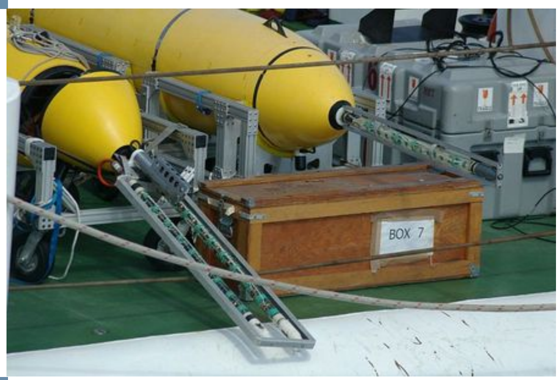
The IvP Helm and New Features of MOOS-IvP 4.1



Michael R. Benjamin
NAVSEA, Division Newport RI

Phone: 401-832-4148
Email: michael.r.benjamin@navy.mil

Acknowledgments

Collaborators:

This work is the product of a multi-year collaboration between the Center for Advanced System Technologies (CAST), Code 2501, of the Naval Undersea Warfare Center in Newport Rhode Island and the Department of Mechanical Engineering and the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology in Cambridge Massachusetts, and the Oxford University Mobile Robotics Group.

Sponsors:

The IvP Helm autonomy software and the basic research involved in the interval programming model for multi-objective optimization has been developed under support from ONR Code 311 (Program Managers Dr. Don Wagner and Dr. Behzad Kamgar-Parsi). Prior prototype development of IvP concepts benefited from the support of the In-house Laboratory Independent Research (ILIR) program at the Naval Undersea Warfare Center in Newport RI.

Outline

- The IvP Helm and Architecture Motivations
- New capabilities in Release 4.1
- Plans for Development

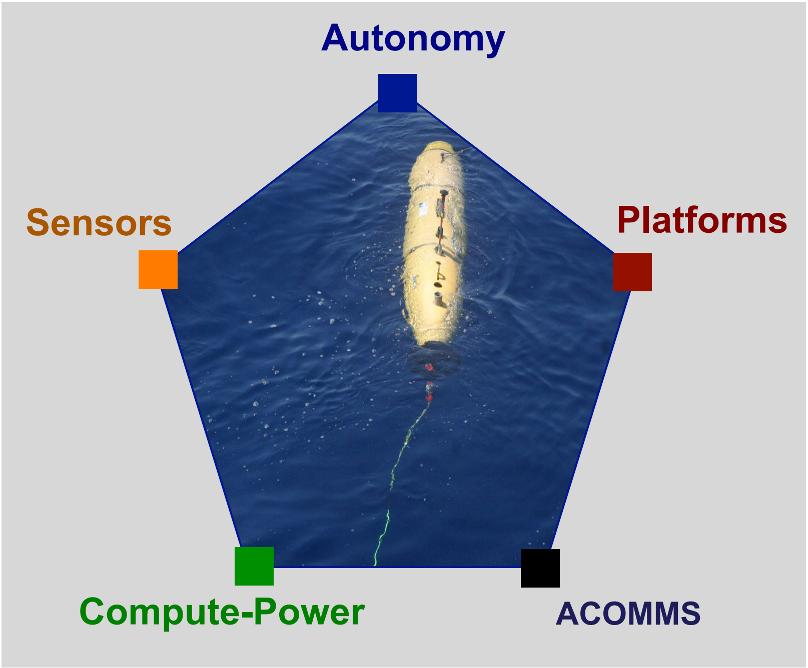
Component Technologies in an Unmanned System

Platforms are becoming cheaper, smaller and able to persist longer.

Sensors are becoming smaller, more capable and cheaper - available on more platforms.

Computing power on-board is making live sensor processing and decision making based on sensors possible.

Acoustic communications between vehicles is making collaboration between vehicles possible and contributing to greater deployment persistence.

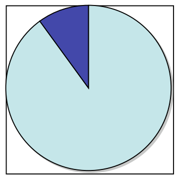


Each trend affects what is required and desired from the ***Autonomy System***

Unmanned Vehicles and System Complexity and Cost

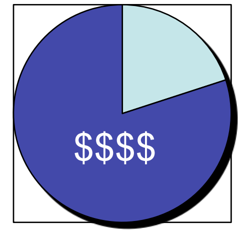
System Complexity and Cost

Autonomy



Analogy to a PC in 1985
- The machine dominated the cost.
- Few choices in software.

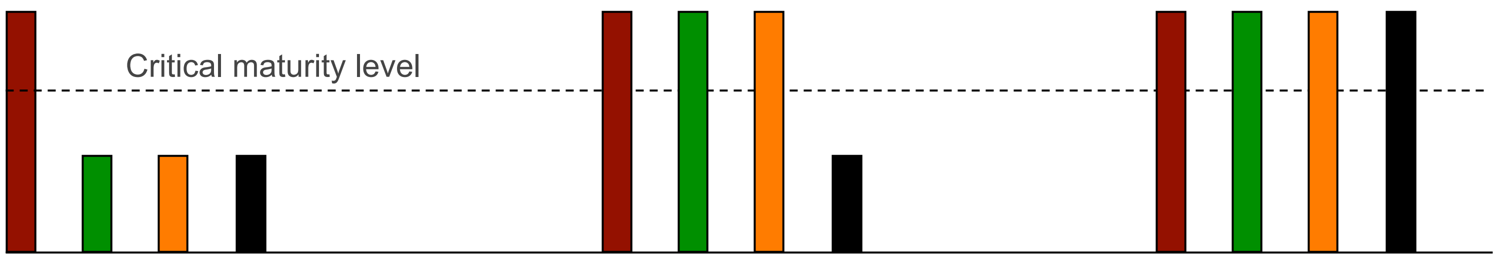
Autonomy



Autonomy Capability:

Deterministic/Canned → Adaptive/Dynamic → Collaborative

Critical Components:



Platforms ACOMMS Sensors Compute-Power | Platforms ACOMMS Sensors Compute-Power | Platforms ACOMMS Sensors Compute-Power

Time: - - - - - - - - - - →

1995 2005 2010

Conclusions Drawn from Observing Current Trends

- **Autonomy/software development needs to be nimble.**
  Why? Platforms, hardware, communications technology and missions evolve quickly.
  - Platform independence is key.
  - Not beholden to any one software provider
  - The infrastructure should be non-proprietary

- **S&T Development and Procurement need to be mindful of software costs.**
  Why? This will be the dominant part of the overall vehicle cost as autonomy matures.
  - cost of initial development
  - cost of upgrades
  - cost of validation
  - cost of not being able to utilize the most effective algorithms due to proprietary issues.

- **Nested Capabilities is key to controlling software costs and rapid development**
  Why? No single organization has the expertise to build the most effective system.
  - Three software tiers: (a) public (b) limited distribution (c) proprietary or classified.
  - This is not just “software re-use”.
  - There is no central policy maker

