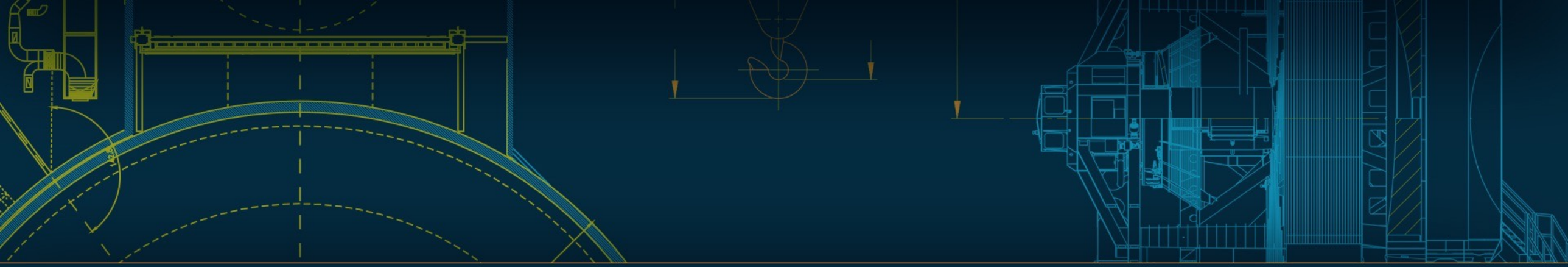
Data Release Production Team Introduction & Development Process

JOHN SWINBANK • SWINBANK@PRINCETON.EDU



LSST

Large Synoptic Survey Telescope

Data Release Production Team Role

1. Define and implement the scientific logic used to generate the annual LSST data releases.
2. Develop the scientific logic used to produce “calibration products” (flats, darks, biases, detector characterization, etc) for both nightly & annual processing.
3. Develop re-usable algorithms & software primitives used in both nightly and annual processing.

Yusra AlSayyad (Princeton)

Bob Armstrong (Princeton, 0.5FTE)

Jim Bosch (Science Lead, Princeton)

Merlin Fisher-Levine (Princeton)

Perry Gee (UC Davis; 0.3 FTE; leaving October 2017)

Mandeep Gill (SLAC; currently on 3 month contract)

Augustin Guyonnet (Harvard)

Vishal Kasliwal (Princeton; 0.5 FTE; leaving mid-2017)

Nate Lust (Princeton)

Lauren MacArthur (Princeton)

Josh Meyers (Princeton/SLAC)

Fred Moolekamp (Princeton)

Tim Morton (Princeton; starting February 2017)

Eli Rykoff (SLAC; 0.5 FTE)

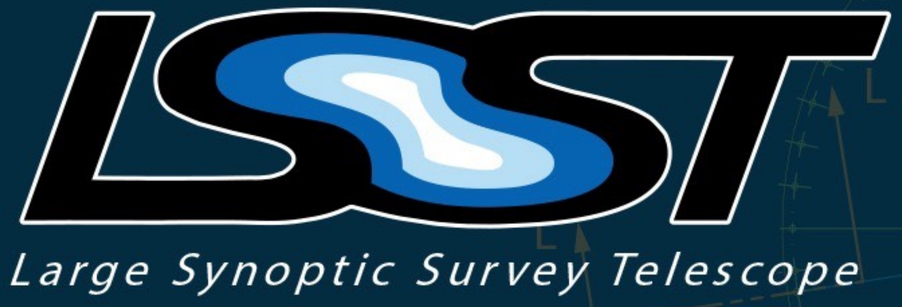
Pim Schellart (Princeton)

John Swinbank (Technical Manager, Princeton)

Honourable Mentions:

- Paul Price (Princeton) – funded by Subaru Hyper Suprime-Cam & Prime Focus Spectrograph
- **Robert Lupton** (Princeton) – funded by DM Project Science

Development Process



Cycle Planning

- Plan for the next 6 (or 3) months of work.
- Identify priorities based on the plan and discussions with stakeholders.
  - Most prominently Robert (Pipelines Scientist) & Jim (DRP Science Lead).
  - Substantial input on priorities from DMLT at May 2016 F2F.
  - [Post replan, “following the plan” should(?) dominate, but it hasn’t historically.]
- Discuss with developers, usually one-on-one, to agree what they’d like to work on and estimate effort.
- Enter plan as a series of “epics” in JIRA and Excel sheet (for PMCS ingest).
  - Most epics have concrete deliverables; some are “buckets”.
  - Requires effort estimate (1.4 SPs/day), start & finish dates.

Defining Stories

- Within an epic, stories are usually defined by some combination of Jim, the developer(s) working on the epic, and me.
- Encourage developers to define their own stories, with oversight from me, when possible; usually, newcomers need more help.
- Define stories as early as possible. Often, a few exploratory stories to start, then some more planning/design review/RFC, then more stories are added later.
- Estimate up-front how long a particular story will take (1.4 SPs/day).
  - Our estimates are improving, but still need more work.
  - Particularly hard to estimate QA-type stories (“something looks weird: figure it out”). Being more fine-grained helps – we are getting better.

