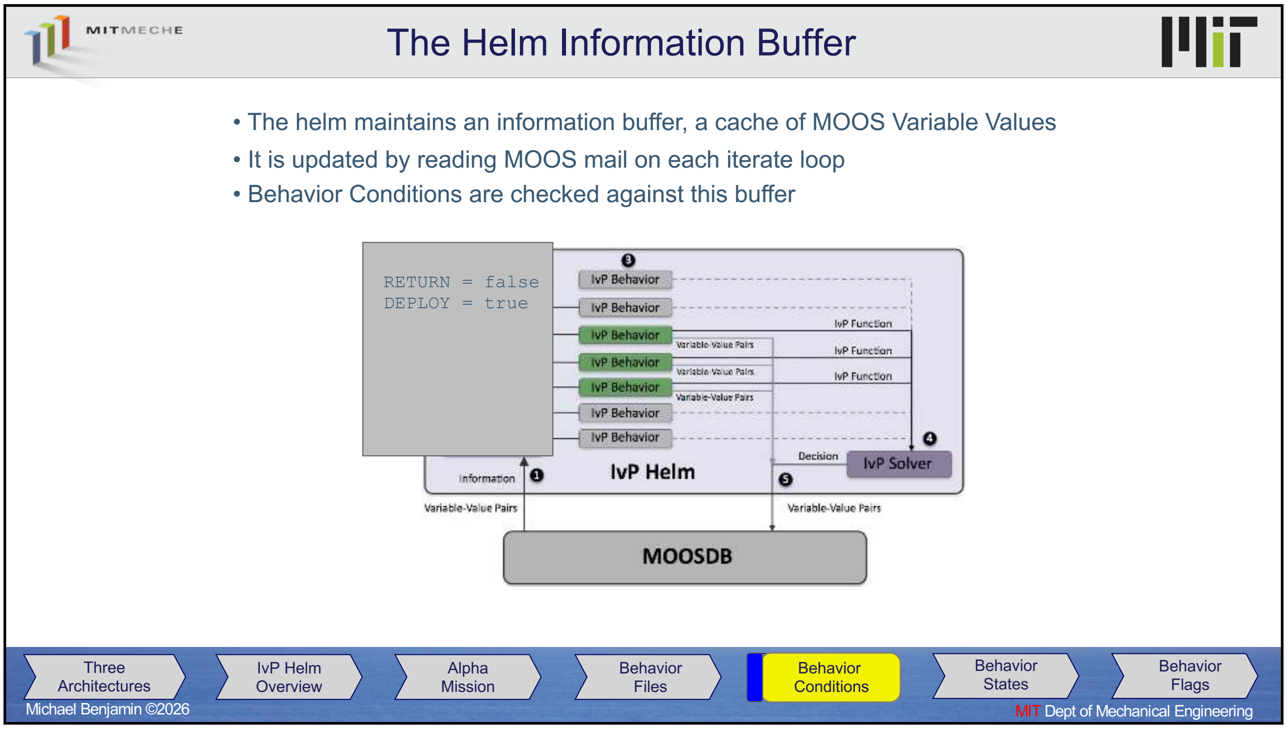
51



The Helm Information Buffer

- The helm maintains an information buffer, a cache of MOOS Variable Values
- It is updated by reading MOOS mail on each iterate loop
- Behavior Conditions are checked against this buffer

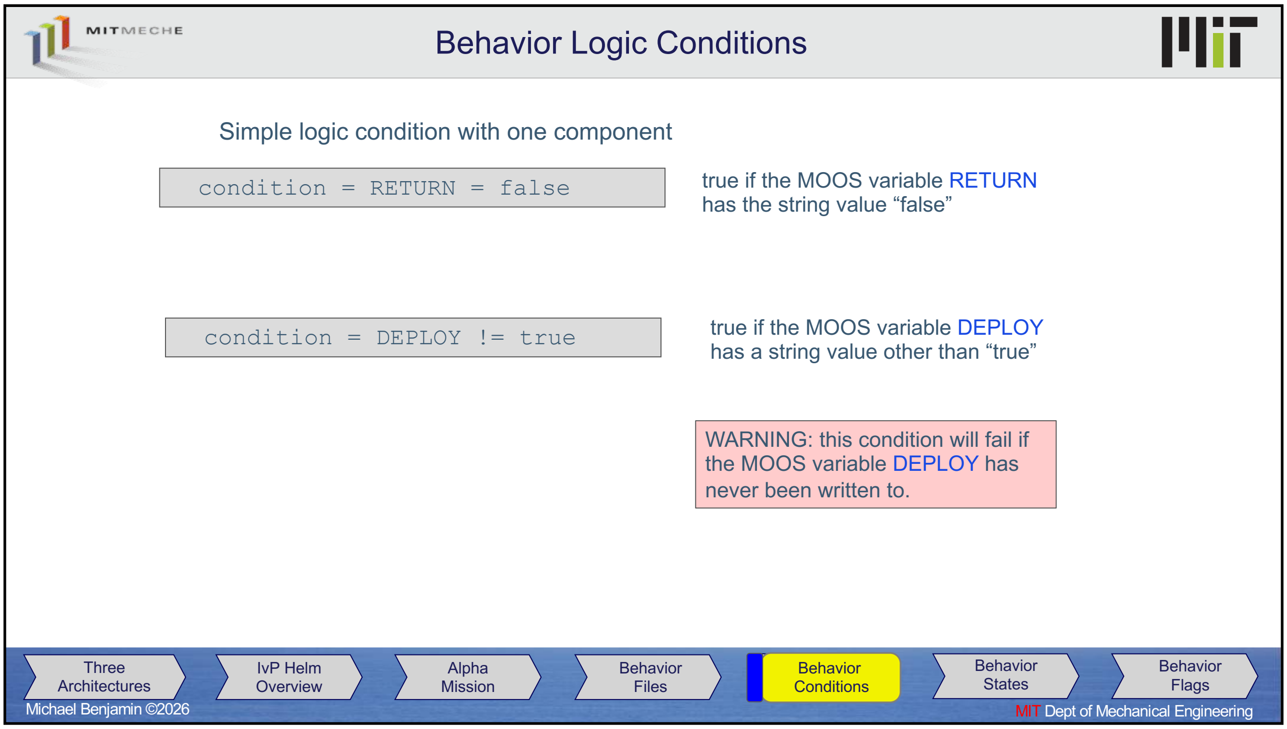
52



The Helm Information Buffer

- The helm maintains an information buffer, a cache of MOOS Variable Values
- It is updated by reading MOOS mail on each iterate loop
- Behavior Conditions are checked against this buffer

53



Behavior Logic Conditions

Simple logic condition with one component

```
condition = RETURN = false
```

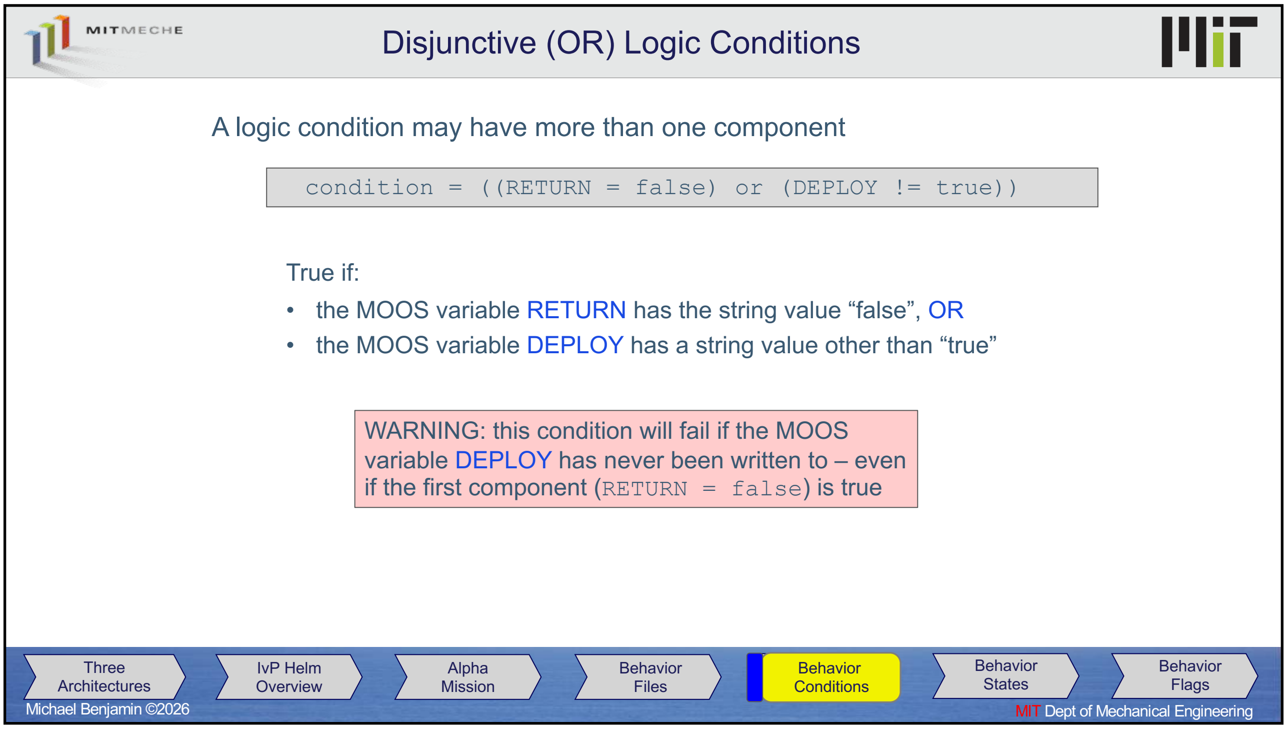
true if the MOOS variable RETURN has the string value “false”

```
condition = DEPLOY != true
```

true if the MOOS variable DEPLOY has a string value other than “true”

WARNING: this condition will fail if the MOOS variable DEPLOY has never been written to.

