

A doctor, architect, and programmer were arguing over which was the oldest profession. The doctor said, "God took Adam's rib to make Eve. This was surgery! Thus, medicine is the oldest profession." The architect said, "No no! God took the Chaos and made the heavens and the Earth, which is sure the work of an architect, which is thus the oldest profession." The programmer looked up from his keyboard, smiled slyly, and said, "And who do you suppose created the Chaos?"

MACS Development

2,230 files, 415,000 LOC

Beware ye who enter here ...

AI Globus, San Jose State University Research Foundation
Rick Jacoby, Dell, Inc.

AIRSPACE OPERATIONS LAB

Basics

- Developer Environment
- Threading and Distributed Processing
- Development Philosophy
- The Big Picture



Developer Environment

- MACS is 100% pure Java (ADRS is C)
 - 2,230 files, 415,000 LOC
- UI in Swing
- Parts are object oriented, others procedural design
- GIT for source code control
 - Each lab has its own branch
 - When stable and tested, merge into integration branch
 - After regression test, merged into master branch
- Mantis for issue and bug tracking
- Log4j for logging, may switch to java util logging
 - Each class (should) have its own logger instance
- There is a unit test (JUnit), but it covers little
- Most AOL developers use Eclipse on Windows
- Libraries for build and run all functionality
 - RXTXcomm, jython, jsapi, freetts, en_us, cmutimelex, cmulex, cmu_us_kal, log4j-1.2.13.jar

Git and Wiki Server Info

- Git server: <https://aolgit.arc.nasa.gov/git/>

Protocol	Access-Type	Example
SSH	Read-Write	<code>git clone git@aolgit.arc.nasa.gov:MacSource</code>
GIT	Read-Only	<code>git://aolgit.arc.nasa.gov/MacSource.git</code>
HTTPS	Read-Only	<code>https://aolgit.arc.nasa.gov/git/</code>

- Wiki server: <https://aol1.arc.nasa.gov:8443>

- Macs-Documentation: End-User documentation

<https://aol1.arc.nasa.gov:8443/display/macs/Home>

- Macs-Development: Eclipse & SmartGit configuration screenshots

<https://aol1.arc.nasa.gov:8443/display/macsdev/Home>

Bug-Tracking, CI Server & Mailing list Info

- Mantis Bug-tracking software
 - <https://ofu.arc.nasa.gov/mantis/>
- Continuous Integration Server
 - <https://aolgit.arc.nasa.gov/hudson/>
- Mailing list info
 - To subscribe, go to:
 - <http://eos.arc.nasa.gov/mailman/listinfo/macshelp>
 - macshelp@eos.arc.nasa.gov
- Contact Vic (vaibhav.kelkar@nasa.gov) for accounts

