Meritocratic governance model

The meritocratic governance model is a commonly found model in which participants gain influence over a project through the recognition of their contributions. The Apache Software Foundation (ASF) is perhaps the most famous example of a large-scale meritocratic community. The foundation operates with an almost completely 'flat' structure, which means that anyone willing to contribute can engage with their projects at any level. At the other end of the 'control' spectrum is the benevolent dictator governance model, which is led by one individual.

The governance model under which a project is run is described in a governance document. In section 2 we provide a template for projects wishing to adopt the meritocratic governance model and create their own governance document. This template can be reused in its entirety or edited to suit individual needs. Like most of our materials, it is available under a Creative Commons licence (see footer for details) and can therefore be reused and modified, as long as attribution to OSS Watch is maintained. For information about the purpose of governance models, or for a discussion of the benefits of one model over another, please see our general document on governance models.

1. Introduction

Despite the apparent structural differences between meritocracies and benevolent dictatorships, both subscribe to the same open source principles of sharing the code and encouraging everyone to contribute back to the community. It is no surprise, therefore, that meritocratic project management committees and benevolent dictators both exercise their decision making power through loyalty rather than legalities. They all know that members are free to take the code and create alternative projects. In fact, this ability to fork is very important to the health of open source communities. This is because it ensures that those involved in project governance strive to make the right decisions for the community, rather than for a single individual or company.

However, there are notable differences between the two models, particularly with

regard to how decision making within the community is carried out. At its most extreme, a meritocratic governance model appears to give away control to community members in response to their contributions to the project. But in reality, control is not handed out without due consideration. A meritocracy is not a democracy; that is, not everyone has a binding vote. Only those who have earned it through positive contribution to the project have a binding vote. This allows project leaders to ensure that only those that share a common vision and, just as importantly, are willing to work towards that shared vision, are given decision making authority.

The 'flatness' of a meritocratic project's structure comes from the fact that once someone has decision making authority, they have exactly the same authority as everyone else. Another aspect of this flatness is that decision making responsibilities are usually reserved for those willing and able to understand, and appropriately represent, the views of the wider community. So, when an important decision is to be made, those with a vote are expected to represent the views of those who have yet to earn a vote.

Meritocratic projects, like all projects, usually start life with a small number of decision makers, possibly even a single person. Consequently, the early stages of project management are little different from those found in the benevolent dictator model. The key difference is that the initial team provides a mechanism for the distribution of control from the 'dictator' to a flat structure in recognition of contribution.

In the next section we provide a template for a governance document for a meritocracy, which projects can use as a basis for drawing up their own governance document. The template can be used in its entirety or customised to suit individual needs. OSS Watch can also help you fine-tune your model.

The template is based on the meritocratic model used in The Apache Software Foundation. It should be noted, however, that individual ASF projects are free to design their own bylaws. Therefore, this model is not likely to be identical to that employed in any given ASF project.

2. Template for a meritocratic governance document

2.1. Overview

This is a consensus-based community project. Anyone with an interest in the project can join the project community, contribute to the project's design and participate in

the decision making process. This document describes how that participation takes place and how to set about earning merit within the project community.

2.2. Roles and responsibilities

2.2.1. Users

Users are community members who have a need for the project. They are the most important members of the community and without them the project would have no purpose. Anyone can be a user; there are no special requirements.

The project asks its users to participate in the project and community as much as possible. User contributions enable the project team to ensure that they are satisfying the needs of those users. Common user contributions include (but are not limited to):

- evangelising about the project (e.g. a link on a website and word-of-mouth awareness raising)
- informing developers of strengths and weaknesses from a new user perspective
- providing moral support (a 'thank you' goes a long way)
- providing financial support (the software is open source, but its developers need to eat)

Users who continue to engage with the project and its community will often become more and more involved. Such users may find themselves becoming contributors, as described in the next section.

2.2.2. Contributors

Contributors are community members who contribute in concrete ways to the project. Anyone can become a contributor and contributions can take many forms, such as those outlined below and in the above section on users. There is no expectation of commitment to the project, no specific skill requirements and no selection process.