54



MITMECHE

Disjunctive (OR) Logic Conditions

A logic condition may have more than one component

```
condition = ((RETURN = false) or (DEPLOY != true))
```

True if:

- the MOOS variable RETURN has the string value “false”, OR
- the MOOS variable DEPLOY has a string value other than “true”

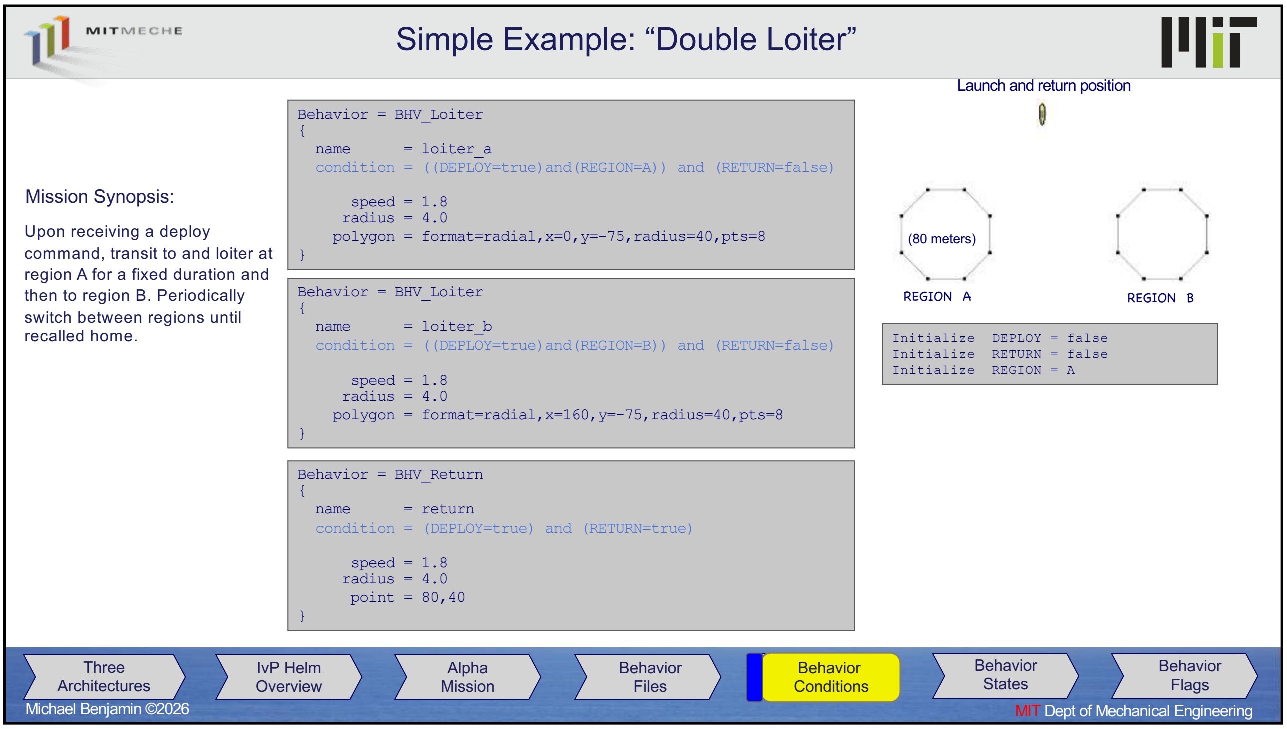
WARNING: this condition will fail if the MOOS variable DEPLOY has never been written to – even if the first component (`RETURN = false`) is true

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

55



MITMECHE

Simple Example: “Double Loiter”

Mission Synopsis:

Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

```
Behavior = BHV_Loiter
{
  name      = loiter_a
  condition = ((DEPLOY=true)and(REGION=A)) and (RETURN=false)

      speed = 1.8
     radius = 4.0
    polygon = format=radial,x=0,y=-75,radius=40,pts=8
}
```

```
Behavior = BHV_Loiter
{
  name      = loiter_b
  condition = ((DEPLOY=true)and(REGION=B)) and (RETURN=false)

      speed = 1.8
     radius = 4.0
    polygon = format=radial,x=160,y=-75,radius=40,pts=8
}
```

```
Behavior = BHV_Return
{
  name      = return
  condition = (DEPLOY=true) and (RETURN=true)

      speed = 1.8
     radius = 4.0
      point = 80,40
}
```

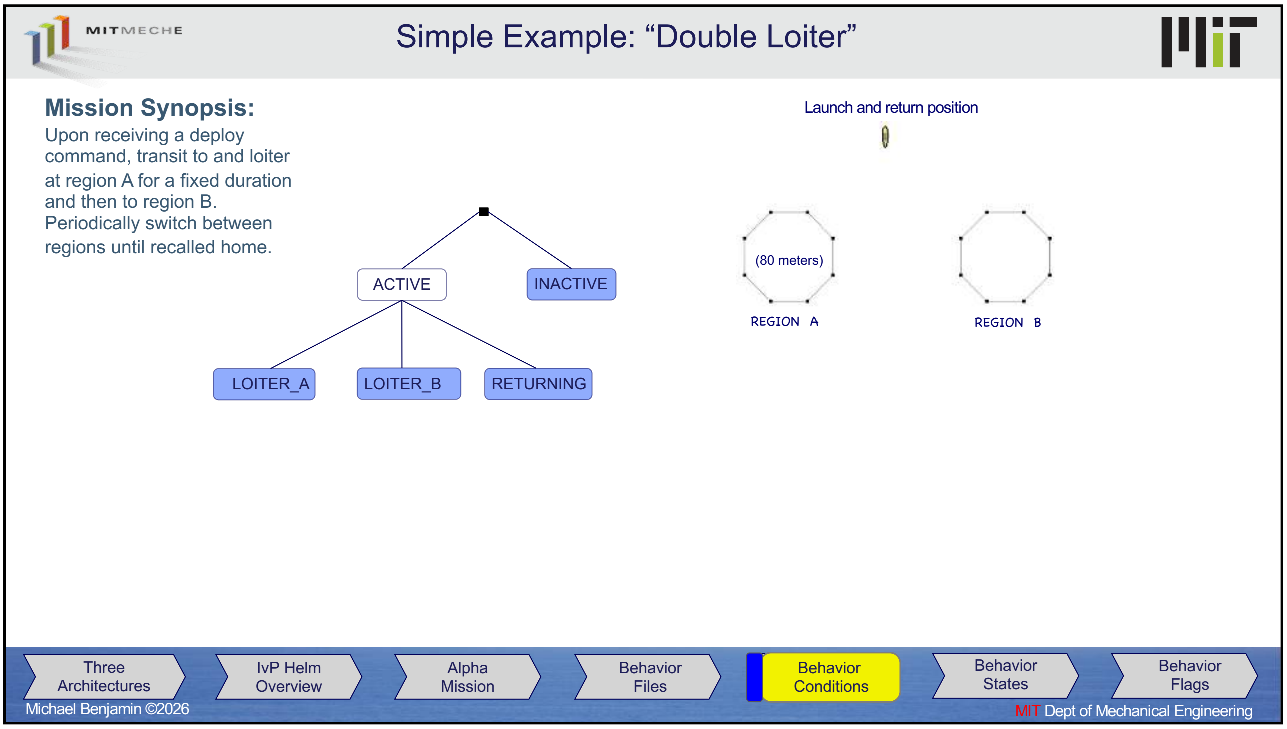
```
Initialize  DEPLOY = false
Initialize  RETURN = false
Initialize  REGION = A
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

56



MITMECHE

Simple Example: "Double Loiter"

Mission Synopsis:

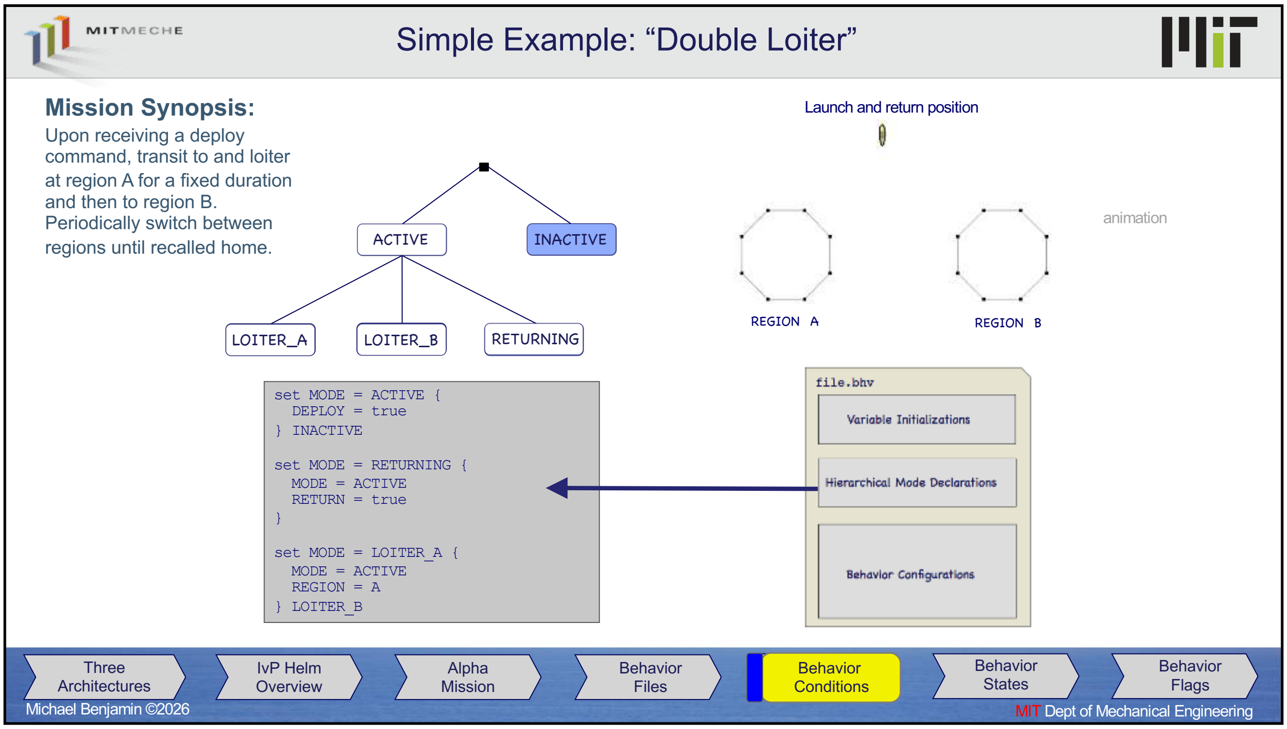
Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

57



MITMECHE

Simple Example: "Double Loiter"

Mission Synopsis:

Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

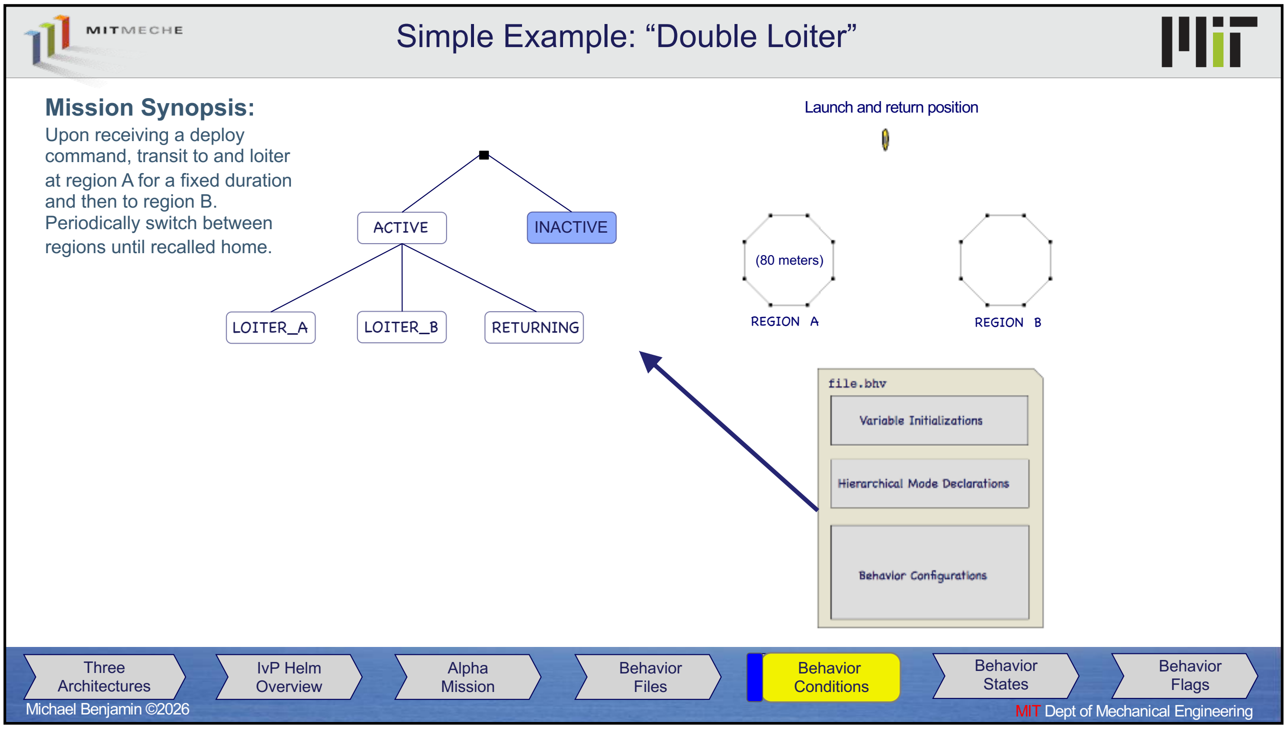
```
set MODE = ACTIVE {
  DEPLOY = true
} INACTIVE