Threading and Distributed Processing

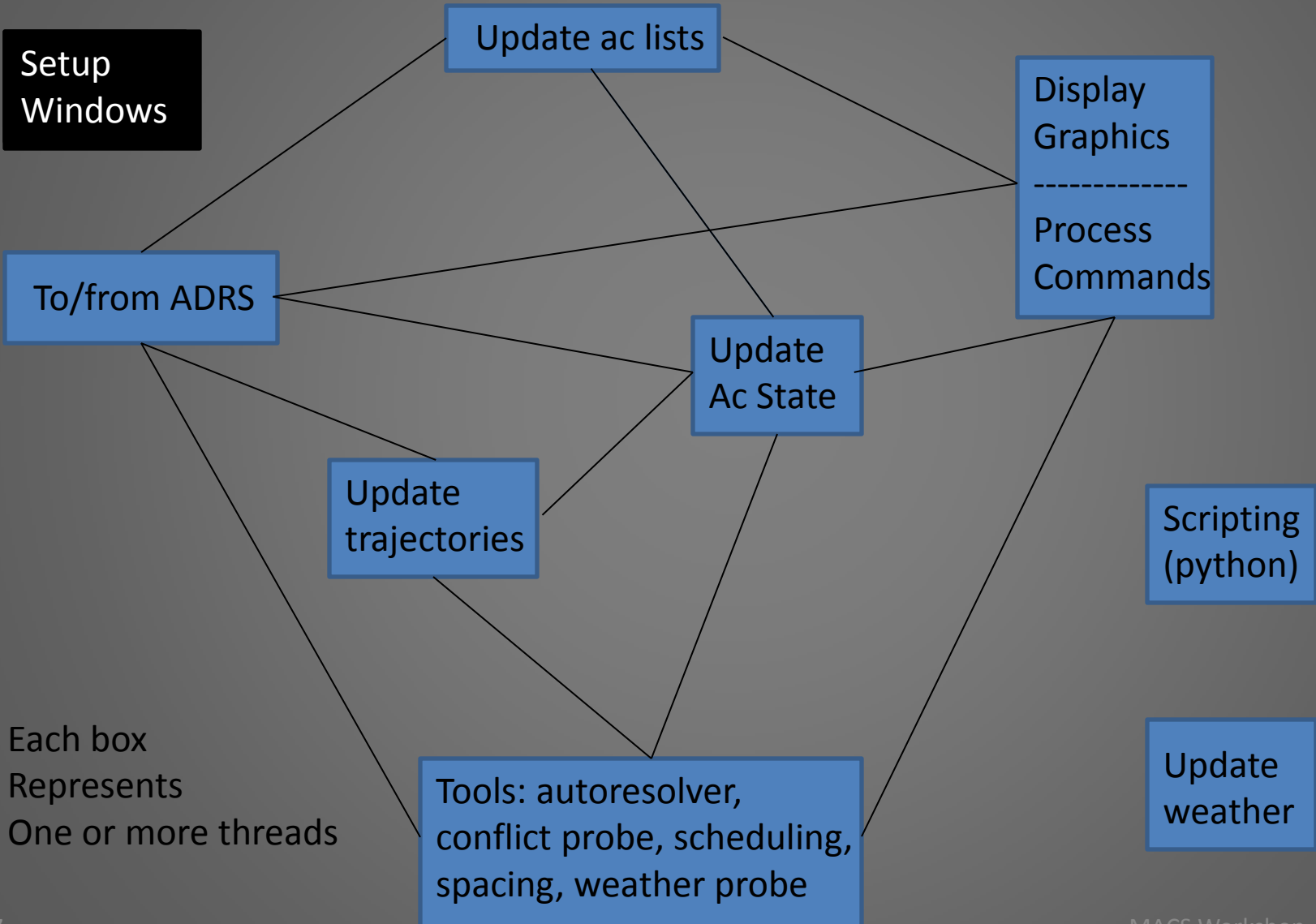
- MACS operationally usually runs on multiple machines
 - ATC stations, Pilot stations, Simulation control, Data collection
 - One jar, many setup options
 - ADRS for communication, usually on dedicated machines
- Each MACS runs many threads (up to 194)
 - Threads catch exceptions in `periodicRunAction()` and restart
 - NOTE:
 - `if (foo != null)`
 - `return foo.aField; // foo may be null!`
 - Synchronization is slow, can lead to deadly embrace
 - Assignment is atomic
 - Vector is synchronized, List is not
 - `for(int i = 0; i < aVector.size(); i++)`
 - `aVector.get(i) // NOT thread safe!`
 - `for(Foo f : Collection<Foo>)` will throw concurrent list modification exceptions
 - See `threads.AcManagerThread.AircraftList` for efficient thread-safe list handling
- MACS can run on a single machine, most development in this mode
 - Add `-adrs=offline -operator=Developer-Lite` to command line
 - Can partially test distributed mode on single machine by running ADRS and multiple MACS on one machine
 - Not recommended for operational use

The Big Picture

Scenario Editor

Simulation management

Setup Windows



Each box
Represents
One or more threads

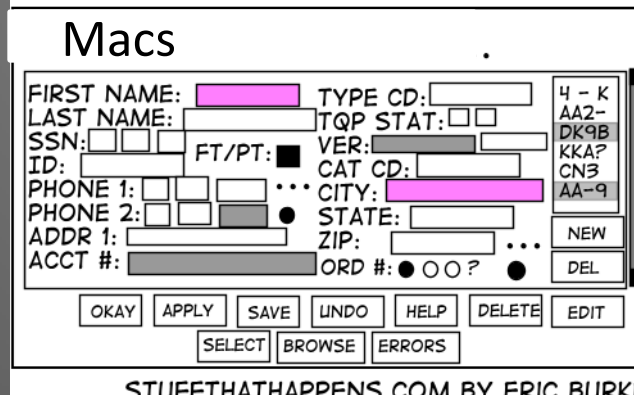
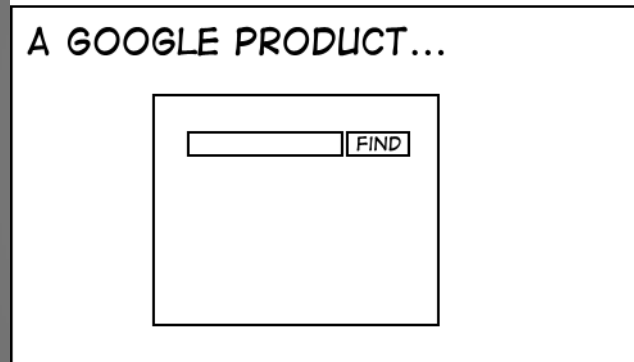
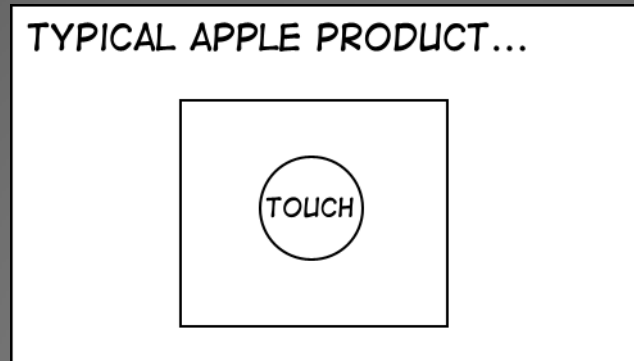
Philosophy