In addition to their actions as users, contributors will also find themselves doing one or more of the following:

- supporting new users (existing users are often the best people to support new users)
- reporting bugs

- identifying requirements
- providing graphics and web design
- programming
- assisting with project infrastructure
- writing documentation
- fixing bugs
- adding features

Contributors engage with the project through the issue tracker and mailing list, or by writing or editing documentation. They submit changes to the project itself via patches, which will be considered for inclusion in the project by existing committers (see next section). The developer mailing list is the most appropriate place to ask for help when making that first contribution.

As contributors gain experience and familiarity with the project, their profile within, and commitment to, the community will increase. At some stage, they may find themselves being nominated for committership, as described in the next section.

2.2.3. Committers

Committers are community members who have shown that they are committed to the continued development of the project through ongoing engagement with the project and its community. Committership allows contributors to more easily carry on with their project related activities by giving them direct access to the project's resources. That is, they can make changes directly to project outputs, without having to submit changes via patches.

This does not mean that a committer is free to do what they want. In fact, committers have no more authority over the project than contributors. While committership indicates a valued member of the community who has demonstrated a healthy respect for the project's aims and objectives, their work continues to be reviewed by the community before acceptance in an official release. The key difference between a committer and a contributor is when this approval is sought from the community. A committer seeks approval after the contribution is made, rather than before.

Seeking approval after making a contribution is known as a commit-then-review process. It is more efficient to allow trusted people to make direct contributions, as the majority of those contributions will be accepted by the project. The project employs various communication mechanisms to ensure that all contributions are reviewed by the community as a whole, but there is no need to detail them here. By the time a contributor is invited to become a committer, they will have become

familiar with the project's various tools as a user and then as a contributor.

Anyone can become a committer; there are no special requirements, other than to have shown a willingness and ability to participate in the project as a team player. Typically, a potential committer will need to show that they have an understanding of the project, its objectives and its strategy. They will also have provided valuable contributions to the project over a period of time.

New committers can be nominated by any existing committer. Once they have been nominated, there will be a vote by the project management committee (PMC; see below). Committer voting is one of the few activities that takes place on the project's private management list. This is to allow PMC members to freely express their opinions about a nominee without causing embarrassment. Once the vote has been held, the aggregated voting results are published on the public mailing list. The nominee is entitled to request an explanation of any 'no' votes against them, regardless of the outcome of the vote. This explanation will be provided by the PMC Chair (see below) and will be anonymous and constructive in nature.

Nominees may decline their appointment as a committer. However, this is unusual, as the project does not expect any specific time or resource commitment from its community members. The intention behind the role of committer is to allow people to contribute to the project more easily, not to tie them in to the project in any formal way.

It is important to recognise that commitership is a privilege, not a right. That privilege must be earned and once earned it can be removed by the PMC (see next section) in extreme circumstances. However, under normal circumstances committership exists for as long as the committer wishes to continue engaging with the project.

A committer who shows an above-average level of contribution to the project, particularly with respect to its strategic direction and long-term health, may be nominated to become a member of the PMC. This role is described below.

2.2.4. Project management committee (PMC)

The project management committee consists of those individuals identified as 'project owners' on the development site. The PMC has additional responsibilities over and above those of a committer. These responsibilities ensure the smooth running of the project. PMC members are expected to review code contributions, participate in strategic planning, approve changes to the governance model and manage the copyrights within the project outputs.

Members of the PMC do not have significant authority over other members of the community, although it is the PMC that votes on new committers. It also makes decisions when community consensus cannot be reached. In addition, the PMC has access to the project's private mailing list and its archives. This list is used for sensitive issues, such as votes for new committers and legal matters that cannot be discussed in public. It is never used for project management or planning.

Membership of the PMC is by invitation from the existing PMC members. A nomination will result in discussion and then a vote by the existing PMC members. PMC membership votes are subject to consensus approval of the current PMC members.

2.2.5. PMC Chair

The PMC Chair is a single individual, voted for by the PMC members. Once someone has been appointed Chair, they remain in that role until they choose to retire, or the PMC casts a two-thirds majority vote to remove them.

The PMC Chair has no additional authority over other members of the PMC: the role is one of coordinator and facilitator. The Chair is also expected to ensure that all governance processes are adhered to, and has the casting vote when the project fails to reach consensus.

