SQuaRE
Team Introduction & Development Process

Frossie Economou • frossie@lsst.org

Large Synoptic Survey Telescope

What John et al. said

This talk is the diff with the SQuaRE branch

Large Synoptic Survey Telescope

SQuaRE's role

Science|Software? Quality and Reliability Engineering

- Automated quality control/testing [cf. LDM-151]
  - Harness for monitoring software and data quality
  - Regression, trending analysis and alerts
- Developer infrastructure supporting software QA (IEEE 730)
  - Documentation
  - Continuous Integration
  - Communication
- Code distribution and Science Platform environment
- No longer doing science verification, KPMs or integration

SQuaRE's people

FTEs: 4.5 EVM (5.4 total) across 7 humans

100% and in Tucson unless otherwise indicated as of Dec 2016

- Jonathan Sick
- Adam Thornton 🆕
- JMatt Peterson

- Frossie Economou (T/CAM, ~90%)
- Angelo Fausti (75%)
- Josh Hoblitt (~50%, remote)
- Michael Wood-Vassey (Acting Science Lead, ~25%, remote)

All construction-era hires: 4 astronomy background, 3 other background, most would now be described as devops/full-stack engineers

Developer services

For SQuaRE, Construction is Operations

Currently in production or upcoming this cycle

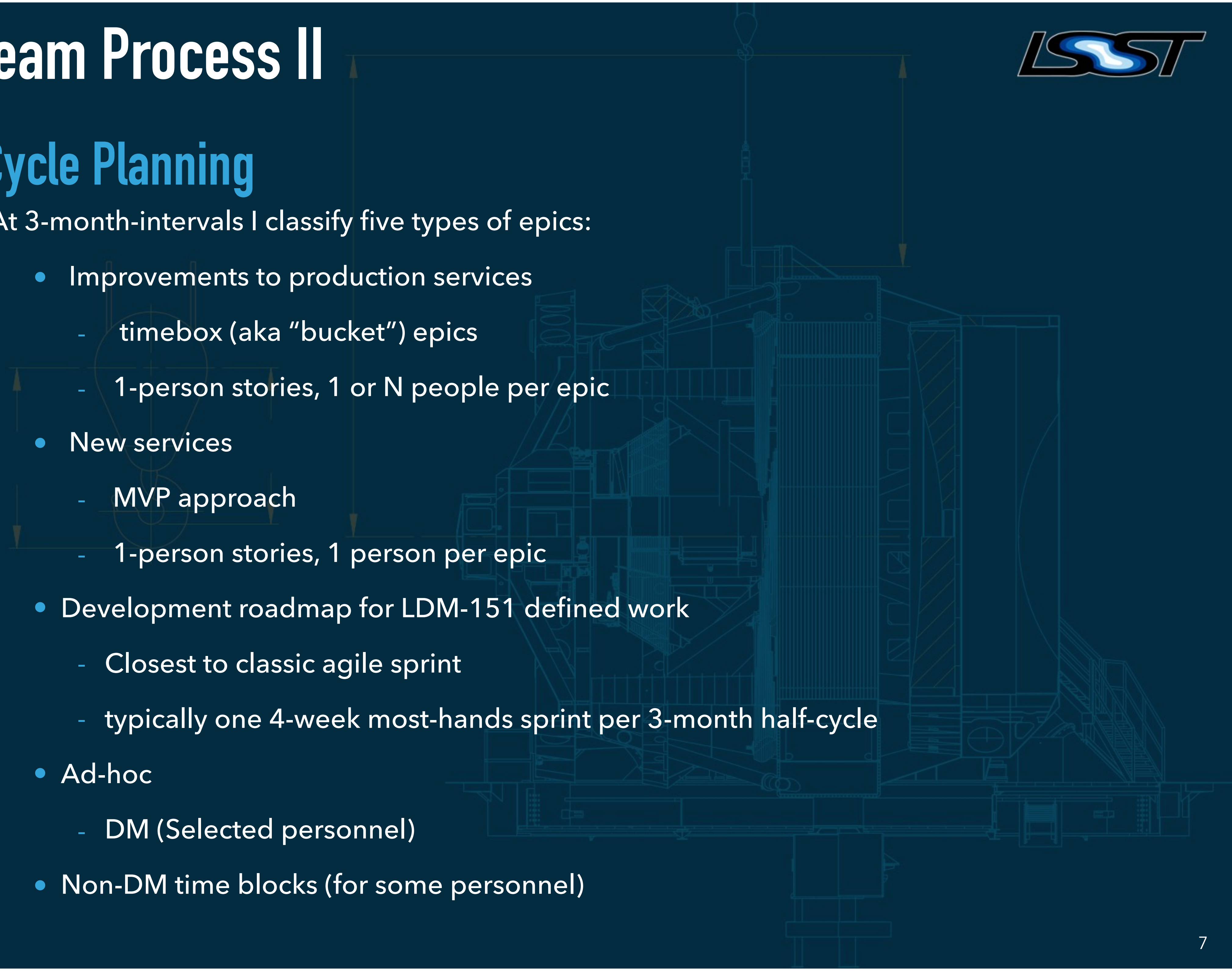
- pipelines.lsst.io - stack release and stack documentation
- developer.lsst.io - developer documentation
- [dmtn|sqr]NNN.lsst.io - technote platform
- ci.lsst.codes - continuous integration platform
- squash.lsst.codes - QC harness and validation framework
- status.lsst.codes 🆕 service status monitoring
- api.lsst.codes [coming soon] 🆕 microservices platform for monitoring etc
- community.lsst.org forum
- slack chatbot 🆕
- .. etc…