- Segregate air and ground side information
 - AcDescriptor.state for air side
 - AcDescriptor.atcData for ground side
 - Communicate through data link, not in the code!
- Check for null references, indices in bounds
- When encounter an error, log it and try to stumble on (if possible)
 - include stack trace: `logger.error("msg", new Exception())`
- When coding follow suit: in each part of MACS, try to be consistent with the existing code. We already have enough inconsistencies!
- Avoid redundant comments, do comment 'why?'
- Add, don't replace functionality

Add, Don't Replace Functionality

- Use the setup files/UI to turn on/off new capabilities
 - Save/restore to setup file
 - macsUtility.XmlUtility
 - displays.atc.atcDisplayData.Foo.getClassSpecificData()/setClassSpecificData()
 - Prompt user to save when closing setup window or exiting MACS
 - Call CommonSetupWindow.applySetupChange()
 - Calls applyChange(), which you write in subclass if needed
 - Be sure UI is updated when a new file is read (updateBody() is called)
- UI pattern
 - Add widgets to CommonWindow.jpanel
 - createBody()/actionPerformed()/updateBody() or updateSettings()
 - displays.common.DispUtils.addGBComponent
 - displays.common.GBPanel.java
 - uses macsUtility.Procedure.java
 - Regenerates panel on file read by calling createBody() from updateBody()
 - displays.common JMacs widgets
 - displays.common.JMacsFileMenu