2.3. Support

All participants in the community are encouraged to provide support for new users within the project management infrastructure. This support is provided as a way of growing the community. Those seeking support should recognise that all support activity within the project is voluntary and is therefore provided as and when time allows. A user requiring guaranteed response times or results should therefore seek to purchase a support contract from a community member. However, for those willing to engage with the project on its own terms, and willing to help support other users, the community support channels are ideal.

2.4. Decision making process

Decisions about the future of the project are made through discussion with all members of the community, from the newest user to the most experienced PMC member. All non-sensitive project management discussion takes place on the project contributors' mailing list. Occasionally, sensitive discussion occurs on a

private list.

In order to ensure that the project is not bogged down by endless discussion and continual voting, the project operates a policy of lazy consensus. This allows the majority of decisions to be made without resorting to a formal vote.

2.4.1. Lazy consensus

Decision making typically involves the following steps:

1. Proposal
2. Discussion
3. Vote (if consensus is not reached through discussion)
4. Decision

Any community member can make a proposal for consideration by the community. In order to initiate a discussion about a new idea, they should send an email to the project contributors' list or submit a patch implementing the idea to the issue tracker (or version-control system if they have commit access). This will prompt a review and, if necessary, a discussion of the idea. The goal of this review and discussion is to gain approval for the contribution. Since most people in the project community have a shared vision, there is often little need for discussion in order to reach consensus.

In general, as long as nobody explicitly opposes a proposal or patch, it is recognised as having the support of the community. This is called lazy consensus - that is, those who have not stated their opinion explicitly have implicitly agreed to the implementation of the proposal.

Lazy consensus is a very important concept within the project. It is this process that allows a large group of people to efficiently reach consensus, as someone with no objections to a proposal need not spend time stating their position, and others need not spend time reading such mails.

For lazy consensus to be effective, it is necessary to allow at least 72 hours before assuming that there are no objections to the proposal. This requirement ensures that everyone is given enough time to read, digest and respond to the proposal. This time period is chosen so as to be as inclusive as possible of all participants, regardless of their location and time commitments.

2.4.2. Voting

Not all decisions can be made using lazy consensus. Issues such as those affecting the strategic direction or legal standing of the project must gain explicit approval in the form of a vote. This section describes how a vote is conducted. Section 2.4.4 discusses when a vote is needed.

If a formal vote on a proposal is called (signaled simply by sending a email with '[VOTE]' in the subject line), all participants on the project contributors' list may express an opinion and vote. They do this by sending an email in reply to the original '[VOTE]' email, with the following vote and information:

- +1 'yes', 'agree': also willing to help bring about the proposed action
- +0 'yes', 'agree': not willing or able to help bring about the proposed action
- -0 'no', 'disagree': but will not oppose the action's going forward
- -1 'no', 'disagree': opposes the action's going forward and must propose an alternative action to address the issue (or a justification for not addressing the issue)

To abstain from the vote, participants simply do not respond to the email. However, it can be more helpful to cast a '+0' or '-0' than to abstain, since this allows the team to gauge the general feeling of the community if the proposal should be controversial.

Every member of the community, from interested user to the most active developer, has a vote. The project encourages all members to express their opinions in all discussion and all votes. However, only committers to the project (as defined above) and/or PMC members have binding votes for the purposes of decision making. It is therefore their responsibility to ensure that the opinions of all community members are considered. While only committers and PMC members have a binding vote, a well-justified '-1' from a non-committer must be considered by the community, and if appropriate, supported by a binding '-1'.

A '-1' can also indicate a veto, depending on the type of vote and who is using it. Someone without a binding vote cannot veto a proposal, so in their case a -1 would simply indicate an objection.

When a [VOTE] receives a '-1', it is the responsibility of the community as a whole to address the objection. Such discussion will continue until the objection is either rescinded, overruled (in the case of a non-binding veto) or the proposal itself is altered in order to achieve consensus (possibly by withdrawing it altogether). In the rare circumstance that consensus cannot be achieved, the PMC will decide the forward course of action.

