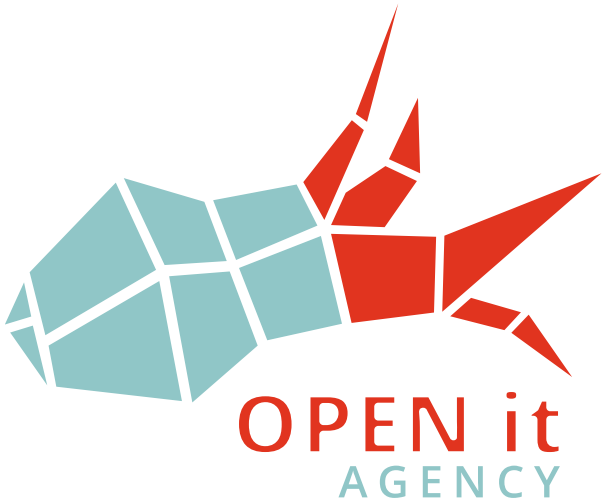

COMMUNITY PLANNER TOOL

VERSION 0.7 26TH NOVEMBER 2014

 OPEN IT AGENCY

1 STOCK

In general, communities need something to grow from. i.e. there should already be something tangible for users to interact with. People gather around resources and real potential - only in the rarest cases is a mere idea enough.

What can also help: signs of success or the participation of well-known people (in terms of attracting attention as well as providing skills) or large institutions etc.

“a necessary pre-condition for success is having ‘something runnable and testable to play with.“
“all communities start with users.“
“tomorrow’s developers are today’s users.“

SUSTAINABILITY AND EXPECTATIONS

Before people or businesses are willing to invest their time, they generally want to know if their investment is secure.

Is the project stable?

Can it be sustained?

Will it still be there tomorrow - reliable, up-to-date, supported and therefore usable?

These concerns show the need for a transparent sustainability plan that lays out answers and gives the users some security.

Where do the resources for operation, maintenance and development come from? This could be a business model, or references to an important funding partner. A business model is mostly self-explanatory, and other sustainability models may also be presented.

Another expectation is that the future of the project should be somewhat foreseeable.

How do you get involved?

Is the direction in which things will develop communicated transparently?

Are the goals clear?

And is the project structure (Project management, governance, IP etc), as well as the resource situation, set up so that this goal seems attainable?

2 STORY

WHAT AND WHY?

Is everything explained clearly?

Does the website explain quickly and clearly, what it's about, in what context it sits and what possibilities are available?

How can these possibilites be described with simple and concrete active words (verbs)?

This is to a large extent basic marketing practice.

Even if you're dealing with a community of contributors who are very comfortable with understanding technical details, it's worth making it as easy as possible to get started. Technically-versed people will be able to click through to subpages for the information which is only relevant for developers.

For a community of contributors & codesigners, what is important is who is behind the project itself. Faces are more memorable than abstract concepts, and it's easier to build a relationship with people than projects. .

Lastly, are there success stories, eg has the product been deployed by large organisations? Tell people about them. Success can play an important part in building trust.

BACKLINK: Does the Story fit the Stock? Is everything true and verifiable?

2 COMMUNICATION DESIGN & PROJECT STRUCTURE

CHOOSING TOOLS

Collaboration requires Tools, Communication channels and structure.

Is the product available? Can it be purchased or self-built?

Is there access to all the necessary tools and means, or are they yet to be produced?

Is it easy for potential vendors to get in contact?

Is all the required knowledge available? eg. the design files, documentation, learning resources, support channels, metainformation about the project. This can be provided in an FAQ, a Mission Statement, a Roadmap or a Governance Model (see below), a Release Management Guide, a Contributor License Agreement (see below) or similar.

There is a wide variety of digital communication tools that can be used for collaboration. In general, the fewer deployed, the better. This means people know where to look to find up-to-date information, and don't miss out on conversations or decisions happening in other channels. Possibilities are websites, online shops, mailing lists, version control systems, wikis, issue trackers, forums for discussion and support, FAQs, IRC channels, support hotlines, project archives, blogs, podcasts, videos, polls, social networks, external Open Innovation or project management platforms.

In addition there are also analog channels which can be deployed, such as events (eg. regular meetups or conferences or irregular hackathons or themed workshops), open workspace/ateliers, a shop or showroom and easy-to-reach contact people.

BACKLINK: Are all the practices in the Story backed up and supported?

"Succesful open source communities know that attracting external contribution depends on the ease by which software code, documentation, project memory and other outputs can be accessed and improved by others."