set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

set MODE = LOITER_A {
  MODE = ACTIVE
  REGION = A
} LOITER_B
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

58



MITMECHE

Simple Example: “Double Loiter”

Mission Synopsis:

Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

Launch and return position

ACTIVE

INACTIVE

LOITER_A

LOITER_B

RETURNING

(80 meters)

REGION A

REGION B

file.bhv

Variable Initializations

Hierarchical Mode Declarations

Behavior Configurations

Three Architectures

IvP Helm Overview

Alpha Mission

Behavior Files

Behavior Conditions

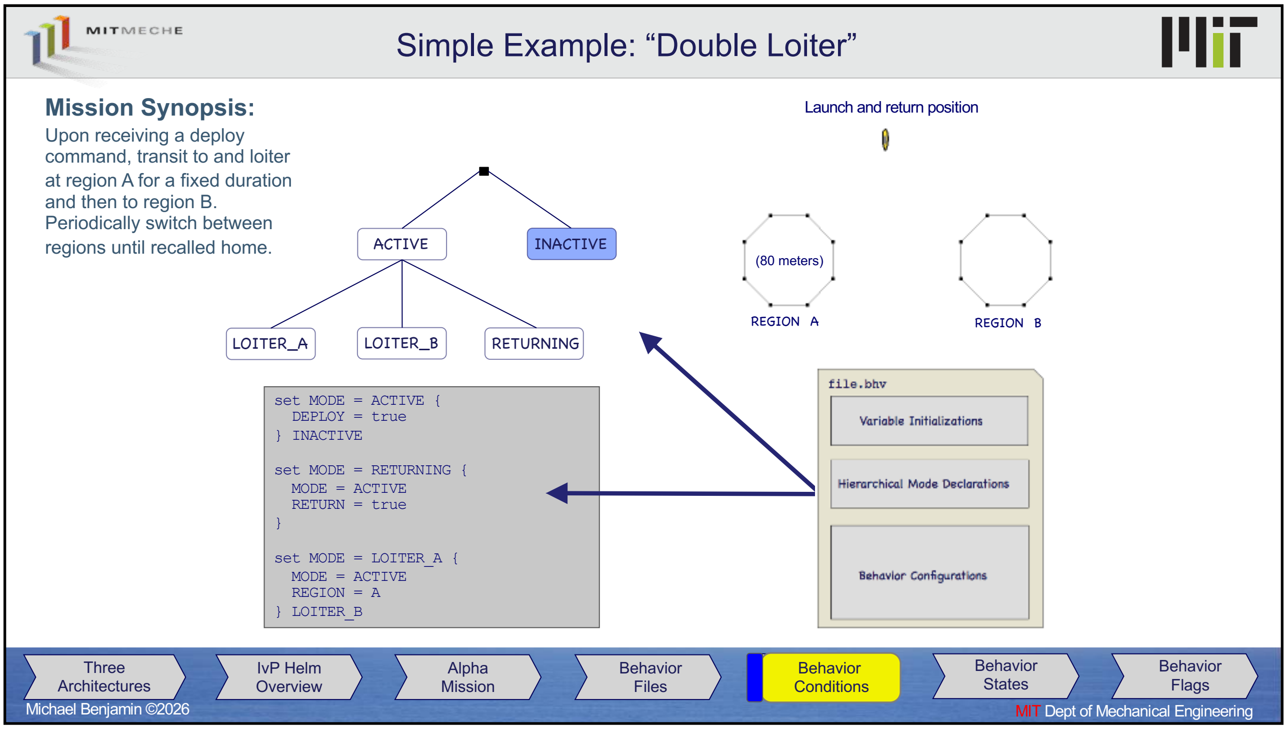
Behavior States

Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

59



MITMECHE

Simple Example: “Double Loiter”

Mission Synopsis:

Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

Launch and return position

ACTIVE

INACTIVE

LOITER_A

LOITER_B

RETURNING

(80 meters)

REGION A

REGION B

```
set MODE = ACTIVE {
  DEPLOY = true
} INACTIVE

set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

set MODE = LOITER_A {
  MODE = ACTIVE
  REGION = A
} LOITER_B
```

file.bhv

Variable Initializations

Hierarchical Mode Declarations

Behavior Configurations

Three Architectures

IvP Helm Overview

Alpha Mission

Behavior Files

Behavior Conditions

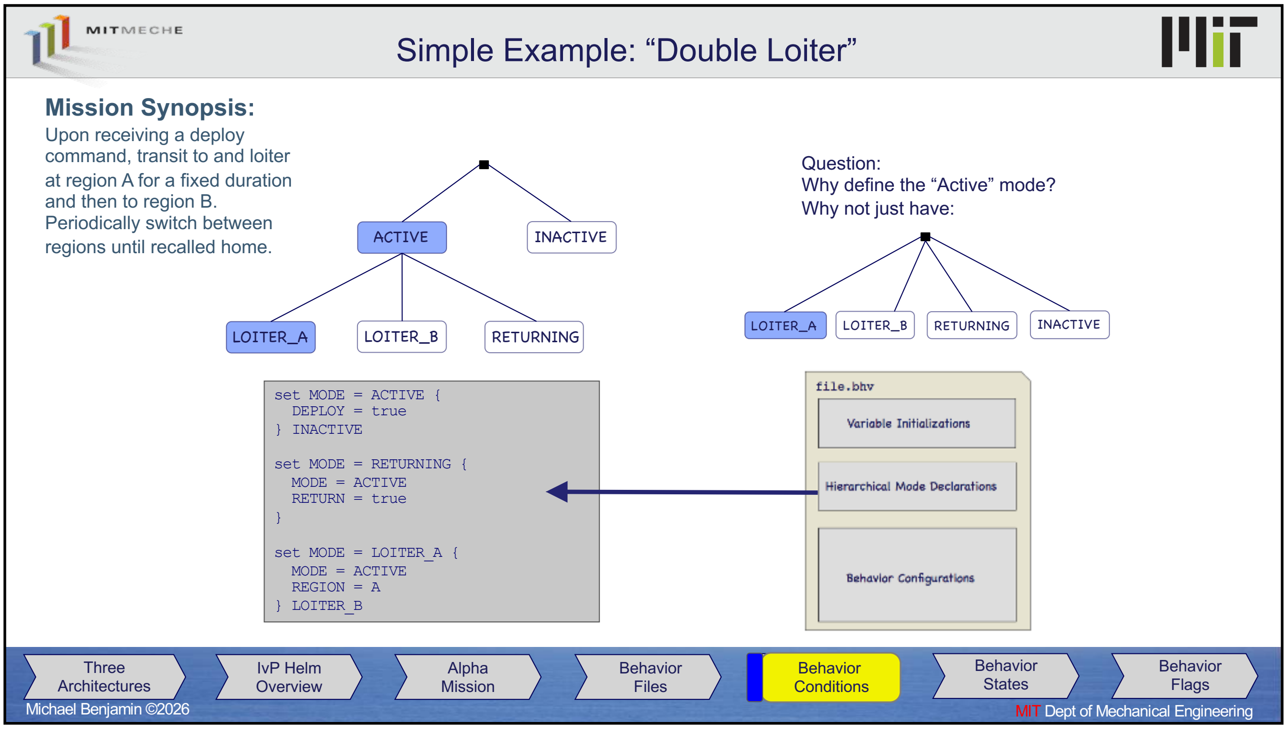
Behavior States

Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

60



Simple Example: “Double Loiter”

