ALMA Software Process Experiences

Alan Bridger, UKATC/STFC
OBSMGT lead;
Lead for ALMA Observation Preparation 2002-

Outline

- Background Context
- Tools
- Day 1 process, its problems and what we did about it
- Main process used in construction
- Current process evolution
- Some lessons

Background Context: 1

- ALMA Construction project began June 2002 –
  - note construction for ALMA encompassed end of PDR design
- Europe/North America 50/50 (ESO/NRAO)
- 8 “Integrated Product Teams” (IPTs): Science, Systems Engineering, Antennas, Front End, Back End, Correlator, Site, … and Computing
- Construction *originally* 2002-2011, “first science” 2007, “full science” 2011.
- Re-baselining 2005
  - Number antennas and receiver bands reduced
  - Construction extended to 2013
  - First (early) science proposal call before 1st April 2011
  - Revised schedule was kept...
- East Asia (initially Japan) joined 2004/5, adding back reciever bands and the Compact Array, and the ACA IPT

Background Context: 2

- Computing comprised:
  - Software Engineering
  - ALMA Common Software
  - Archive
  - Control
  - Correlator (Software interface to)
  - Proposal and Observation Preparation
  - Telescope Calibration
  - Pipeline
  - Integration and Test
  - Observatory Operations (added in 2005)
  - ACA Correlator (2005)
- Has evolved since, of key interest we now have
  - Integration and Test became Integration and Release Management
  - and expand to Software Engineering *and* Quality Management

Background Context: 3

- Development spread over 2 continents (then 3 then 4) and 9 sites (then ~11-12…I think). Approx 50. FTE and approx. 70-80 people.
- Many subsystems wholly within one executive
- But several shared (ITS, ACS, Pipeline) and Japan added FTE to several
- So highly distributed, heterogeneous groups, different development cultures
- Communication & internal interfaces (ICDs) were big issues

Tools: 1

Communication:

- Telecons and finally videocons
- Face to face meetings: leads twice per year, other ad hoc, plus couple of “all-hands” (dropped as too costly)
- Yahoo Messenger (finally moving to Hipchat)
- Twiki
- Skype (later)

Documentation & Requirements Management:

- Twiki (for us – documents attach in structured approach)
- Sitescape Forum (used as little as possible by us)
- DOORS (for a while – we were shielded)

Tools: 2

Development:

- Java, C++, python, UML, XML, (Javascript)
- Eclipse, emacs
- CVS -> SVN (-> git, soon)
- Modified ESO VLT Build system, combined with ant, later maven
- ESO nightly build system (later integrated with/replaced by Jenkins)
- Junit, cppunit, pyunit, integrated with ESO “TAT” system
- Selenium, QFTest
- doxygen
- Remedy *for a very short while,* rapidly replaced by JIRA + JIRA Agile

Day 1 Process

- One major, one minor release per year.
- Yearly planning of features to deliver
  - Adjusted at six-months
- Subsystems deliver features to ITS who integrate, test & release

- In summary: failed…delivered subsystem code could take months to integrate and significant effort on fixes - key issues:
  - Differing ICD interpretations
  - Failed delivery of “part-features”
  - Too many features