ATC Windows & Views

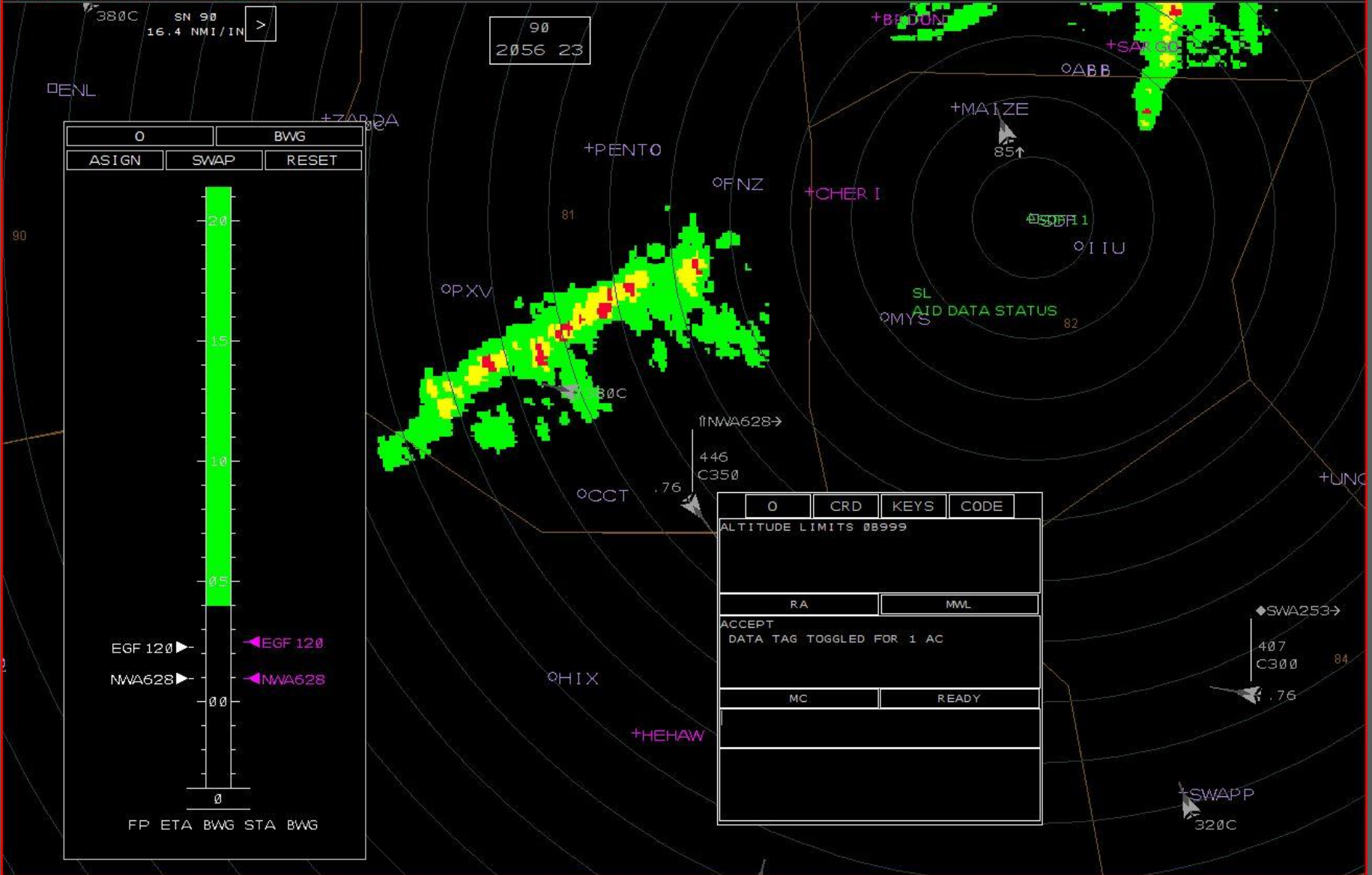


ATC Displays

- Different type of ATC display for each ATC function or domain
- Currently support DSR, STARS, TSD, OCEAN21, ERAM*

DSR

MACHS DSR_1 ATC VIEW -- Configuration: Center



PlanViewSetupWindow



SectorPlanViewSetupWindow

MACS DSR_1 SectorPlanViewSetup

File Address C:\Projects\MacsDatabase\settings\default.DSR_1_sectorPlanView

AC Filter Maps/Airways Optional Views Rng Rngs/Vectors Shortcuts

View	Enable	Display	In Banner	Externalizable
DsrDcView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrCrdView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrTimeView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DsrBannerView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DsrConflictView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrD2View	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DsrFrView	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DSR Timeline TEST1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DSR CmdInputView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrControllerInfoView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrConfigView	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrDIMenuView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrDIStatusView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DsrDIBannerView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RngReadView	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ZoomView	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
AcFilterView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
BoundaryView	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RangeView	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

AtcDisplayObjects

- Package: displays.atcDisplayData
- One for each display characteristic (~100)
- Each ATC display has an AtcConfigMan that references the AtcDisplayData objects
- Object super-class hierarchy
 - AtcDisplayObject
 - AtcVisualObject
 - OptionalViewAdo
 - DisplayChar
 - SpacingAid
- Range, Keyboard, Handoff -- subclass AtcDisplayObject
- Leader, WayPoint, RangeRing -- subclass AtcVisualObject
- TimeLineView, CrdView -- subclass OptionalViewAdo
- TimelineChar, DataBlockChar -- subclass DisplayChar
- Cones, Halos -- subclass SpacingAid

Saving & Restoring AtcDisplayObjects

- Hierarchical saving and restoring
- Saving builds a String
 - Vector getData()
 - getDataCommonData() [name]
 - getDataVisualData() [display it]
 - getDataClassSpecificData() [min/max/current values]
- Restoring uses a string
 - setData(Vector String)
 - setDataCommonData()
 - setDataVisualData()
 - setDataClassSpecificData()
 - Removes Strings as it proceeds

ATC Displays Class Hierarchy

- `CommonWindow` -- extends `JInternalFrame`
 - Screen info: sizing, scrolling, `jpanel`, tool bar. ...
- `CommonPlanViewWindow` -- extends `CommonWindow`
 - Airspace info: sector lists, mouse in NM, `PlanViewSetupWindow`, `SectorPlanViewSetupWindow`, `AtcConfigMan`
- `CommonAtcWindow` -- extends `CommonPlanViewWindow`
 - ATC info: list of views, aircraft, cursors
 - `OverPanel` (Glass pane)
 - `AtcMap` (`paintComponent`, `mouseListeners`)
- `DsrAtcWindow`, `StarsAtcWindow`, `TsdAtcWindow`, `Ocean21AtcWindow`, `EramAtcWindow` -- extends `CommonAtcWindow`
 - Class specific methods