MOOS-IvP
The "3-Architecture" Autonomy Paradigm

#1 – Separation of Vehicle Autonomy from the Physical Platform

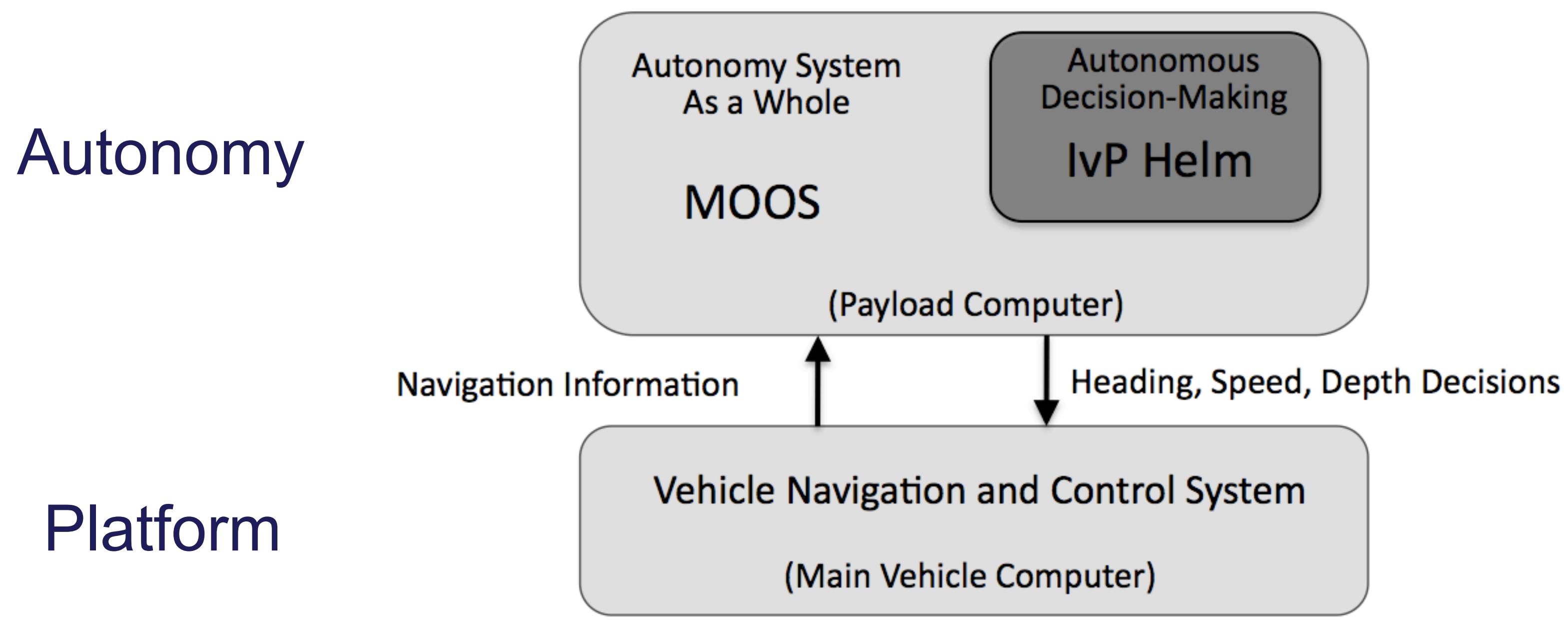
#2 – Separation of the Autonomy System Components

#3 – Separation of Autonomy into Dedicated Distinct Behaviors

The "3-Architecture" Autonomy Paradigm
Principle #1

Architecture Principle #1 – Separation of Vehicle Autonomy from the Physical Platform

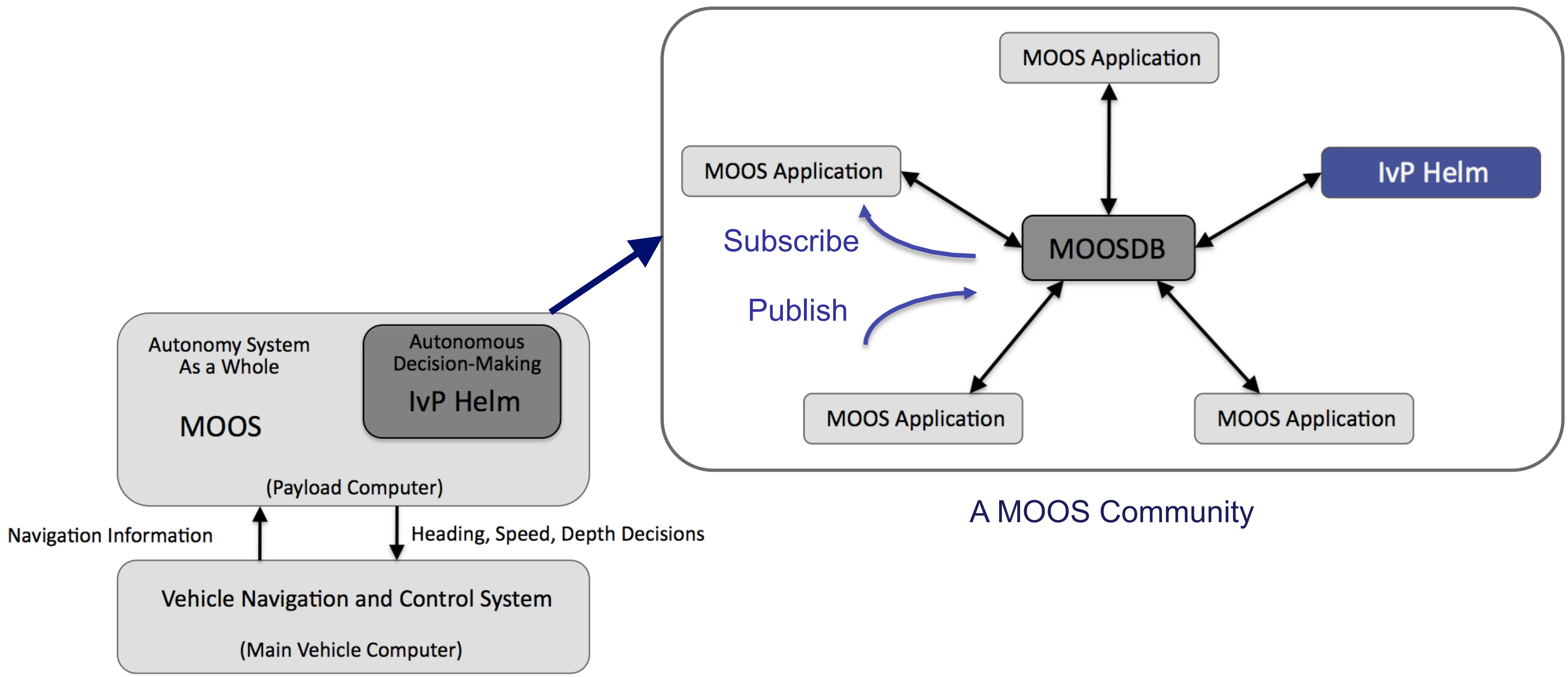
- The autonomy system runs on the vehicle payload computer and provides a series of commands comprised of *heading, speed, depth* values
- The main vehicle computer implements vehicle control (converting heading and speed commands to rudder and thrust actuator commands) and provides the autonomy system with navigation information, and sensor information.



The “3-Architecture” Autonomy Paradigm
Principle #2

Architecture Principle #2 – Separation of Autonomy System Components (MOOS)

- MOOS is middleware built on the publish-subscribe architecture.
- Each MOOS application is a separate process running on the vehicle computer.
- Each process is defined by the messages it publishes and the messages it subscribes for.

