MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY, SENSING, AND COMMUNICATIONS

Lecture 2: Introduction To MOOS

February 5th, 2025

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

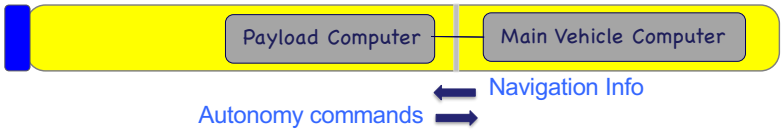
Email:
Mike Benjamin, mikerb@mit.edu

Photo by Arjan Vermeij, CMRE

1

MITMECHE

Payload UUV Autonomy



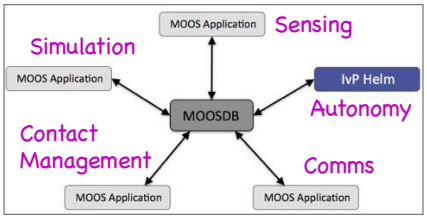
Architecture Principle #1
Payload Autonomy

Decouple the Procurement of Hardware and Software

2

Payload UUV Autonomy

Payload Computer



MOOS Middleware
MOOS Applications

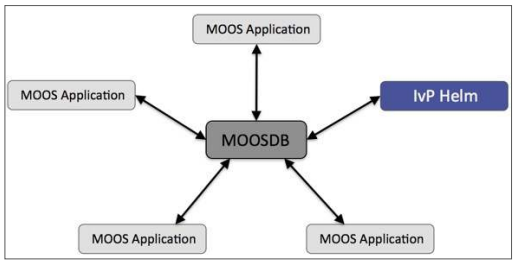
Architecture Principle #2
Autonomy System Middleware

De-couple Software Procurements
Sensing, Autonomy, Simulation, Comms...

3

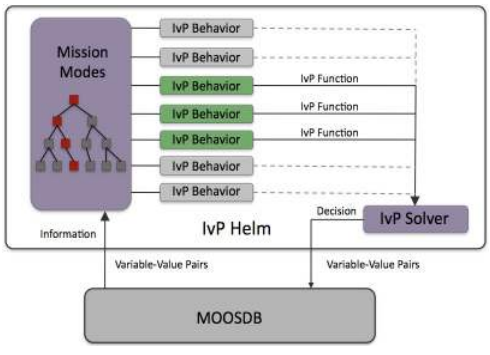
Payload UUV Autonomy

Payload Computer



MOOS Middleware
MOOS Applications

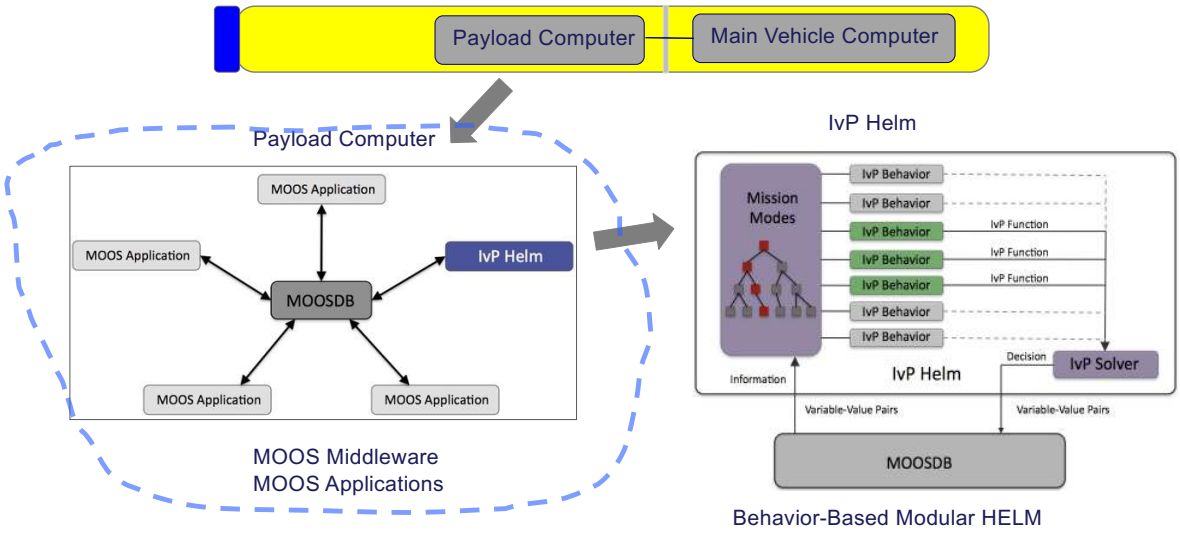
IvP Helm



Behavior-Based Modular HELM

4

Payload UUV Autonomy



MOOS Overview

- MOOS is a Robot Middleware
- Developed by Paul Newman, as an MIT post-doc and now Oxford Professor, Oxbotica Founder
- Initial development 2000-2003 on Bluefin Odyssey II UUV owned by MIT

Italy, Summer 2002

Italy, Summer 2002

MITMECHE

MOOS Overview






7

MITMECHE

Oxa (formerly Oxbotica)

OXA Driver

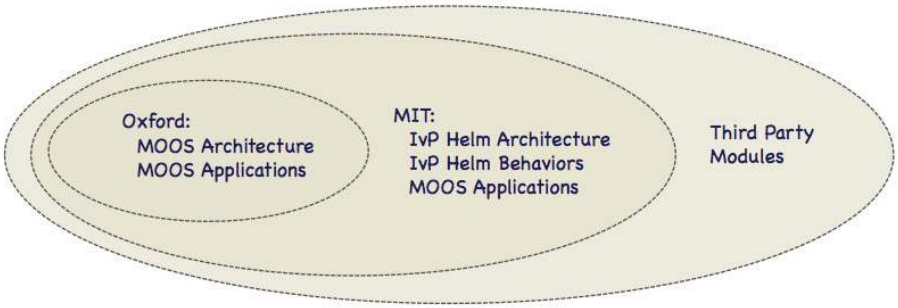
Oxa Driver is a full stack of high-performance software components that enable the safe and efficient self-driving of any vehicle, even in the most challenging of environments.

- Any sensor, vehicle or platform
- Low-energy, high-performance
- Highly accurate and safe
- Integrate the full stack, or embed specific components into other products and technology platforms


8

Nested Repositories

- MOOS, from the Mobile Robotics Group at Oxford
- MOOS-IvP, from the Marine Autonomy Lab at MIT
- 3rd Party (Your) modules.



MOOS Modules in the MOOS-IvP Distribution

The Oxford MOOS tree (11)

- MOOSDB
- pLogger
- pShare
- pAntler
- iMatlab
- pScheduler
- uMS
- jMOOS
- iRemote
- pyMOOS
- uPlayback

8 Work years of development effort

2.8 Mb size
0 dependencies
Download: 2 secs
Build: 7 secs

The moos-ivp tree (57)

- pHelmIvP
- alogavg
- alogbin
- alogcat
- alogcd
- alogcheck
- alogclip
- alogeplot
- alogeval
- aloggrep
- alogheIm
- alogiter
- alogload
- alogmhash
- alogpare
- alogpick
- alogrm
- alogscan
- alogsort
- alogsplit
- alogtest
- alogtm
- alogview
- nspatch
- nsplug
- pickpos
- pluck
- projfield
- projpt
- tagrep
- pAutoPoke
- pMapMarkers
- pMarineViewer
- pMissionHash
- pHostInfo
- uMayFinish
- uFunctionVis
- pMarinePID
- pEchoVar
- pRealm
- pSpoofNode
- uPlotViewer
- nsplug
- uXMS
- uMAC
- iSay
- zaic_hdg
- zaic_peak
- zaic_spd
- zaic_vect
- uFldHazardMgr
- uFldHazardMetric
- uFldNodeBroker
- uFldShoreBroker
- uFldNodeComms
- uFldPathCheck
- uHelmScope
- uTimerScr ipt
- uProcessWatch
- uTermCommand
- pContactMgrV20
- uQueryDB
- uPokeDB
- uFldDelve
- uMemWatch
- uMACView
- pDeadManPost
- pEvalLoiter
- pObstacleMgr
- pMissionEval
- uCommand
- uFldObstacleSim
- uFldCollObDetect
- uFldCollisionDetect
- uFldWrapDetect
- uFunctionVis
- uLoadWatch
- uFldMessageHandler
- uFldContactRangeSensor
- uFldBeaconRangeSensor
- pNodeReporter
- uSimMarineV22

50+ Work years of development effort

According to "sloccount"
www.dwheeler.com/sloccount

CMRE MOOS Modules


The CMRE tree (124 MOOS Applications)

- iCompassSocket
- iDPCAMOOSInterface
- iFLIRRangerHRC
- iFutabaJoyStick
- iHardwareHealthMonitor
- pRadarTrack2Status
- uRangeBearingUUVSim
- iMicrostrainGX1
- pSonarImageProcessing
- iSystempSpcMuscle
- pArraySimToNad
- pLinearTrajectoryGenerator
- pTowedSourceNavEstimator
- pCircleTrajectoryGenerator
- pMaintainRelativeBearing
- pBVBottomTargetDetector
- pBacksteppingController
- iTcpClient
- iTcpServer
- iTelnetClient
- iTisVis
- i3DMGX1
- iAxisM7001
- iSentinel
- iSerialPort
- iPanTilt
- iPlayBack
- iTps730
- iUdpClient
- iUdpGatherer
- iUdpScatterer
- iXBee868
- pAdaptiveSurvey
- pAisToNmea
- iSidusPositioner
- pCommandFilter
- pCompassToNmea
- pDmhtTracker
- pDopplerSim
- pDuripToSlita
- pFLIRPanTilt
- pFSM
- pGetSlitaNAD
- pGpsNodeReporter
- pGuardian
- pHelmToNmea
- pHiPapUvTracker
- pInstroLRF
- pJoyStick
- pKalmanTracker
- pLaserTrack
- pNmeaToXml
- pNull
- pOctaver
- pOENix
- pOEX
- iJoyStick
- iLMSView
- pOEXTel2Status
- pOnte
- iMaxaBeam
- iSncZ20P
- iRosPositioner
- pPersistTracker
- pPIDController
- pPosToNMEA
- pProcessAtlasBB
- pProcessSlitaBB
- pREMUSCodec
- pReplayAtlas
- pScooter
- pSpcMaster
- pVLC
- pWatchDog
- pXBee
- pXmlSerialiser
- pXmlTransformer
- pXmlUnpacker
- uBcSlitaPlayer
- uBcSlitaRcvr
- uLRM2500CI
- uMaxaBeam
- uScooter
- uUVReacquire
- uUvTracker
- pTrackerUUVSim
- pTrackEvaluator
- pSurveyPlanner
- pCartesianToGeo
- iExploreDVL
- pUVDeploy
- pUVNodeReporter
- pTrigger
- pBistaticLocator
- iBlueView
- pNmeaToAis
- pMuscle
- pMaxaBeam
- pUVLink
- iLRM2500CI
- pTrackBuilder
- pIngTimer
- pAlarmBox
- iBVLMS
- pUvTracker
- pVelvet
- uWitty
- pTgtDetector
- pUVReacquire
- pRealAIS2Status
- pRealAIS2Xml
- pHomer
- pLaunchTarget
- pLRAD
- pLuribus
- pAntagonist
- pAnTilter
- pBearingTrack
- pActivePassiveManager
- pArrayNavEstimator
- pBVImgProcessing