"In a community-led project, people participate in order to satisfy their own needs – you might call it 'scratching an itch'. So an individual's main driver will not be to build a community, but to solve a specific problem they face in their job, studies or hobby. For this reason, projects should be run in such a way that process and community overhead do not get in the way of people pursuing their interests."

"The key is to use tools that are appropriate to your community and to keep them to a bare minimum.(...) Tools should facilitate, not dictate."

"Early and frequent releases are crucial for building a sustainable open source project: releases attract users, some of these users become contributors, and more contributors make the project stronger. The downside of releasing early and often is that one needs to manage user expectations. Projects need to be clear about the status of their releases and draw attention to any known bugs in the documentation."

PROJECT STRUCTURE & ONBOARDING

In order to gain a community, you need a clear and understandable project structure.

The fewer tools deployed the better, in general. Clarity is usually better than a complex range of functionality.

An important question is how someone can get involved in the development process. The first steps should be very easy, although the complexity can then slowly increase.

One good way into the project is through the product or service itself - perhaps there's an invitation upon purchase or with the packaging inviting users to come to the site to view special online material such as a video, or to answer a a few questions, give feedback, etc.

Important aids for newcomers can be, for example, an FAQ, a 'How to Get Involved' document, a To-Do list with various roles and tasks listed, or a community manager, whose role it is to welcome newcomers, to help them with questions, and to put community members in touch with others.

"You don't need to be a software developer to contribute to an open source project. The code, documentation and artwork that make up an open source project have all been created, tested, used, discussed and refined by members of the project community. These processes can be broken down into myriad tasks, requiring different skills, levels of involvement and technical expertise."

"The quickest, easiest and most significant way to provide such support in the early stages of your involvement is to answer newcomers' questions. These are often best answered by those who have themselves recently experienced the same issues. By answering questions from newcomers, you will also be helping the project by saving the developers' time."

"Clear documentation explaining how one can move from passive user to contributor, then to senior contributor with commit rights, and eventually to decision-making board member, will be in place. This means that everyone knows what to do if they want to increase their role and responsibility in the project."

COMMUNITY MANAGEMENT

A living community must be carefully nurtured. A few golden rules of Community Management include : Be friendly and a balanced mediator, give feedback, block trolls, use criticism, train coaches/community managers (see Roles) and lastly, maintain an archive.

"Acknowledging contributions immediately and addressing issues in a timely manner are equally important for the future engagement of these users."

"In their early stages, the most significant concern for projects is likely to be dealing with the inevitable support burden. Handled badly, this might, at best, lead to users turning away and, at worst, might lead to the founder giving up. If success is to be achieved, the leader ultimately has to find people to carry out this work. Employing people is one option; another is encouraging users to help out each other by writing documentation and fixing bugs. However, if this is to happen, there must be an infrastructure in place to allow them to do this. Contributions need to be proactively encouraged and leaders also need to ensure that contributions are helpful and of a sufficient quality."

"As a new user, you may feel reluctant to make requests or provide criticism, no matter how constructive, for fear of seeming impolite or ungrateful. But most open source projects will encourage you in every possible way to contribute to discussions on user mailing lists or add feature requests to the issue tracker. At the same time, they will probably make you aware that not all feature requests will be implemented, although every comment will be carefully considered and feedback will be provided as to how important that request is."

4 ROLES & MOTIVATIONS

A community is an open ecosystem in which various actors play different roles.

For community building, it's important to understand which roles occur in particular cases, so that motivations are understood and the community members assuming particular roles can be properly supported. In this way, the ecosystem can grow.

ROLES WITHIN THE WHOLE PROJECT

There are various perspectives to different roles. e.g. there are roles which are professional occupations. Which occupations could work with the product/service/project, or use it for their work? e.g. Carpenter, Gardener, Architect, Producer, Sales Dealer? They are **Stakeholders**, or could become so. **Co-Developers** can recruit themselves from these professions (e.g. a carpenter developing an improved version of an open design chair) or they can come from a hobbyist background. **Funders** could be for example foundations or public institutions. And of course **Customers** and **Users** are also important roles.

With all of these roles, you can be building up a relationship from an early stage in the project's development, in order to understand more about their motivations.

ROLES WITHIN THE CO-DEVELOPMENT PROCESS

A complex open co-development process offers a variety of different roles. A person or institution can therefore play one role in the professional ecosystem around the product, and another in the project's community, for example an engineer working on the mechanics of a product might at the same time assume a community manager role.