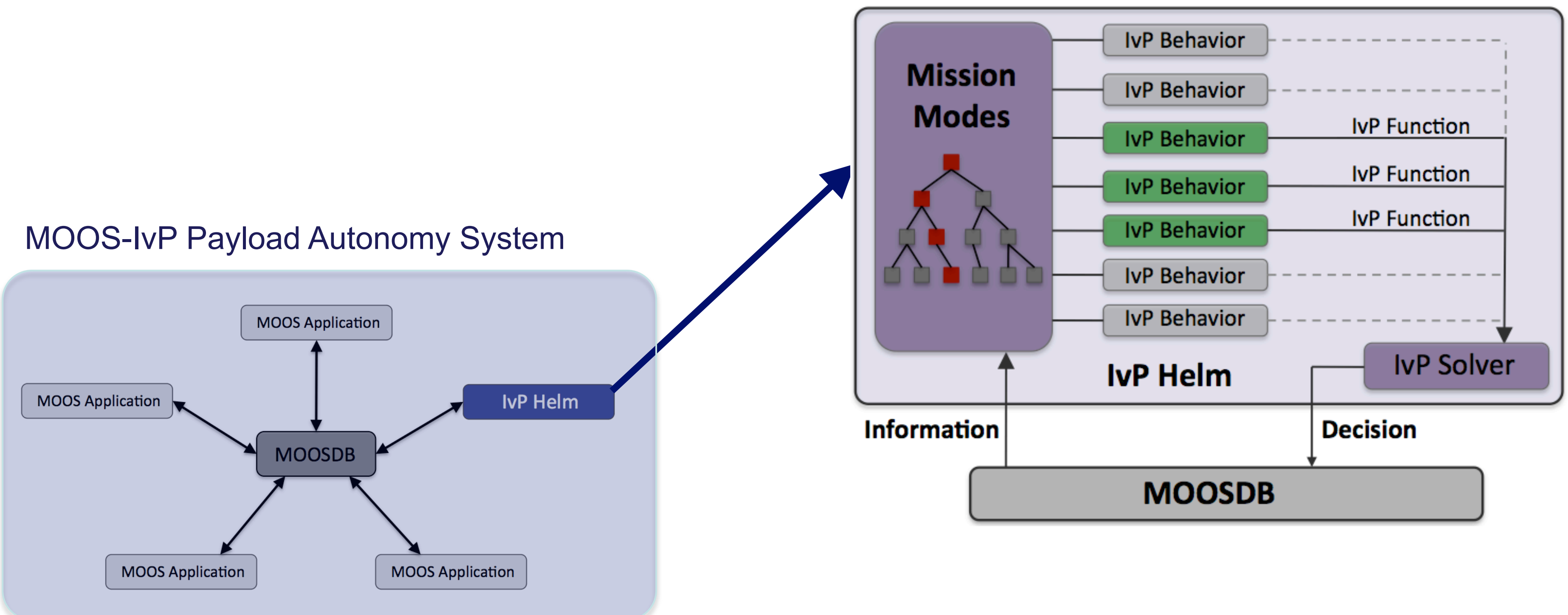


A MOOS Community

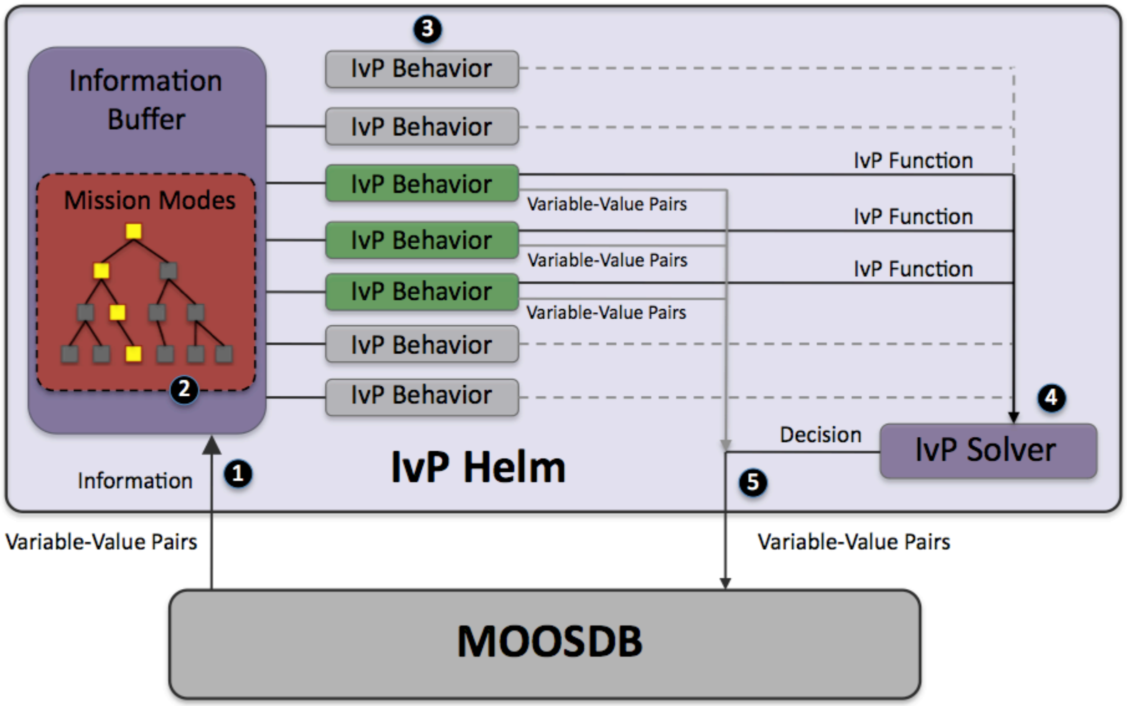
The “3-Architecture” Autonomy Paradigm
Principle #3

Architecture Principle #3 – Separation of Autonomy into dedicated distinct behaviors.

- The IvP Helm is a decision-making engine based on the behavior-based architecture. It is a single MOOS application comprised of multiple specialized behaviors.
- Behaviors are turned on or off based on defined situations (states) and transitions. When multiple behaviors are active, coordination is by multi-objective optimization.
- Interval Programming (IvP) is the technique used for multi-objective optimization.

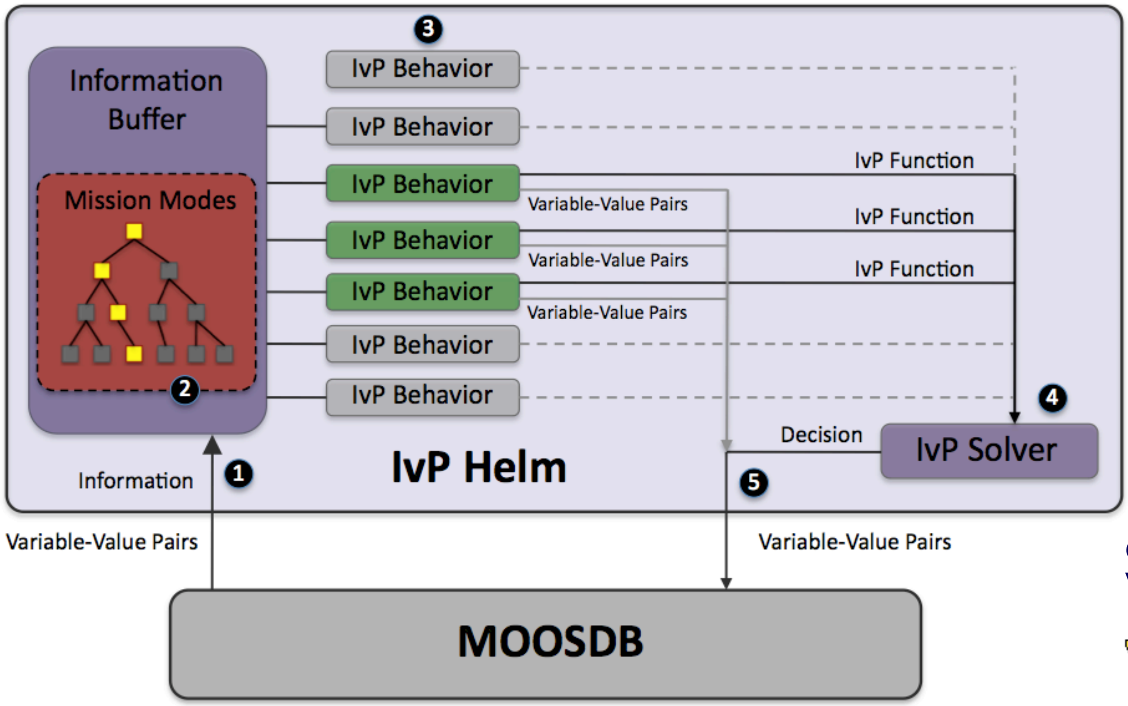


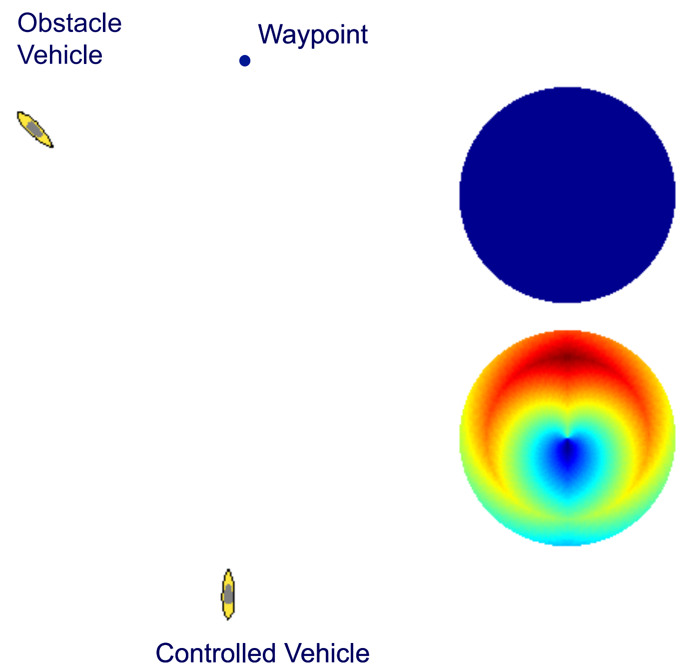
The Helm Iterate Loop



1. Mail is read in the MOOS OnNewMail() function and applied to a local buffer.
2. The helm mode is determined, and 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.

