SRE+UXDD to Improve Productivity of User Support Processes

Prepared for the 2020 Collegeville Workshop on Scientific Software

Mark C. Miller

July 22, 2020

LLNL-PRES-812659

Lawrence Livermore National Laboratory

User support impacts productivity of users <u>and</u> developers

- Each inquiry takes resources
  - reply to email, diagnose and/or reproduce bug, file a detailed issue ticket, …
- Few projects (no DOE projects I am aware of) have dedicated resources
  - When developers are doing this…they are not making progress on project milestones
- Required resources scale with number of users
  - There is a “penalty” for creating widely adopted and used software
- A difficult tradeoff (see [1] and [2])
  - Better software with poorer support
  - Poorer software with better support

Strike the right balance, formalize processes, practice makes perfect

The Fusion of SRE and UXDD

- Site (or software) Reliability Engineering (SRE)
  - Google's answer to ensuring their product (www.google.com) works for users
  - SREs fix problems that crop up and continually make incremental improvements to the [software/system] to make it more reliable[2], more scalable, and more efficient

- User Experience Driven Development (UXDD)
  - Common, though somewhat recent, terminology in the industry
  - Continually feeding back user experience into design, development, testing, etc.

Here, "SRE" means "SRE+UXDD".

The Basic SRE Process

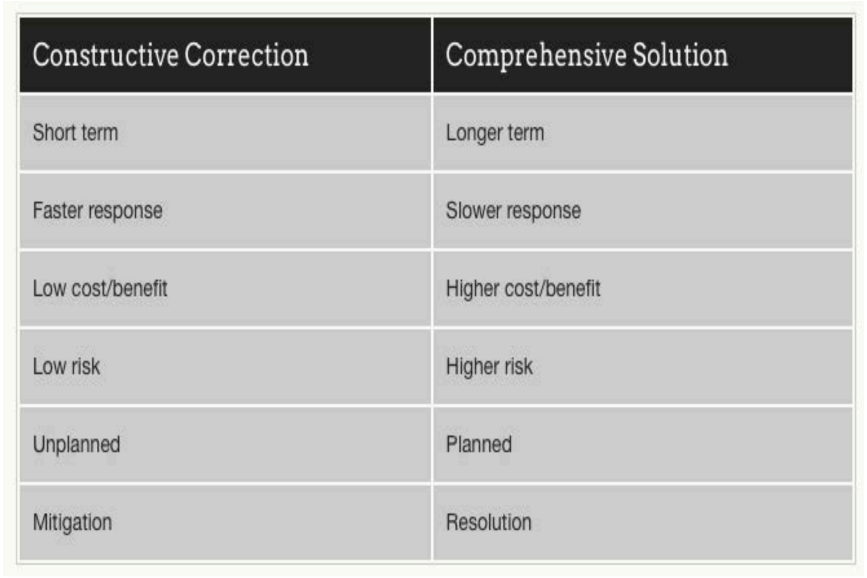
- SRE work allocated and rotated among developers in **shifts**
- During a shift, one developer, the “**SRE primary**”, is responsible for all SRE activity
  - Except for escalations, all other developers are free to ignore
- The SRE primary responds to **all inquiries** within the **response time goal**
  - Note: response != resolution
- SRE primary aims to **resolve all inquiries** during their shift
  - Note: SRE ”resolution” != user’s inquire addressed to satisfaction
  - Handoff unresolved inquiries to next primary

During a shift, no expectation that SRE primary gets any PD done

At idle times they likely can…but no *expectation* of this

SRE vs. Product Development

- In response to user incidents, identify…
  1. Constructive correction (SRE todo list)
  2. Comprehensive solution (PD backlog)
- Constructive correction has value iff
  1. A step towards comprehensive soln,
  2. Substantially reduces impact of issue
  3. Delivered sooner than comprehensive soln.
- Avoid increasing technical debt



Constructive Correction	Comprehensive Solution
Short term	Longer term
Faster response	Slower response
Low cost/benefit	Higher cost/benefit
Low risk	Higher risk
Unplanned	Planned
Mitigation	Resolution