Views & OverPanel

- **CommonAtcView**
 - A view is the internal panel on an ATC display
 - JPanel
 - CommonAtcWindow has list of views
 - All views added to OverPanel
 - (ERAM views not yet a CommonAtcView)
- **OverPanel**
 - JPanel
 - setGlassPane(overPanel)
 - No layout manager
 - Updates views' position, visibility, visuals

Command Processing

```
#include <stdio.h>
int main(void)
{
    int count;

    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
```

3

AMEND 10-3



Command Processing

- Command parts:
 - Keyword(s), parameter(s), aircraft(s)
 - QP J 5 UAL123
 - QP J 6 AAL456 DAL789
- Command status
 - AtcCmdIcd.java
 - enum CmdStatus
 - CMD_ACCEPT
 - CMD_REJECT
 - CMD_PARTIAL_ACCEPT_REJECT

Summary of Adding a New Command

- In `displays.atc.AtcCmd`, add enum
 - `MY_CMD("key words", "methodName", NumAcNeeded.xxx, minArgs, maxArgs, DcEvent)`
 - See `ML_MAN_ASSIGN` for an example
- In `displays.atc.AtcCommandProcessor`, add a method to process the command
 - `static public AtcCommandOutput methodName(AtcCommandInput ci)`
 - See `assignSta(AtcCommandInput ci)` for an example

Adding the Enum to AtcCmd

- 5 command formats (constructor formats)
- “Best” format
 - Command name
 - Command keyword(s)
 - AtcCommandProcessor method
 - Number of aircraft needed
 - Minimum/maximum number or arguments
 - Data collection event
 - Optional: Window type [DSR, STARS, ...]
- Example
 - DLSL ON
 - `DLSL("DLSL", "dlStatusList", NumAcNeeded.ZERO, 1, 1, DcEvent.ATC_SET_DL_STATUS_LIST, WindowType.STARS)`
 - QP C 6 AAL123 DAL456
 - `KB_CONE("QP C", "cone", NumAcNeeded.ZERO_OR_MORE, 0, 1, DcEvent.ATC_CONE),`
- Utility methods also in AtcCmd
 - `acceptsMultipleAc()`
 - `getNumAcNeeded()`
 - `getCmdStr()`

Command Processing Files

- Definitions
 - AtcCmd
 - AtcCmdIcd
- Processing
 - AtcCommandProcessor
 - DsrCommandProcessor
 - Used for all ATC windows
- Input/Output
 - Used by processing method
 - AtcCommandInput & AtcCommandOutput
 - AC list, parameter list, forced, CommonAtcWindow, status, original input
 - AtcCommandOutput
 - Accept & reject lists of AC