The Helm Iterate Loop





1. Mail is read.
2. Helm mode is determined.
3. Behaviors generate their output.
4. Competing behaviors are resolved.
5. The Helm posts its results

Non-Traditional Aspects of Behavior-Based Control in the IvP Helm

- Behaviors have state.
- Behaviors influence each other between iterations.
- Behaviors accept externally generated plans.
- There may be several instances of the same behavior.
- Behaviors may spawn and die dynamically based on events or commands.
- Behaviors may run in a configurable sequence.
- Behaviors rate actions over a coupled decision space (multi-objective optimization)

In short, this is not Rodney Brooks' Behavior Based Control, but the power of independent, incremental development has been retained. Unleashed by the power of Open Source development, and wide, diverse collaborations.

The "3-Architecture" Autonomy Paradigm

#1 – Separation of Vehicle Autonomy from the Physical Platform

#2 – Separation of the Autonomy System Components

#3 – Separation of Autonomy into Dedicated Distinct Behaviors

CHOICES (Is that good or bad?)

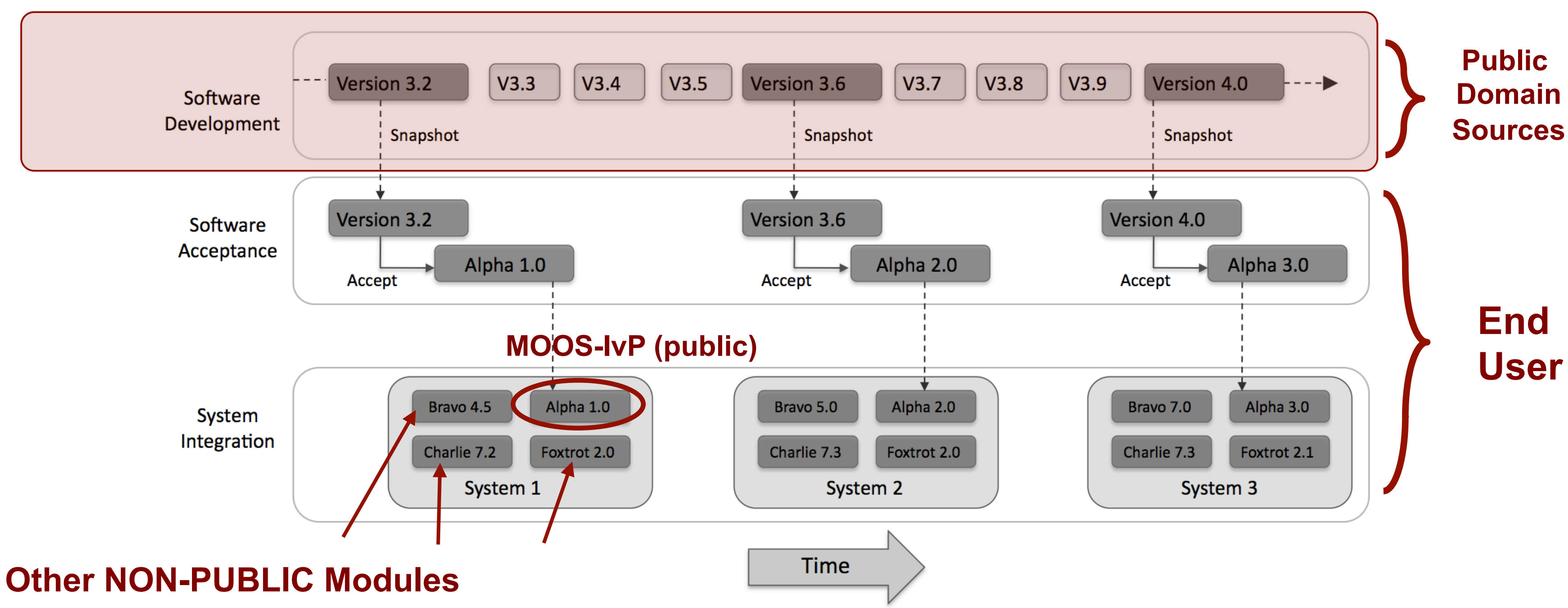
The "3-Architecture" Autonomy Paradigm

#1 – Separation of Vehicle Autonomy from the Physical Platform

#2 – Separation of the Autonomy System Components

#3 – Separation of Autonomy into Dedicated Distinct Behaviors

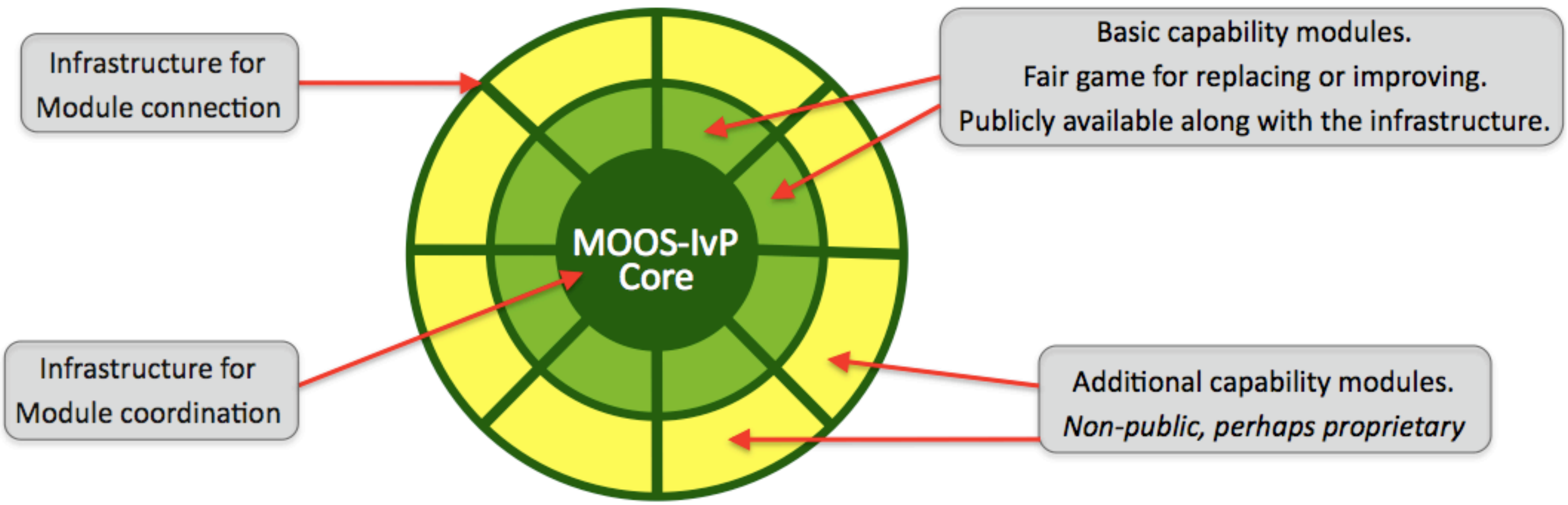
Integration and software development proceed ***independently*** from one another.