Roles in the co-development process could be: **Tester** (Product testing; giving feedback), **User Support** (helping others; answering questions); **Evangelising & Marketing** (explaining the project; recommending/advocating); **Graphics & Design**, writing **Documentation** (Manuals; Meeting Notes; How Tos; FAQs); **Translation**; **Quality control** (keeping an overview of the entire process) and more. These roles can be planned for, outlined and communicated in the project structure.

"As well as looking at the developers and users around a project, it's also important to consider the companies that engage with it. This includes hosting and support providers, consultancy and customisation

services, and companies that bundle the software with other products as part of solutions. The ecosystem around a project is an important indicator as clearly such companies have a strong interest in the sustainability of the software themselves.“

MOTIVATIONS

Where predetermined roles exist, it is worthwhile asking what motivations could drive each role.

What motivates a customer to buy something? This question has long been wrought over in marketing literature, resulting in a wide range of answers. Potential answers are, for example, this product best solves the problem at hand, it helps the customer to improve their status, to earn money, increase desirablity etc, the product is the most reliable, it has the best delivery options, the best range, good service, lowest price, the customer trusts the producer, the producer/vendor is local or is the most convenient to buy from, or the producer was simply the first they found.

What motivates co-developers? see the following table from Lakhani & Wolf: "*Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects* (2005).“ In many areas, direct parallels to open source hardware can be seen.

Motivations to contribute to F/OSS projects

Motivation	% of respondents indicating up to 3 statements that best reflect their reasons to contribute (%)	% volunteer contributors	% paid contributor	Significant diff (t statistic / p value)
Enjoyment based Intrinsic Motivation Code for project is intellectually stimulating to write	44.9	46.1	43.1	n.s.
Like working with this development team	20.3	21.5	18.5	n.s.
Economic/Extrinsic based Motivations Improve programming skills	41.3	45.8	33.2	3.56 (p=0.0004)
Code needed for user need (work and/or non-work)* Work need only	58.7 33.8	- 19.3	- 55.7	- 10.53 (p=0.0000)
Non-work need Enhance professional status	29.7 17.5	37.0 13.9	18.9 22.8	5.16 (p=0.0000) 3.01 (p=0.0000)
Obligation/Community-based Intrinsic Motivations Believe that source code should be open	33.1	34.8	30.6	n.s.
Feel personal obligation to contribute because use F/OSS	28.6	29.6	26.9	n.s.
Dislike proprietary software and want to defeat them	11.3	11.5	11.1	n.s.
Enhance reputation in F/OSS community	11.0	12.0	9.5	n.s.

*Aggregation of responses that indicated needing software for work and/or non-work related need . Not an actual survey question.

How can these particular motivations be supported?

Status can be communicated through badges, stars, points, experience levels or certification, or through the transfer of responsibility. (Responsibility can have similar effects to compensation).

Good community management and a good project structure - making it easy to contribute.

Displaying and rewarding successes, and showing gratitude.

Providing the right conditions for creativity, stimulating exchange and collaboration., and giving feedback.

"If the tools and processes used by project members on a daily basis are made transparent and appealing, and the adoption path into the community is clear, welcoming and informative, new contributors will join in. They will be attracted by the prospect of being part of a potentially successful self-sustainable project. Some of these contributors may be paid by various employers to work on particular aspects of the product. Others may be attracted by the technical challenge, the prospect of acquiring kudos, or simply by the opportunity to improve or showcase their skills. All these and other accepted forms of engaging with external collaborators can be considered and specified in the project's governance document."

"An OSS (open source software) community has no clear distinction between developers and users, because all users are potential developers. There is a large community of programmers trying to essentially outshine or impress their colleagues. They enjoy having other programmers admire their works and accomplishments, contributing to why OSS projects have a recruiting advantage for unknown talent than a closed-source company."

"Though a project may not be associated with a specific individual, the contributors are often recognized and marked on a project's server. This allows for programmers to receive public recognition for their skills, promoting career opportunities and exposure. In fact, the founders of Sun Microsystems and Netscape began as open source programmers."

BACKLINK: Are the Roles and Motivations supported by the **STOCK**, **STORY** and **TOOLS**?

2 GOVERNANCE & INTELLECTUAL PROPERTY

How do decisions come about, and who owns what is produced?

GOVERNANCE

The more complex a project and the more Stakeholders it has, the more important the governance model becomes. From the very start it should be apparent how important decisions are made; who is involved; and what processes are deployed in making them. This can be communicated through a governance document, or through FAQs.

Open Source projects can have various governance models. "Benevolent Dictators“ (Individuals or core teams who make the key decisions) are widespread in the open source software world, as are "Meritocracies“ (who carries out more work, has more of a say.)