SeaRobotics USV


H-Scientific USV

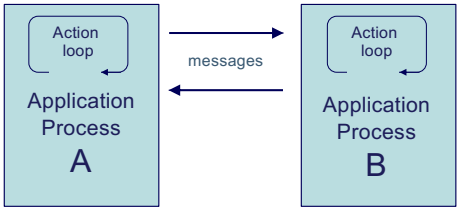

REMUS 100


Ocean Explorer

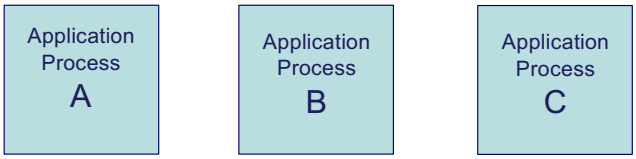

Muscle Vehicle

MOOS Does Two Main Things

1. It enables distinct applications to communicate
2. It enables users to control the frequency of each application's action loop



The Beauty of Separate Processes



On Unix based systems, each process:

- Has a unique Process ID (PID)
- Uses a chunk of computer memory *separate* from all other processes

Advantages:

- A crash in one process will not affect another process
- The OS automatically distributes processes over system CPU cores

MOOSDB is a Process for Communication

- It has its own PID and memory space like any other process
- It maintains a mapping for Variable Names → Values

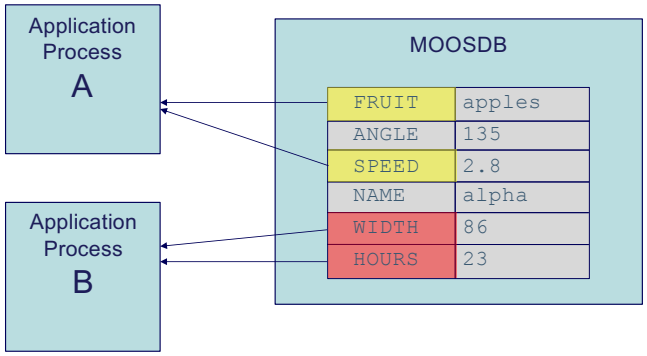
MOOSDB

FRUIT	apples
ANGLE	135
SPEED	2.8
NAME	alpha
WIDTH	86
HOURS	23

Only the most recent value is retained

MOOS Apps Subscribe to the MOOSDB

- An App may register (subscribe for) for any variable
- An App may register any time, but typically during startup
- Multiple apps may register for the same variable



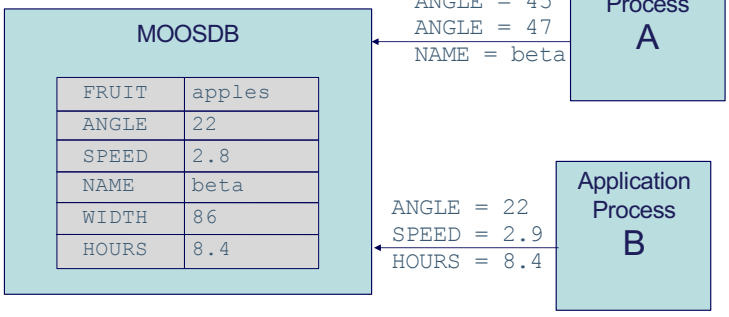
When an App first connects, it gets mail for each registered variable.

(if the variable has ever been written to)

MOOS Apps Publish to the MOOSDB

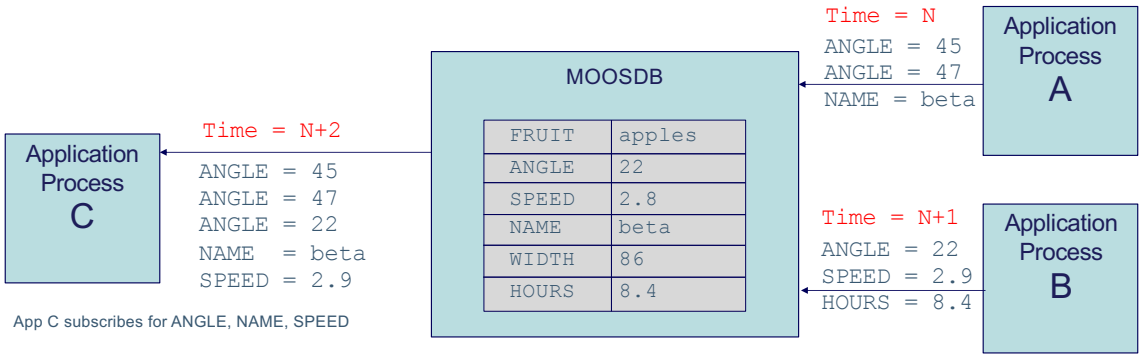
- An App may publish to the MOOSDB any time
- No prior arrangement required

Note: Subscribers will get **all** postings – each as a new piece of mail.



MOOS Apps Publish to the MOOSDB

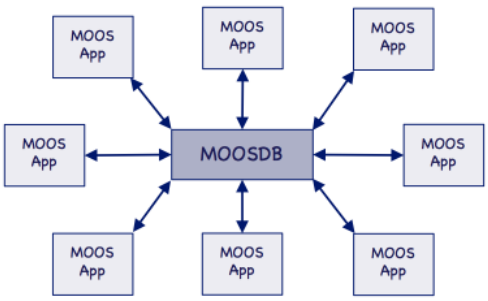
- An App may publish to the MOOSDB any time
- No prior arrangement required



A MOOS Community

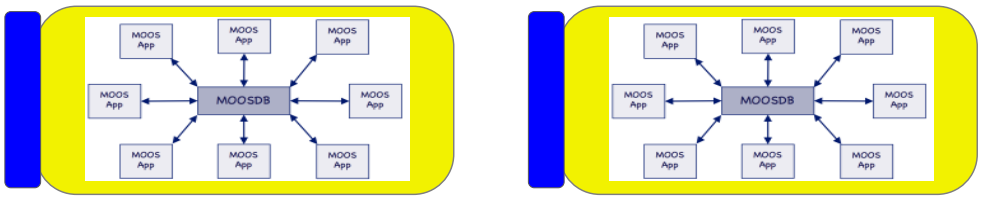
- A MOOS *community* is comprised of one MOOSDB and all connected Apps
- MOOS is described as having a *star* topology.

A community also has a unique
- name
- IP address, Port number



A MOOS Community Per Robot

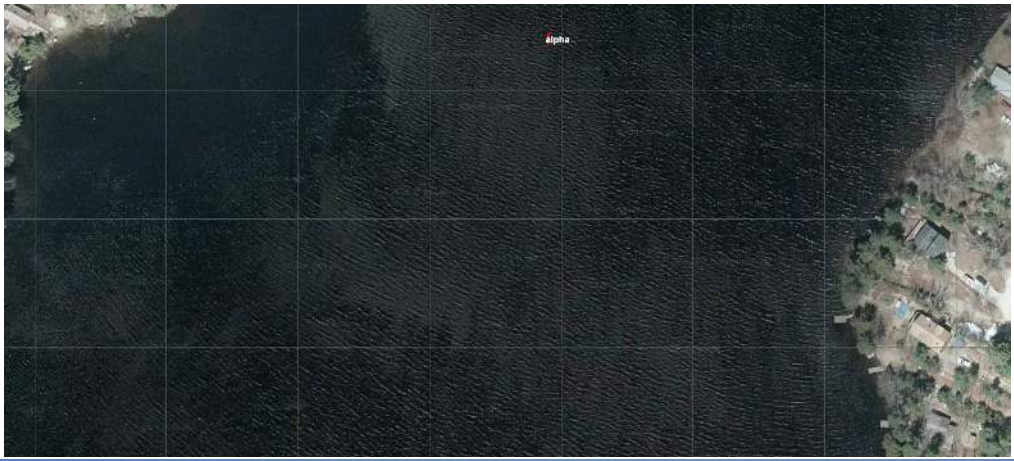
- Typically, one community per vehicle/robot
- Sometimes multiple computers on one vehicle, each with a community