Public Infrastructure – Nested Capabilities

An autonomy system has components with different capabilities, and distribution access.

- Publicly accessible modules providing infrastructure, basic capabilities
- FOUO Modules accessible for isolated developers of a particular project (MCM, ASW)



Autonomy System = Infrastructure + Modules

- Core Infrastructure, tools and autonomy – www.moos-ivp.org.

- Navy/Project specific add-on modules - available via restricted access servers.

What is MOOS-IvP

MOOS – Everything you've come to expect and love from the Oxford distribution.

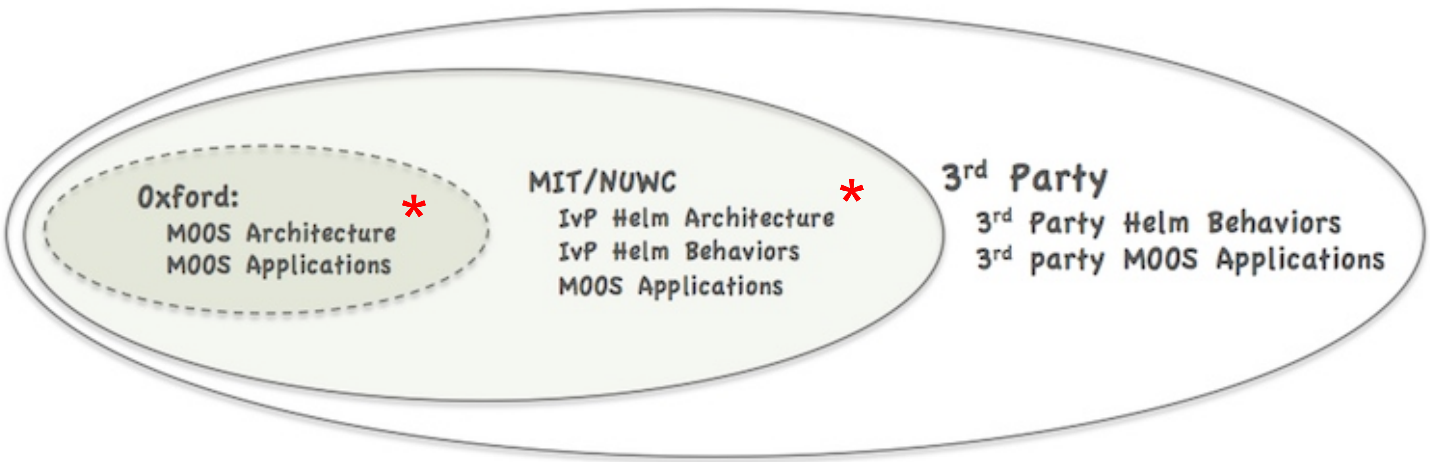
- MOOSDB
- pLogger
- pAntler
- pMOOSBridge
- uMS
- iMatlab
- uPlayback
- pScheduler
- iRemote

- IvP Helm – A behavior based helm and extendable set of behaviors
  - StationKeep
  - PeriodicSurface
  - MinAltitude
  - AvoidCollision
  - ConstantHeading
  - ConstantSpeed
  - ConstantDepth
  - OpRegion
  - PeriodicSpeed
  - CutRange
  - AvoidObstacles
  - MemoryTurnLimit
  - Trail
  - Loiter
  - Timer
  - Waypoint

- IvP Tools – A set of utility applications
  - pNodeReporter
  - uHelmScope
  - pMarineViewer
  - uTimerScript
  - pBasicContactMgr
  - pEchoVar
  - uXMS
  - uProcessWatch
  - uPokeDB
  - uTermCommand
  - Alogscan
  - aloggrep
  - Alogrm
  - Alogclip
  - Alogview
  - aloghelm

Nested Repositories

- MOOS, from the Mobile Robotics Group at Oxford
- MOOS-IvP, from the Laboratory for Autonomous Marine Sensing Systems at MIT
- 3rd Party (Your) modules.



* Architecture Definition and Implementation

Developed MOOS-IvP Modules (173 Modules, 11 Organizations)

Unrestricted (public domain)

Unrestricted (user-to-user)

Restricted (FNC or INP funded)

(As of September 2009)

PUBLIC Modules Oxford (9) (Newman)

- MOOSDB
- pLogger
- iRemote
- uMS
- pMOOSBridge
- uPlayback
- iMatlab
- pAntler
- pScheduler

PUBLIC Modules NUWC/MIT (20) (Benjamin)

- pHelmIvP
- pMarinePID
- uHelmScope
- iMarineSim
- aloggrep
- alogscan
- pMarineViewer
- pNodeReporter
- uTermCommand
- uProcessWatch
- alogrm
- alogclip
- alogview
- uPokeDB
- uXMS
- pEchoVar
- uFunctionVis
- uTimerScript
- geoview
- nsplug

UCCI Modules (26)

(Idaho)
- iSerialPort
- pArtifactPost
- pBlast
- pCommMatrix
- pCrisper
- pEnergyMonitor
- pFleetControl
- pFuzzifier
- pInference
- uFuseGrid

(Idaho)
- pMultiSweep
- pSweepLines

(Panama City)
- pATR
- pNSWC
- pDTSP

(NUWC/ASCM)
- iParserAIS
- iPlaybackAIS
- pASCM_PK
- iRawAIS

(MIT)
- pDR
- pCNEKF
- pSimDVL

(APL-UW)
- iCommStack

(JHU)
- pAutonomyBridge
- pSafety
- iSportScan

PLUS INP MODULES (24)

(MIT -Schmidt)
- pSealab
- pSubIndex
- pTargetOpportunity
- pTrackMonitor
- pTrackQuality
- pCBF
- pBTracker

(MIT - Schmidt)
- pNoiseSim
- pBeamForm
- pPlusnetMessages

(MIT/Metron)
- pNodeStar

(Duke - Kemp)
- pFDM

(MIT -Eickstedt)
- p1HTracker
- pMBTracker
- pSearch
- pMessageSim

(SAIC)
- iVSA
- pAEL

(MIT/SAIC)
- pBearings_VSA
- pBearings_Generic

(MIT/Lincoln)
- pDSPMessanger
- pMultiVSASim

Autonomy, AComms and Sensor Modules - MIT/NUWC (94)

(patrikilakis)
- MOOSBlink
- iPNICompass
- iPWMController
- iWinch
- iAISNMEA
- iCTDSBE49
- pGPSReTx
- MOOSDump
- iOS5000
- iMOOS2Serial
- iGPS_CV
- iGPS
- pSamplingControl
- zlogger
- iModemSim

(schneider)
- pACOMMSHandler
- pACOMMSPoller
- iArduinoDIO
- pCTDCodec
- pCTDLogger
- pREMUSCodec
- iSerialNMEA
- iMOOS2SQL
- iWHOIModem
- uTPB

(schneider)
- iAcousticSim
- pGeneralCodec
- pBTRCodec
- pSoundSpeed
- pTopside2NaFCon
- pCoroner
- pVirtualTether
- iCommander
- iModemComms
- iModemWinch
- iWebsite
- pClusterPriority
- pCoroner
- pHarborTargetSim
- pMailReader
- pXYToLatLon

(battle)
- iHFA
- uMVS_Bluefin

(arjuna)
- iCTD
- iOEX
- iHuxley
- pNaFCon
- pBeamform_SLITA
- pRBTracker

(henrik)
- pArraySim
- pBearingTrack
- pFeatureSim
- pGPSSim
- pGPSOffset
- pGateway
- pLaunch
- pMissionMonitor
- pMultiTargetSim
- pMultiBearingSim
- pTargetSimAIS
- pTrackQuality
- pMultiAcousticSim

(eickstedt)
- pBearingsSim
- pMessageSim
- iMultiStaticSonar
- iGSC_16AO16

(petillo)
- iMseas
- iMseasBathy
- pEnvtGrad
- pHelmToSlocum

(sideleau)
- iGSC
- iOceanServer

(cockrell)
- pSimpleAcousticSim
- pTargetSim
- uNafconSplitter

(joint authors)
- iDAS
- pBearings_DURIP
- pCSVLogger
- pLaunch
- pLoopback
- pNavGlobalize
- uBathy
- uCtdSim2
- uMsgSim
- pRemus
- pActivePassiveMgr

(jshusta)
- pLinkSource
- pLinkSink
- pLinkLog
- pLinkPollSrv

(raylum)
- pBFMF

(leederkerken)

pDR

iXBow440

iSICK

Outline

- The IvP Helm and Architecture Motivations
- New capabilities in Release 4.1
- Plans for Development

New Capabilities in Release 4.1

Highlights:

- Dynamic Behavior Spawning
- Scripting – the uTimerScript application
- Contact Management – the pBasicContactMgr application

Dynamic Behavior Spawning

What is Dynamic Behavior Spawning?