Team Process I

Constraints (general and specific)

- EVM planning cycle
  - High stakes
  - As done by NSF is not matched to Agile process
  - Agile For Government™ can be a workable compromise…
- Many developer-facing services in production already
  - As users take up a service, obvious what features are high priority
  - Nobody wants to tell a dev they have to wait 6+ months for their request to get serviced, but EVM…
- Also have our LDM-151 development capabilities to deliver
- Generalists/devops engineers but small team, risk spreading too thin or context-switching too often

Team Process II



Cycle Planning

At 3-month-intervals I classify five types of epics:

- Improvements to production services
  - timebox (aka "bucket") epics
  - 1-person stories, 1 or N people per epic
- New services
  - MVP approach
  - 1-person stories, 1 person per epic
- Development roadmap for LDM-151 defined work
  - Closest to classic agile sprint
  - typically one 4-week most-hands sprint per 3-month half-cycle
- Ad-hoc
  - DM (Selected personnel)
- Non-DM time blocks (for some personnel)

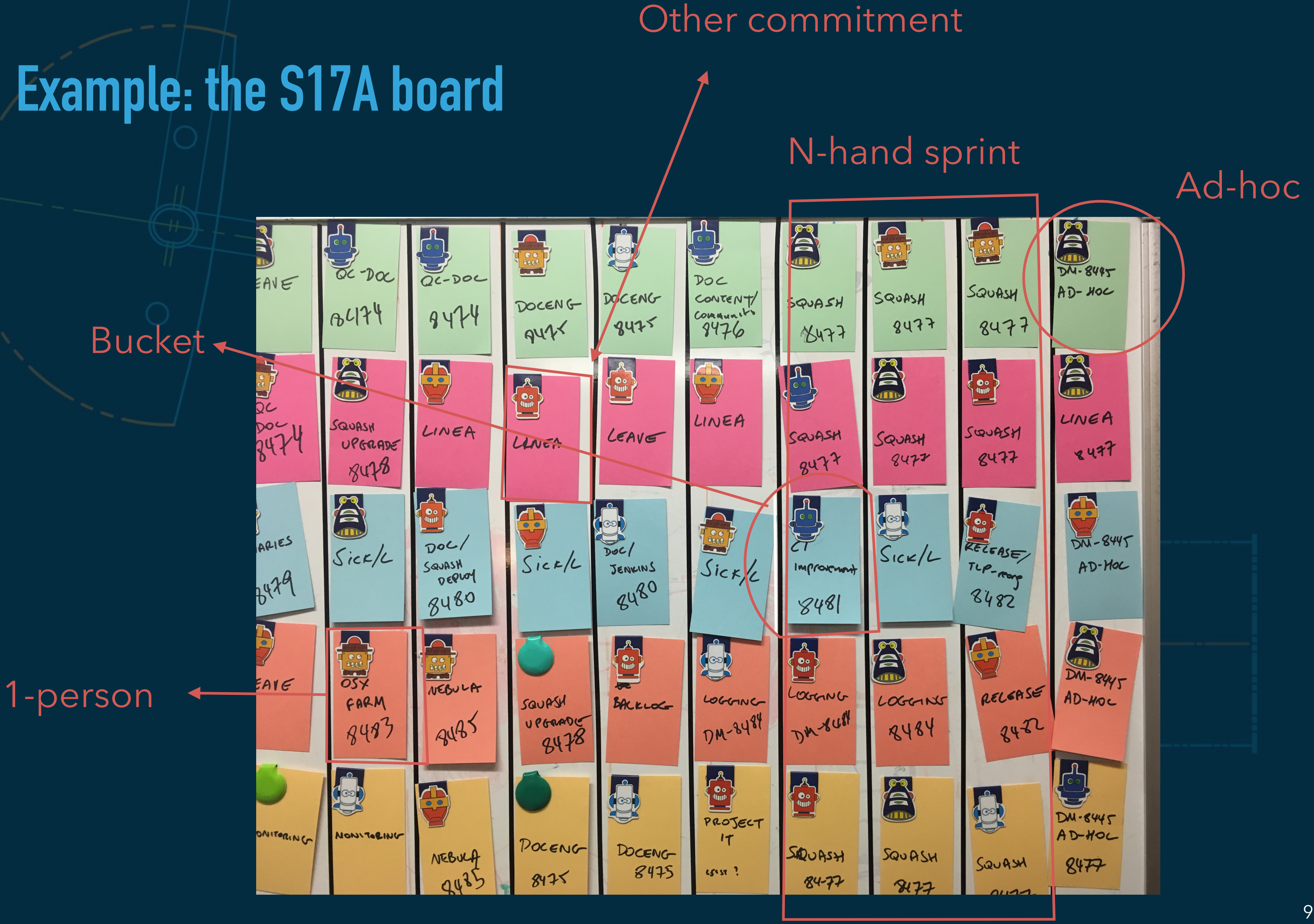
Team Process III

Generating Fully Loaded Cycle Plan

- Minimum 1-week-per-dev epics…
- quantised to units of 1 week-per-dev “cards” [literally]
- fully planned across all weeks in the cycle
- mitigate context switching as much as possible by constraining the technical stack
- cycle plan for Kevin using same spreadsheet as other T/CAMs
- card board used similarly to a Kanban board during the cycle

Team Process IV

Example: the S17A board



Team Process V

In-cycle process

- Every week identify in progress epic card for each dev
- Adjust if it makes sense (eg blocked, urgent issue)
- Discuss scope and technical approach
- Identify stories in that epic for next goal
- Daily not-really-stand-ups to