- Inter-community communications addressed later

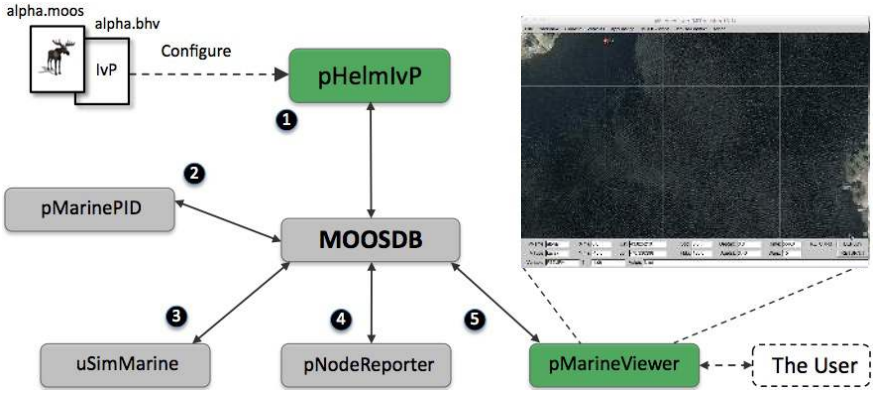
Example: The Alpha Mission

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



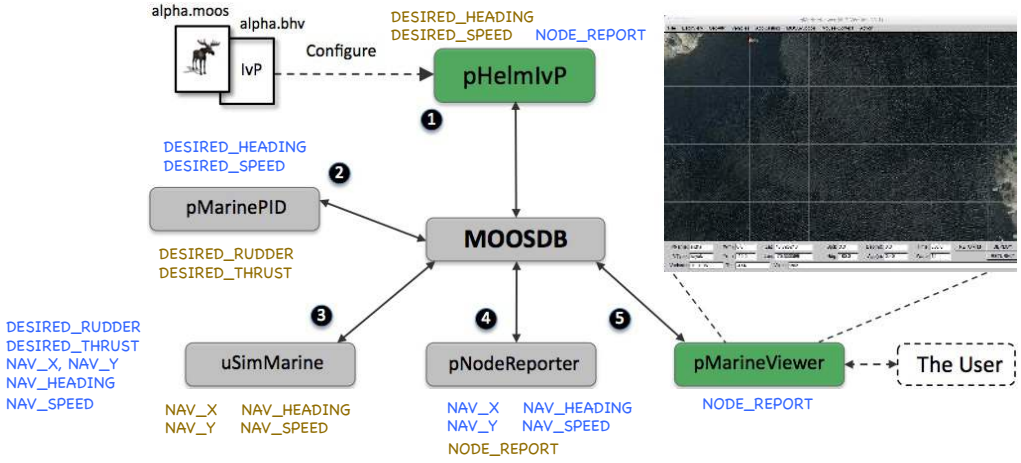
Alpha Mission - Modules

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

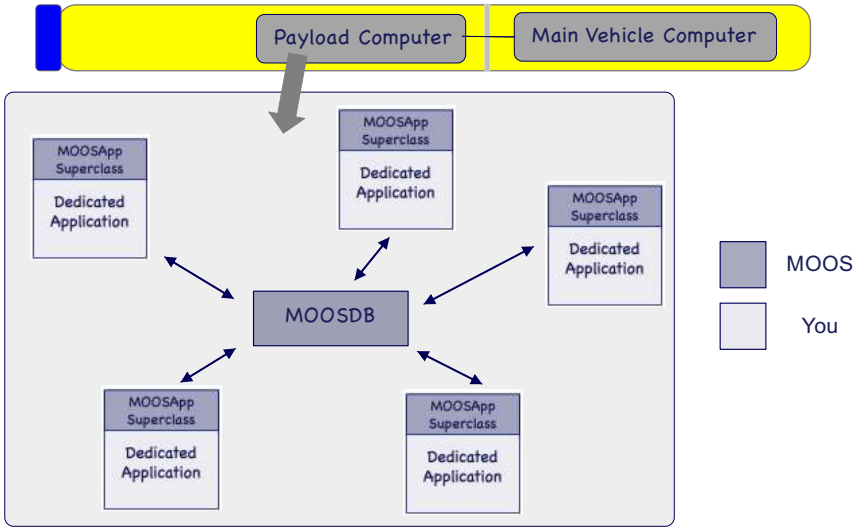


Alpha Mission

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

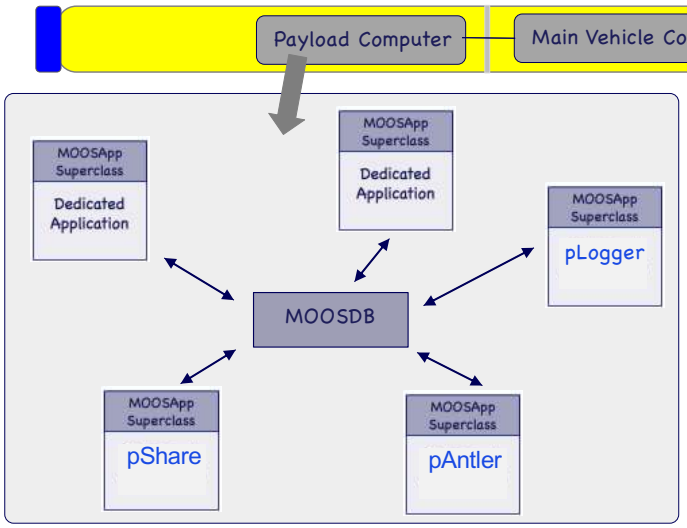


Payload (Backseat) UUV Autonomy



23

Payload (Backseat) UUV Autonomy



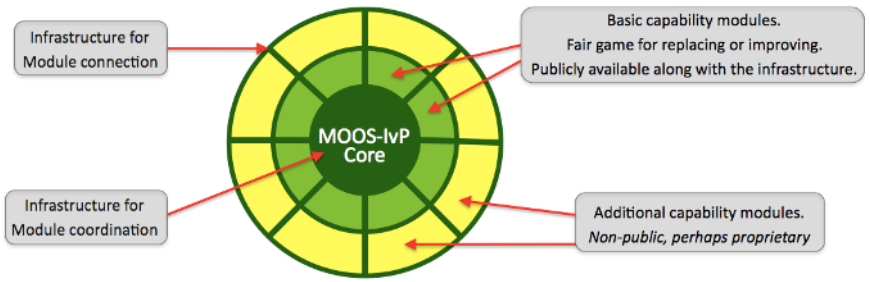
A few important applications come with MOOS:

- pAntler
- pLogger
- pShare

24

Public Infrastructure – Nested Capabilities

An autonomy system has components with different capabilities, and distribution access.
- Publicly accessible modules providing infrastructure, basic capabilities
- Restricted-access modules for developers of a particular project.



Core Infrastructure. Open Source.

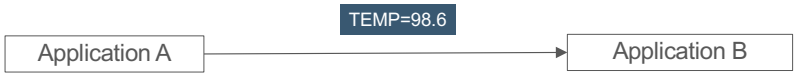
Project specific add-on modules.
Not Open Source.

Autonomy System = Infrastructure + Modules

MOOS Messages

Information Flow

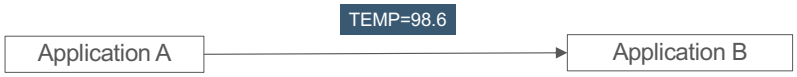
- The various blocks in our architecture need to share information.



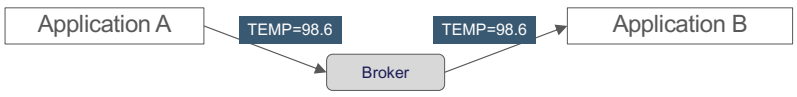
- We want this interface to be as generic as possible
- Ideally Application A and B are designed without ever considering each other

Information Flow

- The various blocks in our architecture need to share information.

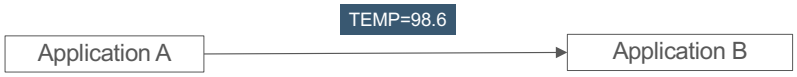


- We want this interface to be as generic as possible
- Ideally Application A and B are designed without ever considering each other



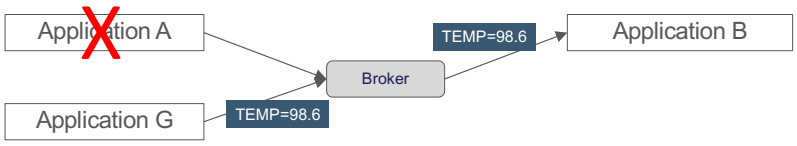
Information Flow

- The various blocks in our architecture need to share information.



- We want this interface to be as generic as possible
- Ideally Application A and B are designed without ever considering each other

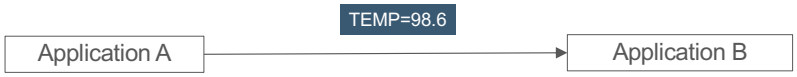
A better version of App A has been developed. Easy to replace.



29

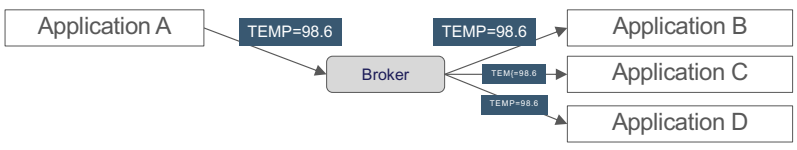
Information Flow

- The various blocks in our architecture need to share information.



- We want this interface to be as generic as possible
- Ideally Application A and B are designed without ever considering each other

Additional apps may want to consume the same information for different purposes.
Easy to add.



(Original purpose)

Some other reason, e.g., visualization

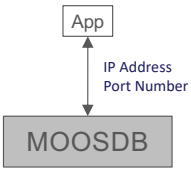
Some other reason, e.g., analysis

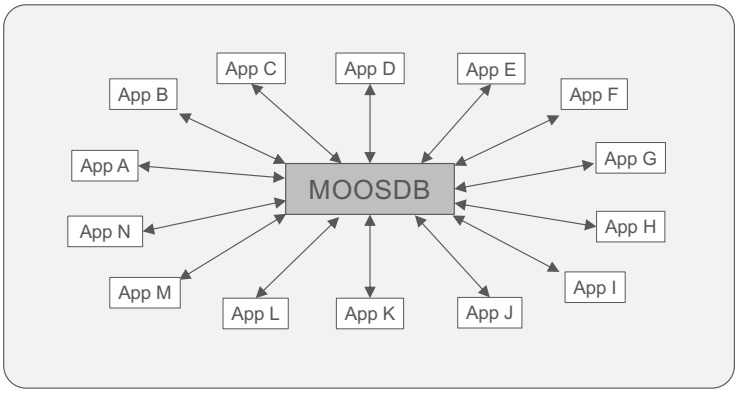
30

A MOOS Community

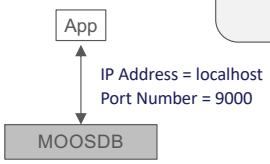
The MOOSDB is a server, providing a service over a configured port, on a particular IP address (not unlike a web server).

An App connects by passing it the IP address and port number where it can expect to find the MOOSDB





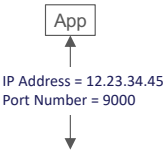
In most situations, the App and MOOSDB are running on the same machine (localhost), and the MOOSDB is using the default port number of 9000.

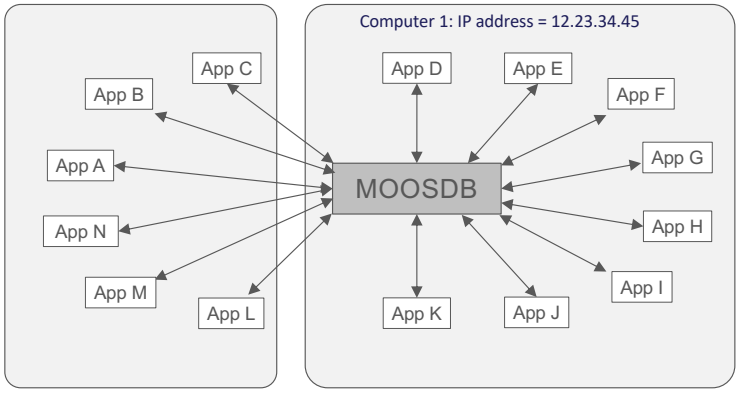


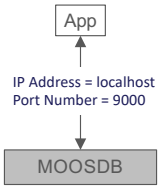
A Distributed MOOS Community

A single MOOS community (having a single MOOSDB), may be distributed over the network. Apps running on the remote machine simply need to know the IP address of the machine running the MOOSDB.

The two machines could be anywhere world-wide but are typically two machines on the same robotic platform, as a way of bringing more computational power.