Mission Synopsis:

Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

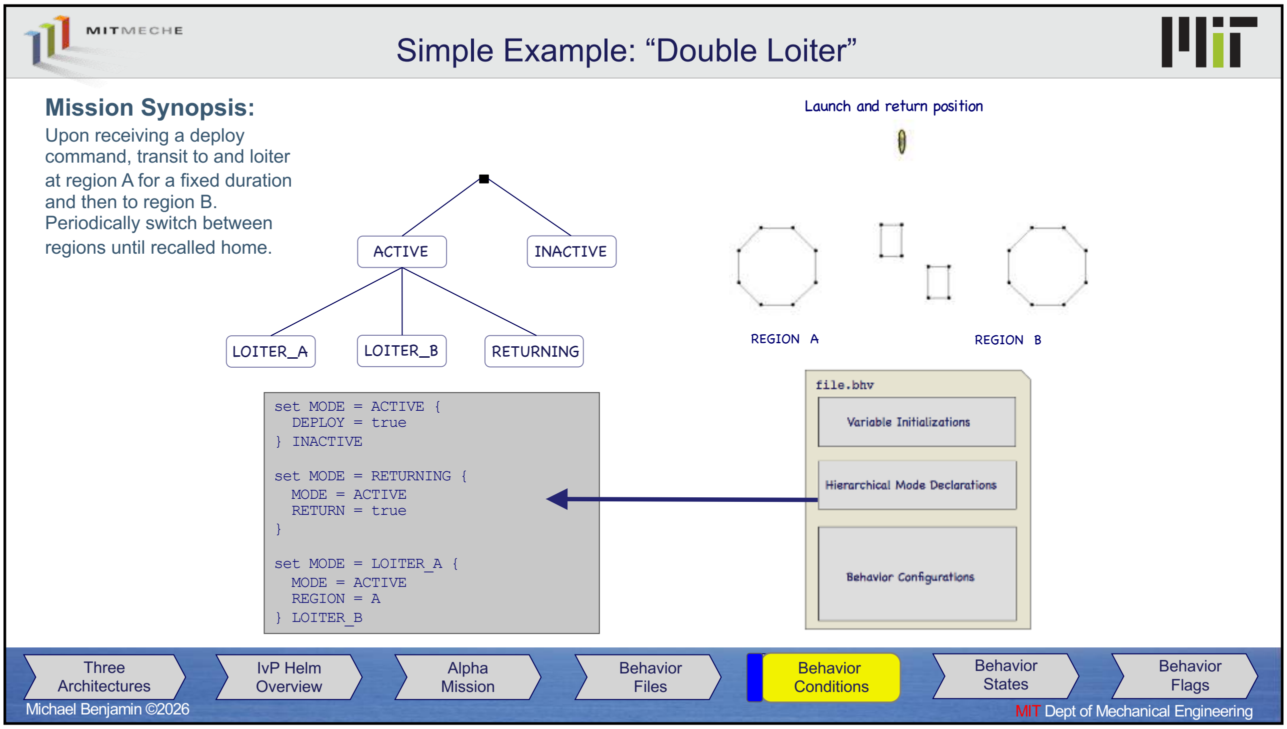
Question:
Why define the “Active” mode?
Why not just have:

```
set MODE = ACTIVE {
  DEPLOY = true
} INACTIVE

set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

set MODE = LOITER_A {
  MODE = ACTIVE
  REGION = A
} LOITER_B
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering



Simple Example: “Double Loiter”

Mission Synopsis:

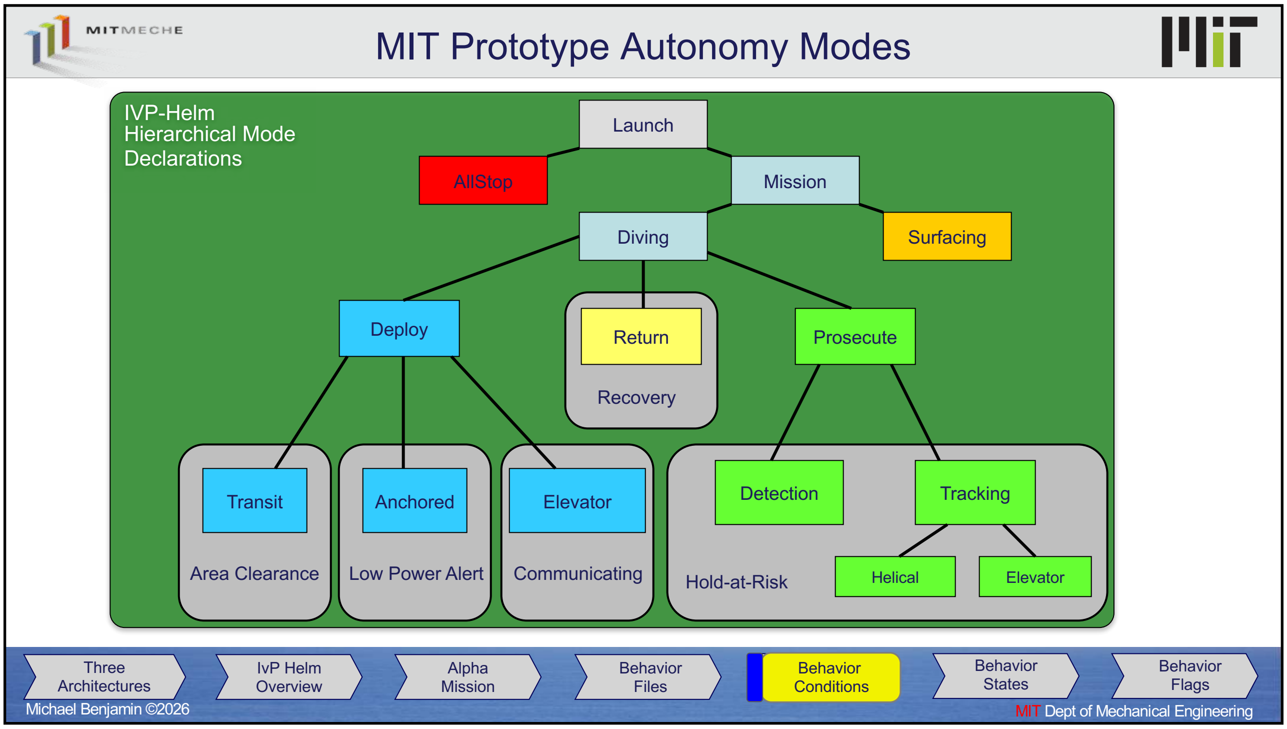
Upon receiving a deploy command, transit to and loiter at region A for a fixed duration and then to region B. Periodically switch between regions until recalled home.

