Score-P Governance Model

Revision: 1675
Date: 2014-11-13 08:28:36 +0100 (Thu, 13 Nov 2014)

David Böhme, Christian Feld, Felix Wolf

1 Mission Statement

The Score-P project develops software for profiling, event-tracing, and online-access performance analysis of massively parallel programs. The project also manages the evolution of the CUBE and OTF2 profiling and tracing file formats, respectively. All results are released as open-source software. The principal guidelines for software development in the Score-P project are production quality, portability to all relevant HPC systems, performance, and scalability.

The project is governed in a meritocratic, consensus-based fashion, where control is awarded in recognition of contribution. Contributions are encouraged by an easy-to-understand and transparent process based on lazy consensus.

2 Scope

This governance model governs

- The project infrastructure
  - Source code repositories for Score-P, OTF2, OPARI2
  - Ticket and wiki system
  - Continuous integration environment
  - Code review systems
- The software
  - Score-P
  - OTF2
  - The CUBE reader and writer component[1]

[1] since version 4

- OPARI2
- Build infrastructure and tests

Furthermore, it defines roles of the people and institutions involved as well as common processes.

3 Roles

We categorize the people and parties involved with the project into various roles. The role of an individual/party reflects their tasks and their level of engagement and responsibility within the project. We define the following roles:

- User
- Contributor
- Approver
- Maintainer
- Chief Maintainer
- Partner (institutional role)
- Spokesman / woman

The following sections define these roles in detail and explain how to obtain them.

3.1 User

Users

- use the software,
- evangelize about the project,
- give feedback,
- and offer moral support.

Anyone can use Score-P, there are no restrictions.

3.2 Contributor

We welcome and encourage participation of community members in the Score-P development. Anyone who contributes, for example by

- suggesting features,
- reporting / triaging / fixing bugs,
- writing code, tests, or documentation,
- conducting tests,
- participating in code reviews,
- organizing meetings,
- participating in the community,
- supporting and coaching new users,
- or improving the contribution process

is a contributor. Again, anyone can become a contributor, there are no restrictions.

3.3 Approver

Contributors with enough experience in the project can become approvers, who evaluate other contributions in the code review system. As such, approvers

- safeguard the long-term success of the project,
- review (code) contributions,
- accept or reject contributions based on
  - technical fit (does the contribution adhere to the project's development guidelines and code quality standards)
  - spirit fit (is the goal and chosen approach of the contribution in alignement with the project's general objectives).

  Rejection of a contribution should be based on and accompanied by constructive feedback.

How to become an approver To become an approver, one needs to be nominated by an existing approver and seconded by another approver (from a different institution, if possible). Other approvers can raise an objection; in this case, the board of maintainers decides.

3.4 Maintainer

To ensure long-term code quality, every part of the Score-P software stack should have a person who is responsible for its maintenance, i.e., a *maintainer*. Hence, maintainers assume responsibility for a *maintenance unit*, for which they should

- report the status,
- ensure continuous beta readiness,
- participate in strategic planning,
- manage IP rights discussions,
- review unreviewed contributions.

Board of maintainers The maintainers organize themselves in the *board of maintainers*.

How to become a maintainer To become an maintainer, one needs to be nominated by an existing maintainer and seconded by another maintainer (from a different institution, if possible). Other maintainers can raise an objection; in this case, a vote in the board of maintainers decides.

3.5 Chief Maintainer

The chief maintainer is responsible for the state and development of the software as a whole. As such, the chief maintainer

- leads the board of maintainers,
- decides if no consensus is reached within the board of maintainers,
- cooperates with the spokesman (see partners / spokesman),
- coordinates and facilitates technical activities,
- oversees contributions from a strategic perspective.

How to become chief maintainer The chief maintainer is appointed by a vote among the maintainers.

Chief maintainer term The chief maintainer is appointed for a single term, but re-election is possible. A chief maintainer term is aligned with the release schedule, and lasts 1 to 1.5 years.