MOOS Messages

- Two primary message components: VARIABLE and VALUE
- Two primary message types: STRING and DOUBLE

MOOSDB

FRUIT	apples
ANGLE	135
SPEED	2.8
NAME	alpha
WIDTH	86
HOURS	23

Name	The name of the data
StringVal	Data in human-readable string format, or raw binary data
DoubleVal	Numeric double float data

MOOS Message Examples

MOOSDB

FRUIT	apples
ANGLE	135
SPEED	2.8
NAME	alpha
WIDTH	86
HOURS	23

Name	FRUIT
StringVal	"apples"
DoubleVal	0
DataType	string

Name	WIDTH
StringVal	" "
DoubleVal	86
DataType	double

MOOS Messages

Each MOOS Message contains additional useful information:

Name	The name of the data
StringVal	Data in human-readable string format, or raw binary data
DoubleVal	Numeric double float data
DataType	Type of data (STRING or DOUBLE or BINARY)
Source	Name of client that sent this data to the MOOSDB
SourceAux	Optional additional information about the source client
Time	Time at which the data was written
Community	The community to which the source process belongs

Posting MOOS Messages

Inside your MOOS application you may post a message with simple line in C++:

```
Notify("FRUIT", "apples");
```

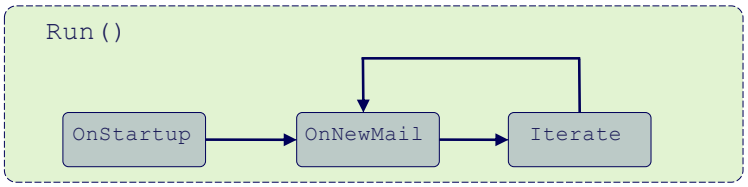
MOOS will automatically fill in the additional fields:

Auto-filled

Name	FRUIT
StringVal	"apples"
DoubleVal	0
DataType	String
Source	pFoobar
SourceAux	
Time	34558.2
Community	alpha

Reading MOOS Messages

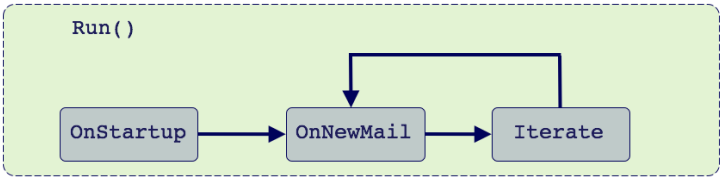
- MOOS Apps read messages inside a mail-handling function
- This function is defined in the MOOSApp superclass for all MOOS Apps



The Flow of Control for all MOOS Apps

A Mail Handling Example

- An example OnNewMail() implementation:




```
bool MyApp::OnNewMail(MOOSMSG_LIST &NewMail)
{
  MOOSMSG_LIST::iterator p;  for(p=NewMail.begin(); p!=NewMail.end(); p++) {
    CMOOSMsg &msg = *p;
    string key = msg.GetKey();

    if(key == "TEMP")
      updateWidth(msg.getDouble());
  }
  return(true);
}
```

Handling a MOOS Message

Other useful functions defined on a MOOS Message:

```
MOOSMsg msg;

string moos_var = msg.GetKey();          // the MOOS variable name

bool is_double = msg.IsDouble();         // true if message content double
bool is_string = msg.IsString();         // true if message content string

double timestamp = msg.GetTime();        // timestamp when message posted

string str_val = msg.GetString();        // the message string content
string dbl_val = msg.GetDouble();        // the message double content

string source  = msg.GetSource();        // who (which app) posted message
string src_aux = msg.GetSourceAux();    // further source information

string community = msg.GetCommunity();  // MOOS community who posted
```


Launching MOOS

Launching MOOS (Bare Bones)

The MOOSDB may be launched from the command line:

```
$ MOOSDB
```

- The new MOOSDB process is the beginning of a MOOS community
- Recall a community has an IP Address, Port Number, Community Name

Terminal output:

```
------------------- MOOSDB V10 -------------------
  Hosting  community                 "#1"
  Name look up is                    off
  Asynchronous support is            on
  Connect to this server on port     9000
--------------------------------------------------
network performance data published on localhost:9020
listen with "nc -u -lk 9020"
```

Default Community Name

Default Port Number

Default IP Address

Launching MOOS (with Mission File)

- The IP Address, Port Number and Community Name may be provided in a mission file.
- The mission file is a command line argument:

```
$ MOOSDB  mission.moos
```

mission.moos

```
Community  = alpha
ServerPort = 9205
ServerHost = localhost
```

```
------------------- MOOSDB V10 -------------------
  Hosting  community                 "alpha"
  Name look up is                    off
  Asynchronous support is            on
  Connect to this server on port     9205
--------------------------------------------------
network performance data published on localhost:9020
listen with "nc -u -lk 9020"
```

Launching MOOS and Mission Configuration

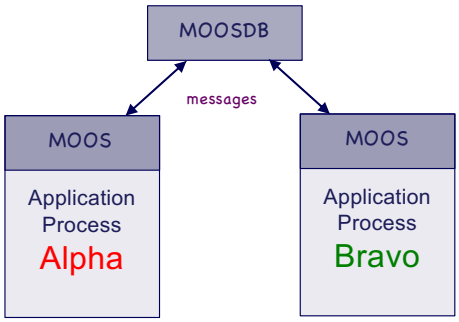
- A mission file may also hold configuration parameters for MOOS apps
- Each application has a dedicated configuration block.

mission.moos

```
Global parameters

ProcessConfig = alpha
{
   alpha parameters
}

ProcessConfig = bravo
{
   alpha parameters
}
```



MOOS Mission Configuration

- Mission configuration is through a single “mission file”, with a .moos extension.
- Each application has a dedicated configuration block.

mission.moos

```
Global parameters

ProcessConfig = alpha
{
   alpha parameters
}

ProcessConfig = bravo
{
   alpha parameters
}
```

“Global parameters”
are accessible to all MOOS applications. They include things like:

- MOOSDB server IP address and port number.
- Local datum (0,0) in lat/lon coordinates.
- Name of the MOOS community.

MOOS Mission Configuration

- Mission configuration is through a single “mission file”, with a .moos extension.
- Each application has a dedicated configuration block.

mission.moos

```
Global parameters

ProcessConfig = alpha
{
   alpha parameters
}

ProcessConfig = bravo
{
   alpha parameters
}
```

“Application parameters”
Accessible only to a particular application.

Application authors implement the handling of parameters upon application startup.

The MOOSApp superclass has a function called OnStartUp() where configuration parameters are handled.

Application authors have access to each line in the application’s configuration block to handle as they see fit.

Scoping MOOS

Scoping MOOS

Scoping the MOOSDB means examining:

- Current values of variables known to the MOOSDB
- Which processes made the most recent post
- When it was posted
- The community of the application making the post.

MOOSDB

FRUIT	apples
ANGLE	135
SPEED	2.8
NAME	alpha
WIDTH	86
HOURS	23

Scoping MOOS

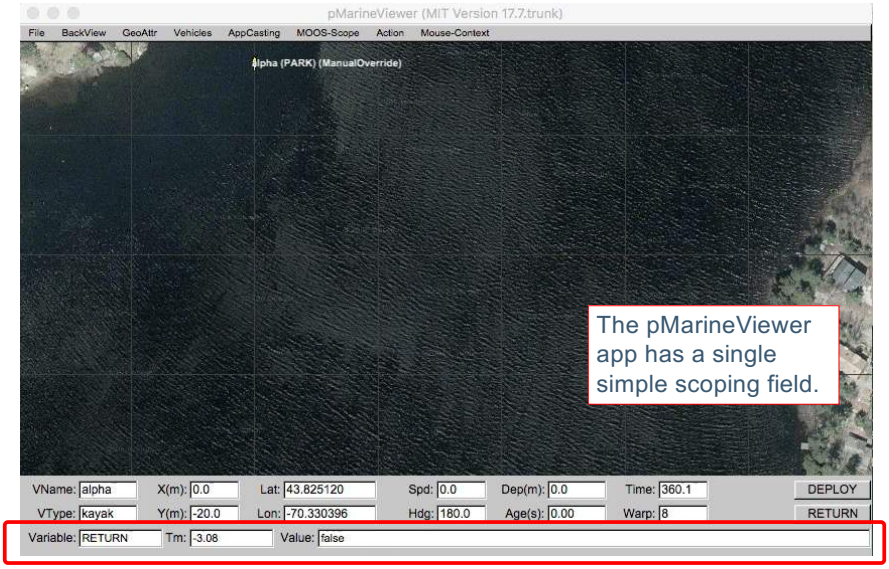
Scoping the MOOSDB means examining:

- Current values of variables known to the MOOSDB
- Which processes made the most recent post
- When it was posted
- The community of the application making the post.

MOOSDB

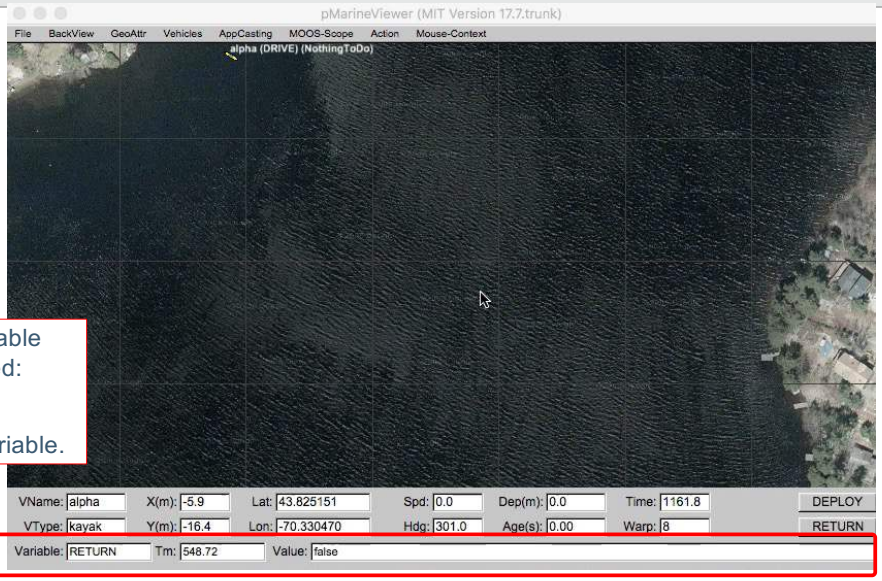
VarName	Source	Community	Time	VarValue
FRUIT	pFruit	alpha	143.21	apples
ANGLE	uMeasure	alpha	1873.24	135
SPEED	uMeasure	alpha	62.11	2.8
NAME	pIdentity	gamma	3.91	alpha
WIDTH	uMeasure	alpha	1873.24	86
HOURS	uMeasure	alpha	1873.25	23

A Simple Single Scope in pMarineViewer



The pMarineViewer app has a single simple scoping field.

Changing the Scope Variable in pMarineViewer



The scope variable may be changed:

Hit CTRL-'A'

Enter a new variable.

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```


The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

- To scope on a MOOSDB, uXMS must connect to the MOOSDB.
- Where is the server?
- It could be anywhere on the internet.
- Exactly where? This is determined by the IP address and the port number, for the MOOSDB server.

