Deliverable 7.6 Governing rules for the non-profit that will oversee Spine development after the project

Revision................................. 1
Preparation date ....................... 2021-04-07 (M43)
Due date ................................. 2021-03-31 (M42)
Lead contractor ......................... VTT

Authors:
Erkka Rinne .............................. VTT
Juha Kiviluoma ......................... VTT
Jody Dillon .............................. ER

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/4.0/.
T7.3 Exploitation and open sourcing planning (M1–M48)

Task leader: VTT; Participants: All

Central element of Task 7.3 is the deployment of the tools to potential users. Through the training and dissemination activities, the aim is also to build an active user community that will continue to use and develop the model also in the future. We will organize training workshops as side events of energy related conferences and workshops such as Wind and Solar Integration Workshops, EERA workshops and IEEE events. In addition, a workshop will be organized in Brussels in M45 as part of one the Spine workshops. Training is also provided through webinars.

Spine project will also produce exploitable results. The task leader together with the relevant project participants will process the case study results into concise conclusions that can be communicated to the stakeholders according to the planned communications (section 2.2.2 of the proposal). The task will also be responsible for carrying out the communications that the exploitation plan has set out. Discussions with policy makers, regulators, transmission system operators, energy companies and toolbox users. The task will also produce and publish replies to the questions posed by Spine Toolbox users. These efforts will be shared between the project partners depending on the expertise needed for each exploitation support.

The utilisation of Spine Toolbox after the project will be ensured through an open source community. The community will consist of Spine project members and other interested parties. The exploitation of Spine Toolbox will continue after the project through an open source community that will consist of Spine project members and all other interested parties. In order to secure and manage the continuous development effort, this task will setup a non-profit organization to oversee the development. All project partners will join the non-profit as initial members. The task will write a deliverable that establishes the governing rules for the non-profit and the rules of entry for new members. In order to assist the future development and maintenance of the model, contribution guidelines will be developed as a deliverable of this task by all project partners together. These will govern how new contributions are to be designed, documented and tested in order to be accepted to the master branch of Spine Toolbox.

All partners of WP7 will participate in deploying the open source tools whenever it concerns the work they are involved with. The open sourcing will include the following elements:

- Deploy documentation and rules for contributions to the open source tools
- Deploy test systems and contribution validation
- Make examples how Spine Toolbox can be used in energy system modelling in collaboration with some of the case studies – and disseminate them
- Generate charts showing how to access model dimensions together with WP3 – and deploy them
D7.6 Governing rules for the non-profit that will oversee Spine development after the project:

The report will set the rules of governance for the non-profit that will oversee Spine Toolbox development after the Spine project is finished. Success criteria: all partners accept the rules.

<table>
<thead>
<tr>
<th>Planned resources PM of T7.3</th>
<th>VTT</th>
<th>UCD</th>
<th>KUL</th>
<th>KTH</th>
<th>ER</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>1.0</td>
<td>17.0</td>
</tr>
</tbody>
</table>

**Comments**

<table>
<thead>
<tr>
<th>V</th>
<th>Date</th>
<th>Authors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2021-04-07</td>
<td>VTT</td>
<td>First draft for internal review</td>
</tr>
<tr>
<td>1.0</td>
<td>2021-04-23</td>
<td>VTT</td>
<td>After internal review, to be submitted</td>
</tr>
</tbody>
</table>

**Disclaimer**

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information as its sole risk and liability.

The document reflects only the author’s views and the Community is not liable for any use that may be made of the information contained therein.
Table of contents

Introduction .................................................................................................................................. 5
  1.1 General roles in open-source software development ........................................................... 5
  1.2 Typical governance models in open-source projects .......................................................... 5

Spine governance model ........................................................................................................... 6

Legal and IPR issues ................................................................................................................. 8

Conclusions .............................................................................................................................. 8

References ................................................................................................................................ 9

Appendix A Spine Governance Model .................................................................................... 10

Appendix B Voting ................................................................................................................... 14
INTRODUCTION

To ensure the continued development of the Spine software, it is important to set rules on how decision making is carried on after the H2020 project ends. Software is seldom finalised to the extent that no further development is required, and the ambition is that the results of the Spine project will live on as community-driven open-source projects. The chosen licensing (LGPL\(^1\)) allows for anyone to copy the source code and start building their own (non-commercial) product. Acknowledging this fact, we wish to encourage interested developers to participate in the project and make Spine software even better.