- Behaviors may be defined as templates - instances spawned upon external events.
- Behavior may be built to die under certain conditions, and post MOOS messages immediately prior to dying.

Motivation:

- For certain behaviors, e.g., collision avoidance, contact tracking, multiple instances of the behavior are required, one for each contact.
- It's virtually impossible to know the amount or type of contacts encountered prior to the start of the mission.

Configuring Behaviors with Dynamic Behavior Spawning

Old Way:

```
Behavior = BHV_AvoidCollision
{
  name        = avd_collision
  pwt         = 200
  condition   = AVOID=true
  updates     = CONTACT_INFO

                     contact = macrura
  active_outer_distance = 50
  active_inner_distance = 20
     completed_distance = 75
     collision_distance = 8
     all_clear_distance = 25
           active_grade = linear
       on_no_contact_ok = true
            extrapolate = true
                  decay = 30,60
}
```

New Way:

```
Behavior = BHV_AvoidCollision
{
  name        = avd_collision
  pwt         = 200
  condition   = AVOID=true
  updates     = CONTACT_INFO
  endflag     = CONTACT_RESOLVED = $[CONTACT]
  templating  = spawn

                     contact = to-be-set
  active_outer_distance = 50
  active_inner_distance = 20
     completed_distance = 75
     collision_distance = 8
     all_clear_distance = 25
           active_grade = linear
       on_no_contact_ok = true
            extrapolate = true
                  decay = 30,60
}
```

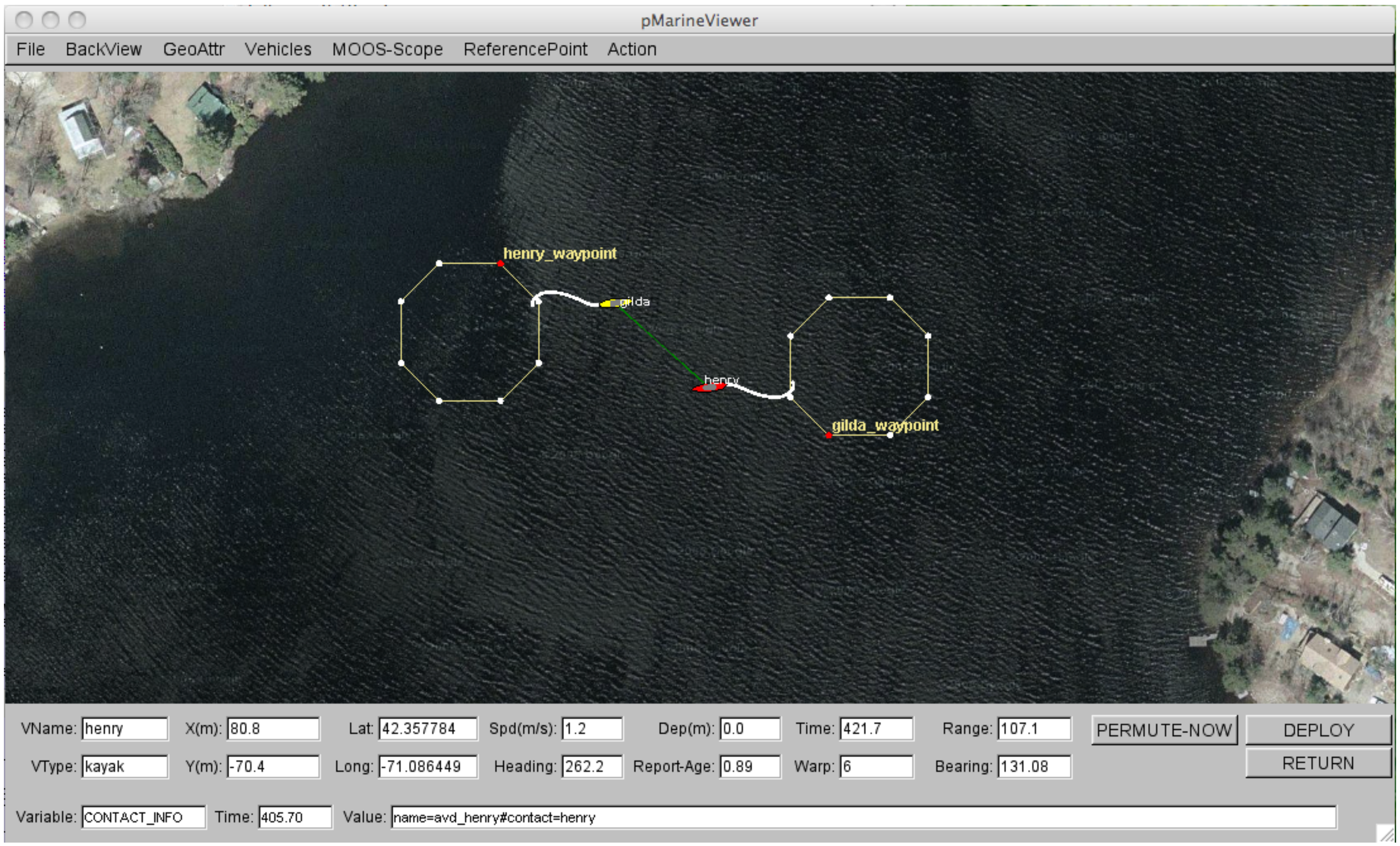
MOOS Post `CONTACT_INFO = “name=avd_macrura # contact=macrura”`

MOOS Post `CONTACT_INFO = “name=avd_henry # contact=henry”`

The Berta Example Mission with Dynamic Behavior Spawning

The Berta example mission:

- In moos-ivp/trunk/missions/m2_berta
- Two vehicles loitering and repeatedly swapping loiter locations
- Each time the vehicles get close, a collision avoidance behavior is spawned.
- After the range opens sufficiently, the collision avoidance behavior dies.



Monitoring Life Events

- A “Life Event” is the *spawning* or *death* of a behavior.
- Life Events may be monitored in a special mode of the uHelmScope MOOS utility:

```
$ uHelmScope --life henry.moos
```

Terminal — uHelmScope — 86×21 — ⌘1

tcsh | uHelmScope

```
      *************************************************************
      *         Summary of Behavior Life Events                   *
      *************************************************************

Time      Iter  Event  Behavior   Behavior Type        Spawning Seed
-------   ----  -----  ---------  ------------------   ----------------------------
 280.50    833  death  avd_gilda  BHV_AvoidCollision
 389.77   1268  spawn  avd_gilda  BHV_AvoidCollision   name=avd_gilda#contact=gilda
 444.52   1415  death  avd_gilda  BHV_AvoidCollision
 557.64   1867  spawn  avd_gilda  BHV_AvoidCollision   name=avd_gilda#contact=gilda
 617.70   2019  death  avd_gilda  BHV_AvoidCollision
 701.60   2355  spawn  avd_gilda  BHV_AvoidCollision   name=avd_gilda#contact=gilda
 758.22   2511  death  avd_gilda  BHV_AvoidCollision
 809.56   2717  spawn  avd_gilda  BHV_AvoidCollision   name=avd_gilda#contact=gilda
 866.61   2863  death  avd_gilda  BHV_AvoidCollision
 971.80   3273  spawn  avd_gilda  BHV_AvoidCollision   name=avd_gilda#contact=gilda
1031.60   3410  death  avd_gilda  BHV_AvoidCollision

Hit 'r' to resume outputs, or SPACEBAR for a single update
```