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

- To scope on a MOOSDB, uXMS must connect to the MOOSDB.
- Where is the server?
- It could be anywhere on the internet.
- Exactly where? This is determined by the IP address and the port number, for the MOOSDB server.

mission.moos

```
ServerHost = localhost
ServerPort = 9005
```


The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

- To scope on a MOOSDB, uXMS must connect to the MOOSDB.
- Where is the server?
- It could be anywhere on the internet.
- Exactly where? This is determined by the IP address and the port number, for the MOOSDB server.

mission.moos

```
ServerHost = localhost
ServerPort = 9005
```

The same information may also be passed on the command line as arguments to uXMS

```
$ uXMS --all --serverhost=localhost --serverport=9005
```

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

By default, the screen will refresh whenever one variable value changes

```
VarName             (S)ource        (T)ime      (C)ommunity   VarValue
----------------    ----------      ----------  ----------    ----------- (7)
DB_CLIENTS          MOOSDB_alpha    106.2       alpha         "uXMS,DBWebServer,"
DB_TIME             MOOSDB_alpha    107.2       alpha         1325701208.08963
DB_UPTIME           MOOSDB_alpha    107.2       alpha         107.20791
FRUIT               pFruit          143.21      alpha         "apples"
ANGLE               uMeasure        107.2       alpha         135
SPEED               uMeasure        107.2       alpha         2.8
NAME                pIdentity       3.91        gamma         "alpha"
WIDTH               uMeasure        1873.24     alpha         86
HOURS               uMeasure        1873.25     alpha         23
-- displaying all variables --
```

All scoped variables

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

By default, the screen will refresh whenever one variable value changes

```
VarName             (S)ource        (T)ime      (C)ommunity   VarValue
----------------    ----------      ----------  ----------    ----------- (7)
DB_CLIENTS          MOOSDB_alpha    106.2       alpha         "uXMS,DBWebServer,"
DB_TIME             MOOSDB_alpha    107.2       alpha         1325701208.08963
DB_UPTIME           MOOSDB_alpha    107.2       alpha         107.20791
FRUIT               pFruit          143.21      alpha         "apples"
ANGLE               uMeasure        107.2       alpha         135
SPEED               uMeasure        107.2       alpha         2.8
NAME                pIdentity       3.91        gamma         "alpha"
WIDTH               uMeasure        1873.24     alpha         86
HOURS               uMeasure        1873.25     alpha         23
-- displaying all variables --
```

Name of the app that *last* published the variable

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

By default, the screen will refresh whenever one variable value changes

```
VarName            (S)ource        (T)ime      (C)ommunity   VarValue
---------------    ----------      ----------  ----------    ---------- (7)
DB_CLIENTS         MOOSDB_alpha    106.2       alpha         "uXMS,DBWebServer,"
DB_TIME            MOOSDB_alpha    107.2       alpha         1325701208.08963
DB_UPTIME          MOOSDB_alpha    107.2       alpha         107.20791
FRUIT              pFruit          143.21      alpha         "apples"
ANGLE              uMeasure        107.2       alpha         135
SPEED              uMeasure        107.2       alpha         2.8
NAME               pIdentity       3.91        gamma         "alpha"
WIDTH              uMeasure        1873.24     alpha         86
HOURS              uMeasure        1873.25     alpha         23
-- displaying all variables --
```

Time of the *last* publication

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

By default, the screen will refresh whenever one variable value changes

```
VarName            (S)ource        (T)ime      (C)ommunity   VarValue
---------------    ----------      ----------  ----------    ---------- (7)
DB_CLIENTS         MOOSDB_alpha    106.2       alpha         "uXMS,DBWebServer,"
DB_TIME            MOOSDB_alpha    107.2       alpha         1325701208.08963
DB_UPTIME          MOOSDB_alpha    107.2       alpha         107.20791
FRUIT              pFruit          143.21      alpha         "apples"
ANGLE              uMeasure        107.2       alpha         135
SPEED              uMeasure        107.2       alpha         2.8
NAME               pIdentity       3.91        gamma         "alpha"
WIDTH              uMeasure        1873.24     alpha         86
HOURS              uMeasure        1873.25     alpha         23
-- displaying all variables --
```

The community of the app that made the *last* publication

The uXMS Scope List

uXMS is a simple scoping utility launched from the command line

```
$ uXMS mission.moos --all
```

By default, the screen will refresh whenever one variable value changes

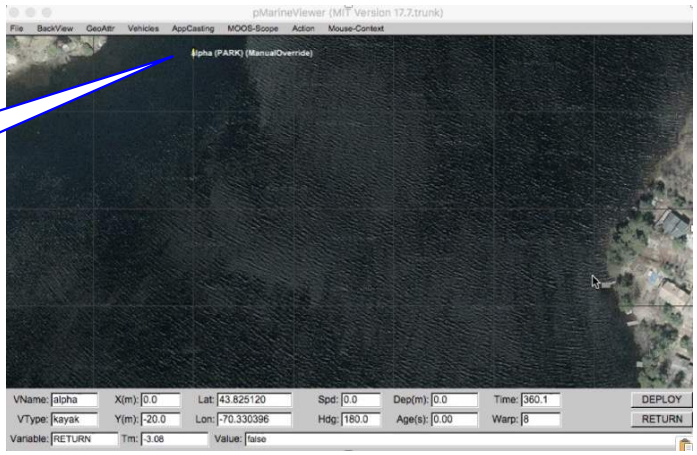
```
VarName            (S)ource      (T)ime      (C)ommunity  VarValue
---------------    ----------    ----------  ----------   ---------- (7)
DB_CLIENTS         MOOSDB_alpha  106.2       alpha        "uXMS,DBWebServer,"
DB_TIME            MOOSDB_alpha  107.2       alpha        1325701208.08963
DB_UPTIME          MOOSDB_alpha  107.2       alpha        107.20791
FRUIT              pFruit        143.21      alpha        "apples"
ANGLE              uMeasure      107.2       alpha        135
SPEED              uMeasure      107.2       alpha        2.8
NAME               pIdentity     3.91        gamma        "alpha"
WIDTH              uMeasure      1873.24     alpha        86
HOURS              uMeasure      1873.25     alpha        23
-- displaying all variables --
```

The value of the *last* publication

Scoping on the Alpha Example Mission with uXMS

Launch the mission

```
$ cd moos-ivp/ivp/missions/s1_alpha
$ pAntler alpha.moos
```



At the start of the mission the vehicle sits motionless at the start position at point (0,-20) in local coordinates.

Scoping on the Alpha Example Mission with uXMS

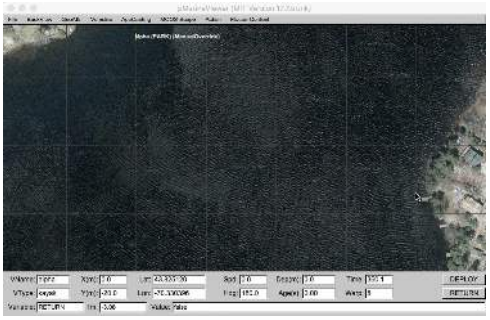
Launch the mission

```
$ cd moos-ivp/ivp/missions/s1_alpha
$ pAntler alpha.moos
```

Launch the scope

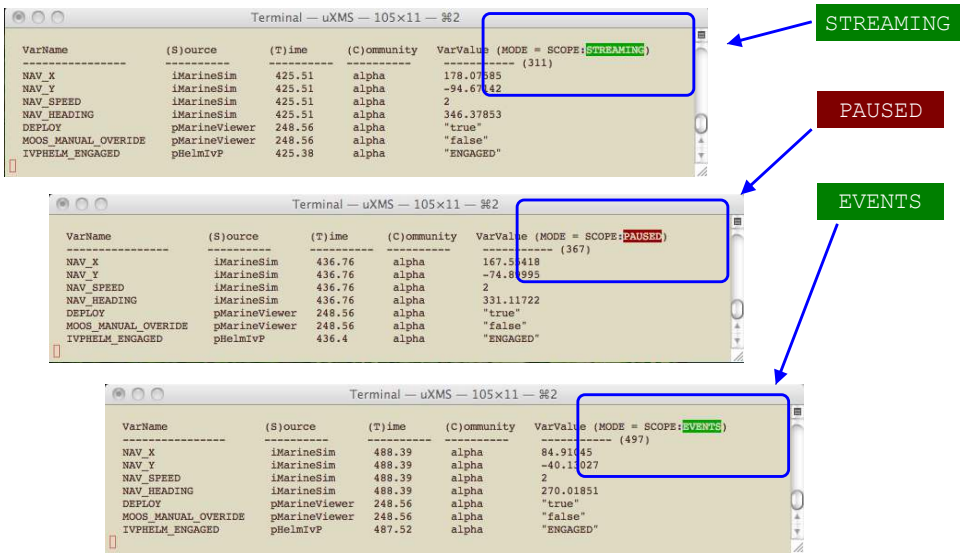
```
$ uXMS alpha.moos. NAV_X NAV_Y NAV_SPEED NAV_HEADING
       DEPLOY IVPHELM_STATE MOOS_MANUAL_OVErRIDE
```

```
============================================================
uXMS_353 alpha                                  0/0(2065)
============================================================
VarName               (S)ource       (T)ime   (C)  VarValue (SCOPING:EVENTS)
--------------------  -------------  -------  ---  --------------------------
DEPLOY                pMarineViewer  1808.98       "true"
IVPHELM_STATE         pHelmIvP       1972.28       "DRIVE"
MOOS_MANUAL_OVERRIDE  pMarineViewer  1808.98       "false"
NAV_HEADING           uSimMarine     1972.20       83.331698
NAV_SPEED             uSimMarine     1972.20       3.58
NAV_X                 uSimMarine     1972.20       70.907074
NAV_Y                 uSimMarine     1972.20       -161.709742
```



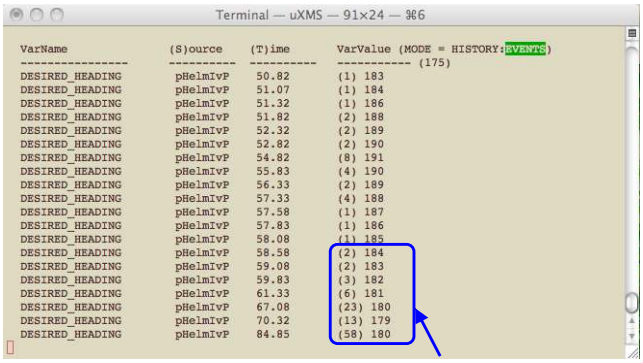
The uXMS Utility Refresh Mode Indicator

The uXMS *refresh mode* is indicated in the top right-hand corner of each report:



The uXMS Utility: The "History" Content Mode

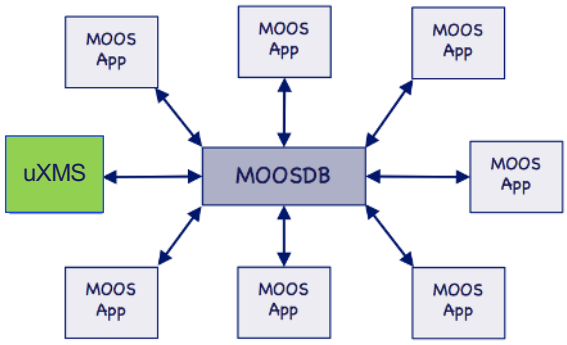
```
$ uXMS mission.moos  --history=DESIRED_HEADING
```



Successive duplicate entries are condensed into a single line with the number of duplicates indicated in parentheses.

Scoping LOCALLY

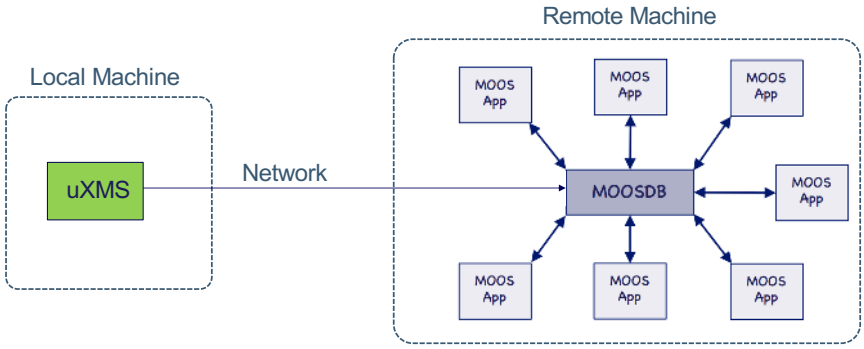
- Typically, a scope is run on the same machine as the rest of the MOOS Community.



