

MIT 2.680
UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 4: Introduction to MOOS Programming

February 22nd, 2024

Web: <http://oceanai.mit.edu/2.680>
Email: [Mike Benjamin, mikerb@mit.edu](mailto:mikerb@mit.edu)

2.681 Spring 2024 – Marine Autonomy – “Programming MOOS Applications” Photo by Arjan Vermeij, CMRE

1


MITMECHE MIT

Big Picture Preview of Today's Lecture and Lab


MOOS Classes MOOS Messages MOOS Mail MOOS App Functions Serialization Time Warp AppCasting MOOS Apps MOOS Conventions

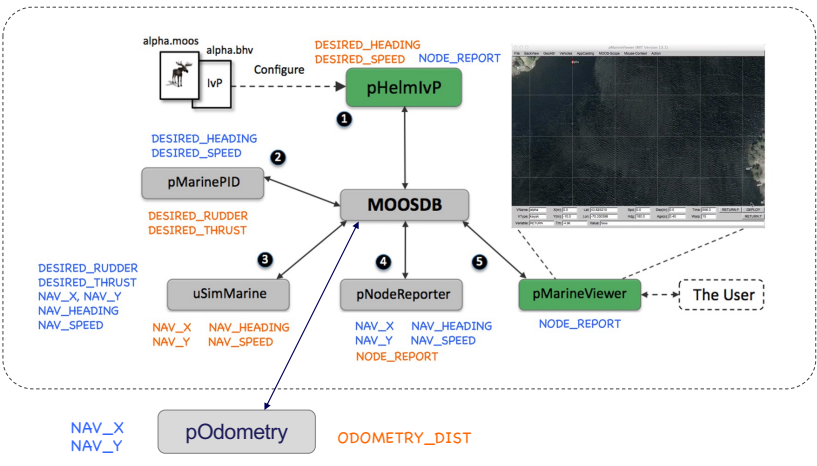
Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

2



Adding an Odometry MOOS App





- In today's lab we will write our first MOOS App, to calculate odometry distance
- We will then use this MOOS app to be involved in the Helm's decision-making

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

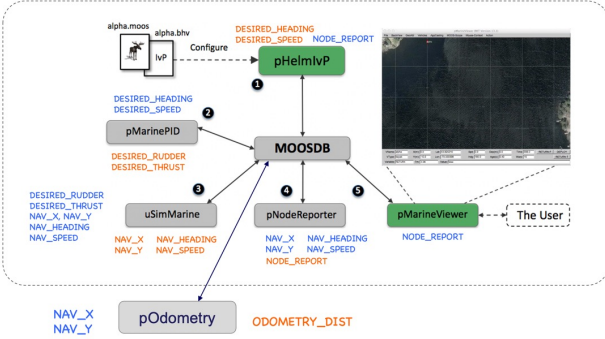
Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering


3



Adding an Odometry MOOS App







Running the Alder Mission (in your moos-ivp-extend tree)

```
$ cd moos-ivp-extend/missions/alder
$ pAntler alder.moos -MOOSTimeWarp=10
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering

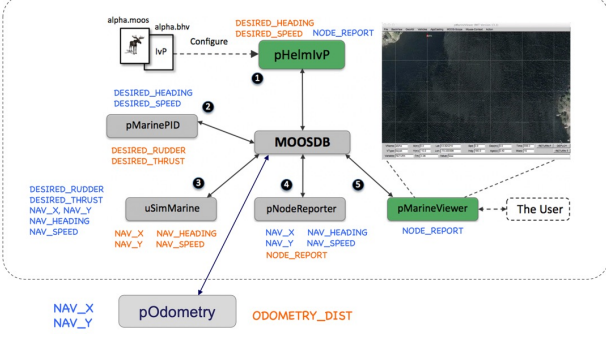
4


2



Adding an Odometry MOOS App







**The Alder Mission
with Odometry**
MIT 2.680

- The Odometry App will publish odometry distance
- The Helm will transition to return home after a certain distance has been achieved.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

5




Adding an Odometry MOOS App





The Alder Mission
MIT 2.680



**The Alder Mission
with Odometry**
MIT 2.680

- The Odometry App will publish odometry distance
- The Helm will transition to return home after a certain distance has been achieved.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

6



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

➔

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

Part 2: Appcasting

- Motivation and How to use Appcasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

Part 3 Good : MOOS App Conventions

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

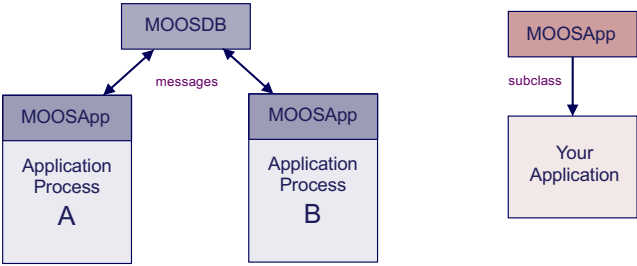
7



MOOS Applications and Inheritance



- In general, MOOS applications are a subclass of the MOOSApp superclass.
- The parent class implementation does most of the work behind the scenes.



```

graph TD
    MOOSDB[MOOSDB]
    subgraph A [MOOSApp Application Process A]
        A
    end
    subgraph B [MOOSApp Application Process B]
        B
    end
    subgraph C [MOOSApp]
        C
    end
    subgraph D [Your Application]
        D
    end
    A -- messages --> MOOSDB
    B -- messages --> MOOSDB
    C -- subclass --> D
    
```

Each application:

- Publishes certain messages
- Subscribes for certain messages

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

8

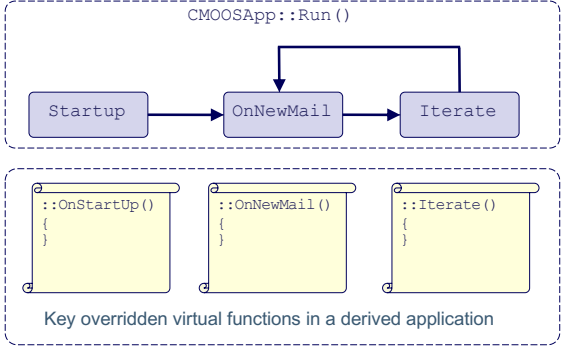
4



The MOOSApp Superclass



Each MOOS application has the option of overriding key parent class *virtual functions*.



Key overridden virtual functions in a derived application

Subscribe
Handle Mail
Process/Publish

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

9



Example Class Definition



The Relayer class definition is in:

```
moos-ivp-extend/
  trunk/src/pXRelayTest/
```

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3      public:
4          Relayer();
5          virtual ~Relayer() {};
6
7          bool OnNewMail(MOOSMSG_LIST &NewMail);
8          bool OnStartup();
9          bool Iterate();
10         bool OnConnectToServer();
11
12         void RegisterVariables();
13
14     protected:
15         // Local member variables
16     };

```

Include CMOOSApp definition and subclass

Declare the constructor
Declare and define the destructor.

Declare the CMOOSApp superclass **virtual functions** for overloading

Declare a utility function where registrations happen

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

10



MOOS Messages



- The form of the data passed between clients is constrained by MOOS:

Name	The name of the data
StringVal	Data in string format
DoubleVal	Numeric double float data
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
DataType	Type of data (STRING or DOUBLE or BINARY)
MessageType	Type of message (usually NOTIFICATION)
Community	The community to which the source process belongs

- Typically, the data type is either a string or a double.
- Binary data may be packed into the string field (images or other data structures etc)

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

11



Posting MOOS Messages (from with an application)



Messages are posted with the `Notify()` function.

```

string moos_var = "WELCOME_MESSAGE";
string moos_msg = "Hello World!";
Notify(moos_var, moos_msg);

```

Caller specified

Left Empty

Automatically Filled-in

Name	The name of the data
StringVal	Data in string format
DoubleVal	Numeric double float data
Source	Name of client that sent this data to the MOOSDB
SourceAux	Optional additional infor about the source client
Time	Time at which the data was written
DataType	Type of data (STRING or DOUBLE or BINARY)
Msg Type	Type of message (usually NOTIFICATION)
Community	The community to which the source process belongs

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

12



Posting MOOS Messages



(from with an application)

Messages are posted with the `Notify()` function.

```
string moos_var = "WELCOME_MESSAGE";
string moos_msg = "Hello World!";
Notify(moos_var, moos_msg);
```

Caller specified

Left Empty

Automatically Filled-in

Name	
StringVal	Data in string format
DoubleVal	Numeric double float data
Source	Name of client that sent this data to the MOOSDB
SourceAux	Optional additional info about the source client
Time	Time at which the data was written
DataType	Type of data (STRING or DOUBLE or BINARY)
Msg Type	Type of message (usually NOTIFICATION)
Community	The community to which the source process belongs

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

13



Posting MOOS Messages



More examples

Posting Literals (string)

```
Notify("WELCOME_MESSAGE", "Hello World");
```

Posting Literals (double)

```
Notify("TEMPERATURE", 98.6);
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

14



MOOS Message Names



By *convention*, MOOS message names are all UPPER CASE letters with numbers and underscores. A further convention is that that begin with a letter.

Good Examples:

```
TEMP
CURRENT_VAL
COMPONENT_1
COMPONENT_278
TIME_TO_COLLISION
```

Meh:

```
Bad-idea
7854
_HELLO?
```

That being said, MOOS will let almost anything through, even white space. Why?

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

15



Posting MOOS Messages



Message Source Information

- When a message is posted to the MOOSDB, the source field is automatically filled in.

If Application pFooBar posts a message:

```
Notify("WELCOME_MESSAGE", "Hello World");
```

↓

Received by another app:

```
MOOSMsg msg;
cout << "Variable: " << msg.GetKey() << endl;
cout << "Value:    " << msg.GetString() << endl;
cout << "Source:    " << msg.GetSource() << endl;
```

The output would be:

```
Variable: WELCOME_MESSAGE
Value:    Hello World
Source:    pFooBar
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

16



Posting MOOS Messages



Specifying the Auxiliary Source

- When a message is posted to the MOOSDB, the **auxiliary source** field is typically left *empty*.