Processing a Command

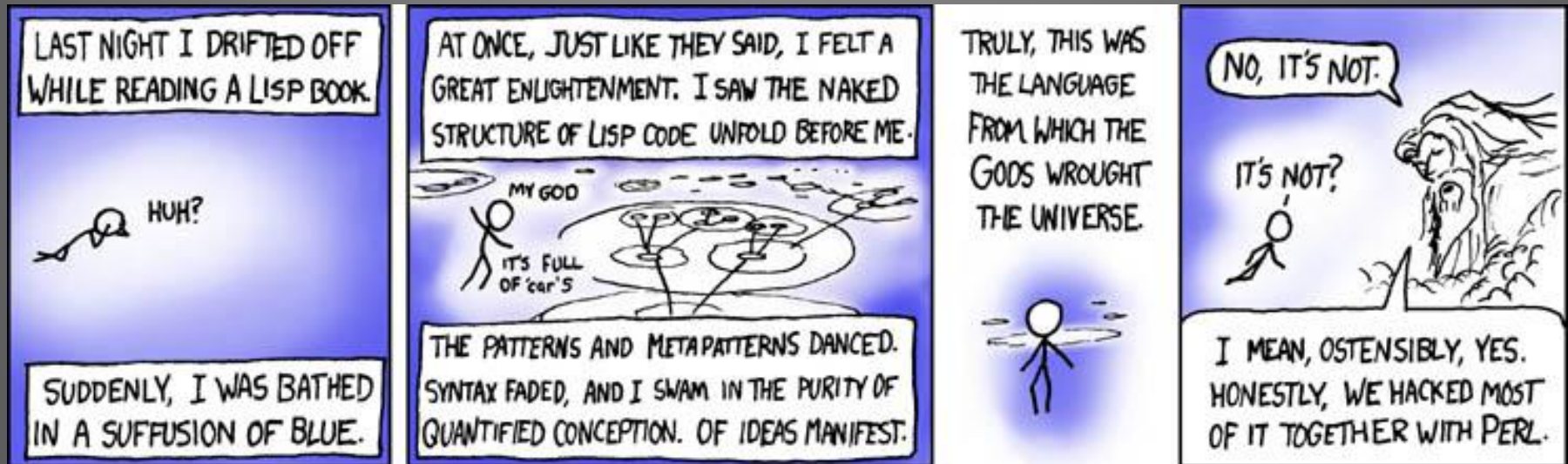
- DsrCommandProcessor
 - All commands for all ATC display windows pass through here
 - Historically processed all commands (parse and process methods)
 - Uses utilities
 - parseCmd(...)
 - ArrayList<String> getAcFromInput(...)
 - AtcCommandInput createCommandInput(...)
 - checkForLocalCmd – debugging, symbolic arguments
 - executeAtcCommand(...)
 - Java reflection by looking for method name in AtcCommandProcessor
 - AtcCmdProcessor.execute(methodName, CI)
 - Still does many, some converted to “sliver” methods
 - return executeAtcCommand(inputCmdStr, "route", atcCmd, tokens, 0, Macs.INT_NOT_SET);
- AtcCommandProcessor
 - Add “process” method for new commands
 - Process methods
 - Input: AtcCommandInput
 - Output: AtcCommandOutput
 - Comment all the input syntaxes for each execute method
 - // AS R <scheduler> <HHMM> <HHMM>
 - // QP J <ac>
 - // QP J <radius> <ac> radius => 1.0 .. 9.9, 10 .. 30

Adding a Data Tag Item

- `commonObjects.TagTextDescr`
 - add field `myField`
 - either run common objects tool or fix up auto-generated methods
 - set in `populateXXXX(arg)` methods
- `displays.datablock.Dblcd.java`
 - add to end of `DB_FIELD_NAME`
 - add to end of `TAG_EXAMPLE`
- `displays.dataBlock.DataTag`
 - add `TAG_MY_ITEM` to static ints
 - set `TAG_MY_ITEM` in `initiateTagItemId()`
 - modify `DataTag.setTagText(TagTextDescr)` to set `tagDisplayText[TAG_MY_ITEM]` from `TagTextDescr.myField`
- NOTE: there is a `TagTextDescr.debugText` that you may set freely

Run Time Inter-Machine Communication

- Data Logging
- ADRS
- macsComm



Data Logging

- Every time a controller or pilot does something, log it!
- Adding an event to existing log file
 - *dataCollection.DcEvent.java*
 - add *MY_EVENT(EventClass, DcTrigger, Text Description)* to enum
 - *EventClass* determines log file
 - Call *dataCollection.DcUtilities.logEvent(MY_EVENT, AcDescriptor, Object, String)* or *logMyClassEvent(...)*
- *To add new information to output*
 - *message.macs* in the appropriate class (e.g., *MacsCustomLogMsgEvent.java*)
 - Add a field to store info
 - In constructor, set the field
 - Append to *toString()*
 - To modify header, add to *DcUtilities.getClassHeaderStr()*
- Creating a new log file
 - *dataCollection.DcLog.java* – add *MY_DC_CLASS_LOG* to enum
 - Add class *message.macs.MacsMyDcClassMsgEvent.java*
 - Add method *message.Msg.macsMyDcClass(args)* -- call *fireMessageEvent(new MacsMyDcClassMsgEvent(args))*
 - Add *getClassHeaderStr()* to *DcUtilities*
 - *dataCollection.DcUtilities* – add call to *macsMyDcClass(args)*
 - *threads.DcThread.java*, add to *handleMessageEvent()*

Debug Logging

- Debugging log(4j)
 - Command line: add `-Dlog4j.configuration="log4j.properties"` before the jar file name
- Sample log4j File
 - `log4j.rootLogger=WARN, A1, FIL1`
 - `log4j.logger.traj4D.Traj4DEvaluator=DEBUG`

 - `log4j.appender.A1=org.apache.log4j.ConsoleAppender`
 - `log4j.appender.A1.layout=org.apache.log4j.PatternLayout`
 - `log4j.appender.A1.layout.ConversionPattern=%-5p\t%s\t%c.%M\t%m\t%d{HH:mm:ss,SSS}\t${macs.host}\t[%t]%n`