Don’t let the perfect become the enemy of the good

- Adopt a practice of identifying constructive corrections wherever possible
- Often even a partial correction goes a long way towards preventing further issues and **engendering good will**
- Add a special label in the product development backlog (e.g. `low-hanging-fruit`)
  - Work that can be completed in ½-day of developer’s time

Response Time and Response vs. Resolution

- In SRE processes, Response != Resolution

- User's value quick response (knowing they've been heard)
  - Some idea if, when how their issue will be resolved

- Setting expectations is critical

- Negotiating priorities (and sometimes even funding) encouraged

Managing SRE Effort and Costs

- Key parameters: a) Coverage hours, b) Shift length, c) Response time goal

- Example: (VisIt w/8 core developers)
  - Coverage hours: West Coast Business Hours (M-F, 8-5)
  - Shift length: One week
  - Response time goal: 4 hours

- Round-robin load balancing SRE activity across development team
  - Many factors complicate this simple approach
  - Every team member becomes competent in SRE

- For team of 6, 17% of budget on SRE might be too much…**so, reduce coverage**

First contacts vs. on-going dialog

- Juggling many communication platforms not practical
  - Confluence, Jira, Slack, MS Teams, Mattermost, GitHub, Email, Twitter, phone, drop-ins

- Must balance **accessibility** for users vs. **productivity** for developers

- Be flexible with first-contacts but restrict on-going communication

- Favor Communication platforms that…

  …Engage "whole" team rather than individual developers

  …Are discoverable from Google

  …Support many features (attachment types, sizes, etc.)

Complicating situations

- Redirecting (to SRE primary) long-time friends and colleagues
- User's who abuse support resources
  - Too much hand-holding
  - Everything is urgent
- VIP users
- Distributed teams and classified computing
- Escalations

A common misconception:
SRE is an *interruption* to product development

- A successful software product involves more than software quality
- It also involves the quality of the user experience (UX)
  - Responsiveness to user's issues, Ease of use, Reliability, etc.
- Ensuring team members become proficient at UX has many important benefits
- Continually addressing UX issues is part of ensuring the product's sustainability
- It is not equal in importance to PD but is still <u>very</u> important (see [1] and [2])

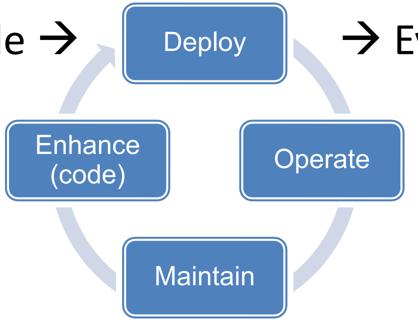
Lawrence Livermore National Laboratory

Site Reliability Engineering (SRE)

(replace “site” with “software”)

- Google’s answer to ensuring their product (www.google.com) works for users

- Traditional SE = Conception → Design → Code → Deliver finished product
  - Development team’s involvement ends with completed, delivered product

- Lifecycle SE = Conception → Design → Code →  → Eventual Replacement

- SREs fix problems that crop up and continually make incremental improvements to the [software/system] to make it more reliable[2], more scalable, and more efficient.

[2]The probability that a user can employ the system to perform a required function without failure under stated conditions

Common actions to resolve an SRE issue

- Answering a question or referring a user to documentation.
- Diagnosing the cause of the user's issue.
- Developing a work-around for users and a reproducer for developers.
- Identifying a constructive correction that would (partially) address the original SRE inquiry and then engaging in the work to resolve it.
  - Common example: Poor or missing documentation
- Identifying a comprehensive solution and filing a new PD issue or determining if the issue is already known and adjusting its priority based on frequency of encountering

Resolution of an SRE issue does not mean user's issue is addressed to satisfaction.

Some of the Goals of SRE Process

- Build/maintain a reputation for **timely** and quality **response** to user’s inquiries.
- Develop practice of **continuous user feedback** and subsequent quality improvements
- Evolve a **database of SRE activity** to inform future development plans
- **Cultivate SRE competence** across whole development team
- **Load balance SRE effort** in an equitable way across the development team.
- **Manage SRE costs/effort** for team as a whole.

Developers are SREs