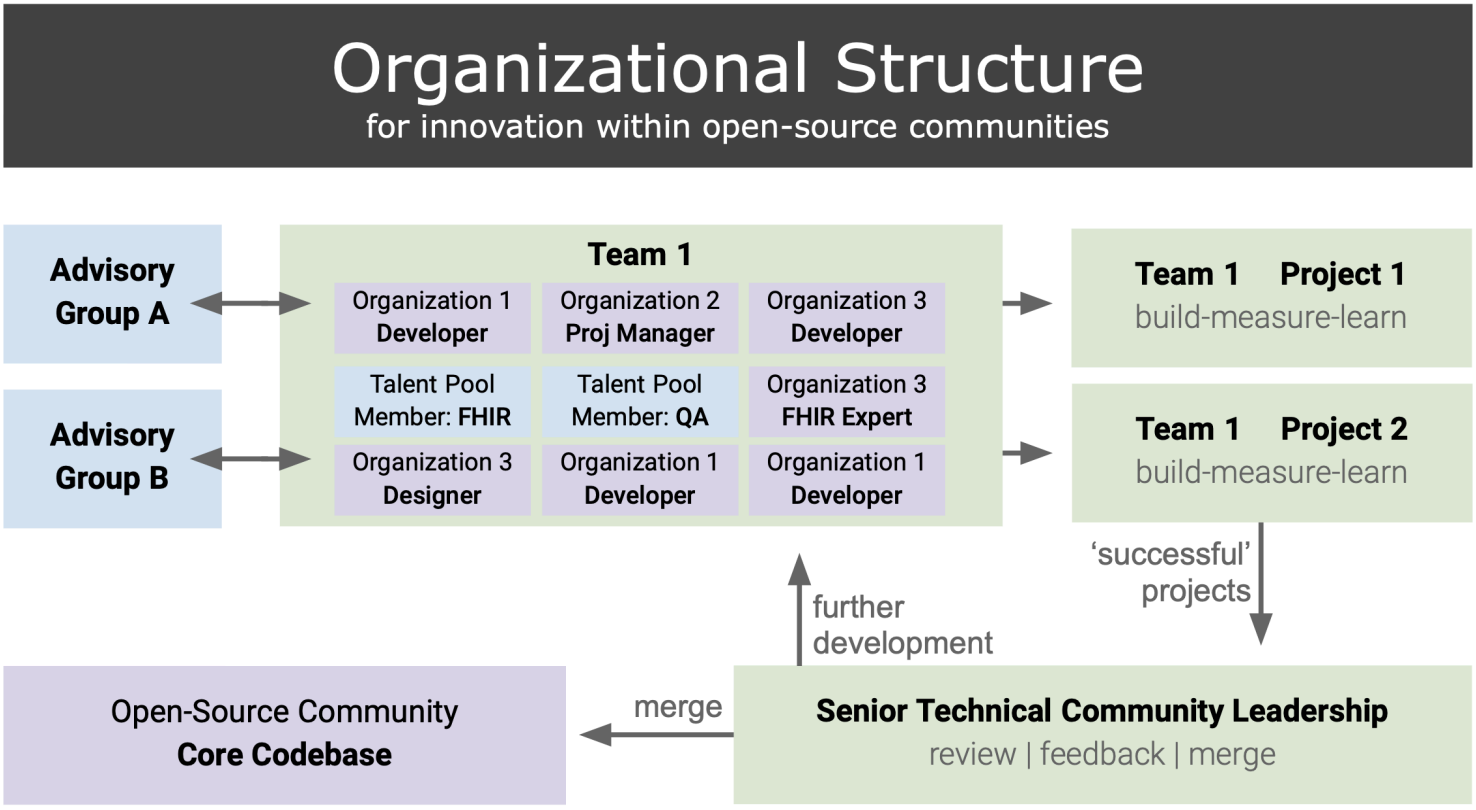
Original Article:
http://www.gregoryschmidt.ca/writing/org-model-for-innovation-within-open-source-community

---

Proposed model for innovation within an open-source community

GregorySchmidt.ca July 2019



How does one bring the buzzword of 'innovation' into an open-source community?