If Application pFooBar posts a message:

```
Notify("WELCOME_MESSAGE", "Hello World", "Special Greeter");
```

↓

Received by another app:

```
MOOSMsg msg;
cout << "Variable: " << msg.GetKey() << endl;
cout << "Value: " << msg.GetString() << endl;
cout << "Source: " << msg.GetSource() << endl;
cout << "SrcAux: " << msg.GetSourceAux() << endl;
```

The output would be:

```
Variable: WELCOME_MESSAGE
Value:    Hello World
Source:   pFooBar
SrcAux:   Special Greeter
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

17



Posting MOOS Messages



Timestamps

- The *timestamp* of a message posted to the MOOSDB, is the time when the message was posted.
- Not the time received by the MOOSDB. Not the time received by the receiving application.

If Application pFooBar posts a message:

```
Notify("WELCOME_MESSAGE", "Hello World");
```

↓

Received by another app:

```
MOOSMsg msg;
cout << "NowTime: " << MOOSTime() << endl;
cout << "Variable: " << msg.GetKey() << endl;
cout << "Value: " << msg.GetString() << endl;
cout << "MsgTime: " << msg.GetTime() << endl;
```

The output would be:

```
NowTime: 1.39273e+09
Variable: WELCOME_MESSAGE
Value:    Hello World
MsgTime: 1.39273e+09
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

18



MOOS Message Functions



(nearly) full list

A summary of functions defined on MOOS messages:

```

MOOSMsg msg;
string msg.GetKey()           // Get the MOOS variable name
string msg.GetName()          // Get the MOOS variable name
bool  msg.IsString()           // true if message type is string
bool  msg.IsDouble()           // true if message type is double
string msg.GetString()         // Get the message string contents
String msg.GetDouble()         // Get the message double contents
string msg.GetSource()         // Get the sender information
string msg.GetSourceAux()      // Get further sender information
string msg.GetCommunity()      // Get the sender community information
double msg.GetTime()           // Get the time message was posted
  
```

There's more. If you want to see for yourself, take a look at:

```

$ cd moos-ivp/MOOS/MOOSCore/Core/libMOOS/Comms/include/MOOS/libMOOS/Comms/
$ emacs MOOSMsg.h
  
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

19



Intro to MOOS Programming Outline



Part 1: General
MOOS App
Concepts

Part 2:
Appcasting

Part 3 Good :
MOOS App
Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

20




Registering for MOOS Mail Messages




(from within an application)

- Messages are registered with the `Register()` function.

```
bool Register(string, double);
```


 Name of the
MOOS Variable


 Min time interval
between notifications

```
Register("WELCOME_MESSAGE", 0);  
Register("GOODBYE_MESSAGE", 0.5);
```

- Incoming mail for `WELCOME_MESSAGE` will be received *each* time another client posts to this variable.
- Incoming mail for `GOODBYE_MESSAGE` to be received at most twice per second.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

21



More on Registering for MOOS Mail



(In case you were wondering....)

Q: How is mail read by an application?

A: In the `OnNewMail()` function. (We'll get to that shortly)

Q: What are legal variables names?

A: Anything but an empty string allowed. By *convention*, variables consist solely of uppercase letters, numbers, and the underscore character.

Q: Are there any ill effects from registering for a variable twice?

A: No. The 2nd registration is just ignored. Even if the min-interval arg is different!

Q: Can an application send mail to itself?

A: Yes, but the app still must register for it like other apps.

Q: Is it possible to un-register for a variable? (Why would one want to?)

A: Yes, the call is `UnRegister(VARNAME);`

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

22



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: Appcasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

23



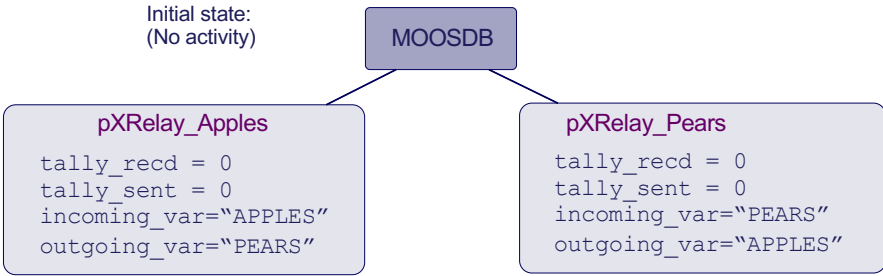
Our Old Friend pXRelay



The pXRelay application works as follows:

- It registers for mail on a given variable (e.g., "APPLES").
- When it receives mail for the variable, it increments a local "received" counter.
- For every mail received, it publishes on another given variable (e.g., "PEARS").

Initial state:
(No activity)



```

graph TD
    MOOSDB[MOOSDB] --- pXRelay_Apples[pXRelay_Apples]
    MOOSDB --- pXRelay_Pears[pXRelay_Pears]
    subgraph pXRelay_Apples_state [pXRelay_Apples]
        t1[tally_recd = 0]
        t2[tally_sent = 0]
        t3[incoming_var="APPLES"]
        t4[outgoing_var="PEARS"]
    end
    subgraph pXRelay_Pears_state [pXRelay_Pears]
        t5[tally_recd = 0]
        t6[tally_sent = 0]
        t7[incoming_var="PEARS"]
        t8[outgoing_var="APPLES"]
    end
  
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

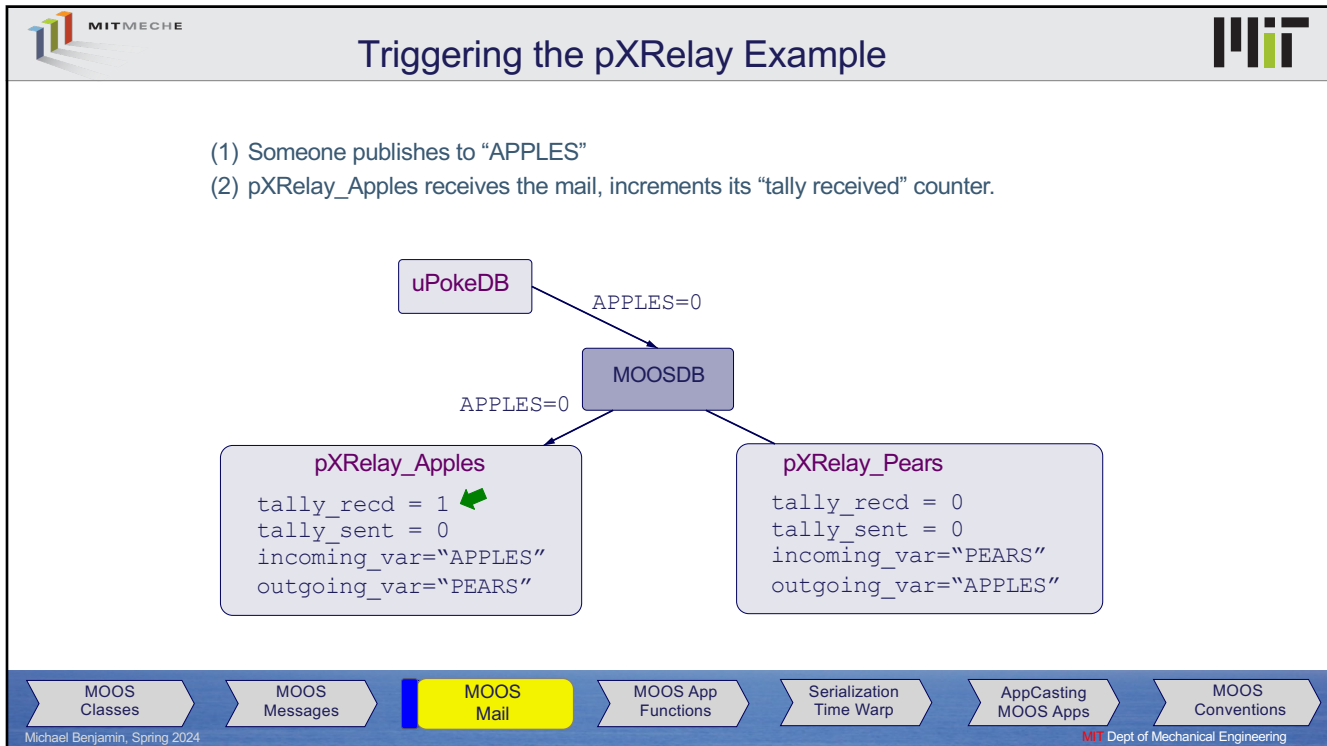
Serialization Time Warp

AppCasting MOOS Apps

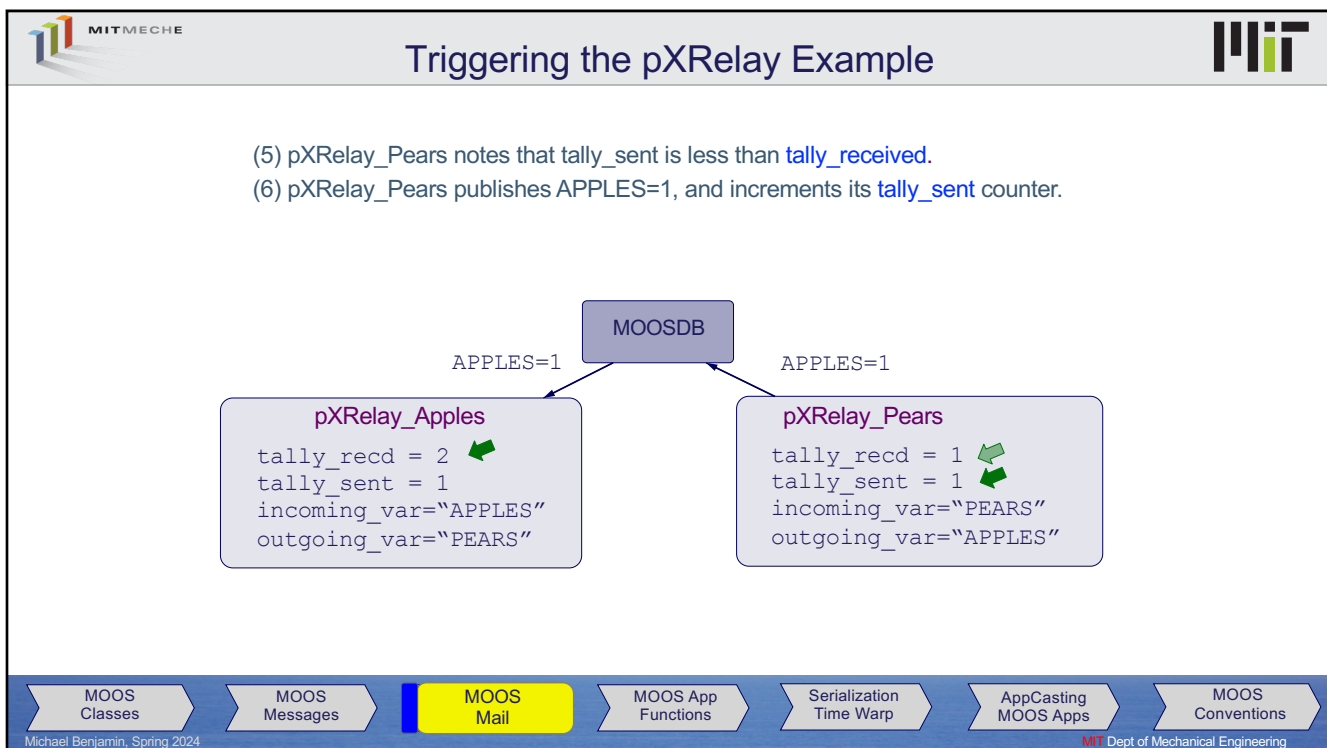
MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering


24




25



26



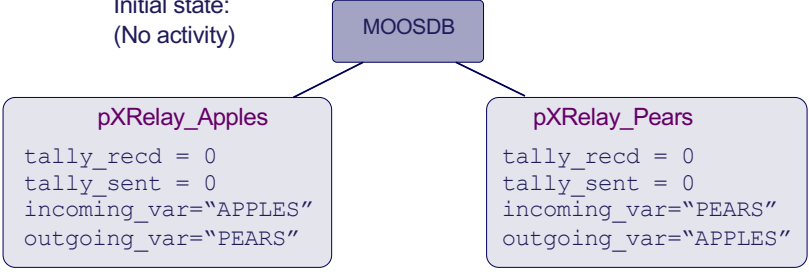
Our Old Friend pXRelay



What's happening inside

- OnStartup(),
- OnNewMail(),
- Iterate()?

Initial state:
(No activity)



MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

27



The "Relayer" Class Definition



The Relayer class definition is in:

```
moos-ivp-extend/trunk/
src/pXRelayTest/
```

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_recd;
16     unsigned long int m_tally_sent;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor
Declare and define the destructor.

Declare the CMOOSApp superclass
virtual functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

28



The “Relayer” Class Definition



The Relayer class definition is in:

moos-ivp-extend/trunk/
src/pXRelayTest/

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor. Declare and define the destructor.

Declare the CMOOSApp superclass virtual functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


Michael Benjamin, Spring 2024

MIT Dept of Mechanical Engineering

29



The “Relayer” Class Definition



The Relayer class definition is in:

moos-ivp-extend/trunk/
src/pXRelayTest/

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor. Declare and define the destructor.

Declare the CMOOSApp superclass virtual functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


Michael Benjamin, Spring 2024

MIT Dept of Mechanical Engineering

30



The “Relayer” Class Definition



The Relayer class definition is in:

moos-ivp-extend/trunk/
src/pXRelayTest/

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor
Declare and define the destructor.

Declare the CMOOSApp superclass **virtual** functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering

31



The “Relayer” Class Definition



The Relayer class definition is in:

moos-ivp-extend/trunk/
src/pXRelayTest/

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor
Declare and define the destructor.

Declare the CMOOSApp superclass **virtual** functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering

32



The “Relayer” Class Definition



The Relayer class definition is in:

```
moos-ivp-extend/trunk/
src/pXRelayTest/
```

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

Include CMOOSApp definition and subclass it

Declare the constructor
Declare and define the destructor.

Declare the CMOOSApp superclass **virtual** functions for overloading

Declare a utility function where registrations happen

Keep track of received and outgoing message counts

Store the user's choice for incoming and outgoing variables.

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

33



The “Relayer” Class Definition



The Relayer class definition is in:

```
moos-ivp-extend/trunk/
src/pXRelayTest/
```

```

0  #include "MOOS/libMOOS/MOOSLib.h"
1  class Relayer : public CMOOSApp
2  {
3  public:
4      Relayer();
5      virtual ~Relayer() {};
6
7      bool OnNewMail(MOOSMSG_LIST &NewMail);
8      bool OnStartup();
9      bool Iterate();
10     bool OnConnectToServer();
11
12     void RegisterVariables();
13
14 protected:
15     unsigned long int m_tally_rcd;
16     unsigned long int m_tally_snt;
17
18     std::string m_incoming_var;
19     std::string m_outgoing_var;
20 };

```

CONVENTION:
 Functions declared before member variables

CONVENTION:
 Class member variables begin with m_

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

34



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relay::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param  = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

35



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relay::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param  = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

36



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relayer::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

37



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relayer::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

38



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relay::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering

39



The pXRelay OnStartup() Method



The task of the OnStartup() method is to grab the two configuration parameters for pXRelay –

The **INCOMING_VAR** and **OUTGOING_VAR**

```

0  bool Relay::OnStartup()
1  {
2      STRING_LIST sParams;
3      m_MissionReader.GetConfiguration(GetAppName(), sParams);
4
5      STRING_LIST::iterator p;
6      for(p = sParams.begin(); p!=sParams.end(); p++) {
7          string line  = *p;
8          string param = MOOSChomp(line, "=");
9          string value  = line;
10
11         if(MOOSStrCmp(param, "INCOMING_VAR"))
12             m_incoming_var = value;
13
14         else if(MOOSStrCmp(param, "OUTGOING_VAR"))
15             m_outgoing_var = value;
16     }
17
18     RegisterVariables();
19     return(true);
20 }

```

Declare function

Get the list of parameters from the .moos file

Iterate through the list of param=value pairs

If param matches, store value in local member var

If param matches, store value in local member var

Now that incoming var is known, register for it!

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
MIT Dept of Mechanical Engineering

40



The pXRelay OnNewMail() Method



The `OnNewMail()` method is called on each application iteration. If there is no mail, it simply returns `true`.

```

0  bool Relayer::OnNewMail(MOOSMSG_LIST &NewMail)
1  {
2      MOOSMSG_LIST::reverse_iterator p;
3      for(p = NewMail.rbegin(); p!=NewMail.rend(); p++) {
4          CMOOSMsg &msg = *p;
5
6          string key = msg.GetKey();
7
8          if(key == m_incoming_var)
9              m_tally_recd++;
10     }
11
12     return(true);
13 }
```

Declare function

Iterate through the MOOS messages

GetKey() returns the MOOS variable

If the key matches the user-specified incoming key, increment the tally

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

41



The pXRelay OnNewMail() Method



The `OnNewMail()` method is called on each application iteration. If there is no mail, it simply returns `true`.

```

0  bool Relayer::OnNewMail(MOOSMSG_LIST &NewMail)
1  {
2      MOOSMSG_LIST::reverse_iterator p;
3      for(p = NewMail.rbegin(); p!=NewMail.rend(); p++) {
4          CMOOSMsg &msg = *p;
5
6          string key = msg.GetKey();
7
8          if(key == m_incoming_var)
9              m_tally_recd++;
10     }
11
12     return(true);
13 }
```

Declare function

Iterate through the MOOS messages

GetKey() returns the MOOS variable

If the key matches the user-specified incoming key, increment the tally

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

42



The pXRelay OnNewMail() Method



The `OnNewMail()` method is called on each application iteration. If there is no mail, it simply returns `true`.

```

0 bool Relayer::OnNewMail(MOOSMSG_LIST &NewMail)
1 {
2     MOOSMSG_LIST::reverse_iterator p;
3     for(p = NewMail.rbegin(); p!=NewMail.rend(); p++) {
4         CMOOSMsg &msg = *p;
5
6         string key = msg.GetKey();
7
8         if(key == m_incoming_var)
9             m_tally_recd++;
10    }
11    return(true);
12 }
13 
```

Declare function

Iterate through the MOOS messages

GetKey() returns the MOOS variable

If the key matches the user-specified incoming key, increment the tally

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

43



The pXRelay OnNewMail() Method



The `OnNewMail()` method is called on each application iteration. If there is no mail, it simply returns `true`.

```

0 bool Relayer::OnNewMail(MOOSMSG_LIST &NewMail)
1 {
2     MOOSMSG_LIST::reverse_iterator p;
3     for(p = NewMail.rbegin(); p!=NewMail.rend(); p++) {
4         CMOOSMsg &msg = *p;
5
6         string key = msg.GetKey();
7
8         if(key == m_incoming_var)
9             m_tally_recd++;
10    }
11    return(true);
12 }
13 
```

Declare function

Iterate through the MOOS messages

GetKey() returns the MOOS variable

If the key matches the user-specified incoming key, increment the tally

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

44



The pXRelay Iterate() Method



The `Iterate()` method is called on each application iteration – after `OnNewMail()` is called.

```

0 bool Relay::Iterate()
1 {
2     unsigned deficit = m_tally_recd - m_tally_sent;
3
4
5     for(unsigned int i=0; i<deficit; i++) {
6         m_tally_sent++;
7         Notify(m_outgoing_var, m_tally_sent);
8     }
9
10    return(true);
11 }

```

Declare function

Determine if we've sent less than we've received

Make N postings where N is the deficit

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

45



The pXRelay Iterate() Method



The `Iterate()` method is called on each application iteration – after `OnNewMail()` is called.

```

0 bool Relay::Iterate()
1 {
2     unsigned deficit = m_tally_recd - m_tally_sent;
3
4
5     for(unsigned int i=0; i<deficit; i++) {
6         m_tally_sent++;
7         Notify(m_outgoing_var, m_tally_sent);
8     }
9
10    return(true);
11 }

```

Declare function

Determine if we've sent less than we've received

Make N postings where N is the deficit

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

46



The pXRelay Iterate() Method



The `Iterate()` method is called on each application iteration – after `OnNewMail()` is called.

```

0 bool Relay::Iterate()
1 {
2     unsigned deficit = m_tally_recd - m_tally_sent;
3
4
5     for(unsigned int i=0; i<deficit; i++) {
6         m_tally_sent++;
7         Notify(m_outgoing_var, m_tally_sent);
8     }
9
10    return(true);
11 }

```

Declare function

Determine if we've sent less than we've received

Make N postings where N is the deficit

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024

MIT Dept of Mechanical Engineering

47



OnStartup() and RegisterVariables() Methods



Variable registration is done both at the end of `OnConnectToServer()` and `OnStartup()`

```

0 bool Relay::OnConnectToServer()
1 {
2     RegisterVariables();
3     return(true);
4 }

```

Declare function

Register here in case not done in OnStartup()

```

0 void Relay::RegisterVariables()
1 {
2     if(m_incoming_var != "")
3         Register(m_incoming_var, 0);
4 }

```

Declare function

Just register for the one user-defined incoming variable

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024

MIT Dept of Mechanical Engineering

48



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: Appcasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

49



General MOOS App Concepts



Message Serialization

- A MOOS Message data type is either a string or a double
- It is common to want to pass a data structure
- For example, if we want to pass a point in 3D space, we could pass `MY_POINT = "22,87,95"`
- The recipient would be responsible for parsing this string back into a 3D points.