```
set MODE = ACTIVE {
  DEPLOY = true
} INACTIVE

set MODE = RETURNING {
  MODE = ACTIVE
  RETURN = true
}

set MODE = LOITER_A {
  MODE = ACTIVE
  REGION = A
} LOITER_B
```

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering



MITMECHE

MIT Prototype Autonomy Modes

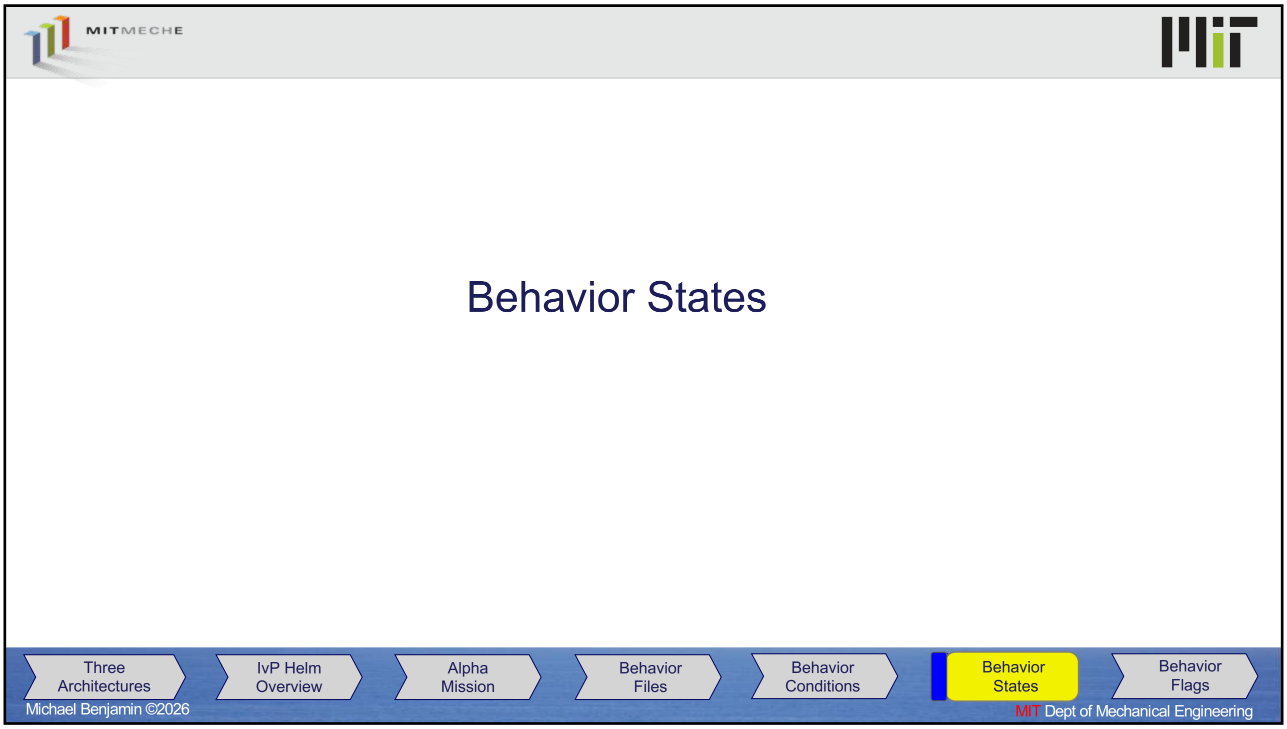
IVP-Helm
Hierarchical Mode
Declarations

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

63



MITMECHE

Behavior States

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

64

MITMECHE

Behavior States

Behaviors may be in one of four states:

Idle | Running Active | Completed

The idle state: a behavior has not met its run condition, as defined by the condition parameter.

The running state: a behavior has met its run conditions

The active state: a behavior is running state and is producing an objective function

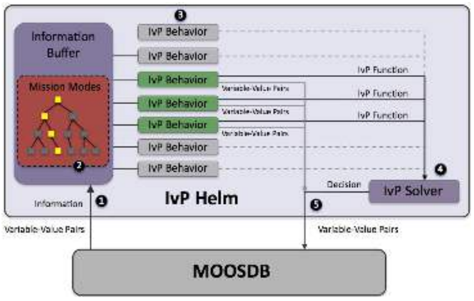
The completed state: Completion is specific to a behavior, or may be due to a duration timeout defined generally for all behaviors.

65

MITMECHE

Active vs. Running States

Idle | Running Active | Completed



The running state: behavior has met its run conditions.
The active state: behavior is running and producing an objective function.

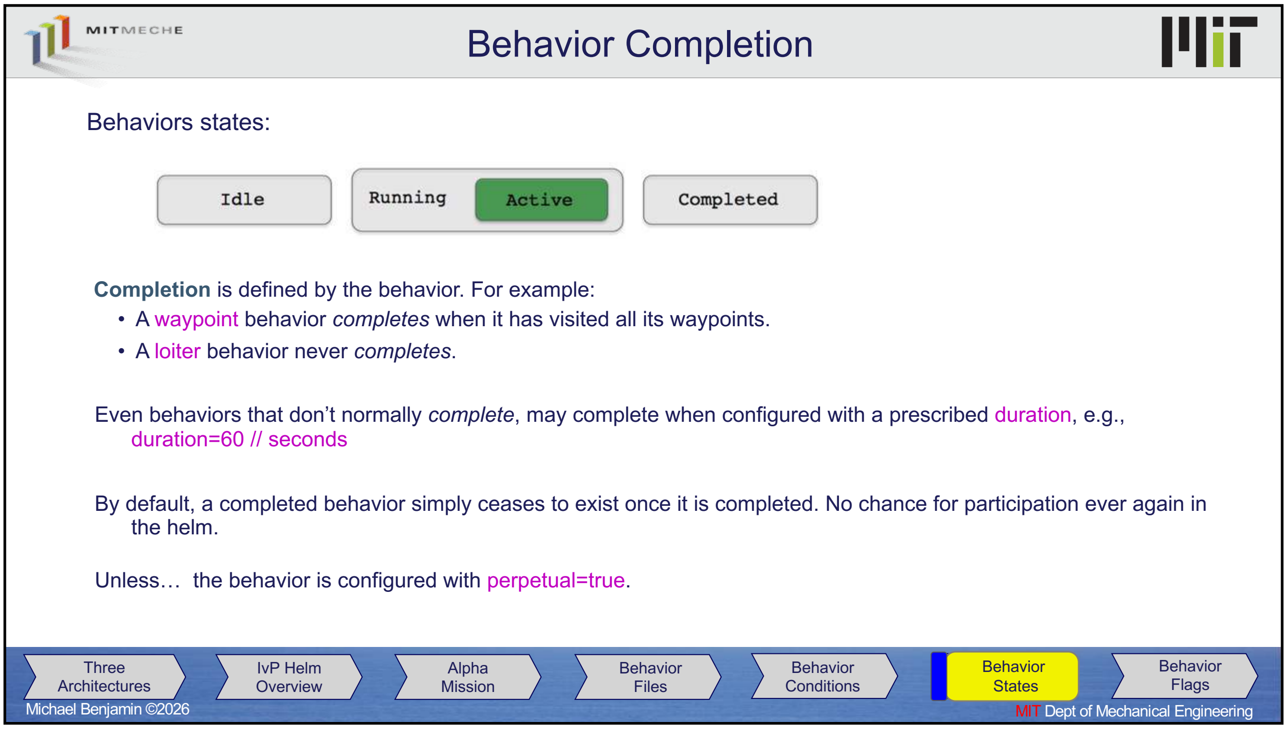
The helm's primary job is to produce a helm decision. A behavior is participating in that decision only if it is producing an objective function.

A behavior may participate in the helm decision based on:

(1) The run conditions (mostly dependent on an external decision process)

(2) The behavior's own logic (a local decision based on a more nuanced understanding of the situation).

66



MITMECHE

Behavior Completion

Behaviors states:

Idle | Running Active | Completed

Completion is defined by the behavior. For example:

- A waypoint behavior *completes* when it has visited all its waypoints.
- A loiter behavior never *completes*.

Even behaviors that don't normally *complete*, may complete when configured with a prescribed duration, e.g., duration=60 // seconds

By default, a completed behavior simply ceases to exist once it is completed. No chance for participation ever again in the helm.

Unless… the behavior is configured with perpetual=true.

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

67

MITMECHE

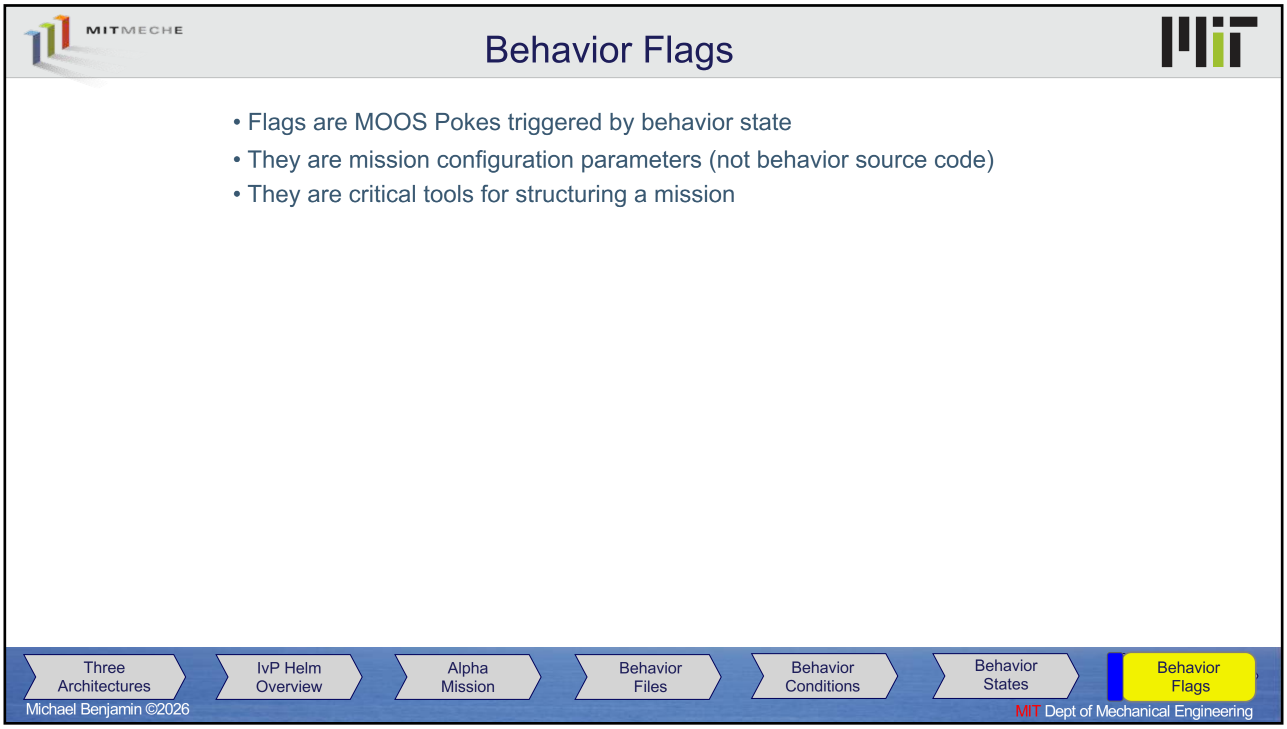
Behavior Flags

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

68



MITMECHE

Behavior Flags

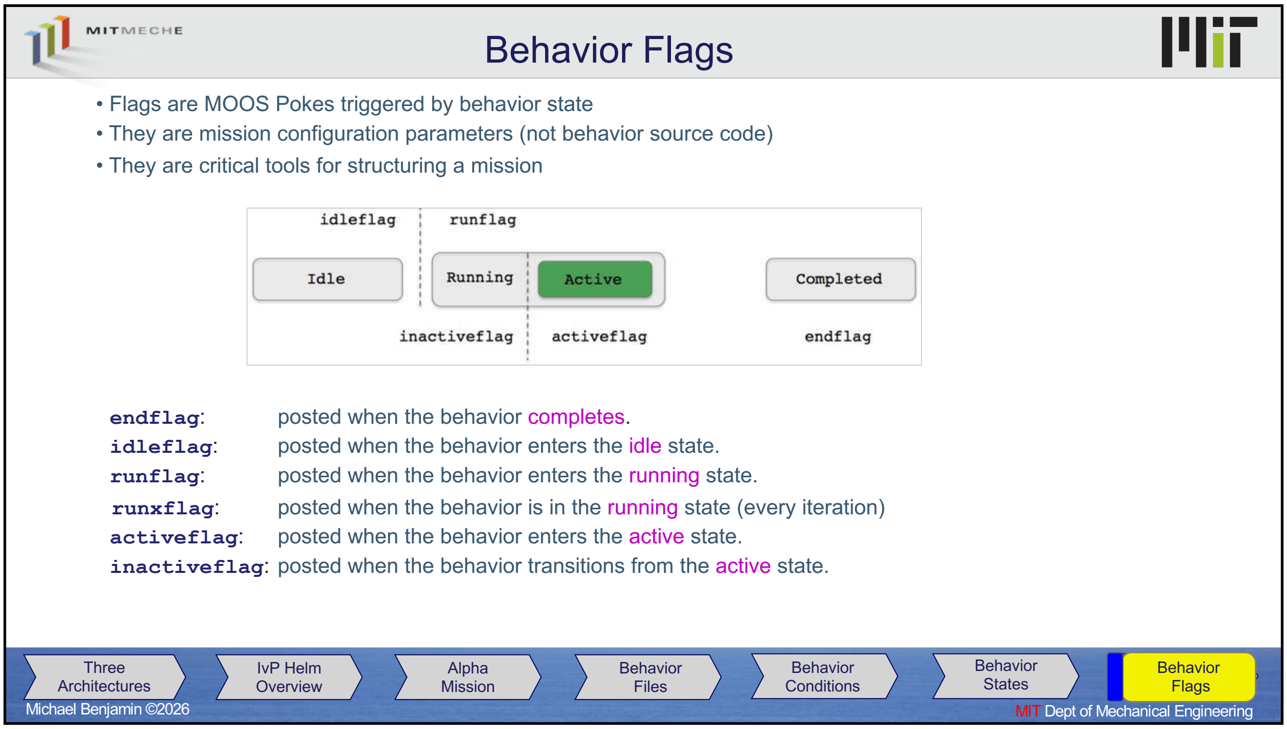
- Flags are MOOS Pokes triggered by behavior state
- They are mission configuration parameters (not behavior source code)
- They are critical tools for structuring a mission

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

69



MITMECHE

Behavior Flags

- Flags are MOOS Pokes triggered by behavior state
- They are mission configuration parameters (not behavior source code)
- They are critical tools for structuring a mission

idleflag runflag

Idle | Running | Active | Completed

inactiveflag activeflag endflag

`endflag`: posted when the behavior completes.

`idleflag`: posted when the behavior enters the idle state.

`runflag`: posted when the behavior enters the running state.

`runxflag`: posted when the behavior is in the running state (every iteration)

`activeflag`: posted when the behavior enters the active state.

`inactiveflag`: posted when the behavior transitions from the active state.

Three Architectures | IvP Helm Overview | Alpha Mission | Behavior Files | Behavior Conditions | Behavior States | Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

70

MITMECHE

End Flags

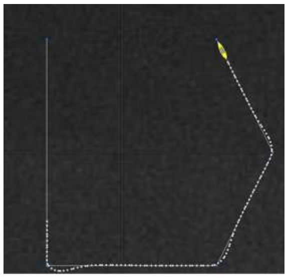
- End Flags are posted when a behavior completes
- An endflag may trigger the condition of another behavior
- Alpha mission as an example. The end of the survey behavior triggers the start of the return behavior.

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