I don't know, but this proposal outlines my current thinking on this topic. Consider it a work in progress, and please consider adding feedback directly into the Google Doc copy of this text.

Listen to the post on your podcast app under: Gregory Schmidt, or the audio in the YouTube version below:

Eric Ries emphases in his book *The Lean Startup* that successful innovation (building a successful business around a new product or service) relies upon being able to cycle through the build-measure-learn feedback loop as quickly as possible in order to test hypothesis via experiments. [I recommend reading the entire book, or at least skimming my summary of it **here**].

What is the right organizational structure within an existing open-source technology community to enable this type of experimentation and innovation discovery?

Relying on everyone in an open-source community to vote on everything is not the right approach to rapid hypothesis-driven experimentation. At the other extreme, rogue autonomous teams may build solutions that don't integrate well 'back into core' (aka back into the main code of the open-source project). This proposal tries to balance these tensions.

This proposal identifies

1. Teams: their formation, community consultation process, and decision-making process.

2. Advisory groups & talent pools: how they help support teams.

3. A way to **merge new code** back into core.

Team Formation

Teams can self assemble as they see fit to solve problems and test new hypothesis.

Teams that form are fully autonomous in their team members, community consultation process, and decision-making process. This is to enable maximal flexibility for innovation and ensure they are 'not bogged' down by top-heavy bureaucracy or complex organizational structures.

Teams can determine the makeup of their collaborators. In the spirit of open-source development, it makes sense that the team has representation and members from different community contributors, but this is up to the team to decide. Some teams may complete a project entirely with one organization; other teams may wait for multiple organizations to join.

Teams can determine their membership size, and the skillsets required. It is recommended that the team is cross-functional - meaning they can move an idea from hypothesis to development to testing without having to rely on other teams. Small (up to 8-12 person) cross-functional teams seem to used often in agile workflows.

There may be cases where a particular project requires several cross-functional teams. These may be considered sub-teams of the larger 'team'. The takeaway summary is that it is encouraged that teams contain the skillsets to test their hypothesis fully.

Building siloed functional teams (eg. a technical team, a design team, a user testing team) is not the preferred method of development at this time. Siloed teams require more complex hand-off between groups. This results in reduced efficiency, slower processes cycles, and less individual ownership in the final product.

Projects

A team may work on a single project, or a single team may work on multiple projects. There may also be settings where a single project involves several sub-teams.

Think of the project as a short-lived iteration of work that the team undertakes. Such as a single MVP or feature or experiment to test. A team may bring in additional members specifically for a particular project.

The team may be considered as a 'longer-term group' of individuals that regularly collaborate on projects.

Community Consultation Process

The team can determine its community consultation process. When should the team seek feedback on their ideas before development? It is up to the team to decide.

Some teams may prefer to remain 'outside the community' for a period of time in order to develop rapidly or via a novel approach. Other teams may prefer to collaborate on every detail with the community.

In many circumstances, it is highly recommended that teams provide a transparent process for all community members to contribute to the discussions occurring within a team. A team is not obligated to 'slow down' their development process for widespread community

consultation; but it is encouraged that the method they choose for project development is transparent so that interested community members may contribute their opinions.

When the best path forward is unclear for the team's voting members, this may be a good opportunity to extend an opportunity for further community and expert consultation. (Though, the final voting rights remain up to the team's decision making process). Community consultation hopefully makes merging back into core more likely, because the final team's output will have more broad appeal.

Decision Making Process

The team can determine its decision-making process. Who will have 'voting rights' within the team? This is up to the team to decide.

Some groups may concentrate all voting power within one individual. Other teams may only make decisions based on full consultation of the open-source community (likely a very slow method).