Scoping REMOTELY

- A scope may also connect to a remote machine
- Need to specify IP Address, Port Number:

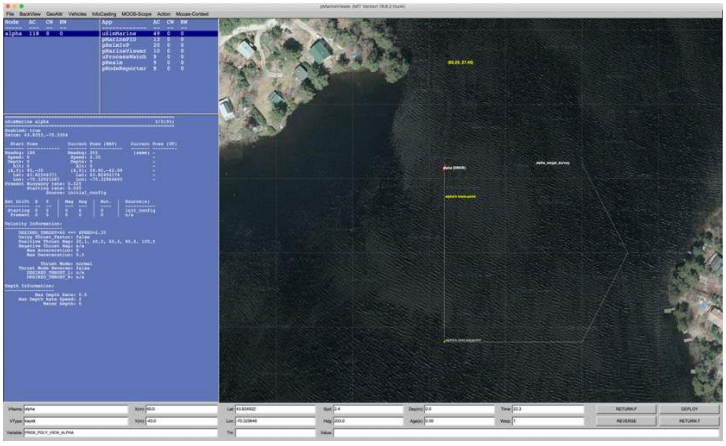
```
$ uXMS mission.moos  --serverhost=18.231.8.45 --serverport=9200
```



Scoping with RealmCasting (Introduced in 2021)

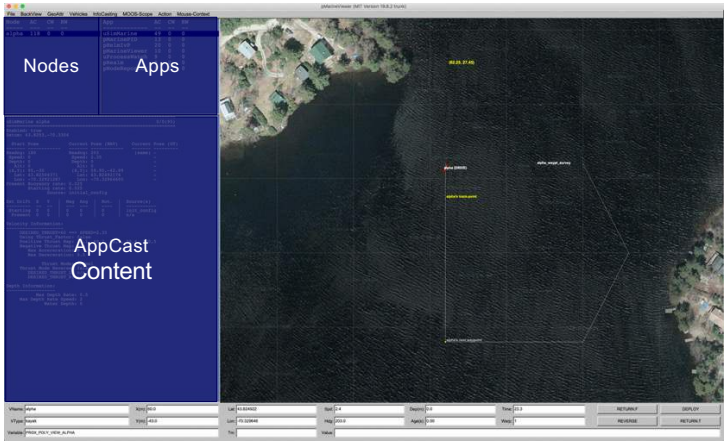
Scoping with RealmCasting

Default window configuration upon launch



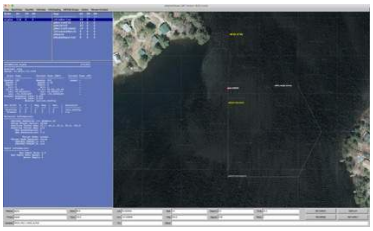
Scoping with RealmCasting

Default window configuration upon launch

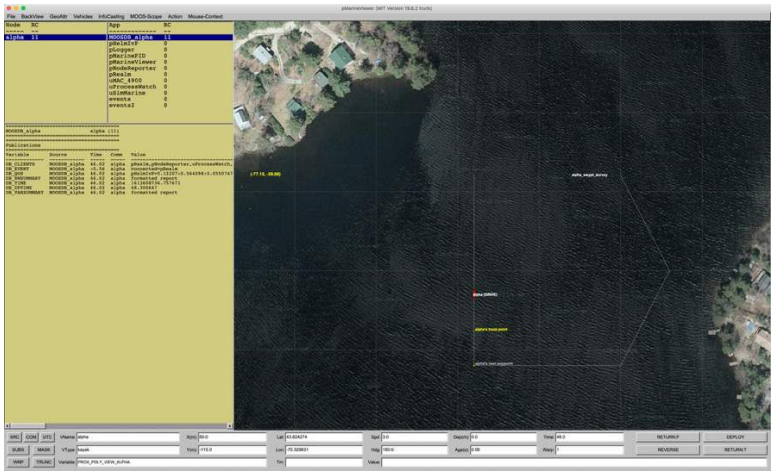


Scoping with RealmCasting

Default window configuration upon launch

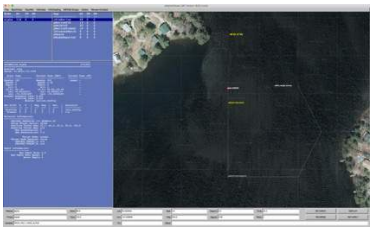


Hit 'a' to Toggle

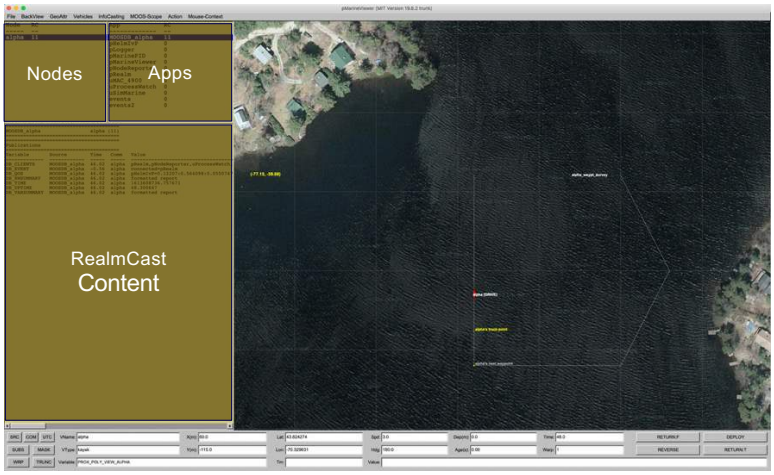


Scoping with RealmCasting

Default window configuration upon launch



Hit 'a' to Toggle

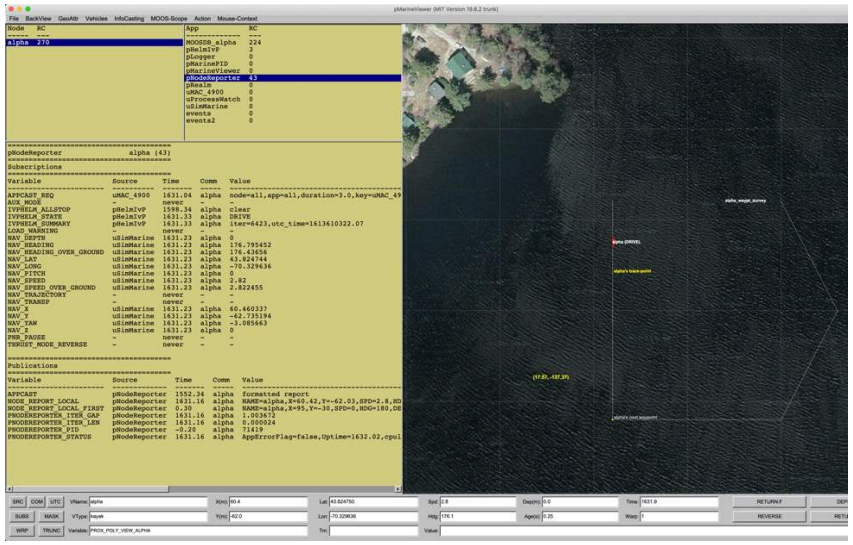


Three Architectures | MOOS Overview | MOOS Messages | Launching Missions | Scoping MOOS | Poking MOOS | Data Logging

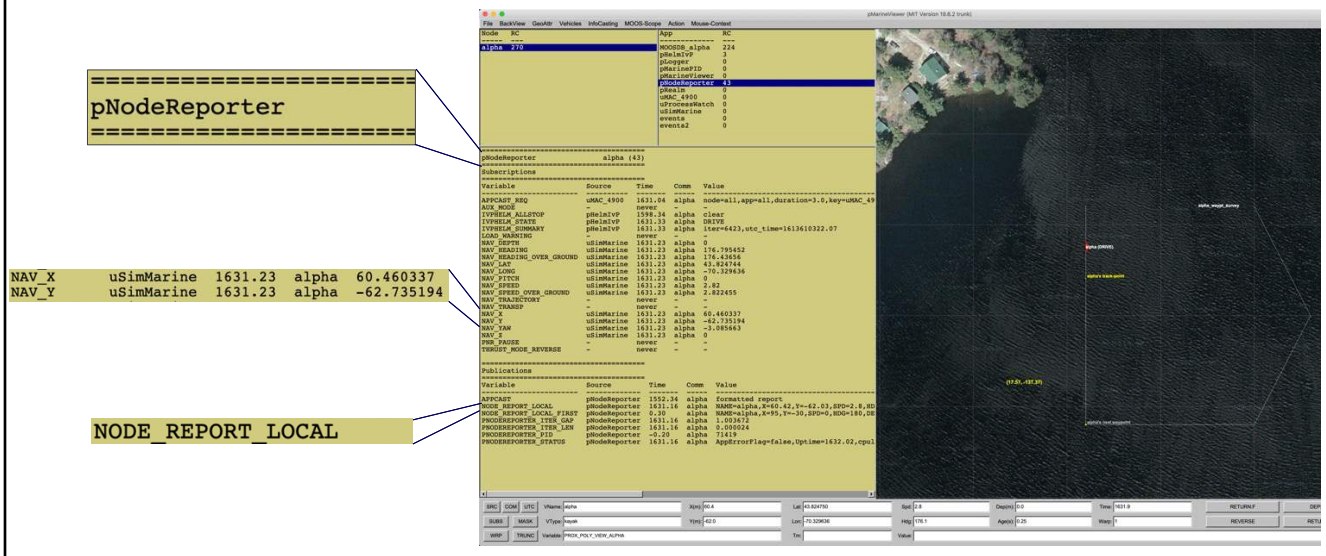
Michael Benjamin, Spring 2026 MIT Dept of Mechanical Engineering

70

Scoping with RealmCasting



Scoping with RealmCasting



MITMECHE

Poking MOOS

73

MITMECHE

Poking MOOS

Poking the MOOSDB :

- A write to the MOOSDB
- Implies that it is outside a typical application write to the MOOSDB