 - `log4j.appender.FIL1=org.apache.log4j.FileAppender`
 - `log4j.appender.FIL1.File=../MacsDebug/${macs.host}_log_${macs.fileTime}.txt`
 - `log4j.appender.append=FALSE`
 - `log4j.appender.FIL1.layout=org.apache.log4j.PatternLayout`
 - `log4j.appender.FIL1.layout.ConversionPattern=%-5p\t%s\t%c.%M\t%m\t%d{HH:mm:ss,SSS}\t${macs.host}\t[%t]%n`

ADRS

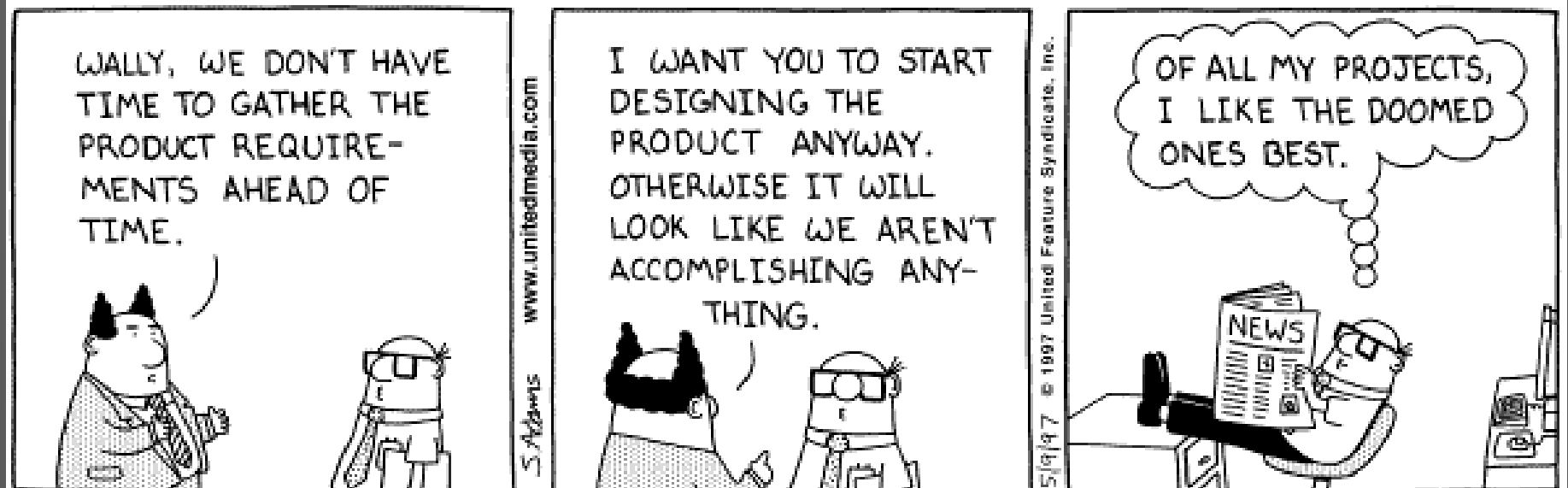
- Used for most inter-machine communication
- Communicates to non-Macs systems
- Written in C
- MACS: mpio.*.java
- What is communicated
 - Aircraft location, can include radar/ADS-B simulation
 - Datalink messages between aircraft and ATC
 - Share controller actions between ATC stations
- Multiple ADRSs common
 - Simulation control and pilots stations must share same ADRS
 - Pilot stations update ac positions for their sector
 - Simulation control updates all other ac

macsComm

- Pairs of controllers sometimes work together on a single sector.
- They are called “R side” and “D-side” meaning Radar and Data.
- macsComm keeps the display of data tags, spacing cones, trajectories synced
- When running two MACS on one machine, this package will give you lots of warning messages. You can ignore them or put
 - `log4j.logger.macsComm.RdCommunicator=ERROR` in your log4j properties file

Things to Know

- Key packages
- Useful stuff
- Traps



Key Packages

- **commonObjects**
 - aircraft, runway, waypoint, flight state, sector, trajectory, etc.
 - Partially generated by custom program: createCommonObjects
 - See commonObjects.Macs for input
 - See comments to know where to put custom code
- **calculators**
- **displays** – almost all the UI code here
- **displays.setup** – most of the setup UI
- **dst**
 - advisories, autoresolution, conflict probe, flight deck spacing, schedulers, atc spacing, traffic analyzer
- **traj4D** – trajectory code
 - Also in commonObjects.Traj4D and TrajPoint
 - Be VERY careful when modifying, tell me what you plan to do
 - "Rapid Generation and Utilization of Four-Dimensional Trajectories for Air Traffic Control and Management Applications in MACS," AI Globus, Richard H. Jacoby, Thomas Prevot, James K. Wong
- **threads** – most of the threads
 - CommonThread
 - DisplayManagerThread is for adding windows
- **NOTE:** junit and log4j are in the source tree