```
Behavior = BHV_Waypoint
{
  name        = waypt_return
  pwt         = 100
  condition   = RETURN = true
  condition   = DEPLOY = true
  endflag     = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
           point = 0,-2
}
```

MITMECHE

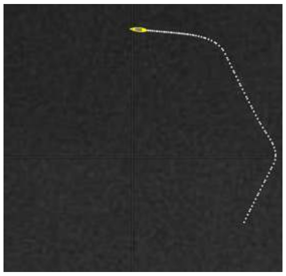
Alpha Mission End Flag Example



1 survey waypoints completes

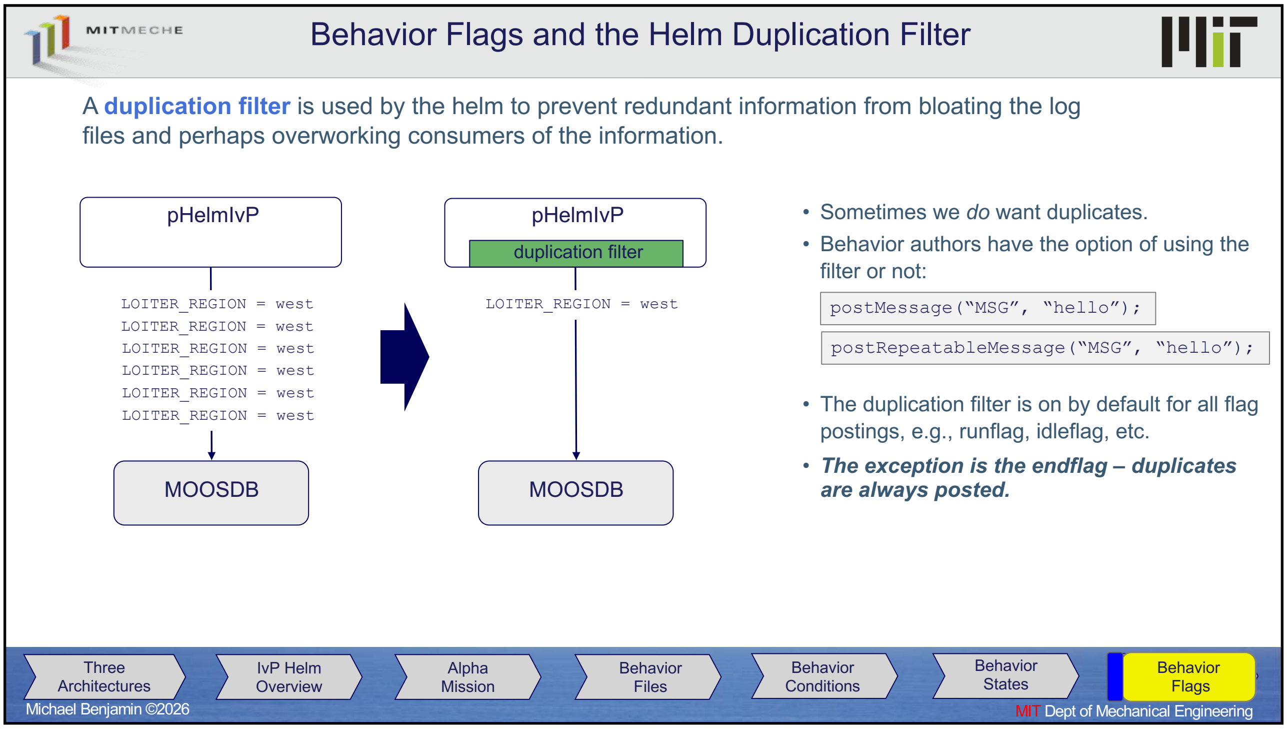
2 endflags posted
RETURN=true

3 return waypoint behavior begins



```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

           speed = 4
  capture_radius = 5.0
     slip_radius = 15.0
         polygon = 60,-40:60,-160:150,-160:180,-100:150,-40
         repeat  = 1
}
```

```
Behavior = BHV_Waypoint
{
  name        = waypt_return
  pwt         = 100
  condition   = RETURN = true
  condition   = DEPLOY = true
  endflag     = DEPLOY = false

           speed = 2.0
  capture_radius = 2.0
     slip_radius = 8.0
           point = 0,-2
}
```



MITMECHE

Behavior Flags and the Helm Duplication Filter

A **duplication filter** is used by the helm to prevent redundant information from bloating the log files and perhaps overworking consumers of the information.

pHelmIvP

```
LOITER_REGION = west
LOITER_REGION = west
LOITER_REGION = west
LOITER_REGION = west
LOITER_REGION = west
LOITER_REGION = west
```

MOOSDB

pHelmIvP

duplication filter