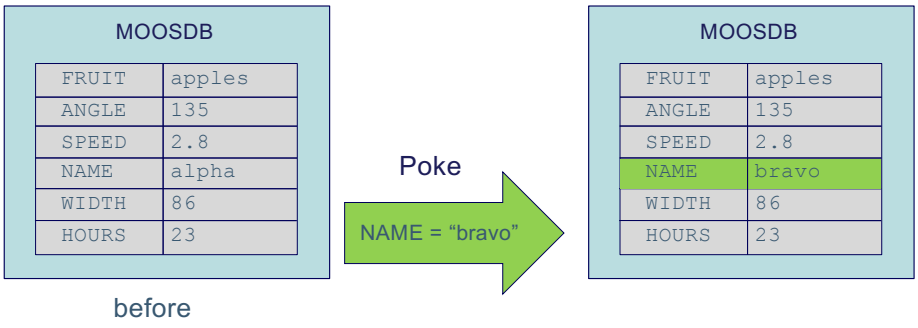
MOOSDB

FRUIT	apples
ANGLE	135
SPEED	2.8
NAME	alpha
WIDTH	86
HOURS	23

74

Changing a Variable Value with a MOOS Poke

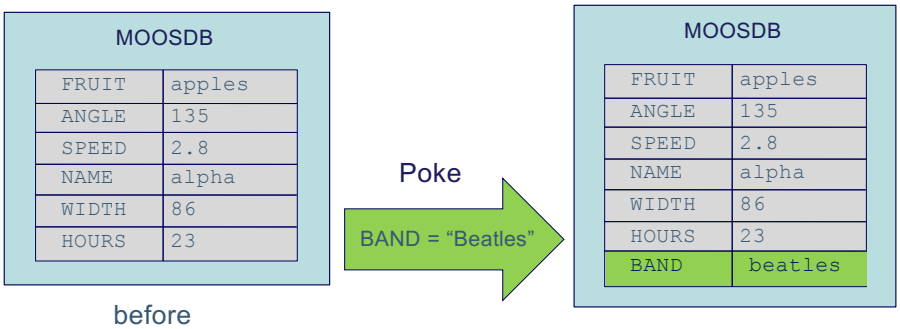
- A poke may simply alter the variable value



before

Publishing a New Variable with a MOOS Poke

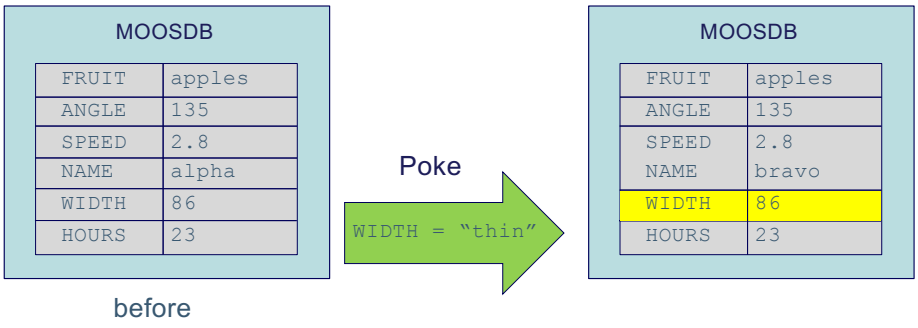
- A poke may write to a new MOOS variable



before

A Poke May Not Change an Existing Variable Type

- Once a variable is of type string – it is always a string
- Once a variable is of type double – it is always a double
- Subsequent pokes are ignored

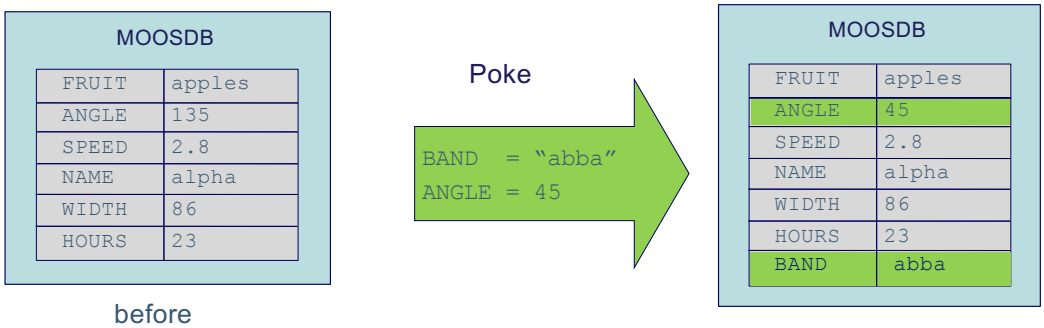


before

Poking with uPokeDB

- uPokeDB is a command line tool for poking the MOOSDB

```
$ uPokeDB mission.moos  BAND="abba"  ANGLE=45
```



before

MOOS Conventions

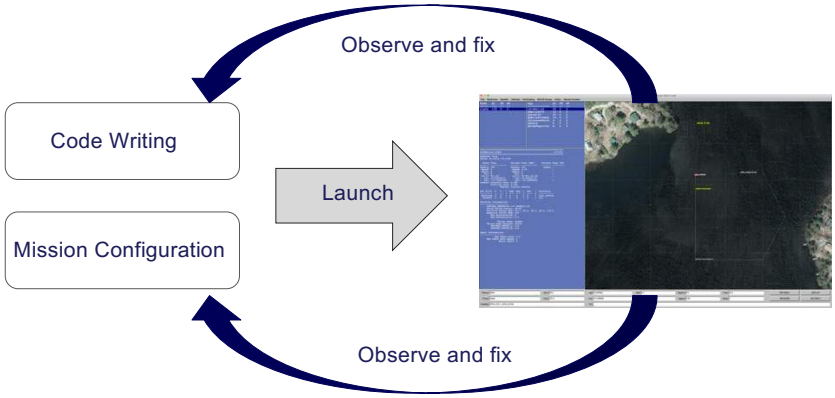
MOOS Variables are
- Typically uppercase
- seldom use numbers
- Never have white space
- Only special character is the underscore '_'

Nice Variables:
- NAV_HEADING
- TOTAL_POINTS
- DESIRED_SPEED
- CLIENTS

Ugly Variables:
- TIME OF DAY
- basic_value
- #ofdays
- SLIP-JOINT

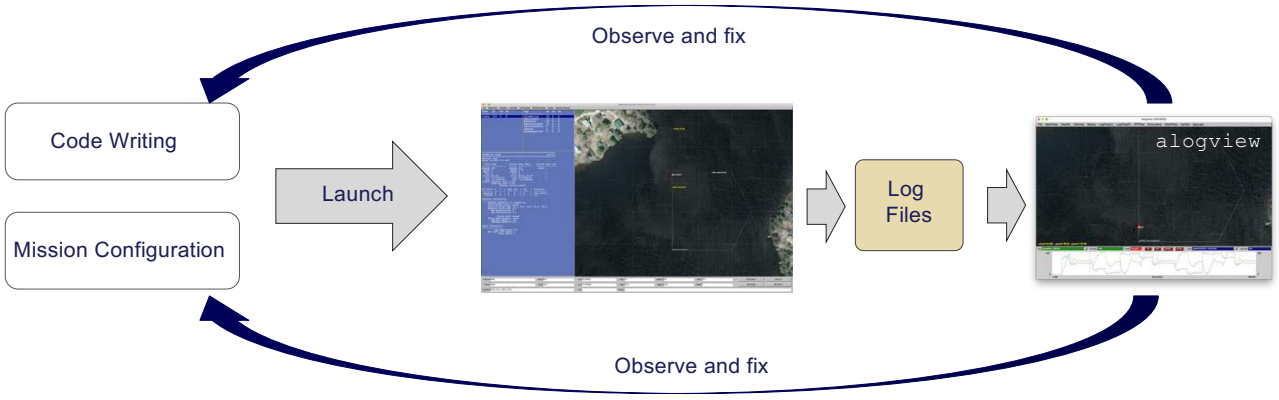
Data Logging

Development Cycle



In many cases, observation of the mission provides enough insight into how to fix either the code or mission configuration.

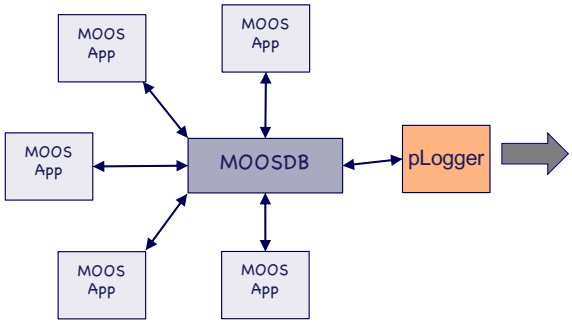
Development Cycle



Log files and log tools are used when the problem too complex to solve from live observation.

Data Logging

pLogger is a MOOS application that logs all or select publications to a file.




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOG FILE:         ./LOG_JAMES/JAMES.alog
%% FILE OPENED ON   Tue Jan 10 22:51:43 2012
%% LOGSTART                 15915046845.4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0.834            DB_UPTIME            MOOSDB_james 8.16382
0.834            DB_CLIENTS           MOOSDB_james
0.994            NAV_Y                uSimMarine -25.00000
0.994            NAV_X                uSimMarine 105.00000
0.994            NAV_SPEED_SOG        uSimMarine 0.00000
0.994            NAV_SPEED            uSimMarine 0.00000
0.994            NAV_LONG             uSimMarine -70.32909
0.994            NAV_LAT              uSimMarine 43.82509
```

file.alog

The logger creates at least four files for each mission:

- `file.alog` – asynchronous log (a new entry any time a post is made)
- `file.slog` – synchronous log (a sampling of variable values at fixed intervals)
- `file._bhv` – a log of critical messages
- `file._moos` – a copy of the mission file used to launch the mission.

Log File Format

The alog file format is meant to be human readable.

file.alog

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOG FILE:         ./LOG_JAMES/JAMES.alog
%% FILE OPENED ON   Tue Jan 10 22:51:43 2012
%% LOGSTART                 15915046845.4   <-- UTC Start Time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0.834            DB_UPTIME            MOOSDB_james 8.16382
0.994            NAV_Y                uSimMarine -25.00000
0.994            NAV_X                uSimMarine 105.00000
0.994            NAV_SPEED_SOG        uSimMarine 0.00000
0.994            NAV_SPEED            uSimMarine 0.00000
0.994            NAV_LONG             uSimMarine -70.32909
0.994            NAV_LAT              uSimMarine 43.82509
```

Time Since UTC Start Time | MOOS Variable | MOOS App Source | Data Value

Configuring the pLogger App

pLogger, like other MOOS Apps, has a configuration block in **mission.moos**.

```
ProcessConfig = pLogger
{
  AppTick       = 10
  CommsTick     = 10

  File          = RED_LOG
  PATH          = ./
  AsyncLog      = true
  FileTimeStamp = true

  // Log it all!!!!!
  LogAuxSrc = true
  WildCardLogging = true
}
```


Wildcard Logging with Finer Control
(Exclusion by Pattern Matching)

- Wildcard logging allows you to capture everything
- Variables or variable patterns may be omitted

```
ProcessConfig = pLogger
{
  AppTick       = 10
  CommsTick     = 10

  File          = BLUE_LOG
  PATH          = ./
  AsyncLog      = true
  FileTimeStamp = true

  WildcardLogging = true
  WildcardOmitPattern = *_STATUS
}
```

Will log all MOOS variables *except* those ending with:

_STATUS

Wildcard Logging – Playing it Safe

- What if a variable was excluded by mistake?
- Use the WildcardExclusionLog to log everything otherwise excluded

```
ProcessConfig = pLogger
{
  AppTick        = 10
  CommsTick      = 10

  File           = GREEN_LOG
  PATH           = ./
  AsyncLog       = true
  SyncLog        = true @ 0.2
  FileTimeStamp  = true

  WildcardLogging = true
  WildcardOmitPattern = *_STATUS
  WildcardExclusionLog = true
}
```

Will log all MOOS variables ending with:

_STATUS

in logfile.xlog

The Alog Toolbox

Tools for Modifying and Analyzing Alog Files

Command-Line log file tools:

- `aloggrep`: Prune an alog file by specifying a set of variables to keep.
- `alogscan`, `aloghelm`: Examine the contents of an alog file in a short summary.
- `alogrm`: Prune an alog file by removing a given set of MOOS variables.
- `alogclip`: Prune an alog file by specifying a min/max timestamp

Each tool is a light-weight single-purpose command-line executable.
Each tool accepts the --help command line option for further usage info.