Voting on important issues can also be deployed. However, a constant voting processes can tend to paralyse a project. One alternative is the "silent consensus". If no objections are made to a suggestion, a silent consensus is reached.

“It is critical that a project clearly communicates its policies and strategies to potential users and developers of the project’s outputs. A clear governance model also allows potential contributors to understand how they should engage with the project, what is expected of them and ***what protections are provided to ensure that their contributions will always be available to them.*** *In addition, it describes the quality control processes that help to assure potential users of the viability of the project“*

“It is never too soon to define a suitable governance model. Without one, the chances of third parties wishing to contribute are considerably reduced. This is for a number of reasons: potential contributors will not know how to contribute; they will not be sure what will happen to their contribution; the project will not look serious about engaging with third parties; there is no visible assurance that contributions will be managed in such a way that they will remain of value to the original contributor.“

"Potential deveolpers can see how the project is run and predict how it will react to their contributions before expending any significant effort on that work."

"There is no open development if I cannot influence and have my say in what we do and what we are trying to build. That is not open development, that is free labour!"

"The more complicated these habits and procedures become, the more important it becomes to aid newcomers with instructions on how they can begin to take part and have a say in the decision making process. Young communities might fall back on the body of knowledge built up and captured in mailing list threads but this does not always help newcomers and can leave them confused. What is needed is something written down, a 'governance model', to capture this shared understanding in a concise documentary form. Formalising arrangements helps ensure the community has a life of its own – independent of any one individual – that can survive and flourish for as long as there is a genuine sustained need for the project's outputs."

"In the long run, communities need to have open development mechanisms in place to ensure that when key contributors, including the founders, move on, their roles are adopted by others."

"Benevolent dictators need not possess the greatest technical skills, but they will, according to Fogel, have 'the ability to recognise good design'. Fogel goes on, however, to make the point that their main responsibility is ensuring participants continue to share the belief that they 'can do more in concert than individually'. Developers will only remain if their leader can make the project a place they want to keep coming back to. This means rewarding hard work by giving credit where it is due and, for those that want it, responsibility for more significant pieces of work. Management of open source projects has been described as active, informal, and low-key."

IP – INTELLECTUAL PROPERTY

Who actually owns products of collective collaboration?

Clarity about any "intellectual property" which is produced is important for the motivation of contributors. Contributors usually have to either transfer or give up their individual copyright claims in order for a project to function as a community operation.

According to the type and complexity of the project, there are important decisions to take into account (e.g. the choice of license) and agreements with contributors that must be made clear (eg. with a "Contributor License Agreement".

Licensing is a complex field - the process of choosing and communicating the license and the rules of the Contributor License Agreement should not be taken lightly. Particularly for larger projects, professional legal advice can be very helpful in this regard.

"In order to be safe, one should always make sure that agreements or contracts specify who will own the intellectual property that results from any collaboration, consortium or contract work"

"If you intend to work on a project as part of your day job, it is vital to check your employment contract for intellectual property rights (IPR) clauses to ensure that you are permitted to participate in an open source project. Since software code is copyright protected, only the owner of that code – or someone authorised by the owner – may legally license it for others to use. Many employers recognize the value of staff participating in the development of software that is used within the firm. Some employers have taken the next step and mark this recognition within institutional IPR policies and/or employment contracts. If you are an educator planning to involve students in an open source project, you should also check that their student agreement permits this type of involvement and that the students understand the implications."

"It is vital that you know who owns the copyrights in your project. To do this, you need to be able to trace every contribution back to its original author. In the case of contributions that directly affect your project outputs, such as programme code or documentation, this is most efficiently managed through a version control system coupled with a clearly defined Contributor Licence Agreement and a submission process using an issue tracker."

"Still it can be complex to ascertain who should make a legal complaint if someone decides to use the program in a way that violates its licence. To avoid this issue, some open source projects ask that contributors explicitly assign the copyright in their contributions to a body that administers the project, thus keeping ownership centralised and making enforcement of the licence easier. An alternative approach is to have contributors license their contributions to the project's administrative body under a licence agreement that permits the body to relicense the contribution."

"The issues of licence compatibility and of complex multiple ownership of intellectual property mean that it is desirable, if not essential, for programmers and their managers to keep detailed records. Version control systems provide some of this record-keeping automatically, recording who made changes to the code and what they did. To complement this information, managers should keep records of the contractual and licensing status of contributors in order to establish who owns their work. They should also require and store explicit agreements from copyright owners that their contributions may be licensed and distributed under the licence selected for the project as a whole. Where code is brought in from existing open source software, the details of the relevant licence must be recorded (having first established that this licence is compatible with the project's overall licensing policy)."