

## **SKA LMC Workshop**

#### **Assembly Integration and Verification**

Craig Haskins | Software Engineer, ASKAP 25<sup>th</sup> March 2015

CSIRO ASTRONOMY AND SPACE SCIENCE www.csiro.au





#### **Speaker Introduction**

- Developed EPICS IOCs for ASKAP
- Developed testing framework for IOCs in Python
- Developed Engineering GUIs (Control Systems Studio)
- Involved with system commissioning
- Software Improvements for ASKAP Design Enhancements (ADE)
- Software Representative for AIV





#### Boundaries of Responsibility during Construction Phase





SKA LMC Workshop : AIV | Craig Haskins

### **Assembly, Integration & Verification**

- AIV is responsible for verification of L1 requirements at site
- AIV supports the creation of an Integration Test Facility (ITF) where LMCs and hardware can be integrated, tested and debugged prior to installation on site.
- AIV does have an interest in suggesting the use of common software, frameworks methodologies and processes for testing and verification, even if the LMC specific test plans are not written and executed by AIV
- Common software reduces downstream integration risk



#### **Development & Integration Cycle**





SKA LMC Workshop : AIV | Craig Haskins

### **Development Stage Guidelines**

- Use a common LMC framework
- Use common software libraries (3rdParty, SKA libraries)
- Test driven development
- Hardware emulation (external emulators, HAL emulation)
- Integrate early and often
- Design in low level interface to LMCs
- Engineering level GUIs
  - Feature enable / disable
  - Debug features
  - Logging configuration



#### **Example : ASKAP Development**

- EPICS Framework + select support libraries
- CPP Unit, JUnit, Nose Test (Python)
- Hardware Abstraction Layer (HAL)
- Target & emulated HAL
- Device emulators
- Common software libraries, common EPICS databases.
- Log4epixx (log4cxx with EPICS Wrapper, GUI)



### **Continuous Integration**

- Continuous Integration can assist downstream system integration by finding issues early on & reduces integration risk
- Provided test coverage is adequate
- Requires good unit & functional testing
- Requires good Regression testing
- When a bug is found, write a test against it
- Downside of Continuous Integration: Continuous Noise
- Multi-Stage Continuous Integration might be applicable



#### **Example: ASKAP Continuous Integration**

- Continuous Integration using Jenkins
- Build & testing triggered by revision control commits
- EPICS IOCs tested against emulated hardware