1.1 General roles in open-source software development

In open-source projects, anyone can contribute to the development of the software. However, it is important to note that contributions can also be in forms other than code, e.g. providing thought leadership and strategic direction, scientific contributions and development of methodologies, participating in discussion, filing bug reports, advocating or presenting the software or managing the repositories. All people involved in the development of a project in any way can be called contributors. [1] Usually not all contributors have direct access to the source code or other material and thus need to seek approval before contributing [2]. In the GitHub\(^2\) environment this is done using pull requests.

Those contributors with direct write access to the project material have more control – and responsibility – over the project. The term committer is sometimes used in this case [1]. In comparison to other contributors, committers should seek approval of their contributions from the community afterwards [3].

Maintainer is a broad term used for various roles in different projects. In some projects they are the only people with commit access. On the other hand, maintainers do not need to be someone who actively contributes code to the project but also someone who make the software more accessible to others. In any case, maintainers can be understood as being responsible for the organisation of the software and the repositories. [1]

1.2 Typical governance models in open-source projects

The most common governance structures used in open-source projects are the so called Benevolent Dictator For Life (BDFL), meritocratic governance and the liberal contributions models [1], [2]. A major part of the governance is to decide on who has access (see roles above) and how important or difficult decisions are made.

In the BDFL approach one person, usually the original developer, has the final say on all major decisions. Small projects are basically by default governed like this. [1]

In a meritocratic governance structure, decision making is based on voting by merited members of the community, and major guidance is provided by a Project Management Committee (PMC). All Apache Software Foundation\(^3\) projects are governed using a meritocratic structure [4].

\(^1\) GNU Lesser General Public License, https://www.gnu.org/licenses/lgpl-3.0.en.html

\(^2\) Spine source code is hosted at https://github.com/Spine-project

\(^3\) https://www.apache.org/
On the other end of the spectrum, projects with more liberal contributions do not have formal decision-making roles but instead rely on discussion and finding consensus among all active developers. Examples of liberal projects are Node.js\(^4\) and Rust\(^5\). \([1]\)

In any case, it is important to document openly how the project is governed and who are the responsible persons. This ensures also that new contributors can join the project and feel they have some ownership of it.

**SPINE GOVERNANCE MODEL**

Spine will be organised as two separate – although highly interrelated – projects: Spine Toolbox and SpineOpt. The Spine Toolbox project controls the Spine Toolbox application as well as its auxiliary packages Spine Engine, Spine Items and Spine Database API. The SpineOpt project develops the SpineOpt.jl modelling package as well as the SpineInterface.jl auxiliary package. See Figure 1 for an overview. The split into two projects allows to better target their relevant audiences. Spine Toolbox can be used for modelling in general while SpineOpt is a modelling tool specifically for energy systems research.

![Figure 1: Spine development is organised into two interrelated projects: Spine Toolbox and SpineOpt.](image)

A meritocratic governance model approach will be used in both projects. Meritocracy is chosen as the governance model to ensure wide enough representation from the original H2020 project consortium as well as to allow new contributors to join the development as time passes.

Generally, it is desirable to seek consensus on significant issues and every effort should be made to achieve this goal for the greater benefit of the community and the software itself. When this is not possible, the voting mechanism can be utilised where each community member can express their opinion, but only **voting members** have a binding vote in a project. A vote can be either an upvote (agree) or a downvote (disagree), or the person can also abstain from the vote.

\(^4\) [https://nodejs.org/](https://nodejs.org/)

\(^5\) [https://www.rust-lang.org/](https://www.rust-lang.org/)
Voting members have write access to the project repositories (cf. committers in Section 1.1), but access to certain important areas can be limited (e.g. the main branch). Some voting members can be nominated as maintainers who have additional responsibility and rights over the code repositories. Being a voting member is not an obligation but a right; there is no need to be active.

A Project Management Committee (PMC) oversees each of the projects and decides on voting members, governance rules, contribution guidelines and standards, and strategic choices. The PMC also resolves any disputes. Each PMC has a chair chosen among the committee. See detailed rules on decision making and voting in Appendix A. The first PMC members will be chosen from volunteers through consensus before the Spine H2020 project ends.