Ways of handling this:

Poorest way: Passing the string "22,87,95" and just have an informal agreement that the first field is X, second field is Y, third field is Z.

Better way: Passing the string "x=22, y=87, z=95".
The fields are self-labelled and the message is easily extendable. Still have to have an informal agreement on the field labels.

Even Better way: Create a Class or Structure for holding this information.
Class implements an instance-to-string function.
Class implements a string-to-instance function.

Best(?) way: Use a general-purpose scheme like Google Protocol Buffers to define your data structure in a meta-file and auto-generate the class.

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

50



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: Appcasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- ➔ Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

51



General MOOS App Concepts



Time Warp

- MOOS implements a function called `MOOSTime()`.
It returns the number of seconds since the start of Jan 1st, 1970.
- MOOS simulations may be run with TimeWarp
- In your .moos configuration file:


```
MOOSTimeWarp = 20
```
- From the command-line:


```
pAntler --MOOSTimeWarp=20
```
- From within your app, MOOS implements `GetMOOSTimeWarp()`
You may want to slow down any terminal debug output at high time warps.
- TimeWarp accepts values less than 1, if you want to actually slow things down.

"Time Warp Compliance":
Your app may be "time warp compliant" if it only gets its time from MOOSTime()

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

52



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: Appcasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

➔ **Motivation and How to use AppCasting**

• How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

53

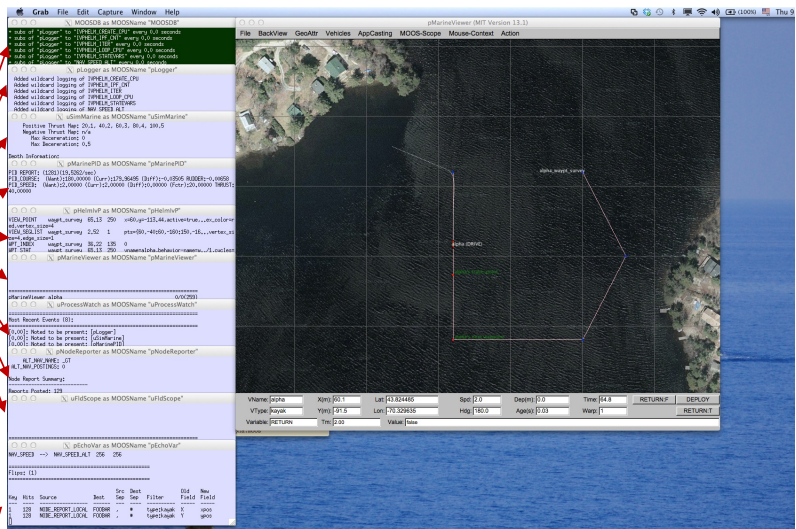


Life Before AppCasting



• One window for each MOOS Application:

10 windows



MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

54



AppCasting in MOOS



AppCasting was motivated by a few observations:

- The biggest headache of users new to MOOS (students in MIT 2.680) was the derailment of a mission due to an unnoticed configuration or runtime error.
- Debugging typically involves re-launching with app terminal windows open and analyzing expected vs. observed output.
- Deploying multiple vehicles each with multiple MOOS Apps means a lot of terminal windows are open.
- On a remotely deployed vehicle, one cannot ssh in and see any application terminal output at all!
- Since terminal output is rarely viewable for the above practical reasons, apps are rarely designed with much thought put into their terminal output.

... Introducing AppCasting in MOOS

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

55



With AppCasting



File BackView GeoAttr Vehicles AppCasting MOOS-Scope Mouse-Context Action

Name	HC	LP	HP	App	HC	LP	HP
alpha	85	0	0	relative	50	0	0
				phelminv	10	0	0
				sfProcessWatch	3	0	0
				sfProcessWatch	3	0	0
				sfProcessWatch	3	0	0
				sfProcessWatch	3	0	0
				sfProcessWatch	3	0	0

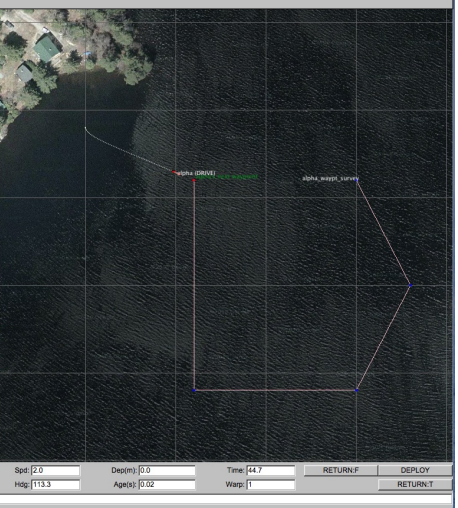
```

=====
phelminv alpha 0/0(179)
=====
Helm Iteration: 122
Helm Functions: 1
Node(s):
SolveTime: 0.00 (max=0.00)
GreedyTime: 0.01 (max=0.01)
LoopTime: 0.01 (max=0.01)
Halted: false (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
speed = 2
course = 113
Behaviors Active: ----- (1)
waypt_survey [30-41] [gpr=100] [pos=6] [cpu=0.14] [upd=0/0]
Behaviors Running: ----- (0)
Behaviors Idle: ----- (1)
waypt_return[always]
Behaviors Completed: ----- (0)

Variable Behavior Time Iter Value
-----
BVV_STATUS waypt_return 14.59 1 name=waypt_return,pos=352388..rue,stat
CYCLE_INDEX waypt_survey 14.59 1 0
VIEW_POINT waypt_survey 14.59 1 x=60,y=-40,active=false,lab..ex_color
VIEW_SEGMENT waypt_survey 14.59 1 pta=(60,-40;60,-160;150,-160;vertex
WPT_INDEX waypt_survey 14.59 1 0
WPT_STAT waypt_survey 45.00 122 vname=alpha,behavior=name=..0/0,cycl

Most Recent Events (1):
[14.59]: var=MOOS_MANTAL_OVERRIDE[matter=true, skew=0.22

```



MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

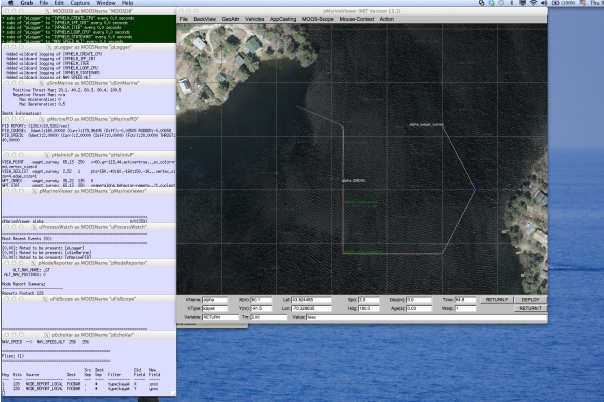
56



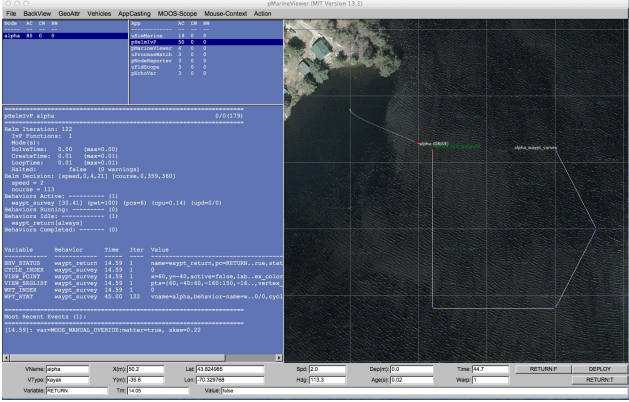
AppCasting



Without



With Appasting



MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

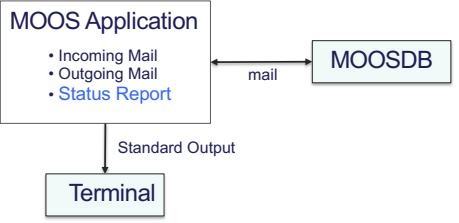
57



MOOS Application I/O



- A typical MOOS application interacts by way of mail and the MOOSDB.



```

graph TD
    subgraph MOOS_Application [MOOS Application]
        direction TB
        IM[Incoming Mail]
        OM[Outgoing Mail]
        SR[Status Report]
    end
    MOOS_Application <-->|mail| MOOSDB[MOOSDB]
    MOOS_Application -- Standard Output --> Terminal[Terminal]
  
```

- Most applications also produce debugging/status info to the terminal.
- Often this format is an afterthought.
- Often this content is out of sight, if a terminal is not open.

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

58



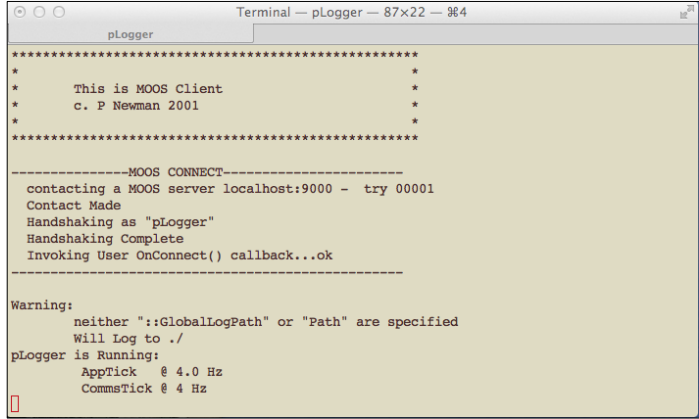
Typical Terminal Output



Typical terminal output of a MOOSApp will show:

- Startup summary and health status,
- A simple heartbeat character or other simple health indicator.

pLogger



MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

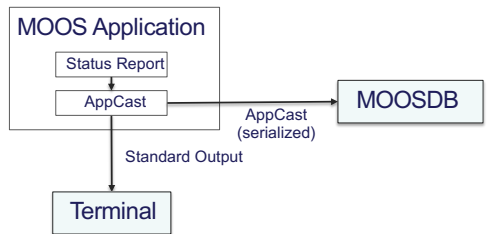
Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

59



Introducing AppCasting





```

graph TD
    subgraph MOOS_Application [MOOS Application]
        SR[Status Report]
        AC[AppCast]
    end
    AC -- "Standard Output" --> T[Terminal]
    AC -- "AppCast (serialized)" --> MOOSDB[MOOSDB]
  
```

An AppCast-Enabled MOOS App:

- Generates an AppCast representing its status report.
- The AppCast is sent to the terminal standard output.
(From the user's perspective it looks like any other MOOS application.)
- The AppCast is also serialized and sent to the MOOSDB.