```
LOITER_REGION = west
```

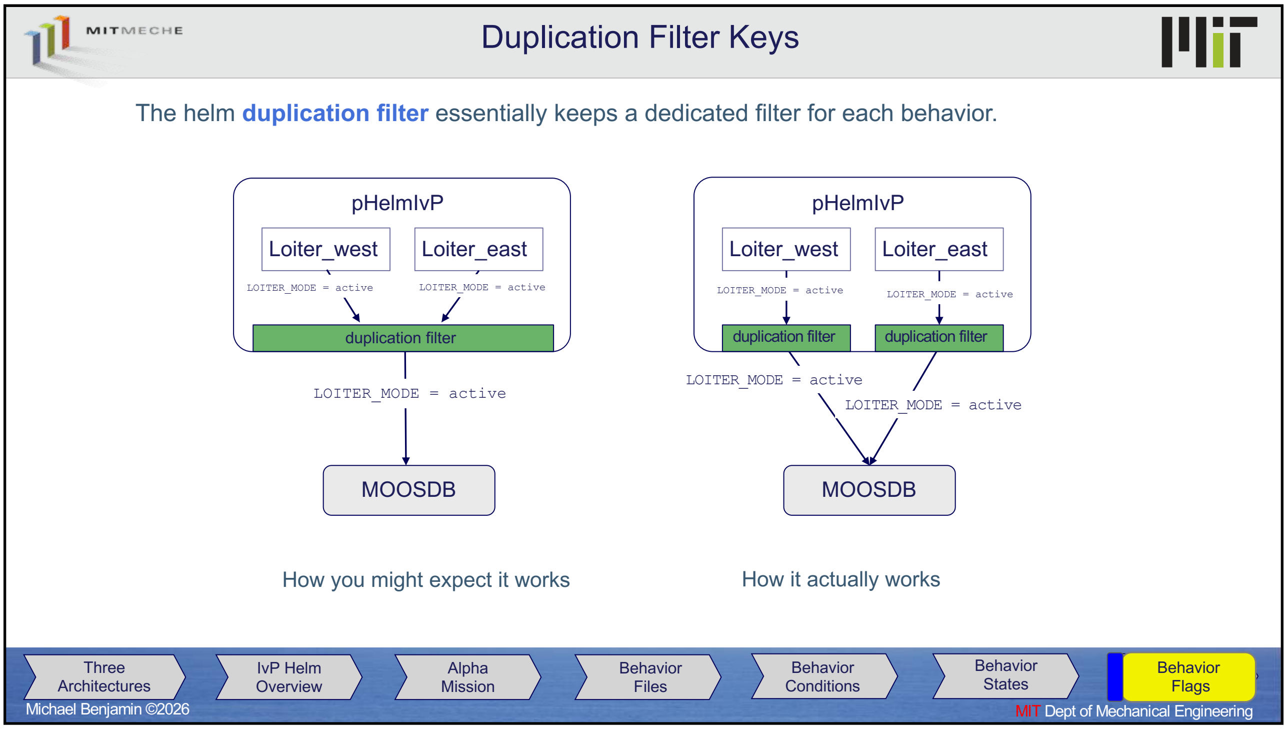
MOOSDB

- Sometimes we *do* want duplicates.
- Behavior authors have the option of using the filter or not:

```
postMessage("MSG", "hello");
```

```
postRepeatableMessage("MSG", "hello");
```

- The duplication filter is on by default for all flag postings, e.g., runflag, idleflag, etc.
- ***The exception is the endflag – duplicates are always posted.***



MITMECHE

Duplication Filter Keys

The helm **duplication filter** essentially keeps a dedicated filter for each behavior.

pHelmIvP

Loiter_west

Loiter_east

```
LOITER_MODE = active
LOITER_MODE = active
```

duplication filter

```
LOITER_MODE = active
```

MOOSDB

How you might expect it works

pHelmIvP

Loiter_west

Loiter_east

```
LOITER_MODE = active
LOITER_MODE = active
```

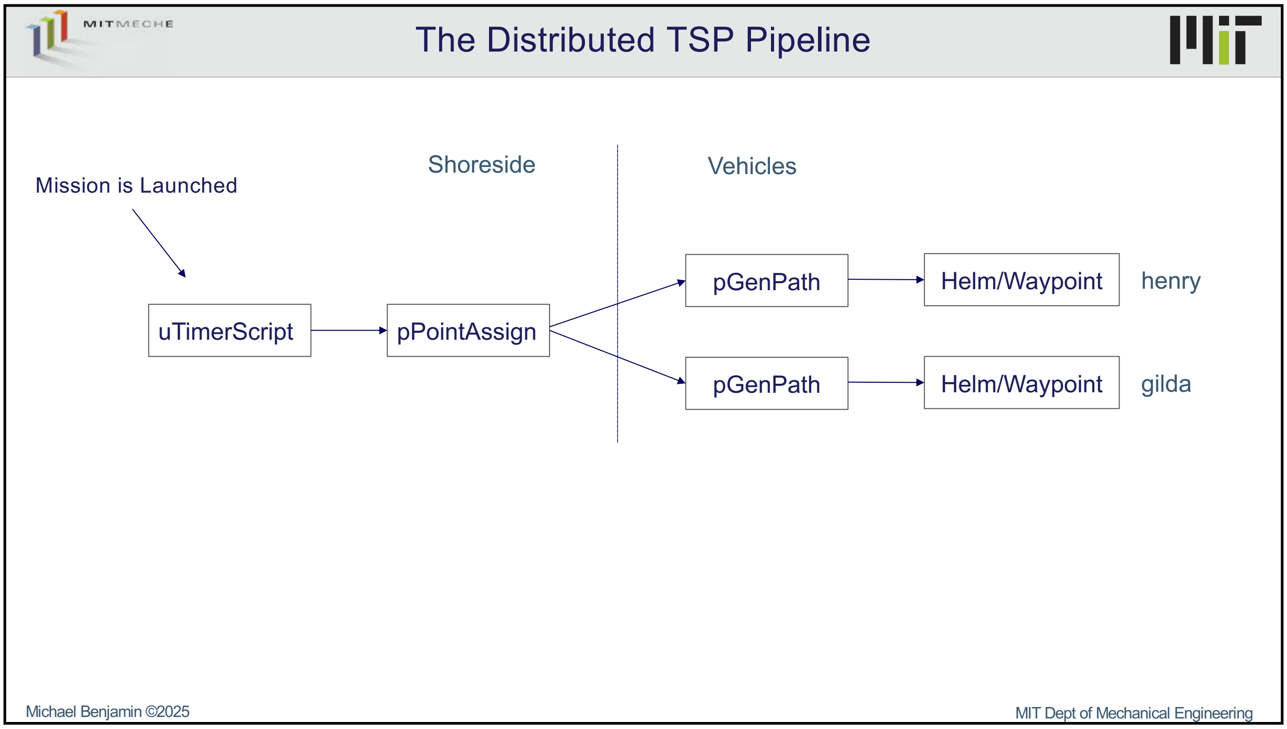
duplication filter

duplication filter

```
LOITER_MODE = active
LOITER_MODE = active
```

MOOSDB

How it actually works



The Distributed TSP Pipeline

Shoreside

Vehicles

Mission is Launched

uTimerScript

pPointAssign

pGenPath

Helm/Waypoint

henry

pGenPath

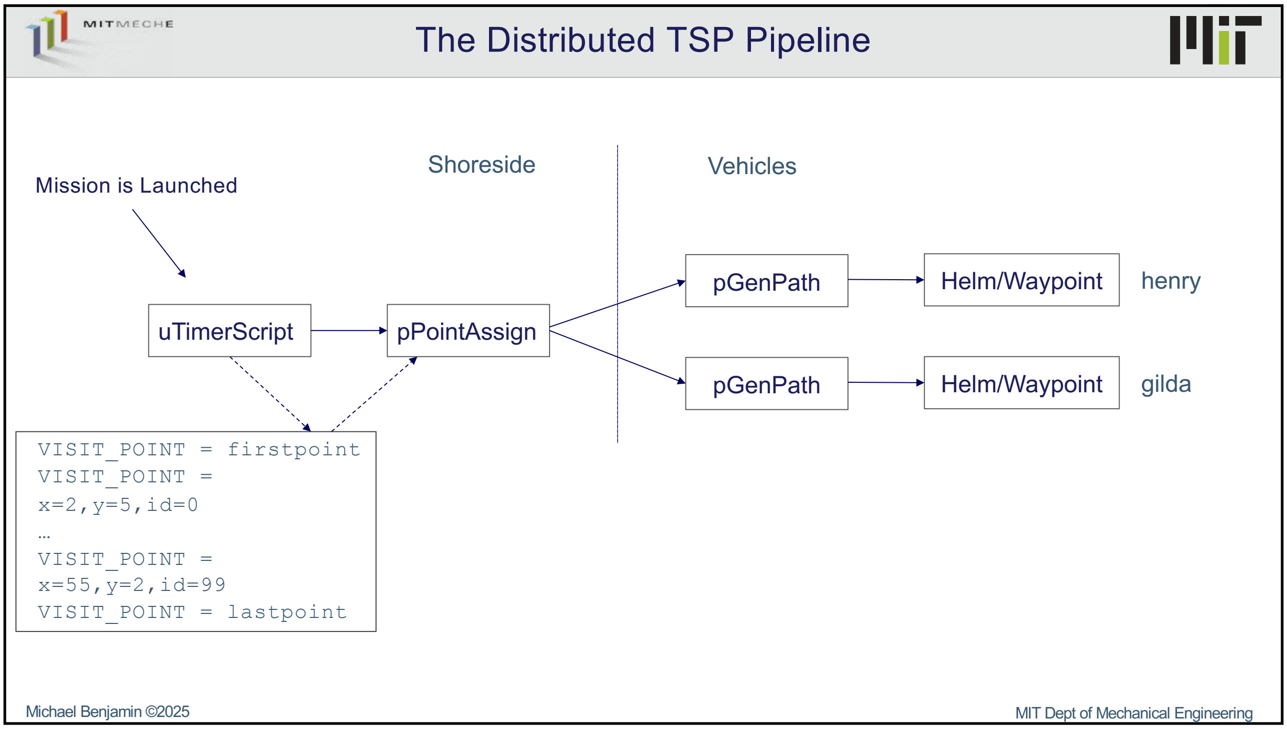
Helm/Waypoint

gilda

Michael Benjamin ©2025

MIT Dept of Mechanical Engineering

75



The Distributed TSP Pipeline

Shoreside

Vehicles

Mission is Launched

uTimerScript

pPointAssign

pGenPath

Helm/Waypoint

henry

pGenPath

Helm/Waypoint

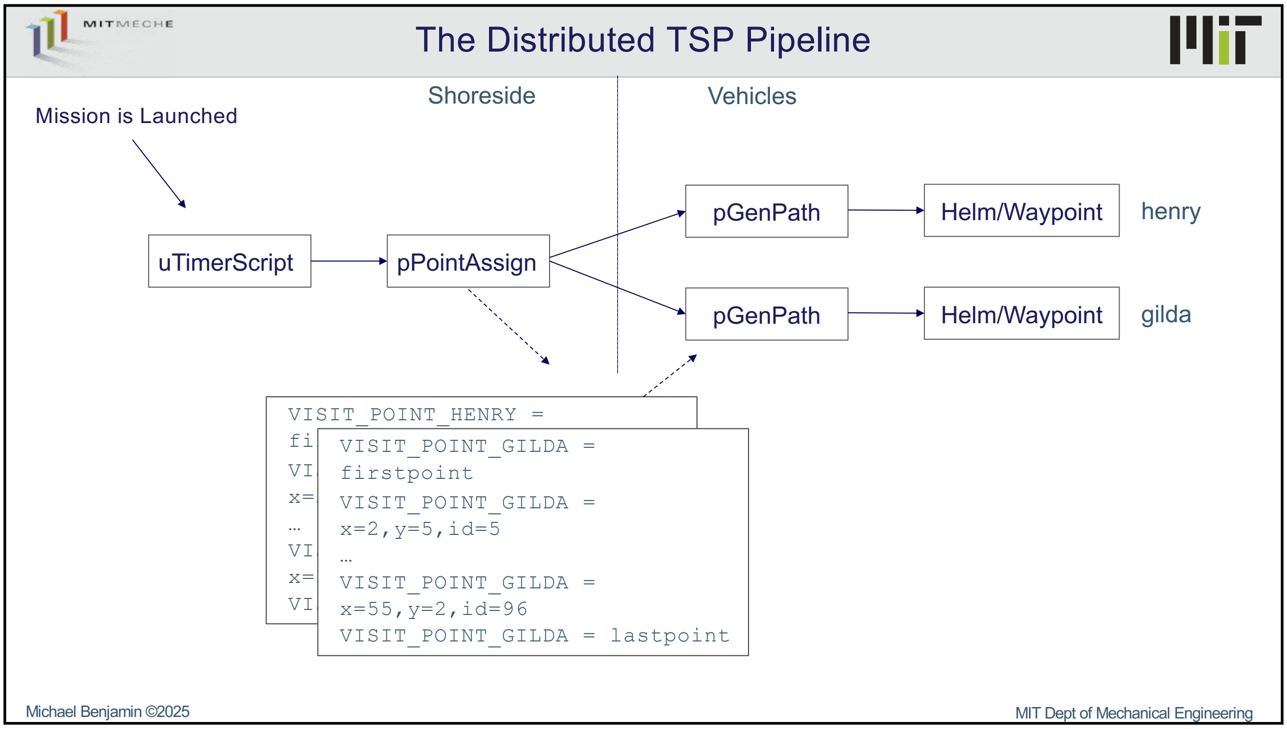
gilda