    - round table on status
    - co-ordinate work with team-mates working with same card
    - informally peer-review new technical approaches
    - raise potential threats to estimate
    - drink coffee
- Normal DM process (ticket branches, review etc for most tickets)
- Cowork session one afternoon a week
- I review and sign off before epic can be closed

Commentary I

The Good, the Bad & the Ugly

Nobody in their right mind would choose this over an agile methodology. That said:

- It's actually not horrific. Team devs are shielded from most of the details and focus on opening and closing their tickets
- I do have estimation feedback at the Epic level
- "Staying in JIRA" is a godsend (thx Kevin!)
- Move to half-cycles doubled work but increased accuracy
- You sometimes have to take variance on the chin to do the right thing (eg. allow a dev with momentum to do one more feature before losing their context) - cycle end is always Solomon's judgement
- Unplanned situations make for hard choices
- Disconnect with folks in LoE mode over the realities is stressful

Commentary II

Is "The Process" a problem?

- Sure, it's "ditch-digging" work for T/CAMs
- But it's not that much more work over normal technical planning and reporting in normal agile environments (~0.15 FTE more maybe)
- Team is not far from peak efficiency in many contexts
- However I am spending twice as much time managing a much smaller team than I did in a typical agile environment
  - Lots of higher management requests (LDM-151, WBS, Planning Packages, re-plan, slides, review materials etc etc)
  - Inefficient decision-making frequently wreaks havoc with finishing things (too many cooks, hard to find someone to just call it)
  - Poorly defined internal interfaces result in too much P2P negotiation
  - We're not leveraging the stuff we do (eg. monthly report) outside T/CAMs