Useful Stuff

- `macsUtility`
 - Particularly `Utility.java`
- `commonObjects.Tools.java`
- `displays.common`
 - Particularly `displays.common.DispUtils.java`
- `test.*` -- JUnit unit testing
- NOTE: Macs runs Python scripts at sim time
 - `simulation.scripting`
 - Windows->Setup Panels->Scripting Setup

Traps

- macsUtility and gov.nasa.alsUtility have many of the same classes, use macsUtility
- What you think is incorrect behavior may well be the setup
 - I've spent days looking for bugs that turned out to be correct behavior given the setup options
- There's a fair amount of unused code
- There are incorrect comments
- There's an ample supply of bugs
- MACS generates a great deal of sometimes pointless output
 - Tsafe generates a lot of output to avoid JVM bug
- The scenario editor does things much differently than the other plan view windows

Testing



Bug Bash by Hans Bjordahl

Copyright 2005 Hans Bjordahl

<http://www.bugbash.net/>

Testing

- Test in the same airspace and setup as the next experiment
- Check the debug logs after shakedowns
- A (very) Minimal Regression Test
 - `test.AllTests.main(String[] arguments)`
 - StabilitySensorTest may give false positives
 - Fire up Macs with a bunch of aircraft
 - Wait a little while
 - ENTER on an ac symbol, data tag should appear
 - PICK on the portal, trial plan should appear
 - Click-Move-Click or Drag a point on the trial plan
 - Type 'TT', trial plan should disappear
 - Click on call sign, filed route should appear
 - Type ZZZZ and filed route should disappear
- A true regression test for the integration branch is in work

Good Luck!

- Al Globus – aglobus@arc.nasa.gov, x3819
- Rick Jacoby – richard.h.jacoby@nasa.gov, x0023
- James Wong – james.k.wong@nasa.gov, x6313

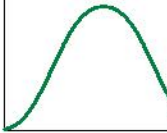


Classical learning curves for some common editors

Notepad

Pico

Visual Studio



MACS DSR_1 ATC VIEW -- Configuration: Center

204 SN 204
0141 30 34.1 NMI/IN

SWA751 →
310 450
02:18:17 LAX

SWA40 →
310 450
02:15:41 LAX

SL
AID DATA STATUS

O	CRD	KEYS	CODE
RNG BRG	DL	DS	NASA EXP
INT	PVD	ALT	METER LIST
			EMERG CHECK
			POS CHECK

RA MML

MC READY

O	DC
FILTER	FDB FIELDS
HOST	NO H/E SYNC
	HIST VECTOR
	DATA LINK
	BRIGHT VOLUME
	ALL

87 →
471 3
9:33 LAX

SWA806 →
165 370
01:48:00 LAX

HAL334 →
310 507
01:52:00 LAX

HAL346 →
210 431
01:46:59 LAX

AAL941 →
17 30
01:46:58 LAX

SWA40 →
320 405
02:14:43 LAX

ZLA1 →
162 371 296
DAL 3 3 5 9 4 3 1 7

VRD112 →
320 405
01:58:58 LAX

997 →
385
04:11 LAX

877 →
406 320 405 374
00:26 01:40 02:00 02:00

TEST1

40
35
30
25
20
15
10
05
00
-05
-10
-15
-20
-25
-30
-35
-40
-45
-50
-55
-60
-65
-70

FP ETA FP ETA



Adding a Command

- `displays.atc.AtcCmd.java` add to enum
 - Write `MY_CMD("UsersName", "methodName", NumAcNeeded.?, minArgs, maxArgs, DcEvent)`
 - See, for example, `ML_MAN_ASSIGN`
- `AtcCommandProcessor.java`
 - Write static `public AtcCommandOutput methodName(AtcCommandInput ci)`
 - See, for example, `assignPlannedStas`
- Macs uses reflection on the `methodName` to execute the command. Can be tricky to debug.
- NOTE: `displays.atc.dsr.DsrCommandProcessor.java`
 - misnamed and misplaced. Handles commands for all plan view windows.
 - Also includes implementations of many commands