Hipparchus Governance

Revision History

VERSION	DATE	REASON
1	2016/06/05	Creation

Table of contents

1. Introduction

Hipparchus is a library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language.

Hipparchus is developed under APACHE V2.0 license. All contributions to Hipparchus have to be compliant with the terms and conditions of this license. This license belongs to the "Business Friendly" category. It allows use, modification and redistribution. It does not impose further obligation to derived products.

The following document describes the governance model under which Hipparchus is run. It has been established on the meritocratic governance model proposed by OSS Watch, under Creative Commons License: Attribution Share Alike.

Read more: http://www.oss-watch.ac.uk/resources/meritocraticGovernanceModel.xml

2. Overview

Hipparchus is a consensus-based community project. Anyone with an interest in Hipparchus can join its community, contribute to its 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 Hipparchus community.

2.1 Roles and responsibilities

2.1.1 Users

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

Hipparchus asks its users to participate in the project and community as much as possible. The objective of the users' contributions is to enable Hipparchus to be in phase with the needs of the users. Common user contributions include (but are not limited to):

- ✓ evangelizing about Hipparchus (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 Hipparchus and its community will often become more and more involved. Such users may find themselves becoming contributors, as described in the next section.

2.1.2 Contributors and active contributors

Contributors are community members who contribute in concrete ways to Hipparchus. 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 must agree that their patches will be integrated in Hipparchus under APACHE V2.0 license and under the terms of the Hipparchus Contributor License Agreement.

Contributors engage with Hipparchus 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.

An active contributor is a contributor who has:

- ✓ a valid e-mail address attached to his profile
- ✓ in the last 12 months contributed in concrete ways to Hipparchus
- ✓ a maximum delay of reactivity of 3 weeks when contacted individually

2.1.3 Committers

Committers are community members who have shown that they are committed to the continued development of Hipparchus through ongoing engagement with Hipparchus and its community. Committers must also have earned technical merit based on the quality of their contributions. While no specific mathematical or programming knowledge is required and it is not expected that committers understand the entire Hipparchus code base, it is expected that committers have made significant contributions to at least one Hipparchus module, algorithm or implementation class. Committers must also have demonstrated that they work well within the Hipparchus community and can serve in the important role of welcoming new contributors and reviewing and incorporating their contributions. Committership allows contributors to more easily carry on with Hipparchus related activities by giving them direct access to Hipparchus resources. That is, they can make changes directly to Hipparchus sources, without having to submit changes via patches.

This does not mean that a committer is free to do whatever s/he wants. In fact, committers have no more authority over Hipparchus than contributors. While committership indicates a valued member of the community who has demonstrated a healthy respect for Hipparchus 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. Hipparchus 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 is no affiliation, qualification or other special requirements other than those described above. There is no minimum time of contribution before someone may be nominated to become a committer, though sustained engagement in the community is a factor the PMC may consider in evaluating candidates for committership.

To obtain rights for repository access, committers will have to sign a commitment (Individual Contributor License Agreement -ICLA- or Corporate Contributor License Agreement -CCLA-) indicating that every contribution to Hipparchus they will commit is covered by APACHE V2.0 license.

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, if the vote is successful, the new committer is privately informed and, assuming s/he accepts, the new committer is announced on the public mailing list.

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 recognize 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 Hipparchus, 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.

2.1.4 Project management committee (PMC)

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.

Any member of the PMC is free to leave after having informed the PMC of his or her decision.

2.1.5 PMC Chair

The PMC Chair is a single individual, elected by the existing PMC members. Once someone has been appointed Chair, the PMC Chair remains in that role until he or she chooses to retire, or the PMC casts a two-thirds majority vote to remove the PMC Chair.

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 deciding vote when the project fails to reach consensus.

2.1.6 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 recognize 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.

2.2 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.

2.2.1 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 Hipparchus 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 recognized 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 Hipparchus. 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 a 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.

It is a judgement call to determine what kinds of actions require discussion and the 72-hour lazy consensus process. Committers may commit patches directly that lead to discussion after a change has been made. There is nothing wrong with this, as long as consensus is eventually reached and, if necessary, the changes are either improved or reverted to reflect that consensus.

2.2.2 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'.

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 rescinded 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.

2.2.3 Types of approval

Different actions require different types of approval, ranging from lazy consensus to a majority decision by the PMC. These are summarized 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 more binding +1 votes than binding -1 votes and at least three binding +1 votes.	72 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 code base to replace an existing product).	120 hours

2.2.4 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.	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.	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.	2/3 majority of the PMC
Committer removal	When removal of commit privileges is sought.	2/3 majority of the PMC
PMC member removal	When removal of PMC membership is sought.	2/3 majority of the community limited to active contributors, committers and PMC members
PMC rules changes	When the rules defined in this document are changed	2/3 majority of the PMC

2.3 Contribution process

Anyone can contribute to Hipparchus, 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 developer mailing list is the most appropriate place for a contributor to ask for help when making their first contribution.

2.3.1 Contributor License Agreements

The following agreements have been established using models proposed by the Apache Foundation, under Copyright © 2010 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.

All contributors of ideas, code or documentation to Hipparchus complete, sign, and submit (via postal mail or email) an Individual Contributor License Agreement (ICLA) . The purpose of this agreement is to clearly define the terms under which intellectual property has been contributed to Hipparchus and thereby allow the PMC to defend the project should there be a legal dispute regarding the software at some future time. A signed ICLA is required to be on file before an individual is given commit rights to Hipparchus project.

For a corporation that has assigned employees to work on Hipparcchus, a Corporate CLA (CCLA) is available for contributing intellectual property via the corporation, that may have

been assigned as part of an employment agreement. Note that a Corporate CLA does not remove the need for every developer to sign their own ICLA as an individual, to cover any of their contributions which are not owned by the corporation signing the CCLA.

CLAs may be submitted by emailing a scan of the signed copy to admin@hipparchus.org. It is also possible to fill out the PDF document or edit the text document, create a detached GPG signature, and send both the document and the detached signature via email to the admin@hipparchus.org.

2.3.2 Software Grants

When an individual or corporation decides to donate a substantial body of existing software or documentation to the Hipparchus project, they need to execute a formal Software Grant Agreement (SGA) with the Hipparchus PMC. The PMC must VOTE to accept such contributions