If the objective of a team is to be able to rapidly test hypothesis through the build-measure-learn feedback loop a decision-making process that involves those who have "skin in the game" (aka, are actively contributing resources to the team) is likely a reasonable start.

Advisory Groups

Over time expert advisory groups should be available to help support teams in their development process. These advisory groups are built around different domains of knowledge and functional work.

For instance, a design advisory group, a technical advisory group, a FHIR advisory group, a terminology & concepts advisory group, a security advisory group, etc.

Advisory groups are comprised of members from within the open-source community, as well as external experts. The exact formation of these expert advisory groups is unclear. Likely a process of invitations, self-identification, and self-enrollment will work at the start. At the start, it may be easiest if community assigns a single go-to person or chair for each group.

The role of expert advisory groups is to help provide high-level advice to the autonomous teams. Teams may choose to follow, or not follow the advice the advisory groups provide.

The hope is that advice will help the teams arrive at solutions that are more likely to work, and in turn more likely to be merged into the community's core codebase.

Members of an expert advisory group may become 'hands-on' in particular teams or projects. However, it is not expected that expert advisory group members provide ongoing hands-on support to teams. Their role is more to provide expert opinion, rather than hands on work.

Over time there is the potential for each advisory group to form within itself its own community of best practice. This may be of interest to the expert advisory individuals within a group as they can share their experience and learning from their broader work outside of OpenMRS.

Talent Pool

Connected with each advisory group is a talent pool. The talent pool is comprised of community members (volunteer and paid) who are able to make significant contributions to projects.

Cross functional vertical teams will need different members depending on the work that is being accomplished. Sometimes team members for this work are already pre-determined by the collaborating teams. At other times they may seek additional members to join their team.

As each advisory group is connected with a talent pool, this talent pool can may be a source of potential people to become involved with a team or its project.

Members of the talent pool may be junior members who seek hands-on project experience, and mentorship of the advisory group. Talent pool members may also be experts themselves who have the capacity through their own project funding or time to do more prolonged work with teams and projects.

When talent pool members engage in projects they often will join the cross-functional team leading the project, or the team working on a specific project that they are required on.

Unlike advisory group members who provide more opinions, members from the talent pool actually ‘get the job done’ and are engaged much more closely with teams and projects.

(I’m not a huge fan of the term ‘talent pool’. Alternative name suggestions?).

Merging with core

The process of being able to innovate within an open-source community, and the process for reintegrating the best new projects into the core codebase are two separate yet related tasks.

The tasks are separate because the above development process emphasizes autonomy among teams. Teams can set their own rules, and play by their own rules. However, when a team is working on something it does not mean their work will de-facto reintegrate into the core codebase.

A small group of senior technical leaders, perhaps with a longstanding history within the open-source project, may be the right group who decide what is eligible to be merged back into the core codebase. Perhaps this group also sets their own principles on how they operate and how they decide what to merge. For particular merge requests, the group may develop a method of soliciting feedback from the community and its key members. These senior technical leaders may make suggestions back to teams on what changes should be made prior to their approval to merge to core.

The hope with the development process outlined in this post is that - as autonomous teams have the freedom and flexibility to work on new ideas; talk with the community; talk with expert advisory groups within the community; and build MVPs that can gather real-world metrics about their ideas - the process of re-integration into the core for successful projects will be data-driven and rather straightforward.

I hope this proposal will help enable innovation within the structure of a larger open-source community.

Doing so will enable an organization to move from 'single-threaded decision making' to 'multi-threaded decision making' (to borrow Paul Biondich's phrase).

This proposal is based on considerations from:

> *The Lean Startup* by Eric Ries [A summary of that book is in the prior blog post]
>
> My experience closely working within my brother's startup

My more limited experience working within the OpenMRS community

This proposal is missing further reading from the areas of:

Lean Impact by Anne Mei Chang

The Innovator's Dilemma by Clayton Christensen

Books on open-source communities (suggestions welcome for texts that specifically address innovation within open-source communities).