Software development with distributed teams in large astronomy projects: The LSST experience (so far)

Frossie Economou frossie@lsst.org
Science Quality and Reliability Engineering team
LSST Data Management

Fact

Opinion
(Fiction?)

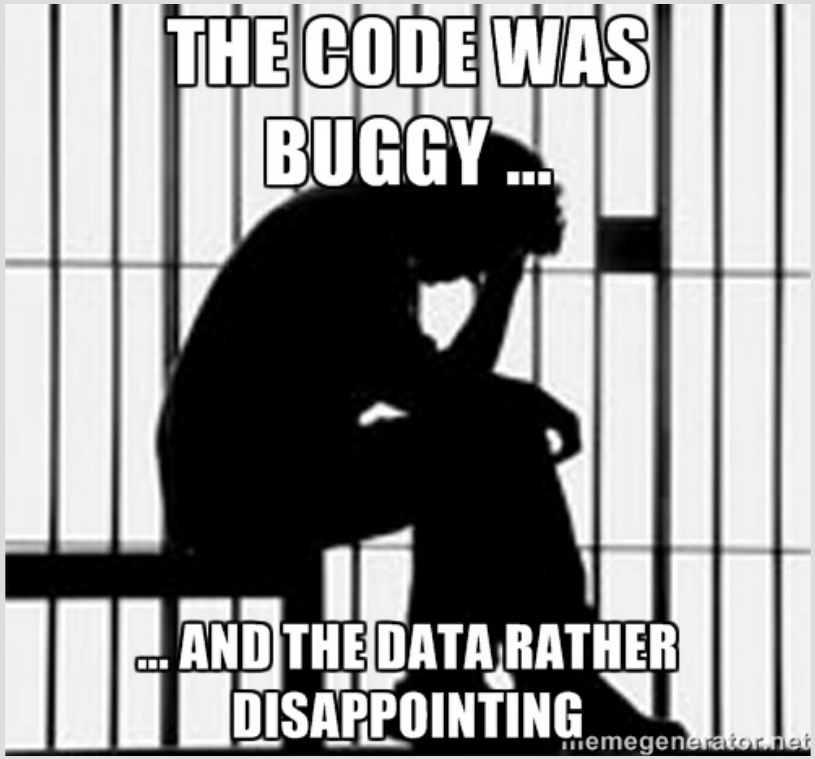
Data Management

- 9 main WBS elements
  - Alert Production
  - Data Release Production
  - Science User Interface
  - Data Access
  - Databases
  - Process Middleware
  - Infrastructure
  - SQuaRE
  - Long Haul Networks

- 6 institutions
  - U of Washington
  - Princeton
  - IPAC
  - SLAC
    "
  - NCSA
    "
  - AURA/LSST
    "

What is SQuaRE's organisational purpose?

- Provide scientific and technical feedback to the LSST DM Manager that demonstrates LSST/AURA DM is fulfilling its commitments as expected by the NSF with regards to: science quality and software performance and reliability



… but not like this

An ounce of prevention…

Science QA

Developer Support

Stability & Performance

Software Distribution

We only have one problem

- How do you build a single, coherent, performant, **maintainable** system…

- … with 80+ people at 6+ institutions?

- (and that's just Data Management)

We only have one problem

- How do you build a single, coherent, performant, **maintainable** system…
  I. software development tooling and services
  II. communication and documentation services
  III. science quality analysis services
- … with 80+ people at 6+ institutions?
  IV. agile and devops practices
- (and that's just Data Management)
  V. fostering a positive open-source team culture

I. Software Development & Tooling Services

Tech

- Repository Management (Github)
- Continuous Integration (Jenkins, Travis CI)
- GitLFS
- (Build)
- (Test)
- Deployment (Vagrant, AWS, OpenStack, Docker)
- Portability testing
- Release Engineering

developer.lsst.io

Process

- Tickets for almost all work
- Branches for almost all work
- Code review for ALL work
- (… but do it right)
- developer.lsst.io

Good tools == Happy Devs

- Community & Best-of-breed tooling
- SaaS with easy migration paths
- Stick to Open Source conventions
- No special snowflakes
- Tool integration via APIs, not walled gardens

II. Communication & Documentation Services

Communications

- Communication is the #1 overhead
- Distributed teams are people too
- Not everyone consumes information in the same way
- Nice tools make people happy
- Documents are communication too
- … but are a lot more
- Project documentation != Software Documentation
- sqr-011.lsst.io

Platform		Audience		
		DM	Project	External
Chat	HipChat	Y	~	N
	Slack* (RFC-140)	Y	~	?
Forum	Community/Discourse	Y	Y	Y
Wiki	Confluence	Y	Y	read
Bugs, Patches	JIRA	Y	Y	read
	GitHub	N	N	Y
Doc Indexes	DocHub* API	Y	Y	Y

Software Documentation

- Needs to be part of the process
- Let devs use their toolchains
- Web-native
- CI the docs
- Embrace Jupyter

III. Science Quality & Analysis Services

SQuaRE's QA activities

- Leverage automation/DevOps techniques to execute, monitor and analyze the vast majority automatically (*we don't do the QA, we write software that does QA*)
- "QA as a service"
- Bring continuous integration processes to data as well as software
- Deploy anywhere, any number of times (scalability)
- "Bring your code to the data"
- No special snowflakes

QA-0

- Continuous Integration Services
- Test execution harness
- Validation Metrics Code
- Computational Metrics
- Curated Datasets
- SQUASH - Science Quality Analysis Harness

QA-1

- AlertQA
- Validation Metrics Performance
- Dome/Operator Displays
- Telescope Systems
- Camera Calibration
- Engineering and Commissioning