3.6 Partner

A partner is an organization that contributes (significant) resources (financial, infrastructure, etc.) to the project. As such, it

- sends a delegate to the board of partners,
- participates in the long-term strategic planning.

Partners should always provide up-to-date contact information.

Board of partners The partners organize themselves in the *board of partners*, where they participate in decision making through their delegates. The delegate role is not fixed, a partner can name any person as its delegate at any time. However, only one vote per partner is allowed for any decision.

How to become a partner To become a partner, the organization needs to be nominated by an existing partner and seconded by another partner. Other partners can raise an objection; in this case a vote in the board of partners decides.

3.7 Spokesperson

The spokesperson leads the board of partners. In that role, he/she

- decides if no consensus is reached,
- cooperates with the chief maintainer,
- and coordinates and facilitates non-technical activities, such as
  - acquisition of funding,
  - public relations.

How to become spokesperson The spokesperson is appointed by a vote among the board of partners.

Spokesperson term The spokesperson is appointed for a single term, but re-election is possible. As for the chief maintainer, a spokesperson term lasts 1 to 1.5 years.

3.8 Role Expiration

Approver and maintainer status automatically expires after 1 year of inactivity. Partner status expires automatically after 2 years of inactivity. Organized by the chief maintainer (for developers) and the spokesperson (for partners), the lists of developers and partners should be regularly (at least annually) examined for inactive members. The following steps should be followed:

1. Identify inactive members
2. Notify members who have been inactive for 6 months (developers) or 1 year (partners) about impending role expiration
3. Remove members who have been inactive for 1 year (developers) or 2 years (partners)

3.9 Role Revocation

Anyone can voluntarily step down from a role. Maintainers and partners are asked to announce their intention to step down some time in advance. In extreme cases, role priviliges can be revoked.

3.9.1 Revoking developer priviliges

Revocation of someone's developer priviliges can be requested by any maintainer. In this case, the Chief Maintainer initiates a vote in the board of maintainers. Approver priviliges will be revoked if 2/3 of the maintainers agree. Maintainer priviliges will be revoked if 2/3 of both the maintainers and the partners agree.

3.9.2 Revoking partner status

Revoking partner status from an organization can be requested by any partner. The spokesperson then initiates a vote among the board of partners. Partner status will be revoked if 2/3 of the partners and 2/3 of the maintainers agree.

4 Communication channels

Various communication channels serve discussions between users, between developers, and for information exchange from users to developers and vice versa. The following community communication channels should be established:

- Mailing lists
  - User support list (users to approvers).
    Serves as a feedback channel for users to developers, e.g. for reporting bugs. The communication is one-way, i.e. users don't see other user's mails.
  - Approver list (restricted to developers on approver level or above).
    A mailing list for developer-internal discussions.
  - Partner & maintainer list (restricted to maintainers and partners).
    A mailing list for discussions concerning project governance / strategic decisions for maintainers and partners.
  - Announcement list (chief maintainer to users).
    A news channel, allows us to announce e.g. new releases to the public.

- Wiki
- Ticketing and Code Review System.
  The wiki and the ticketing/review systems are usually the place to go for basic technical discussions.
- Teleconferences

5 Processes

Development and evolvement of the project happens through various recurring standard processes. The governance model defines basic guidelines for development and decision-making processes.

5.1 Development Processes

The governance model provides guidelines for two often recurring development processes, the release planning and feature development.

5.1.1 Release Planning

The integration work required for the next release(s) should be discussed in a biannual, strategic release-planning meeting between maintainers. The community is welcome to participate, and should be asked for input and feedback.

5.1.2 Software Development

All development activities should be managed in the source code repositories.

Trunk and release branches The development trunk is used for integrating new features and small bugfixes only. Release branches are stable branches with a fixed feature set from which releases tarballs are created. Feature integration into trunk undergoes pre-commit reviews, smaller changes undergo post-commit reviews. Checkins to release branches are triggered by the maintainer of the affected component.