In summary:

- Those who don't agree with the proposal and think they have a better idea should vote -1 and defend their counter-proposal.
- Those who don't agree but don't have a better idea should vote -0.
- Those who agree but will not actively assist in implementing the proposal should vote +0.
- Those who agree and will actively assist in implementing the proposal should vote +1.

2.4.3. Types of approval

Different actions require different types of approval, ranging from lazy consensus to a majority decision by the PMC. These are summarised in the table below. The section after the table describes which type of approval should be used in common situations.

Type	Description	Duration
Lazy consensus	An action with lazy consensus is implicitly allowed, unless a binding -1 vote is received. Depending on the type of action, a vote will then be called. Note that even though a binding -1 is required to prevent the action, all community members are encouraged to cast a -1 vote with supporting argument. Committers are expected to evaluate the argument and, if necessary, support it with a binding -1.	N/A
Lazy majority	A lazy majority vote requires more binding +1 votes than binding -1 votes.	72 hours
Consensus approval	Consensus approval requires three binding +1 votes and no binding -1 votes.	72 hours
Unanimous consensus	All of the binding votes that are cast are to be +1 and there can be no binding vetoes (-1).	120 hours
2/3 majority	Some strategic actions require a 2/3 majority of PMC members; in addition, 2/3 of the binding votes cast must be +1. Such actions typically affect the foundation of the project (e.g. adopting a new codebase to replace an existing product).	120 hours

2.4.4. When is a vote required?

Every effort is made to allow the majority of decisions to be taken through lazy

consensus. That is, simply stating one's intentions is assumed to be enough to proceed, unless an objection is raised. However, some activities require a more formal approval process in order to ensure fully transparent decision making.

The table below describes some of the actions that will require a vote. It also identifies which type of vote should be called.

Action	Description	Approval type
Release plan	Defines the timetable and actions for a release. A release plan cannot be vetoed (hence lazy majority).	Lazy majority
Product release	When a release of one of the project's products is ready, a vote is required to accept the release as an official release of the project. A release cannot be vetoed (hence lazy majority).	Lazy majority
New committer	A new committer has been proposed.	Consensus approval of the PMC
New PMC member	A new PMC member has been proposed.	Consensus approval of the community
Committer removal	When removal of commit privileges is sought.	Unanimous consensus of the PMC
PMC member removal	When removal of PMC membership is sought.	Unanimous consensus of the community

2.5. Contribution process

Anyone can contribute to the project, regardless of their skills, as there are many ways to contribute. For instance, a contributor might be active on the project mailing list and issue tracker, or might supply patches. The various ways of contributing are described in more detail in a separate document.

The developer mailing list is the most appropriate place for a contributor to ask for help when making their first contribution.

3. Conclusion

A clear and transparent governance document is a key part of any open development project. It defines the rules of engagement within the community and describes what level of influence a community member can expect to have over a project. In addition, it enables members to decide their level of involvement with that community. In the case of a meritocracy, it also provides a clear way to contribute and a highly visible reward system.

This example governance model comes from the more democratic end of the spectrum of control found within open development projects. But it is not a democracy. It is a model that passes control to those who are most likely to wield it for the benefit of all, rather than a minority interest. This document's partner, Benevolent dictator governance model, describes a model in which control is held by a single individual. This individual strives to ensure that the project represents a chosen stakeholder group, rather than the most active, as is the case with the meritocratic model.

4. Further reading

Links:

- The Apache Software Foundation

Related information from OSS Watch:

- What is a software patch?
- Governance models
- Benevolent dictator governance model
- Meritocrats, cluebats and the open development method: an interview with Justin Erenkrantz
- How to build an open source community
- Roles in open source projects

Date: Tue, 08 Feb 2011 (first published February 2010)
Last reviewed: February 2011
Author: Ross Gardler, Gabriel Hanganu

Unless otherwise indicated, this page is © 2010-2011 University of Oxford.


It is licensed under the Creative Commons Attribution-ShareAlike 2.0

England & Wales licence.

 OSS Watch is funded by the Joint Information Systems Committee.

OSS Watch values your input and questions.
If you have feedback on this document, or any OSS Watch activity, please send it to:
info@oss-watch.ac.uk
Follow our tweets via the OSS Watch twitter account at http://twitter.com/osswatch