Monitoring Life Events

- A "Life Event" is the *spawning* or *death* of a behavior.
- Life Events may be monitored in a special mode of the uHelmScope MOOS utility.
- The Life Event may also be examined post-runtime from the MOOS log files:

```
$ aloghelm --life henry_logfile.alog
```

Terminal — tcsh — 94×31 — ⌘3

sh | pMarinePID | tcsh

```
     ****************************************************
     *        Summary of Behavior Life Events           *
     ****************************************************

Time     Iter  Event  Behavior      Behavior Type      Spawning Seed
-------  ----  -----  ------------  -----------------  ----------------------------
   0.00     1  spawn  loiter        BHV_Loiter         helm startup
   0.00     1  spawn  waypt_return  BHV_Waypoint       helm startup
   0.00     1  spawn  station-keep  BHV_StationKeep    helm startup
  94.57   247  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 152.94   398  death  avd_gilda     BHV_AvoidCollision
 222.32   677  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 280.50   833  death  avd_gilda     BHV_AvoidCollision
 389.77  1268  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 444.52  1415  death  avd_gilda     BHV_AvoidCollision
 557.64  1867  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 617.70  2019  death  avd_gilda     BHV_AvoidCollision
 701.60  2355  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 758.22  2511  death  avd_gilda     BHV_AvoidCollision
 809.56  2717  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
 866.61  2863  death  avd_gilda     BHV_AvoidCollision
 971.80  3273  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
1031.60  3410  death  avd_gilda     BHV_AvoidCollision
1164.46  3927  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
1226.22  4061  death  avd_gilda     BHV_AvoidCollision
1341.33  4503  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
1400.98  4638  death  avd_gilda     BHV_AvoidCollision
1520.36  5108  spawn  avd_gilda     BHV_AvoidCollision name=avd_gilda#contact=gilda
leonardo:missions/m2_berta(trunk)%
```

The uTimerScript Utility:

Overview

uTimerScript

What is uTimerScript?

- A tool that allows the user to script a set of pre-configured events (pokes) to a MOOSDB.
- Each event can be configured to happen after a specified amount of elapsed time.
- Enables us to fake incoming command-and-control messages, sensor events etc.

A simple example:

```
ProcessConfig = uTimerScript
{
  AppTick   = 4
  CommsTick = 4

  EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10
  EVENT = var=DEPLOY, val=true, time=15
}
```

```
EVENT = var<variable>, val=<value>, time=<delay>
```

This simple script will launch the Alpha or Berta missions automatically.

The uTimerScript Utility:
Starting and Pausing the Script

uTimerScript

When does the script start?

- By default the script starts when uTimerScript connects to the MOOSDB and begins to Iterate().
- It may be configured in the "paused" mode
- It may be configured to include a delay once it has started.
- It may be configured to require conditions be met before starting.

Starting the script in the PAUSED mode, with a DELAY.

```
ProcessConfig = uTimerScript
{
  AppTick    = 4
  CommsTick  = 4

        EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10
        EVENT = var=DEPLOY, val=true, time=15
  CONDITION = ALPHA != 20
  DELAY_START = 30
  PAUSED = true
}
```

Script will be paused if `ALPHA=20`.
(uTimerScript will register for `ALPHA`).

The script may then be un-paused by posting to the MOOSDB:
`UTS_PAUSE=false`.

The uTimerScript Utility:
Randomizing the Event Times

uTimerScript

Random event scheduling:

- Events may be configured to occur at a random time in a given interval.
- Random events are useful in testing the robustness of algorithms in varying situations.

The same example script with events randomized:

```
ProcessConfig = uTimerScript
{
  AppTick    = 4
  CommsTick = 4

   EVENT = var=MOOS_MANUAL_OVERRIDE, val=false, time=10:20
   EVENT = var=DEPLOY, val=true, time=10:20
  PAUSED = true
}
```

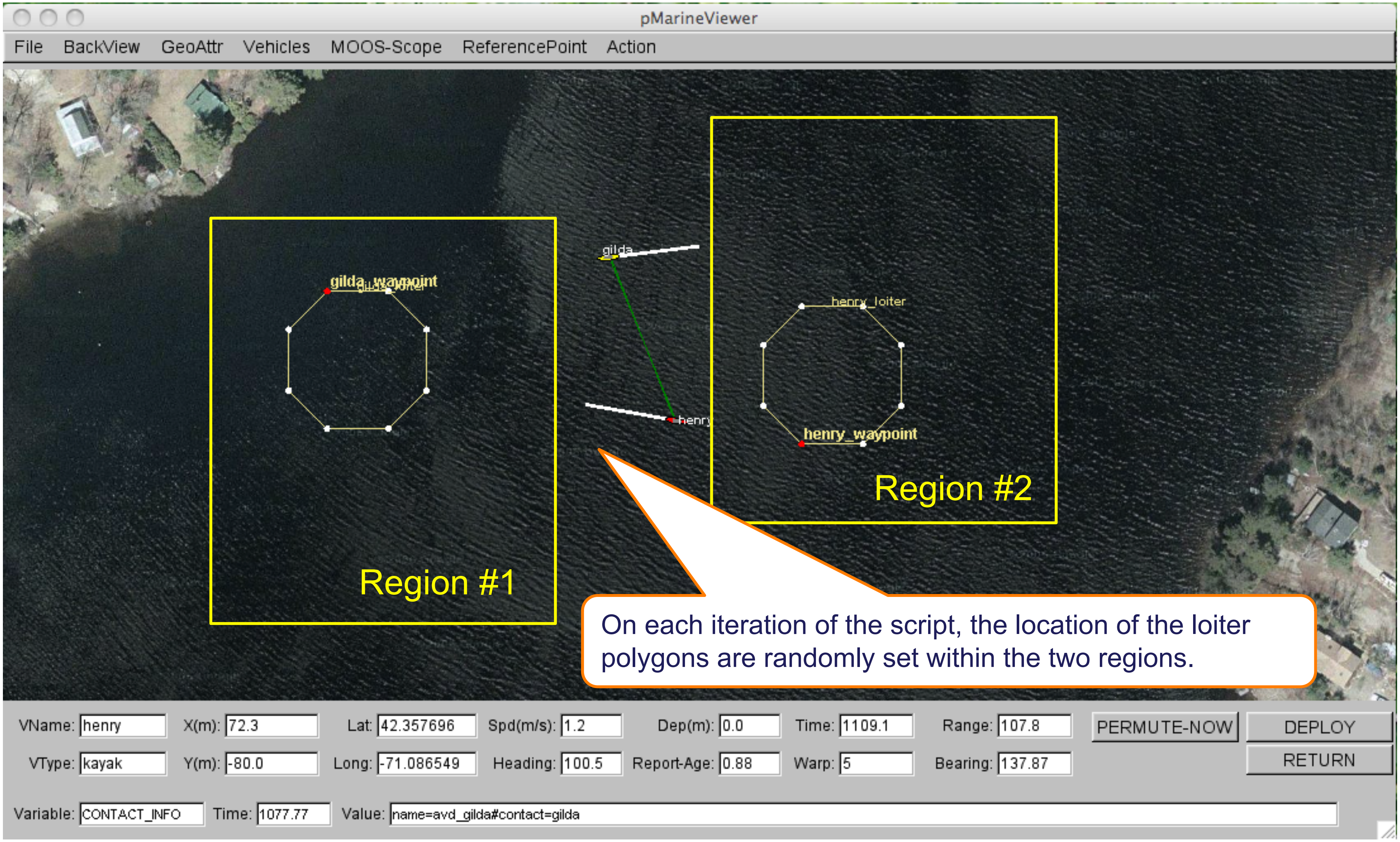
Event occurs between 10 and 20 seconds after the script begins

Event times are chosen with uniform probability.