QA-2

- DRP-specific dataset
- Release data product editing tools (including provenance tracking)
- Interfaces to Workflow and Provenance System(s)
- Output Interface to Science Pipelines
- Comparison tools for overlap areas due to satellite processing
- Metrics/products for science users to understand quality of science data products
- Characterization report for Data Release

CI ➔ QA ➔ Commissioning ➔ L3

developers ➔ project science ➔ collaborations

job control ➔ pipeline ➔ workflow

virtualisation ➔ containerisation ➔ orchestration

inspection ➔ monitoring ➔ notification

logs ➔ dashboard ➔ visualisation

code+data ➔ facility ➔ science

fixed ➔ flexible ➔ arbitrary

SQuaRE QA and user services tech stack

- AWS (EC2, S3, RDS etc)
- Google Compute Engine
- Openstack
- Fastly / Varnish
- S3 / Swift
- Kubernetes (Swarm? Mesos?)
- Puppet & Ansible
- Elasticsearch / (LogstashlfluentdlRiemann)/ Kibana
- Docker / virtualisation
- vagrant / packer

Building on

- Documentation example

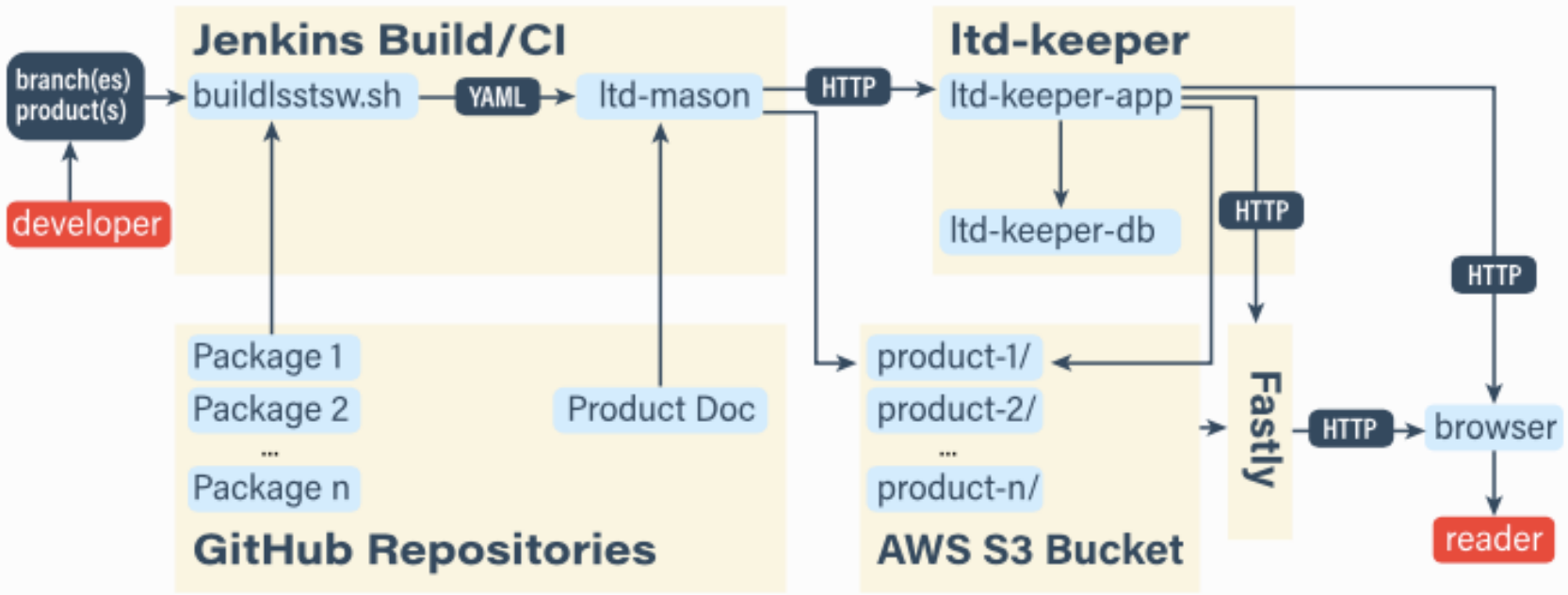


Figure 1 Architecture of LSST the Docs (LTD).

IV. Agile & Devops Practices

Why Agile Works

- Reduce the cost of change
- Reduce the cost of mistakes
- Iterate to the right solution
- Humanity sucks at macroplanning

Agile for Government™: Techniques

- Define work in terms of epics
- Break epics into stories (1 SP = 0.5 idealized days)
- Timed or time-boxed sprints
- Backlog grooming
- Testing & release
- Scrum or Kanban boards

V. Fostering a Positive Open-Source Team Culture

Here Be Dragons 

- Beware the silos:
  - Institution versus institution
  - Programmers versus non-Programmers
  - Algorithms versus Plumbing
  - Managers versus Scientists versus Engineers
  - Data Management versus The World
  - Construction versus Operations

Decisions are hard, m'kay?

- Decision-making can create gross inefficiencies
- Clamp down on bikeshedding
- Good technical teams need technical agency
- Buy-in comes from transparency
- Avoid uniformity for the sake of it
- Standing committees almost always a bad idea
- LSST DM's process (RFC): http://ls.st/o05

This is only the beginning

- Construction is not an end to itself
- Critical transitions are for people as well as systems
  - R&D -> Construction
  - Construction into Commissioning
  - Commissioning into Operations
- Software is never finished. Plan for it.
- Never descope: Deprioritise.
- **If it's not fun, something is wrong**
  - Don't burn your people out.
  - Beware engineered emergencies

Hiring is even harder

- Budget two years
- Favour generalists
- Hire for culture fit
- Hire for skill rather than knowledge
- … except for Python

Questions?
(answers not covered by warranty)