🎅 Jenkins						🔍 search		👷 Jenkins		Q	earch		0	log in i sign up
enkins )								Jenkins i System Verification i verify-ade i #54	i Test Results i osl				(1943)	LE AUTO REPORTAL
2 Paonie		ASKAP	Softwa	re Build and Test Server				Back to Project	Test Result : osl					
- · · · · · · · · ·		Act	tive	All Full_Builds Science Data Processor	System Verification	Telescope Operating System		Q Status	O faiburar (-1)					
Build History		s	w	Name 1	Last Success	Last Failure	Last	Changes	0184048 (0)				134 1	tests (+131)
					22200		2200	Console Output					Took 6	I min 0 sec.
Build Queue (3)		•	9.33	Analysis	16 hr - #2090	2 days 16 hr - #2088	25 m		AU 7					
TOS documentation		0		adiagonate build entry	40 hr 8407	01 days #119	110	View Build Information	All lests					
askapsoft build mavericks 64 bit		•		askapsop pullo cray	12 11- 8431	24 days - #413	1.02	K History	Class	Duration Fail	(dill) Skip	(st) Pass	(oii) Total	d (alto
askapsoft build wheezy 64 bit		0	-	askapada huild maundaka 64 hit	E dain 4 hr . #210	1 mo 6 days - #204	1 br	Text Dente	osl a abf init 01	9.8 sec	0	0	7 +7	7 +7
		•		askapsop_bono_mavenoks_e4_bit	5 days + m - +310	1 mb 6 days	1.00	Test Hesuit	osl a abf set port map 02	77 ms	0	0	2 +2	2 +2
Build Executor Status	-			and an and a local strategy of a local	A		1.64	👍 Previous Build	osl a abf set weights 03	4.1 sec	0	0	2 +2	2 +2
		-		askapsop build wheezy 64 bit	5 days 5 hr - #319	2 mo 1 day - #2/U	1.00		osl a acx init 01	6.4 sec	0	0	6 +6	6 +6
I master		-	-	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		1000	14.1	Next Build	osl a ade init 01	15 sec	0	0	4 +4	4 +4
an moster		-		askapsdp_docs_nightly	11 hr - #38	24 days - #14	49 m		ost a ade init 04	17 sec	0	0	4 +4	4 +4
1 TOS documentation	#4								osl a ade init 07	18 sec	0	0	4 +4	4 +4
		•		askapeoft build mavericks 64 bit	2 hr 48 min - #835	4 days 17 hr - #821	1 hr		osl a ade init 08	20 sec	0	0	4 +4	4 +4
# aktos01									cel a ade init 10	20 sec	0	0	4 +4	4 +4
1 Idle				askapsoft build wheezy 64 bit	2 hr 44 min - #1455	6 days 3 hr - #1430	1 hr		osl a ade init 13	15 sec	0	0	4 +4	4 +4
2 Idle									osi a ade shutdown 07	14 sec	0	0	3 +3	3 +3
				askapsoft documentation	10 hr - #1847	15 days - #1829	1 hr		osi a ade simulate overtemp 02	12 sec	0	0	4 +4	4 +4
hrage (offline)									osi a ade simulate overtemp 03	57 ms	0	0	2 +2	2 +2
Wahaa				CSS-3.2.x	1 mo 17 days - #115	N/A	8 mil		osi a ade simulate overtemp 05	18 sec	0	0	4 +4	4 +4
m chan		-							osi a ade simulate overtemp 05	56 ms	0	0	2 +2	2 +2
1 askapsoft build mavericks 64 bit	#836			DataChallenge1A	4 days 15 hr - #220	23 days - #214	1 hr		osl a ade simulate overtemp 08	14 sec	0	0	4 +4	4 +4
and the second se		-							osl a ade simulate overtemp 09	73 ms	0	0	2 +2	2 +2
# gijane		•	- 244	open-monica	1 mo 10 days - #548	N/A	32 m		osi a ade_simulate_overtemp_11	14 sec	0	0	4 +4	4 +4
1 Idle		-	- T		CONTRACTOR CONTRACTOR				osi a ade simulate overtemp 12	58 ms	0	0	2 +2	2 +2
		•	100	Belease Documentation	11 br - #480	24 days - #485	2 hr		ost a ade wait for loca 01	35 sec	0 -1	0	3 +3	3 +2
# gijoe		-	-	Charles cover in the set	TT THE BASE	Li daju <u>Live</u>			osi a adx get coarse spectra 04	56 sec	0	0	3 +3	3 +3
1 askapsoft build wheezy 64 bit	#1456		1	Cuelhasia	15 br - #2005	24 days - #2061	A he		osl a adx get histograms 05	43 sec	0	0	3 +3	3 +3
		-	Solutions and a solution of the solution of th	Syllingers	TO TH - ERSON	24 Gays - FROM	- 10		cel a edx init 01	9.1 sec	0	0	7 +7	7 +7
m mtos1		-		100 1 3	14 14 181	24 4500 421	2 he		osl_a_adx_set_dds_noise_03	0.59 sec	0	0	2 +2	2 +2
1 Idle		-	1.00	103-13	1418 . 822	24 days - 201	- III.		osi a adx set port map 02	0.14 sec	0	0	2 +2	2 +2
2 Idla		-	ale.	700 4 8 4-1-1-1	0 days 00 by 100		220		ost a pat init 01	12 sec	0	0	8 +8	8 +8
		-	100	103-1.3-0ebian	8 days 23 nr - 220	1 mo 17 days - #22	3.1.0		osi_a_trd_init_01	0.85 sec	0	0	5 +5	5 +5
# mtos2									osl s abf verify 05	0 ms	0	0	3 +3	3 +3
1 idle				TOS_documentation	N/A	N/A	N/A		ost s acx verify 05	0 ms	0	0	1 +1	1 +1
2 Idle		loop: 1	CMI						Col. 5. DOD. YOUNY	0 ms	0 -2	0	9 49	9 +7
		Journ. 5	M 222 -		Legend 5	RSS for all RSS for failures	RSS		ost s ade verity overtemp shutdown 09	0 ms	0	0	13 +13	13 +13
Canton W					angeor (		Carlo		out a set weity of	0 ms	0	0	0 +0	0 +5
									Mar a bai Aprilik Ma	0 ms			1 11	1 +1



### **System Integration**

- Bringing Element LMCs together
- Integration tests derived from commissioning tests
- Software only integration test using emulators
- Hardware Integration at Integration Test Facility



### **Example : ASKAP System Integration**

- Hardware Abstraction Layer (HAL)
  - Target HAL and Emulated HAL For Most EPICS IOCs
  - Switchable at Runtime
- Protocol emulators for simple devices (e.g. weather station)
- OSL (Operator Script Library)
  - Python scripting for EPICS IOCs
  - Scripting Framework for creating Automated Tests
  - Generates web based reports & junit xml for CI tool
- OSL Used for
  - Automated system integration tests with Emulated HAL
  - Manual System commissioning test with Target HAL + Hardware



## **AIV Software Deployment**

- Software packaging
- Dependency management
- Software configuration management
- Flexible deployment
- Automated testing of deployment



#### **Example: ASKAP Software Deployment**

- Using APT/Debian package management
- Single package per LMC
- Runtime configuration on installation via machine name
- Deploy to virtual machines
- Deploy to a test facility
- Deploy to site

#### **Need to Define a Software Release Process**

- What are the deliverables?
- What are the processes?
  - How to manage hand-over of SW to AIV
  - Version control / Numbering
  - Quality gates
  - Issue resolution
  - Regression testing
- Roles and Responsibilities



#### **High Level Software Release Process**

TM and other consortia testing their elements together at an Integration Test Facility (against L2 and L3 reqs). Informal support is provided by AIV

**Issues**, Patches?

By this testing, TM and other consortia confirms integration readiness of their elements.

Formal handover to AIV via a quality gate

AIV accepts the elements / modules and integrates them into the telescopes for formal System Level Verification against L1 requirements



# Thank you

**CSIRO Astronomy and Space Science** Craig Haskins

Software Engineer

- t +61 2 9372 4308
- e Craig.Haskins@csiro.au
- w http://www.atnf.csiro.au

www.csiro.au