```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```

Michael Benjamin ©2025

MIT Dept of Mechanical Engineering

76



The Distributed TSP Pipeline

Mission is Launched

Shoreside

Vehicles

uTimerScript

pPointAssign

pGenPath

Helm/Waypoint

henry

pGenPath

Helm/Waypoint

gilda

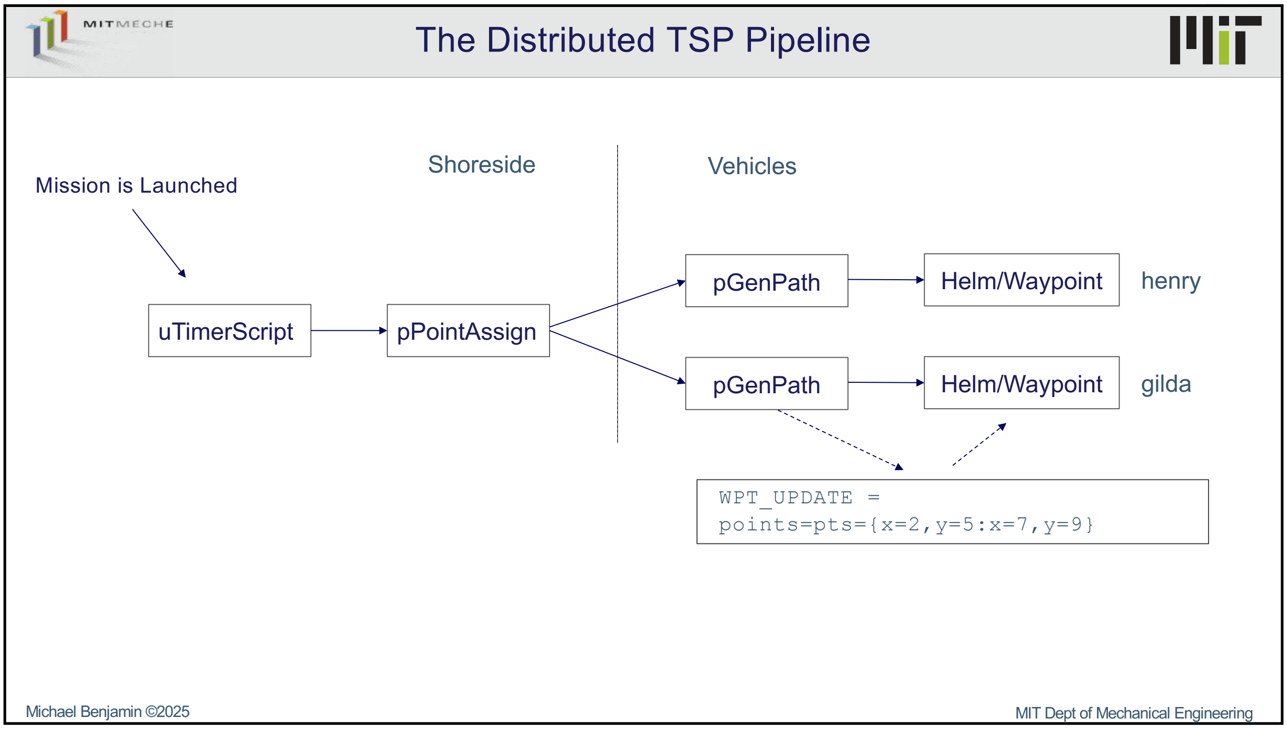
```
VISIT_POINT_HENRY =
fi
VI
x=
…
VI
x=
VI
```

```
VISIT_POINT_GILDA =
firstpoint
VISIT_POINT_GILDA =
x=2,y=5,id=5
…
VISIT_POINT_GILDA =
x=55,y=2,id=96
VISIT_POINT_GILDA = lastpoint
```

Michael Benjamin ©2025

MIT Dept of Mechanical Engineering

77



The Distributed TSP Pipeline

Mission is Launched

Shoreside

Vehicles

uTimerScript

pPointAssign

pGenPath

Helm/Waypoint

henry

pGenPath

Helm/Waypoint

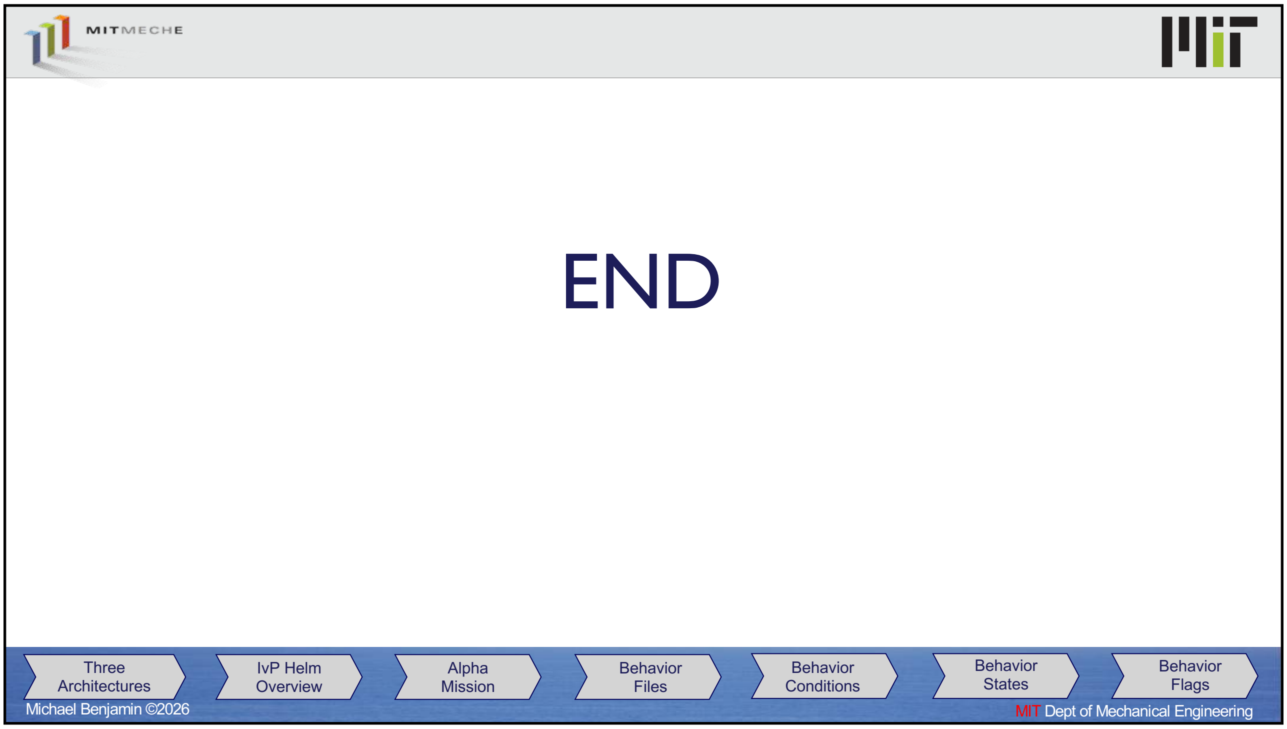
gilda

```
WPT_UPDATE =
points=pts={x=2,y=5:x=7,y=9}
```

Michael Benjamin ©2025

MIT Dept of Mechanical Engineering

78



MITMECHE

END

Three Architectures

IvP Helm Overview

Alpha Mission

Behavior Files

Behavior Conditions

Behavior States

Behavior Flags

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

79