Monthly Sprints

- Developers work on a sprint cycle aligned with the calendar month.
- We hold a sprint planning meeting towards the end of the previous month to discuss what will be worked on.
  - In practice, we don't go into that cold: it's rare that an epic fits within one month, so usually the developer is rolling on with existing work.
  - Even if not, I will generally have discussed with the developer before the meeting what their goals are.
- Experience shows that trying to use JIRA directly in the meeting is clunky. Usually discuss in general terms what developers are aiming to achieve, and program into JIRA later.

Weekly “Standup” Meeting

- Developers all report on progress over previous week and plans for the next.
- Encouraged to bring up any issues which are blocking them.
- Again, do this largely without reference to JIRA.
  - I will have a list of tickets that I know people are (or should be) working on ready to discuss with them
- Takes an hour or so – not really a “standup”.

Completing Work

- A story is complete when it passes code review.
  - NB not the same as Gaia's code reviews.
  - I monitor JIRA, and try to make sure that this happens at reasonably high quality (ie, including tests, documentation, etc).
- An epic is complete when the developer convinces me that it's done & I push the relevant button in JIRA.
  - Which likely involves discussion with stakeholders, ensuring that everybody who has expressed an interest is happy with the outcome.
- There is no formal review/sign-off/acceptance testing of epic deliverables.

Commentary

Sprint?

- Not a “cross-functional team”.
- Rather than the whole team agreeing on a goal for the sprint, we have different developers working on distinct parts of the codebase for the long term.
- Individuals build up expertise on deblending, modelling, etc etc.
- Goals tend to break down as “individual (or small group) aims for the cycle”, rather than “whole team aims for the sprint”.
- Lax about adding stories mid-sprint.
  - Impact is limited to the individual doing the work, rather than disrupting the team’s goals.

Reviews & Demos

- Basically don't happen.
- This is a real problem:
  - Easy for quality-of-implementation issues to slip by unnoticed.
  - Easy for us to claim (EV) credit for work which isn't really done.
- We have tried sprint demos in the past (Summer 2015) with … difficulties.
  - Developers very unhappy: felt they were being unfairly attacked for delivering work as specified. Reviewers — obviously — disagreed.
  - Re-introduction will take careful management & cultural change.
- Could be at a per-sprint or a per-epic level.

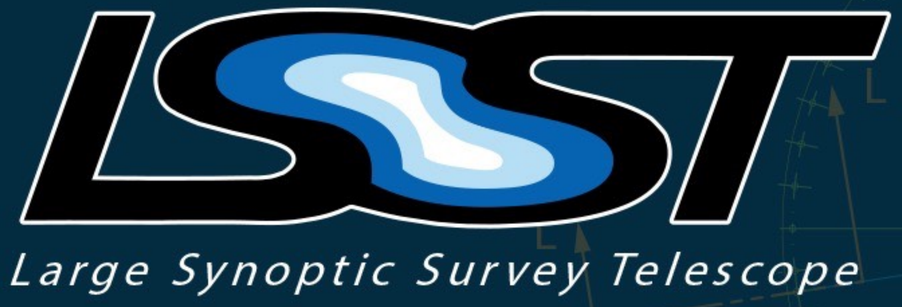
Prioritization, Planning & Responsibility

- Scrum tells us the Product Owner creates stories and prioritizes the backlog.
- Who is that? (Is it the same person across all teams?)
  - DM Project Scientist?
  - …delegating to institutional Science Lead?
  - …but T/CAMs are supposed to be producing the (re-)plan.
  - …and other DMLT members clearly (feel they) should have authority here.
- The aim is not to be dogmatic: at some level, the terminology is unimportant, but we need a clear hierarchy of who is setting priorities and who can call on what effort.

Design Review & Work Authorization

- Before a major new piece of work is started, the pundits should agree with the developer in question the approach to be taken.
- We handle this through RFCs at the moment, but that's not adequate:
  - Need to agree what counts as a "major" piece of work.
  - Need to get buy-in from relevant pundits.
  - Need to cover developers' backs in demos.
  - Need to build the best thing possible.
- Requires a significant time investment from reviewers.
  - Endemic DMLT busyness is a problem.

Extra Slides




What I need is agility and flexibility!
You'll have it!
The waterfall model is obsolete! It's time to listen to user feedback, we're ready for short sprints and continuous development!
That's the principle of Agile, we couldn't agree more!
We shouldn't restrict developers with deadlines and schedules, we need to be free to adapt to end up with a really awesome product
Wow, I'm really happy to hear that! It's rare to find customers who really get the philosophy!
Yeah! However, for the budget we'll need a single fixed price with the details of what you're gonna provide. We need to plan, you know?
CommitStrip.com

Development Challenges

- In R&D & on other projects (HSC), involved with implementation of "middleware" (e.g. task framework, workflow system).
  - Now in the purview of other groups (not always clear where, certainly not pre-replan).
  - Still adjusting to this.
- Long staff ramp up time.
  - Slow to staff up (and still below the staff we need: more tomorrow).
  - Complex codebase inherited from R&D, poorly documented rules & conventions.
  - Major algorithmic achievement requires significant skill & experience.

Cycle Planning #2

- While executing the cycle, JDS tracks whether epics are starting on time and if others are over-running.
- In extremis, can drop an epic from the cycle by notifying Kevin Long in the month before it starts.
  - In our current way of working this has no earned value consequence(!)
- If we don't drop an epic, but aren't ready to start it, we will show a variance until it is done.
  - Particularly tiresome when the epic is not urgent: can leave it sitting on the back-burner while we fight fires elsewhere, but will show an EV variance forever.