The uTimerScript Utility:

Usage in the Berta Example Mission

uTimerScript



The pBasicContactMgr Utility:

What it is, and is not

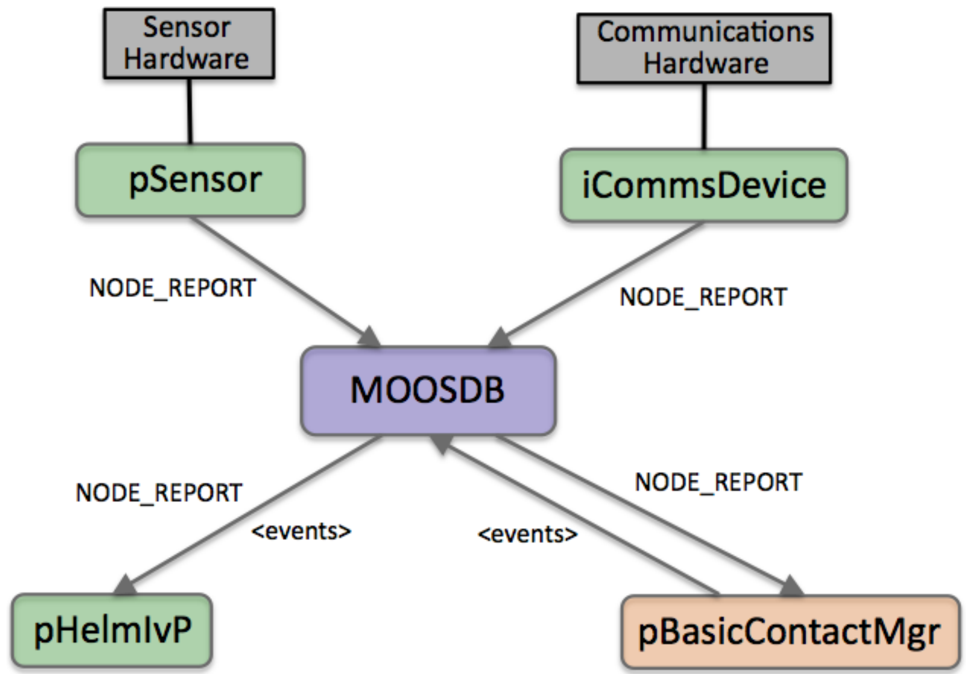
pBasicContactMgr

What is pBasicContactMgr?

- A tool for managing node reports and generating conditional events.
- It posts summary reports for all known contacts.
- It posts events, i.e., alerts, about contacts based on the range to the contact.
- Designed with the IvP Helm in mind to allow the helm to spawn contact-related behaviors dynamically as they become known.

What pBasicContactMgr is NOT:

- It is not a sensor application.
- It does not perform sensor fusion.
- It does not represent or reason about areas of uncertainty associated with contact position.



Variables Published:

- `CONTACTS_LIST`
- `CONTACTS_RECAP`
- `CONTACT_ALERTED`
- `CONTACTS_UNALERTED`
- `CONTACTS_RETIRED`
- `CONTACT_MGR_WARNING`

The pBasicContactMgr Utility:

Contacts, Alerts, Record keeping

pBasicContactMgr

The following are reported (Posted to the MOOSDB) on each iteration:

`CONTACTS_LIST:`	comma-separated list of contacts.
`CONTACTS_RECAP:`	A comma-separated list of contact summaries.
`CONTACT_ALERTED:`	A list of contacts for which alerts have been posted.
`CONTACTS_UNALERTED:`	A list of contacts for which alerts are pending, based on the range criteria.
`CONTACTS_RETIRED:`	A list of contacts removed due to the information staleness.
`CONTACT_MGR_WARNING:`	A warning message indicating possible mishandling of or missing data.

Examples:

- `CONTACTS_LIST:` = "delta,gus,charlie,henry"
- `CONTACT_ALERTED:` = "delta,charlie"
- `CONTACTS_UNALERTED:` = "gus,henry"
- `CONTACTS_RETIRED:` = "bravo,foxtrot,kilroy"
- `CONTACTS_RECAP:` = "name=delta,age=11.3,range=193.1 # name=gus,age=0.7,range=48.2 #name=charlie,age=1.9,range=73.1 # name=henry,age=4.0,range=18.2"

The pBasicContactMgr Utility:

Alert Triggers

Alerts are triggered by range. Configured in the MOOS configuration file:

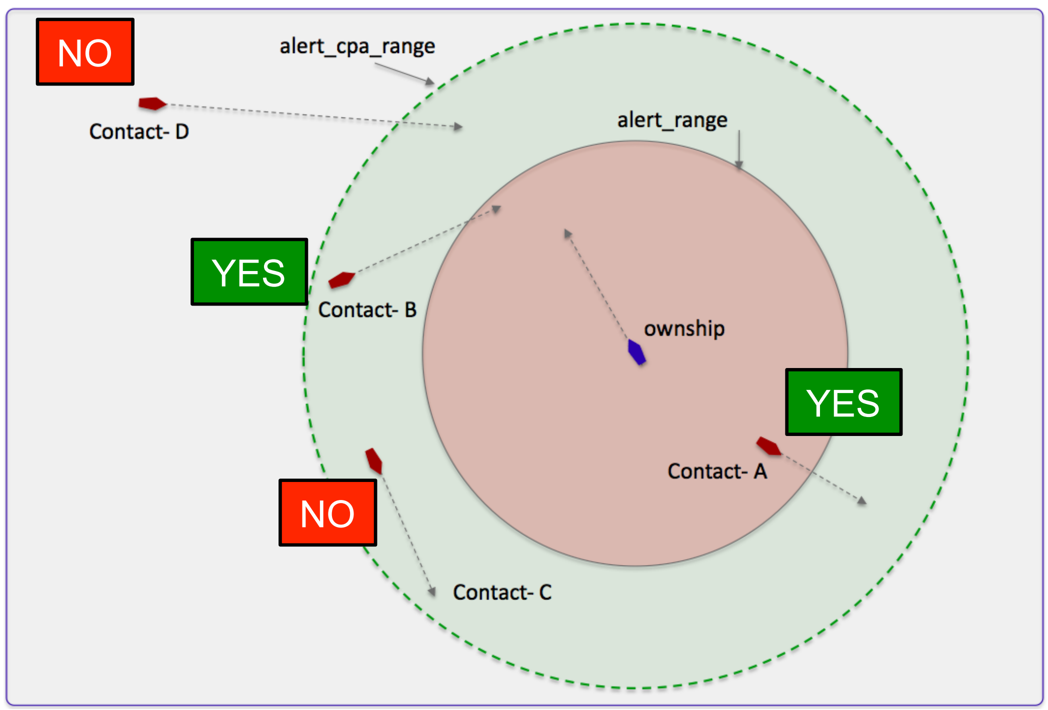
```
    ALERT_RANGE = <distance>    // meters
ALERT_CPA_RANGE = <distance>    // meters
 ALERT_CPA_TIME = <duration>    // seconds
```

`ALERT_RANGE` – when a contact is within this range an alert is generated.

`ALERT_CPA_RANGE` – when a contact is within this range and its closest point of approach (CPA) is within the alert range, an alert is generated.

`ALERT_RANGE` – The time used for CPA calculation.

Examples:



Outline

- The IvP Helm and Architecture Motivations
- New capabilities in Release 4.1
- Plans for Development

Plans for Development

FY11 Planned activities:

- *Opportunistic Function Generation*

  Modification of the Helm and IvPBehavior super class to allow behaviors to re-submit IvP functions from prior iterations if deemed sufficiently similar between iterations.

- *Helm-Accessible Approximate Vehicle Dynamics*

  Identify concise representations and approximations of vehicle dynamics for behaviors to better evaluate candidate helm decisions.

- *Integrated Scheduling with Behavior-Based Control*

  Investigation of hybrid approaches of combining scheduling and planning techniques with traditional behavior-based reactive decision-making.

- *Mixed Human-Machine Competition Scenarios*

  Exploration of competition scenarios for development of behaviors and interfaces to human operators based on win/lose evaluation metrics.