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

60



An Example AppCast

From the uProcessWatch MOOSApp



List of Strings

List of Events (Limited)

```

uProcessWatch henry (160)
Summary: All Present

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName      Watch Reason  Status
-----
pBasicContactMgr  ANT      DB      OK
pHelmIvP          ANT      WATCH DB      OK
pHostInfo         ANT      DB      OK
pLogger           ANT      DB      OK
pMarinePID        ANT      WATCH DB      OK
pNodeReporter     ANT      WATCH DB      OK
pShare            ANT      DB      OK
uFldMessageHandler ANT      DB      OK
uFldNodeBroker    ANT      DB      OK
uSimMarine        ANT      WATCH DB      OK

Most Recent Events (8):
[4.01]: Resurrected: [uFldMessageHandler]
[2.01]: PROC_WATCH_EVENT: Process [uFldMessageHandler] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]

```

Application Iteration Counter

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

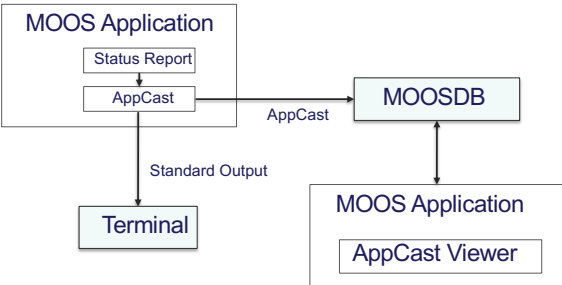
61



AppCast Viewing



- A separate MOOS utility application may be run to view AppCasts from any AppCast-enabled application.



```

graph TD
    subgraph MA1 [MOOS Application]
        SR[Status Report]
        AC1[AppCast]
    end
    subgraph MA2 [MOOS Application]
        ACV[AppCast Viewer]
    end
    MOOSDB[MOOSDB]

    SR --> T[Terminal]
    AC1 --> MOOSDB
    AC1 --> T
    MOOSDB <--> ACV

```

- Now a user can see application output even if an application terminal output is otherwise suppressed (e.g., no open terminal or I/O re-directed to /dev/null).

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

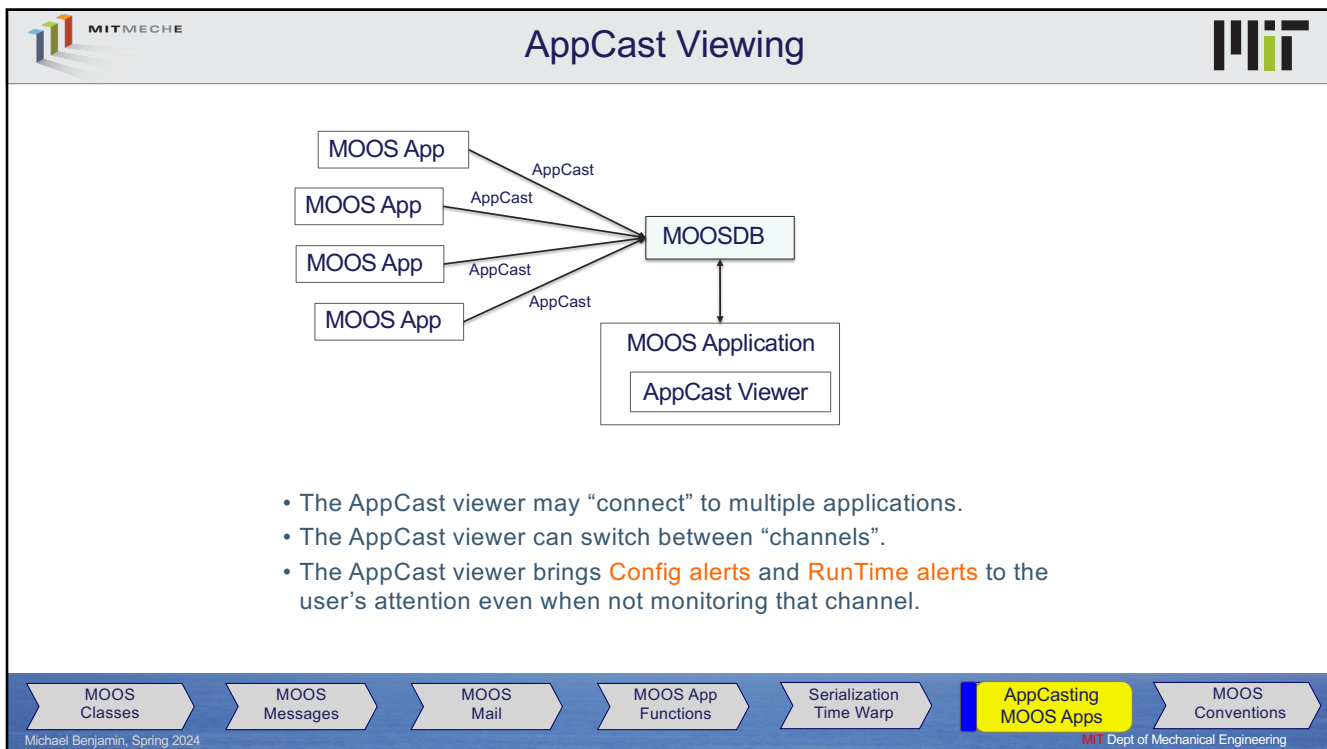
Serialization Time Warp

AppCasting MOOS Apps

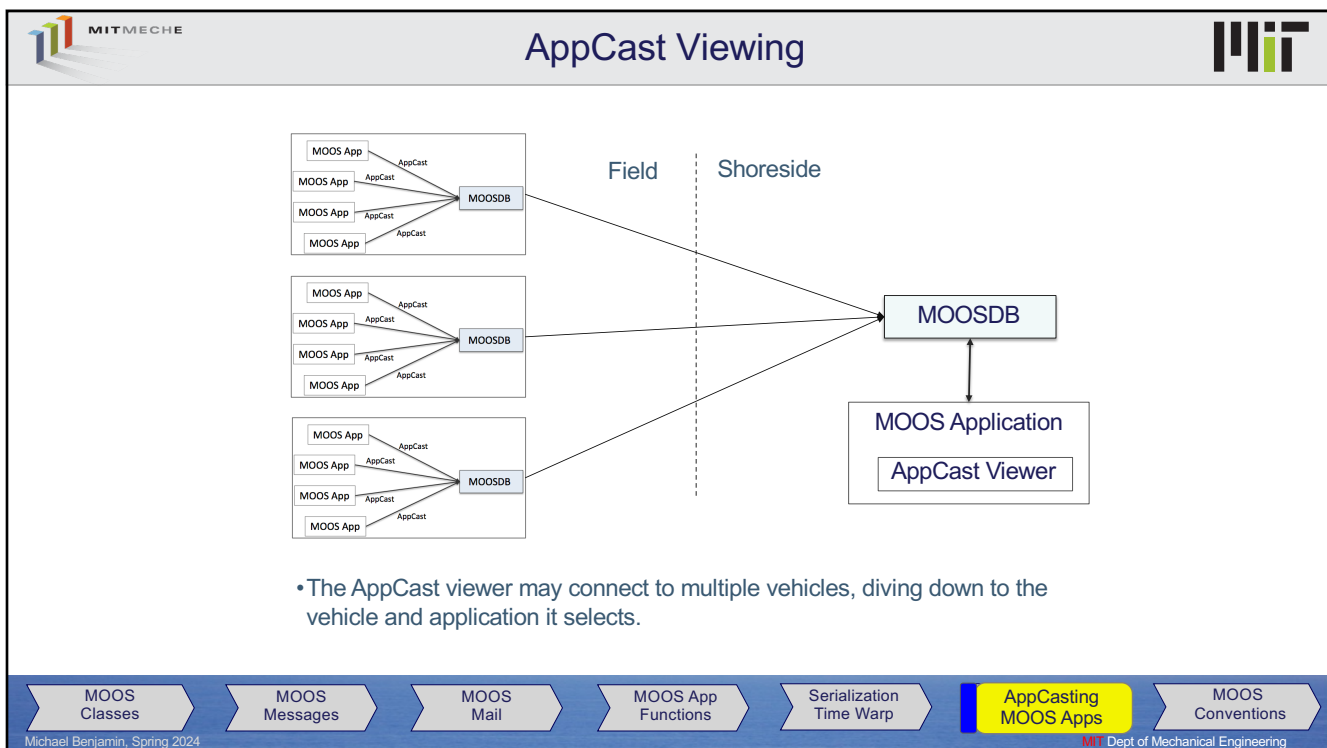
MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