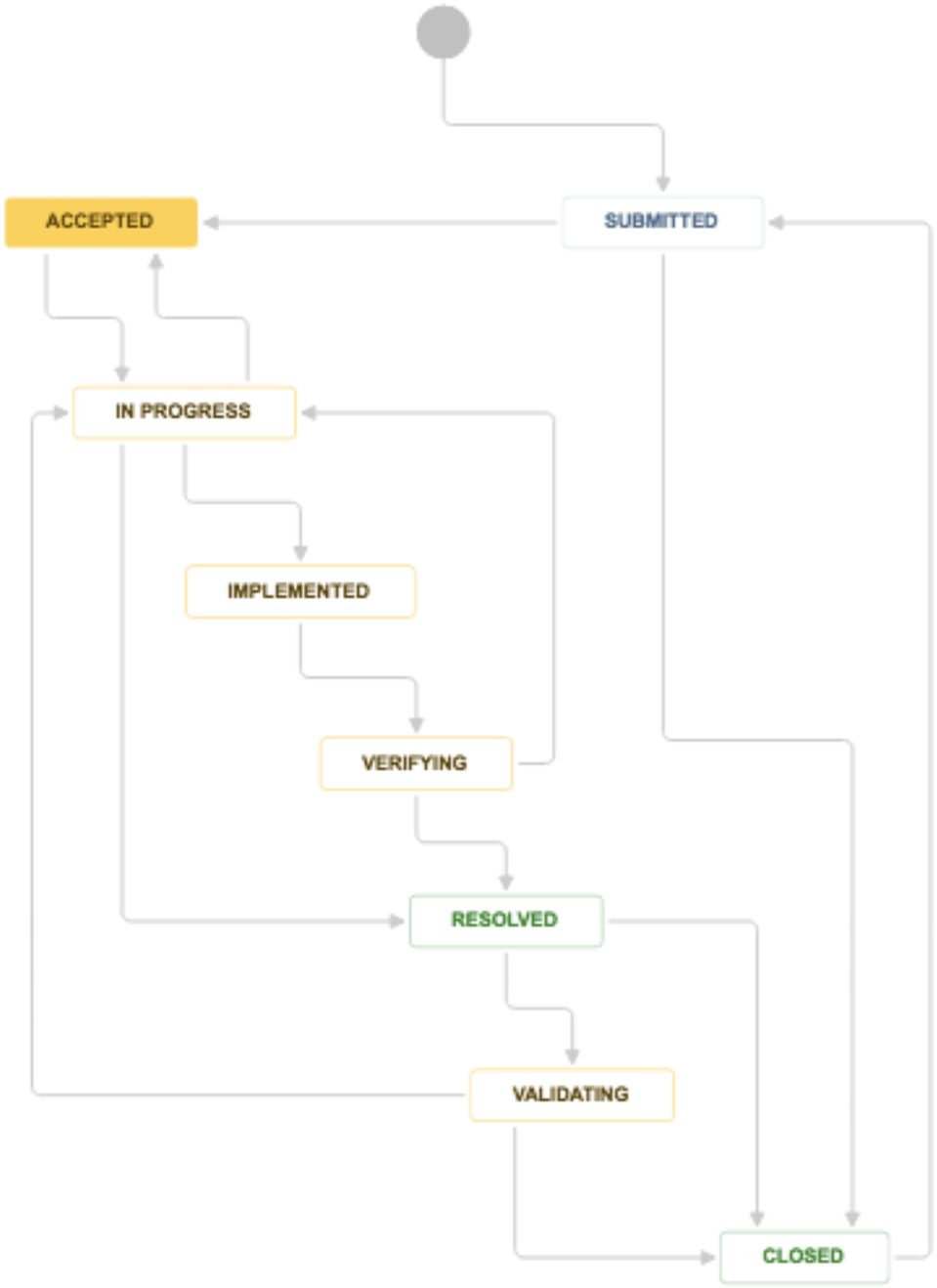
Function Based Teams & Other improvements

- Introduction (by UKATC!) of FBTs to help with first two issues:
  - Cross subsystem teams formed to tackle single functions
  - Work on goal acheivable in 4-8 weeks
  - Frequent meetings (usually 1-2 face to face)
  - Defined deliverable & report
- Made significant improvement to the outcome of releases
- cf. agile sprints

- But 6 monthly releases still too much: too many features
  - Evolved towards releases every 4-6 weeks.
  - With clearer emphasis on developer tests

Process for key construction stages/early operations

- Annual planning with science
  - features required for 6-12 months + longer term
- “Dot” releases every 4-6 weeks
  - Used by commissioning
- 3 Phase testing: Developer (unit tests + feature test, test team (verification), science (validation)
  - Integrated tests – automated regression
  - Bug fixes during verification and validation
- Selected releases -> accepted releases for operations deployment
  - Further testing, user, end-to-end, formal acceptance, SCCB-managed
- Heavy use of/reliance on JIRA(-agile)
  - Feature development/bug fixes
  - Science discussions for feature specification development
  - Formal patch requests
  - Formal change requests


ACCEPTED
SUBMITTED
IN PROGRESS
IMPLEMENTED
VERIFYING
RESOLVED
VALIDATING
CLOSED

Current Situation

- Moving towards monthly releases, staggered by subsystems
- Retain 3 Phase testing
- Reduce number of accepted releases
- Moving towards a continuous integration model and a git-like approach to merging features (prior to switching to git)
- Features not passing verification in 1-week period dropped

Some concerns about this:

- Dropping features will sometimes simply not be possible
- Staggered subsystems -> concerns about dependencies

Some Lessons: 1

- Six-monthly “big-bang” integrations bad, certainly in context of widely distributed teams
  - Concerns about my perception of SKA release plans
- Control subsystem had no requirement to provide a simulator, so it didn’t
  - This was a really major omission
  - One was provided late built “in spare time”

Some Lessons: 2

- JIRA good! I can't live (my ALMA life) without it.
  - Do integrate it with the repository
- CI is not only Continuous Integration, it is also Continuous Improvement:
  - Don't change for change sake, but always evaluate your processes, what works in construction may not suit operations.
- Communication is vital: messaging, conference calls, face to face, wikis, plus of course formal.
  - Do whatever it takes to talk to each other!
  - Early use of yahoo was not a "management" choice – it was developers/ leads
  - Always remember: communication is actually hard