Community Resource Management II

Development Cycles for Open-source
While open source can help achieve your development goals, it is also worth considering when open source might actually be a hindrance. There are several scenarios where open source can potentially interfere with your development cycle.

- Development is too rapid for open source affordances (docs, Wiki, general understanding) to keep up.
- Missing transparency (issues not clear, information not shared) breaks link between developers/team and contributors.
- Openness clashes with privacy and confidentiality.

When Open Source Software Comes with a few Catches
https://www.wired.com/story/when-open-source-software-comes-with-catches/

Open-source Sabotage and Protestware
https://www.wired.com/story/open-source-sabotage-protestware/

Dealing with Open Source Conflict
https://codeengineered.com/blog/2018/open-source-conflict/

Effects and Antecedents of Conflict
https://dl.acm.org/doi/10.1145/2818048.2820018

Understanding the open source software life cycle
https://www.redhat.com/en/resources/open-source-software-life-cycle-brief

A review of open source software development life cycle models
https://www.researchgate.net/publication/289328296_A_review_of_open_source_software_development_life_cycle_models

No-code Development
No-code approaches allow an internal development team or open-source development team to create new features or extensions of the existing codebase without having to directly manipulate the code. This might involve interacting with an interactive flowchart

interface or aesthetic interface. The effects of no-code development are potentially revolutionary, and a way to get a broad audience

No-code Development (Wikipedia)
https://en.wikipedia.org/wiki/No-code_development_platform

Future of No-code Development
https://medium.com/@sannanmalikofficial/the-future-of-no-code-development-40dac8cee756

A simple guide to the no-code movement
https://webflow.com/no-code

How No-code Tools are Democratizing 3D Design (Verizon)
https://events.verizon5glabs.com/parttwothreeddesign

Benefits of a No-code/Low-code Development Platform
https://www.informationweek.com/software/the-benefits-of-adopting-a-low-code-no-code-development-platform

Process Flow Diagram
https://en.wikipedia.org/wiki/Process_flow_diagram

Aesthetic Computing
https://medium.com/@isohale/aesthetic-computing-f715cbf06430

User Path Analysis
A more quantitative way to understand pathways is to create a user path analysis. A user path analysis involves evaluating the different paths through your community's various resources most frequented by contributors and participants.

For example, a common pathway might be from Onboarding Guide to Wiki to Guidance Tree to Github Repository. You might use this information to optimize this pathway and expect a large audience at every step. Alternatively, this pathway might be the most common, but is only taken by about 10% of your contributors. In such cases, you might look to cultivate alternate pathways.

Path analysis can also help prioritize which community resources need to be maintained on a high-frequency basis, and which resources can be maintained less frequently (or replaced by different resources).

User Path Analysis (Google Analytics)
https://support.google.com/analytics/answer/9317498?hl=en

6 issues that user path analysis can uncover
https://www.moengage.com/blog/issues-user-path-analysis-can-uncover/

Path Analysis in Computing
https://en.wikipedia.org/wiki/Path_analysis_(computing)

Is path analysis a good use of your time?
https://www.kaushik.net/avinash/path-analysis-a-good-use-of-time/

Where are the opportunities for expansion, education?

Targeting in a Layered Open source community
https://publish.illinois.edu/bradly-alicea/2020/12/10/targeting-in-a-layered-open-source-community/