The aloggrep Tool

- The aloggrep tool is passed an alog file and list of variables to *keep*
- Output is to the terminal window

```
$ aloggrep file.alog  NAV_X NAV_Y
```

- If provided the name of a new alog file, the new file is created
- The new file is a syntactically complete alog file (retaining header info)

```
$ aloggrep file.alog  NAV_X NAV_Y   newfile.alog
```

Hint: We often use this tool to help us create a focused set of data for debugging.

The alogrm Tool

- The alogrm tool is passed an alog file and list of variables to *remove*
- Output is to the terminal window

```
$ alogrm file.alog  DB_STATUS
```

- If provided the name of a new alog file, the new file is created
- The new file is a syntactically complete alog file (retaining header info)

```
$ alogrm file.alog  DB_STATUS  newfile.alog
```

Hint: We often use this tool to reduce unnecessary variables to reduce alog file size

The alogclip Tool

- The alogclip tool is passed an alog file and start and end time
- *All entries in this time window will be kept.*
- Output is to the terminal window

```
$ alogclip file.alog  200 1200
```

- If provided the name of a new alog file, the new file is created
- The new file is a syntactically complete alog file (retaining header info)

```
$ alogclip file.alog  200 1200  newfile.alog
```

Hint: We often use this tool to reduce alog file size

Example *alogscan* Output

Terminal — tcsh — 111×30

Variable Name	Lines	Chars	Start	Stop	Sources
DB_CLIENTS	181	16315	-0.57	363.44	MOOSDB_alpha
DB_TIME	358	6086	0.46	363.00	MOOSDB_alpha
DB_UPTIME	358	3119	0.46	363.00	MOOSDB_alpha
VIEW_POINT	859	123241	36.14	363.07	pHelmIvP:waypt_survey
VIEW_SEGLIST	2	130	36.14	36.14	pHelmIvP:waypt_survey,pHelmIvP:hsline
DEPLOY	1	5	12.77	12.77	pHelmIvP
DESIRED_HEADING	1295	11324	12.77	363.07	pHelmIvP
DESIRED_SPEED	1295	9065	12.77	363.07	pHelmIvP
HELM_IPF_COUNT	1293	9051	36.40	363.07	pHelmIvP
PLOGGER_CMD	1	27	12.77	12.77	pHelmIvP
LOGGER_DIRECTORY	36	1152	0.72	355.24	pLogger
DESIRED_RUDDER	6241	47868	9.86	363.36	pMarinePID
DESIRED_THRUST	6248	49976	9.86	363.36	pMarinePID
MOOS_DEBUG	15	319	9.78	36.12	pMarinePID,pHelmIvP
NODE_REPORT_LOCAL	702	105140	6.71	362.92	pNodeReporter
PID_OK	1	4	18.83	18.83	uProcessWatch
PROC_WATCH_FULL_SUMMARY	1	64	18.83	18.83	uProcessWatch
PROC_WATCH_SUMMARY	68	680	18.83	357.93	uProcessWatch
UPW_EVENT	1	38	18.83	18.83	uProcessWatch
NAV_DEPTH	1419	9933	4.66	363.31	uSimMarine
NAV_HEADING	1419	12454	4.66	363.31	uSimMarine
NAV_SPEED	1419	9933	4.66	363.31	uSimMarine
NAV_X	1419	11671	4.66	363.31	uSimMarine
NAV_Y	1419	13056	4.66	363.31	uSimMarine

Total variables: 24
Start/Stop Time: -0.57 / 363.44
ptsur:sl_alpha/MOOSLog_12_1_2012_____06_57_53(42kool)%

Will report behavior sources on helm output.

Will report multiple sources if applicable.

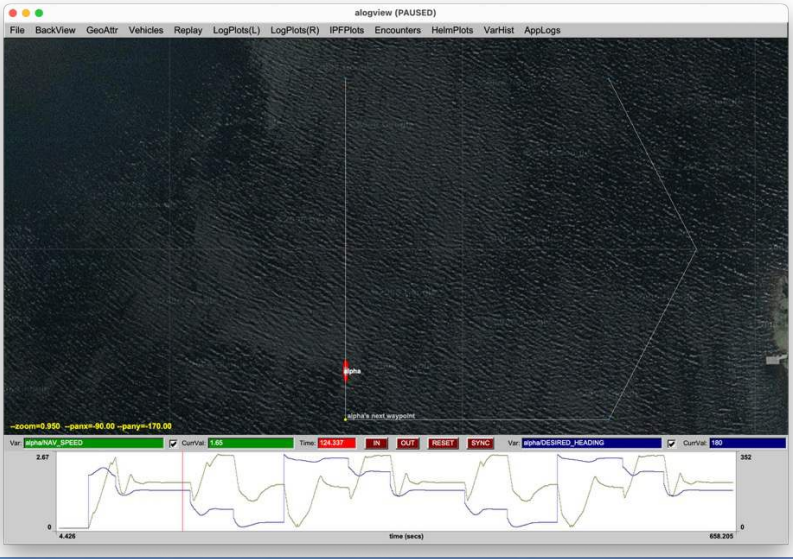
The alogview Replay Utility

Notables:

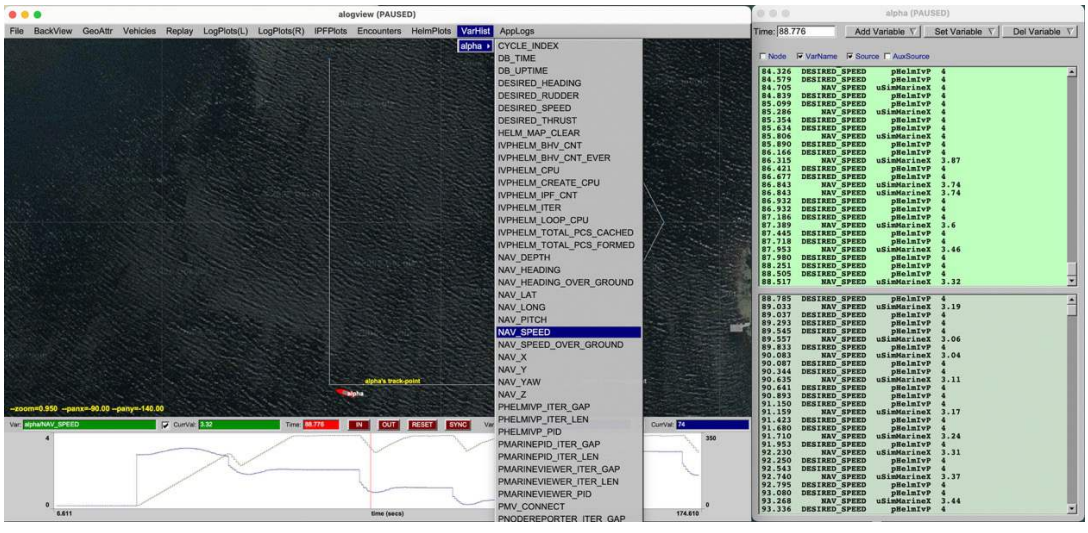
- Replay the mission, start, stop, or jump around in time.
- Replay any number of vehicles, auto synchronized.
- View logged data plotted vs. time
- View Helm objective functions, helm state, stepping through time.
- View terminal output produced by any application.

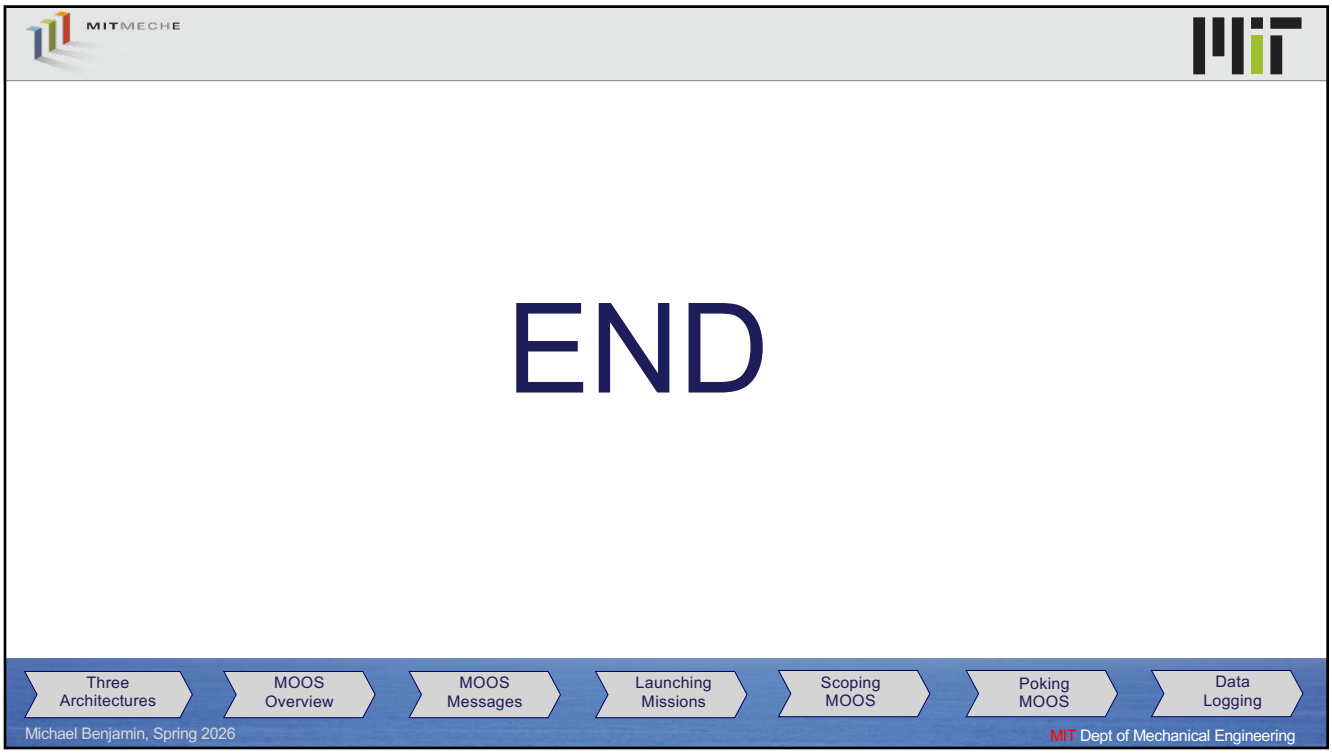
```
$ alogview file.alog
```

```
$ alogview file1.alog. \
file2.alog ... fileN.alog
```



Variable History in Alogview




MITMECHE
END
Three Architectures
MOOS Overview
MOOS Messages
Launching Missions
Scoping MOOS
Poking MOOS
Data Logging
Michael Benjamin, Spring 2026
MIT Dept of Mechanical Engineering

95