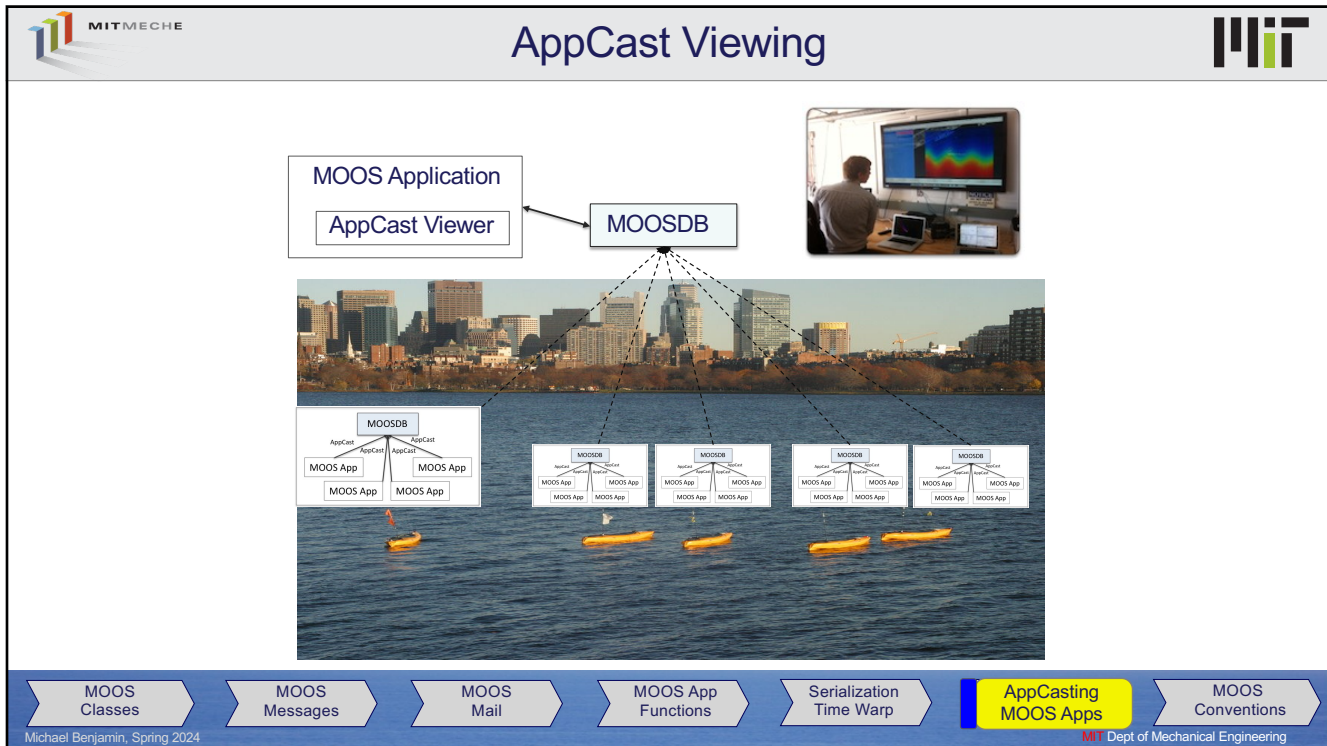
62



63



64



65

AppCast Viewers

What does an AppCast Viewer do?

- Sends AppCast requests to clients.
- Renders received AppCasts.
- Allows the user to select/switch between different MOOSApps and vehicles

Currently there are three AppCast Viewer applications:

(1) uMAC

Terminal
(good for ssh'ing into a remote vehicle)

(2) uMACView


GUI (fttk)

(3) pMarineViewer


GUI (fttk)

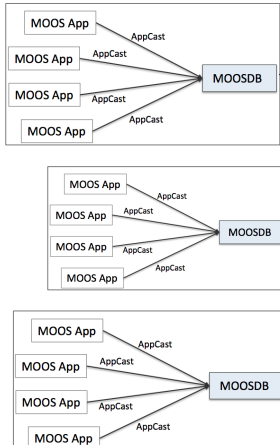
Michael Benjamin, Spring 2024

66




AppCast Viewing with the uMACView Tool






Field

Shoreside



Select the Vehicle/Node → Select the MOOSApp → View the AppCast



Node	AC	CW	HW	App	AC	CW	HW
shoreside	79	0	0	uflMessageHandler	4	0	0
archie	16	0	0	uflMarine	2	0	0
david	44	0	0	pBelmInv	2	0	0
archie	16	0	0	pNodeReporter	2	0	0
charlie	15	0	0	uProcessWatch	28	0	0
betty	16	0	0	pBasicContactMgr	2	0	0
grey	12	0	0	uflNodeBroker	2	0	0
				pHostInfo	2	0	0

- uMACView is a stand-alone, GUI-Based Viewer
- Launch from the command-line or with `pAntler`.

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

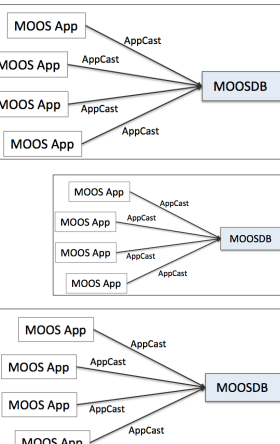
Michael Benjamin, Spring 2024
 Dept of Mechanical Engineering

67




AppCast Viewing with uMAC





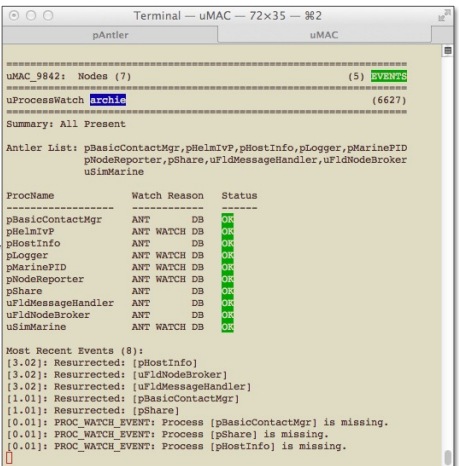
Field

Shoreside



Terminal interface provides most of what the GUI tools provide.

Primary advantage: When a remote vehicle is not sending AppCasts to a shoreside, user can ssh into the vehicle and launch uMAC to debug.



```

uMAC_9842: Nodes (7) (5)
uProcessWatch archie (6627)
Summary: All Present
Antler List: pBasicContactMgr, pBelmInv, pHostInfo, pLogger, pMarinePID, pNodeReporter, pShare, uflMessageHandler, uflNodeBroker, uflMarine

ProcName      Watch Reason Status
-----
pBasicContactMgr ANT DB OK
pBelmInv        ANT WATCH DB OK
pHostInfo       ANT DB OK
pLogger         ANT WATCH DB OK
pMarinePID      ANT WATCH DB OK
pNodeReporter   ANT WATCH DB OK
pShare          ANT DB OK
uflMessageHandler ANT DB OK
uflNodeBroker   ANT DB OK
uflMarine       ANT WATCH DB OK

Most Recent Events (8):
[3.02]: Resurrected: [pHostInfo]
[3.02]: Resurrected: [uflNodeBroker]
[3.02]: Resurrected: [uflMessageHandler]
[1.01]: Resurrected: [pBasicContactMgr]
[1.01]: Resurrected: [pShare]
[0.01]: PROC_WATCH_EVENT: Process [pBasicContactMgr] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pShare] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pHostInfo] is missing.
  
```

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
 Dept of Mechanical Engineering

68



The AppCast Structure



AppCast Config Warnings

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Configuration warnings:

 - Usually created at App startup time.
 - Unlimited in quantity.

Config Warnings

Run Warnings

General messages

Events

```

uProcessWatch gilda 1/1 (150)
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.
Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIVP, pHostInfo, pLogger, pMarinePID
pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
uSimMarine, uXMS

ProcName      Watch Reason  Status
-----
pBasicContactMgr  ANT      DB      OK
pHelmIVP          ANT WATCH DB      OK
pHostInfo         ANT      DB      OK
pLogger           ANT      DB      OK
pMarinePID        ANT WATCH DB      OK
pNodeReporter     ANT WATCH DB      MISSING
pShare            ANT      DB      OK
uFldMessageHandler ANT      DB      OK
uFldNodeBroker    ANT      DB      OK
uSimMarine        ANT WATCH DB      OK

Most Recent Events (8):
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

69



The AppCast Structure



AppCast Run Warnings

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Run warnings:

 - Created typically well after launch time when something goes wrong.
 - Limited in quantity.(don't want the size of an appcast to grow unbounded)
 - Provisions are made in AppCast Viewers to ensure RunWarnings come to the user's attention.

Config Warnings

Run Warnings

General messages

Events

```

uProcessWatch gilda 1/1 (150)
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.
Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIVP, pHostInfo, pLogger, pMarinePID
pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
uSimMarine, uXMS

ProcName      Watch Reason  Status
-----
pBasicContactMgr  ANT      DB      OK
pHelmIVP          ANT WATCH DB      OK
pHostInfo         ANT      DB      OK
pLogger           ANT      DB      OK
pMarinePID        ANT WATCH DB      OK
pNodeReporter     ANT WATCH DB      MISSING
pShare            ANT      DB      OK
uFldMessageHandler ANT      DB      OK
uFldNodeBroker    ANT      DB      OK
uSimMarine        ANT WATCH DB      OK

Most Recent Events (8):
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

70



The AppCast Structure



AppCast Messages

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Messages:

 - Free in format. Up to the user to pick the information and layout.
 - Typically “cleared” on each generation of an appcast.
 - Just a list of strings.
 - Tables, columns etc. done by the user.

Config Warnings

```
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra
```

Run Warnings

```
Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter
```

General messages

```
Antler List: pBasicContactMgr, pHelmIVP, pHostInfo, pLogger, pMarinePID
pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
uSimMarine, uXMS
```

ProcName	Watch Reason	Status
pBasicContactMgr	ANT	DB OK
pHelmIVP	ANT WATCH	DB OK
pHostInfo	ANT	DB OK
pLogger	ANT	DB OK
pMarinePID	ANT WATCH	DB OK
pNodeReporter	ANT WATCH	DB MISSING
pShare	ANT	DB OK
uFldMessageHandler	ANT	DB OK
uFldNodeBroker	ANT	DB OK
uSimMarine	ANT WATCH	DB OK

Events

```
Most Recent Events (8):
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024
 Dept of Mechanical Engineering

71



The AppCast Structure



AppCast Events

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Events:

 - Created typically after launch time when something “notable” happens.
 - Limited in quantity.(don’t want the size of an appcast to grow unbounded)
 - Each event is just a string with a timestamp.

Config Warnings

```
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra
```

Run Warnings

```
Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter
```

General messages

```
Antler List: pBasicContactMgr, pHelmIVP, pHostInfo, pLogger, pMarinePID
pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
uSimMarine, uXMS
```

ProcName	Watch Reason	Status
pBasicContactMgr	ANT	DB OK
pHelmIVP	ANT WATCH	DB OK
pHostInfo	ANT	DB OK
pLogger	ANT	DB OK
pMarinePID	ANT WATCH	DB OK
pNodeReporter	ANT WATCH	DB MISSING
pShare	ANT	DB OK
uFldMessageHandler	ANT	DB OK
uFldNodeBroker	ANT	DB OK
uSimMarine	ANT WATCH	DB OK

Events

```
Most Recent Events (8):
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]
```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

Michael Benjamin, Spring 2024
 Dept of Mechanical Engineering


72

36



The AppCast Structure

AppCast Events



An APPCAST message:

```
proc=pHelmIvP!@#iter=5!@#node=alpha!@#iter=5!@#messages= Comms Policy: open!@ Helm
Iteration: 0!@ IvP Functions: 0!@ Pieces (Formed): 0!@ Pieces (Cached):
0!@ Mode(s): !@ SolveTime: 0.00 (max=0.00)!@ Create
Time: 0.00 (max=0.00)!@ LoopTime: 0.00 (max=0.00)!@ Halted: false (0
warnings)!@ Active Goal: false!@Helm Decision: []!@Behaviors Spawnable: -----
(0)!@Behaviors Active: ----- (0)!@Behaviors Running: ----- (0)!@Behaviors
Idle: ----- (0)!@!@Behaviors Completed: ----- (0)!@Hold-On-Apps:
none!@!@!@Variable Behavior Time Iter Value!@----- ----- -----
!@!@#!@#events_total=0!@#!@#run_warning_total=0!@#
```

(Still more or less human readable)

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

MIT Dept of Mechanical Engineering


Michael Benjamin, Spring 2024

73



The AppCast Structure

AppCast Events



An APPCAST message:

```
proc=pHelmIvP!@#iter=5!@#node=alpha!@#iter=5!@#mess
ages= Comms Policy: open!@ Helm
Iteration: 0!@ IvP Functions: 0!@ Pieces
(Formed): 0!@ Pieces (Cached): 0!@ Mode(s):
!@ SolveTime: 0.00 (max=0.00)!@ Create
Time: 0.00
(max=0.00)!@ LoopTime: 0.00 (max=0.00)!@ Hal
ted: false (0 warnings)!@ Active
Goal: false!@Helm Decision: []!@Behaviors
Spawnable: ----- (0)!@Behaviors Active: -----
--- (0)!@Behaviors Running: -----
(0)!@Behaviors Idle: ----- (0)!@!@Behaviors
Completed: ----- (0)!@Hold-On-Apps:
none!@!@!@Variable Behavior Time Iter Value!@--
-----
!@!@#!@#events_total=0!@#!@#run_warning_total=0!@#
```

```
pHelmIvP iter=5
Comms Policy: open
Helm Iteration: 0
IvP Functions:
Pieces (Formed): 0
Pieces (Cached): 0
Mode(s):
SolveTime: 0.00 (max=0.00)
Create Time: 0.00 (max=0.00)
LoopTime: 0.00 (max=0.00)
Halted: false (0 warnings)
Active Goal: false
Helm Decision:
Behaviors Spawnable:
----- (0)
Behaviors Active:
----- (0)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (0)
@Behaviors Completed: ----- (0)
Hold-On-Apps: none
Variable Behavior Time Iter Value
----- -----
events_total=0!
run_warning_total=0
```

(Still more or less human readable)

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

74



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: AppCasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

75



How do you make an “AppCast-Enabled” MOOS application?



MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

76



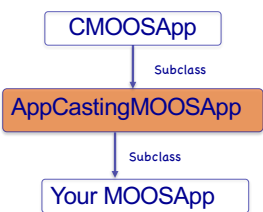
On-Demand AppCasting



To implement on-demand appcasting, a few things need to be done **in each application**.

http://oceanai.mit.edu/ivpman/appcast_enable

- Apps must register for APPCAST_REQ mail.
An AppCast request will renew a token for some number of seconds
Until the token expires, the app generates an appcast repeatedly.
- Even while appcasting, the app only generates an AppCast every N secs.
The app keeps track of the last real-time appcast generation.
- Each app handles a config setting indicating whether an xterm is open.
This setting is a global variable in the .moos config file.



```

graph TD
    CMOOSSApp[CMOSSApp] -- Subclass --> AppCastingMOOSSApp[AppCastingMOOSSApp]
    AppCastingMOOSSApp -- Subclass --> YourMOOSSApp[Your MOOSSApp]
  
```

So, a new generic “AppCastingMOOSSApp” class is used:

Minimizes boilerplate in individual apps.

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


 Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

77



Using the AppCastingMOOSSApp Superclass



Six Steps

Step 1: Subclass the AppCastingMOOSSApp Superclass

Step 2: Invoke two superclass methods in your `Iterate()` method.

Step 3: Invoke a superclass method when you register variables.

Step 4: Invoke a superclass method during `OnNewMail()`.

Step 5: Invoke a superclass method during `OnStartup()`.

}

Trivial, 1-2 line
changes in each
case

Step 6: Implement your `buildReport()` function.

}

This is where you
get to be creative
about what your
app reports.

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions

Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

 Dept of Mechanical Engineering

Michael Benjamin, Spring 2024


78

39



Using the AppCastingMOOSApp Superclass

Your Class Definition



Step 1: Subclass the AppCastingMOOSApp Superclass

```

0 #include
1 "MOOS/libMOOS/Thirdparty/AppCasting/AppCastingMOOSApp.h"
2 class YourMOOSApp : public AppCastingMOOSApp
3 {
4     // All your normal class declaration stuff
5
6     bool buildReport();
7 };

```

The `buildReport()` function is a virtual function in the superclass. It is where you can do the work of constructing an AppCast.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps

MOOS
Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

79



Using the AppCastingMOOSApp Superclass

Modifying Your Iterate() and Registrations



Step 2: Invoke two superclass methods in your `Iterate()`

```

0 bool YourMOOSApp::Iterate()
1 {
2     AppCastingMOOSApp::Iterate();
3
4     // Do all your normal Iterate stuff
5
6     AppCastingMOOSApp::PostReport();
7     return(true);
8 };

```

Updates the current MOOSTime, and # of iterations.

Determines if an AppCast is warranted, and invokes `buildReport()` if so.

Step 3: Invoke a superclass method when you register variables.

```

0 void YourMOOSApp::registerVariables()
1 {
2
3     AppCastingMOOSApp::RegisterVariables();
4
5     // Do all your other registrations
6 }

```

The superclass will register for `APPCAST_REQ`, indicating another app, like uMAC, is interested in appcasts from this app.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions


Serialization
Time Warp

AppCasting
MOOS Apps


MOOS
Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

80



Using the AppCastingMOOSApp Superclass



Modifying Your OnNewMail() and OnStartup()

Step 4: Invoke a superclass method when you handle mail.

```

0 bool YourMOOSApp::OnNewMail(MOOSMSG_LIST
1 &NewMail)
2 {
3     AppCastingMOOSApp::OnNewMail(NewMail);
4
5     // Do all your other normal mail handling.
6 }
```

The superclass will handle the `APPCAST_REQ` mail.

Step 5: Invoke a superclass method during `OnStartup()`

```

0 void YourMOOSApp::OnStartup()
1 {
2     AppCastingMOOSApp::OnStartup();
3
4     // Do all your other startup stuff
5 }
```

The superclass will register for `APPCAST_REQ`, indicating another app, like `uMAC`, is interested in appcasts from this app.

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

81



Intro to MOOS Programming Outline



Part 1: General MOOS App Concepts

Part 2: AppCasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: `OnNewMail()`, `OnStartup()`, `Iterate()`
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use AppCasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

82



Good Conventions for MOOS Programming

Command-Line Switches



- Most MOOS apps are launched from pAntler, but a number of common command-line switches a common, good practice.
- Add these to your own apps – Remember they are there for the apps you use.

-v, --version:

- Provides the version information for the given app
- Always provide this when/if submitting a bug report.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

MIT Dept of Mechanical Engineering


Michael Benjamin, Spring 2024

83



Good Conventions for MOOS Programming

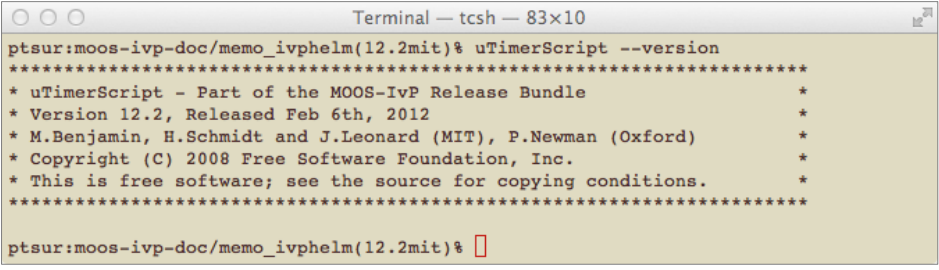
Command-Line Switches



- Most MOOS apps are launched from pAntler, but a number of common command-line switches a common, good practice.
- Add these to your own apps – Remember they are there for the apps you use.

-v, --version:

- Provides the version information for the given app
- Always provide this when/if submitting a bug report.



```

Terminal — tcsh — 83x10
ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)% uTimerScript --version
*****
* uTimerScript - Part of the MOOS-IvP Release Bundle                *
* Version 12.2, Released Feb 6th, 2012                             *
* M.Benjamin, H.Schmidt and J.Leonard (MIT), P.Newman (Oxford)      *
* Copyright (C) 2008 Free Software Foundation, Inc.                *
* This is free software; see the source for copying conditions.    *
*****
ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)%

```

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

84



Good Conventions for MOOS Programming



Command-Line Switches

- Most MOOS apps are launched from pAntler, but a number of common command-line switches a common, good practice.
- Add these to your own apps – Remember they are there for the apps you use.

-v, --version:

-h, --help:

- Provides the version information for the given app
- Always provide this when/if submitting a bug report.
- Short synopsis of what the app is intended to do.
- Other command line switches are available
- Other usage tips

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

85



Command Line Switches



```
$ uTimerScript --help
```

```

Terminal -- tcsh -- 75x39
ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)% uTimerScript -h

=====
Usage: uTimerScript file.moos [OPTIONS]
=====

SYNOPSIS:
-----
Allows the user to script a set of pre-configured pokes to a
MOOSDB with each entry in the script happening after a speci-
fied amount of time. Script may be paused or fast-forwarded.
Events may also be configured with random values and happen
randomly in a chosen window of time.

Options:
--alias=<ProcessName>
  Launch uTimerScript with the given process
  name rather than uTimerScript.
--example, -e
  Display example MOOS configuration block
--help, -h
  Display this help message.
--interface, -i
  Display MOOS publications and subscriptions.
--shuffle=Boolean (true/false)
  If true, script is recalculated on each reset. If event
  times configured with random range, the ordering may
  change after a reset. The default is true.
--verbose=Boolean (true/false)
  Display script progress & diagnostics if true.
  The default is true.
--version, -v
  Display the release version of uTimerScript.

Note: If argv[2] does not otherwise match a known option,
then it will be interpreted as a run alias. This is
to support pAntler launching conventions.

ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)%

```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

86



Good Conventions for MOOS Programming



Command-Line Switches

- Most MOOS apps are launched from pAntler, but a number of common command-line switches a common, good practice.
- Add these to your own apps – Remember they are there for the apps you use.

-v, --version:

-h, --help:

-i, --interface:

- Provides the version information for the given app
- Always provide this when/if submitting a bug report.

- Short synopsis of what the app is intended to do.
- Other command line switches are available
- Other usage tips

- Short synopsis of what the app is intended to do.
- The variables to which this app **subscribes**, and example values
- The variables to which this app **publishes**, and example values

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

87



Command Line Switches

-- interface



```
$ uTimerScript --interface
```

Terminal — tcsh — 70x32

```
ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)% uTimerScript --interface

=====
uTimerScript INTERFACE
=====

SYNOPSIS:
-----
Allows the user to script a set of pre-configured pokes to a
MOOSDB with each entry in the script happening after a speci-
fied amount of time. Script may be paused or fast-forwarded.
Events may also be configured with random values and happen
randomly in a chosen window of time.

SUBSCRIPTIONS:
-----
UTS_NEXT    = next
UTS_RESET   = reset
UTS_FORWARD = 10
UTS_PAUSE   = true

PUBLICATIONS:
-----
The primary publications are the events configured by the
user-defined scripts.

UTS_STATUS = name=RND_TEST, elapsed_time=2.00, posted=1,
              pending=4, paused=false, conditions_ok=true,
              time_warp=3, start_delay=0, shuffle=false,
              upon_awake=reset, resets=0/4

ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)%
```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps


MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

88



Good Conventions for MOOS Programming



Command-Line Switches

- Most MOOS apps are launched from pAntler, but a number of common command-line switches a common, good practice.
- Add these to your own apps – Remember they are there for the apps you use.

-v, --version:

-h, --help:

-i, --interface:

-e, --example:

- Provides the version information for the given app
- Always provide this when/if submitting a bug report.
- Short synopsis of what the app is intended to do.
- Other command line switches are available
- Other usage tips
- Short synopsis of what the app is intended to do.
- The variables to which this app subscribes, and example values
- The variables to which this app publishes, and example values
- An example MOOS configuration block.
- Description of MOOS parameter meanings
- Example values and default values

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp


AppCasting MOOS Apps

MOOS Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

89



Command Line Switches



-- example

\$ uTimerScript --example

```

=====
uTimerScript Example MOOS Configuration
=====
Blue lines:   Default configuration

ProcessConfig = uTimerScript
{
  AppTick      = 4
  CommaTick    = 4

  // Logic condition that must be met for script to be unpaused
  condition    = WIND_GUSTS = true
  // Seconds added to each event time, on each script pass
  delay_reset  = 0
  // Seconds added to each event time, on first pass only
  delay_start  = 0
  // Event(s) are the key components of the script
  event        = var=SBR_RANGE_REQUEST, val="name=archie", time=25:35
  // A MOOS variable for taking cues to forward time
  forward_var   = UTS_FORWARD // or other MOOS variable
  // If true script is paused upon launch
  paused        = false // or {true}
  // A MOOS variable for receiving pause state cues
  pause_var     = UTS_PAUSE // or other MOOS variable
  // Declaration of random var macro expanded in event values
  randvar       = varname=ANG, min=0, max=359, key=at_reset
  // Maximum number of resets allowed
  reset_max     = nolimit // or in range [0,inf)
  // A point when the script is reset
  reset_time    = none // or {all-posted} or range (0,inf)
  // A MOOS variable for receiving reset cues
  reset_var     = UTS_RESET // or other MOOS variable
  // If true script will complete if conditions suddenly fail
  script_atomic = false // or {true}
  // A hopefully unique name given to the script
  script_name   = unnamed
  // If true timestamps are recalculated on each script reset
  shuffle       = true
  // If true progress is generated to the console
  verbose       = true // or {false}
  // Reset or restart script upon conditions being met after failure
  upon_awake    = n/a // or {reset, restart}
  // A MOOS variable for posting the status summary
  status_var    = UTS_STATUS // or other MOOS variable
  // Rate at which time is accelerated in executing the script
  time_warp     = 1
}

ptsur:moos-ivp-doc/memo_ivphelm(12.2mit)%

```

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions

Serialization Time Warp



AppCasting MOOS Apps

MOOS Conventions

MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

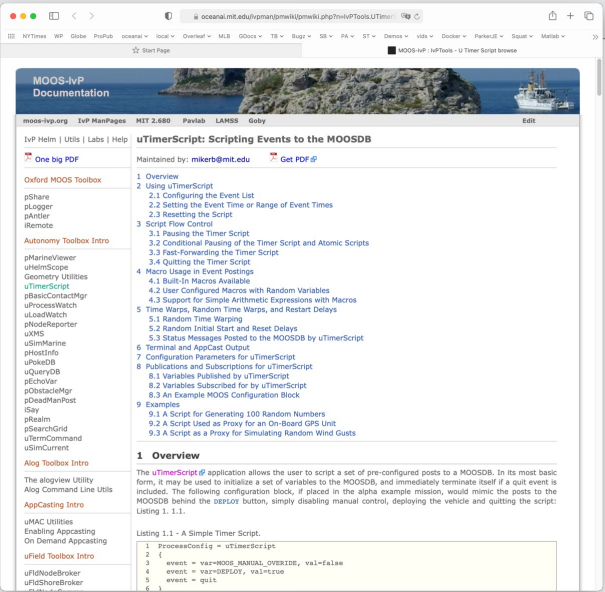
90

Command Line Switches

--web, or -w

```
$ uTimerScript --web
```



MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions



Serialization Time Warp

AppCasting MOOS Apps

MOOS Conventions

Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

91

Intro to MOOS Programming Outline

Part 1: General MOOS App Concepts

Part 2: Appcasting

Part 3 Good : MOOS App Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartUp(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use Appcasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS Classes

MOOS Messages

MOOS Mail

MOOS App Functions


Serialization Time Warp

AppCasting MOOS Apps

MOOS Conventions


Michael Benjamin, Spring 2024 MIT Dept of Mechanical Engineering

92



Good Conventions for MOOS Programming

Documentation (Why and How)



- Types of documentation for your MOOS App:
 - Document your code, inside your code with comments, or Doxygen tags
 - Document your application with command-line switches
 - Document your application with a manual-like PDF, with example missions.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

MIT Dept of Mechanical Engineering


Michael Benjamin, Spring 2024

93



Good Conventions for MOOS Programming

Documentation (Why and How)



- Types of documentation for your MOOS App:
 - Document your code, inside your code with comments, or Doxygen tags
 - Document your application with command-line switches
 - Document your application with a manual-like PDF, with example missions.
- Why do we document our code and application. Three Reasons:
 - So users know what it does and how to use it effectively.
 - So developers may continue development someday when you're not around.
 - What is the third reason? (Perhaps the most important)

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

MIT Dept of Mechanical Engineering


Michael Benjamin, Spring 2024

94



Good Conventions for MOOS Programming

Documentation (Why and How)



- Types of documentation for your MOOS App:
 - Document your code, inside your code with comments, or Doxygen tags
 - Document your application with command-line switches
 - Document your application with a manual-like PDF, with example missions.
- Why do we document our code and application. Three Reasons:
 - So users know what it does and how to use it effectively.
 - So developers may continue development someday when you're not around.
 - What is the third reason? (Perhaps the most important)
 - **Answer: So that YOU write better code!**

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions

 Dept of Mechanical Engineering


Michael Benjamin, Spring 2024

95



Good Conventions for MOOS Programming

Documentation (Why and How)



- Types of documentation for your MOOS App:
 - Document your code, inside your code with comments, or Doxygen tags
 - Document your application with command-line switches
 - Document your application with a manual-like PDF, with example missions.
- Why do we document our code and application. Three Reasons:
 - So users know what it does and how to use it effectively.
 - So developers may continue development someday when you're not around.
 - What is the third reason? (Perhaps the most important)
 - **Answer: So that YOU write better code!**
- Don't leave documentation until the end.
 - By documenting as you go along, you will write better code.
 - By documenting as you go along, you are less likely to "run out of time"

MOOS
Classes

MOOS
Messages


MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


 Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

96



Good Conventions for MOOS Programming



Documentation (Why and How)

- Types of documentation for your MOOS App:
 - Document your code, inside your code with comments, or Doxygen tags
 - Document your application with command-line switches
 - Document your application with a manual-like PDF, with example missions.
- Why do we document our code and application. Three Reasons:
 - So users know what it does and how to use it effectively.
 - So developers may continue development someday when you're not around.
 - What is the third reason? (Perhaps the most important)
 - **Answer: So that YOU write better code!**
- Don't leave documentation until the end.
 - By documenting as you go along, you will write better code.
 - By documenting as you go along, you are less likely to "run out of time"
- For MOOS Apps, document:
 - What it does
 - The publish-subscribe interface
 - An example app configuration
 - An example mission using the application

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

97



Intro to MOOS Programming Outline



Part 1: General
MOOS App
Concepts

Part 2:
Appcasting

Part 3 Good :
MOOS App
Conventions

- MOOS App Class Hierarchy
- MOOS Messages and Posting to the MOOSDB
- Registering for and Publishing Mail
- Key Overloadable Functions: OnNewMail(), OnStartup(), Iterate()
- Serializing and De-Serializing Messages
- Time Warp

- Motivation and How to use Appcasting
- How to convert an existing MOOSApp to an AppCastingMOOSApp

- Command-Line Switches
- Documentation
- Pros and Cons of Branching, How to Branch

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

98



Branching - Why and When?



- A powerful feature of MOOS and Open Source code: If you don't like what the application does, you have options.
- Branching here refers to (a) copying an existing module (b) modify for a different or augmented use, (c) making it available for others to use.
- Pros of Branching:
 - You are free to innovate! (From a decent baseline case)
 - Authors are motivated to maintain their code (or else it will be replaced)
 - Authors have limited liability – (hey, if you don't like it, build your own!)
- Cons of Branching:
 - Lots of choices between trivially different modules is confusing for other users.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp


AppCasting
MOOS Apps

MOOS
Conventions


MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

99



The Etiquette of Branching



- If you have a minor bug fix or feature request, give the maintainer a chance to maintain the code. Don't branch trivially.
- If you do branch, definitely, absolutely, give it a different name!
- Give credit where credit is due to the original author.

MOOS
Classes

MOOS
Messages

MOOS
Mail

MOOS App
Functions

Serialization
Time Warp



AppCasting
MOOS Apps

MOOS
Conventions

MIT Dept of Mechanical Engineering

Michael Benjamin, Spring 2024

100



END

MOOS
Classes

MOOS
Messages

MOOS
Mail


MOOS App
Functions

Serialization
Time Warp

AppCasting
MOOS Apps

MOOS
Conventions

Michael Benjamin, Spring 2024

 Dept of Mechanical Engineering