Feature development New, larger features or other development effort with a significant impact (e.g., refactoring, infrastructure modification, ...) should occur in a feature branch. Feature branches should undergo regular post-commit reviews to ensure consistent quality early on.

New feature branches should be proposed with a *letter of intent* to the developers, e.g. on the approver mailing list or at a developer meeting / conference. The letter of intent briefly states the purpose and planned development work to be performed in the new branch. This allows us to facilitates the coordination of new features, detect

conflicts with existing and/or future functionality, and discuss possible implementation approaches early on.

Hence, the basic workflow for feature development is:

1. Letter of intent
2. Development in feature branch with post-commit reviews
3. Integration in trunk with pre-commit review

Experimental work In addition to reviewed branches, it is always possible to create non-reviewed experimental branches for work which will not directly be integrated into the trunk.

License Developers must make sure that all contributions can be released under the license used by Score-P, and that all copyright holders are listed in the affected source files and the top-level COPYING document.

Reviewers and maintainers must check for compliance with the license and the presence of the copyright information in the source code.

5.1.3 Inventory cleanup

The software repositories should be examined regularly for obsolete items and cleaned up. The chief maintainer organizes the cleanup. The following steps should be followed:

1. Identify obsolete items.
2. Inform developers about items to-be-removed via an E-Mail on the approver list. Developers can object the proposed removal of an item.
3. After the 2-week lazy consensus decision time frame, non-disputed items should be removed.

5.2 Decision-Making

The following decision process should be followed when consensus within the community is required. It should be noted that the decision-making process is by a *veto process*: community members can veto proposals; however, they cannot enforce others to do something.

In short, the general decision-making process is:

1. Proposal
2. Discussion
3. Decision

5.2.1 Proposal

Developers or partners must be notified about proposals that require community consensus. The following form should be followed:

- For small technical changes:
  Implicit notification through ticket or review system.
- For new features or large-scale refactoring:
  Letter of intent on approver list
- For non-technical topics:
  Proposal on partner and maintainer-readable mailing list

5.2.2 Discussion

The discussion of a proposal can move to another communication channel, e.g. the wiki.

5.2.3 Decision

By default, decisions are reached by lazy consensus. That is, proposals are implicitly accepted when there are no objections. The following rules apply:

- A minimum two-week timeframe should pass between proposal submission and acceptance by lazy consensus
- A reminder E-Mail with the text `[Consensus]` in the subject line and a summary of the proposal should be sent to the appropriate (approver or partner) mailing list two working days before the implicit proposal acceptance

When no consensus is reached (i.e., an approver or partner objects the proposal), the following rules apply:

- For proposals that affect a single component:
  The component maintainer decides
- For proposals that affect multiple components or no specific component:
  The chief maintainer decides
- For non-technical proposals:
  The spokesperson decides
- Any maintainer or partner can also request an explicit vote of the boards on the proposal. Then, approval of the proposal requires a simple majority in both boards.

5.3 Voting process

Votes in the maintainer and partner boards are required for role revocations and changes of the governance model. In addition, votes can be requested by a maintainer or partner for other proposals.

When a vote is required, the chief maintainer initiates and organizes the vote among the maintainers, and the spokesperson initiates and organizes the vote among the partners. The following rules apply:

- A maintainer has a single vote, even if he/she maintains more than one maintenance unit
- Votes are cast by E-Mail on the closed maintainer/partner mailing lists
  - Votes *for* a proposal should contain the text `[+1]` in their subject line
  - Votes *against* a proposal should contain the text `[-1]` in their subject line
  - Explicit abstains should contain the text `[0]` in their subject line
- The voting period lasts between two and four weeks, depending on the requirements
- Chief maintainer and spokesperson count the votes
  - Acceptance within a board requires the sum of +1/-1/0 votes to be greater than 0
  - The chief maintainer and spokesperson break ties in the maintainer and partner boards, respectively

5.4 Change of the governance model

Changes of the governance model require a vote in both boards. A 2/3 majority in each board must approve the change; other than that the regular voting rules apply.