Data Access and Database (DAX)
Team Introduction & Development Process

Fritz Mueller • fritzm@slac.stanford.edu

Large Synoptic Survey Telescope

DAX: Who We Are

- Team of 4 full-time engineers, 4 part-time engineers, 1 T/CAM, (5.8 FTE), located at SLAC
- Mandate: design and implementation of non-trivial database systems, data archives, and associated access systems, as required in support of the mission. Recently: plus some misc. frameworks/middleware. WBS: 01.02C.06
- Expertise: scientific data management, databases at extreme scale, distributed storage systems, backend service architectures
- Experience: blend of high energy physics, observational astronomy, and industry

DAX: Who We Are

Full-time

- Igor Gaponenko
- John Gates
- Nate Pease
- Kenny Lo

T/CAM

- Fritz Mueller

Part time

- Andy Hanushevsky
- Fabrice Jammes
- Andy Salnikov
- Brian Van Klaveren

Collaborator

- Vaikunth Thukral (X-SWAP)

DAX: Cycle Planning

- Plan 6 months at a time; opportunity to tweak at 3 mo. point
- T/CAM does preliminary load of cycle-planning Google sheet
  - Work chosen according to long-term plan + carry-over + some standard “bucket” epics
  - Epics created if not already in backlog
  - Preliminary resource balancing
- Discussions with individual devs to sanity check effort estimations, order of work, epic descriptions/scope
- Discussions with T/CAMS to sanity check inter-team dependencies
- Google sheet to Kevin and thence to PMCS

DAX: Sprint Execution

- 1 month sprints, synchronized to calendar months
- No daily standups
- Weekly team meeting – primary weekly commitment for devs
  - Project news to team
  - Round of scrum-like status: done in week, issues, intentions for upcoming week
  - Group hack session: ad-hoc design discussions, collaborations, whiteboard time, coffee & snacks
- Currently no demos or retrospectives at end of sprint, though we do review the Google sheet to stay aware of cycle targets

DAX: EVM vs. Agile

- Currently the project uses JIRA story points for EVM effort estimation; obviates velocity feedback advantages of Agile
- EVM planning packages drive cycle loading a la waterfall; coarse grained and don’t account for emergent work, so:
- Some effort reserved for “bucket” epics; we execute small tasks within this effort budget in a more Agile fashion
- No way to stop/pause/segment planning packages once touched; cumulative and somewhat spurious variances

DAX: Would Like to Improve

- Estimation is poor; could we implement velocity feedback to help?
- Can we find a better way to manage delayed/alternative/re-scoped planning packages, resulting in more meaningful variance reporting?
- Should we add retrospectives/demos, and if so how can these be done with low productivity-impact on devs?
- EVM long-term planning a la waterfall is extremely expensive, and there is a danger that it will go instantly out of date.
- In general, much need for architecture specification beyond verbal/gestalt.