To ensure the continued compatibility of Spine Toolbox with the SpineOpt.jl modelling package, the SpineOpt voting members have a perpetual right to nominate one Spine Toolbox PMC representative if Spine Toolbox PMC does not have a member from the SpineOpt PMC.

Each main component (repository) of a project should have at least one named maintainer chosen by the PMC. Maintainers ensure the quality of code and other development practises. Other responsibilities and rights of the maintainers are decided by the PMC. Push access to the main development branch could be limited to maintainers only.

Repositories not falling directly under either of the projects as well as the common landing web page source code (spine-project.github.io) is managed together by the PMC’s (chairs or other representatives). All PMC members are GitHub organisation owners (administrators).

Spine Advisory Board (AB) follows Spine development as a whole. The board’s main function is to maintain informal collaboration of a strategic and scientific nature at the ‘Spine level’. It can make suggestions, but has no formal decision-making role in projects. Each original project partner organisation has a seat in the AB at the end of the Spine H2020 project. Additionally, external interested parties can be invited from e.g. NREL, G-PST, the Spine H2020 project Advisory Board etc. Members of the AB can invite others and membership is decided by consensus within the board. Other tasks of the board include organising (facilitating) annual events etc.

See Figure 2 for an overview of the different roles in the Spine community and how they overlap. Appendices A and B give more details on the governance structure and decision-making process.

Figure 2: Overview of community roles in Spine projects
GREECE

D7.6 Governing rules for the non-profit that will oversee Spine development after the project

Page 8 of 16

LEGAL AND IPR ISSUES

Copyright and ownership of the source code and other material stays with the original contributors. All contributors per project shall be listed for clarity. All source code will be licensed under LGPL version 3 or later and other material under Creative Commons Attribution-ShareAlike 4.0 International. A standing item in the AB agenda is to discuss and decide on the creation of a foundation or similar non-profit organisation to have financial resources for maintenance and development. The non-profit would also own the trademarks for Spine Toolbox and SpineOpt.

CONCLUSIONS

This deliverable describes the governing rules for Spine Toolbox and SpineOpt development after the H2020 project Spine has ended. It is to be decided by the Spine Advisory Board if a non-profit organisation is founded to support the development financially and in legal matters.

---

6 CC BY-SA 4.0, https://creativecommons.org/licenses/by-sa/4.0/
REFERENCES


Appendix A Spine Governance Model

Adapted from Meritocratic Governance Model by Ross Gardler and Gabriel Hanganu [3].

Overview
This is a meritocratic, consensus-based community project. Anyone with an interest in the project can join the community, contribute to the project 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.

Roles and Responsibilities

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 reasonable. 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):

- communicate 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. There are no special rules for becoming a contributor as anyone can contribute to the project.

Contributors are community members who contribute in concrete ways to the project. Anyone can become a contributor, and contributions can take many forms, as detailed in a separate document. 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 may 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 discussion board, 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 voting members (see next section). The developer discussion board 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 voting membership.

**Voting members** are community members who have shown that they are committed to the continued development of the project through ongoing engagement with the community. Voting membership allows contributors to carry on with their project related activities more easily 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 voting member is free to do what they want. In fact, voting members have no more authority over the project than contributors. While voting membership 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 voting member and a contributor is when this approval is sought from the community. A voting member 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. By the time a contributor is invited to become a voting member, they will have become familiar with the project’s various tools as a user and then as a contributor.

Anyone can become a voting member; 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 voting member 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 voting members can be nominated by any existing voting member. Once they have been nominated, there will be a vote by the project management committee (PMC; see below). Voting member voting is one of the few activities that takes place on the PMC’s private discussion board. 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 discussion board. 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 voting member. 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 voting member 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 voting membership 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 voting membership exists for as long as the voting member wishes to continue engaging with the project.

A voting member 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.
**Project Management Committee**

The project management committee consists of those individuals identified as ‘project admins’ on the development site. The PMC has additional responsibilities over and above those of a voting member. 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 voting members. It also makes decisions when community consensus cannot be reached. In addition, the PMC has access to the project’s private discussion board and its archives. This list is used for sensitive issues, such as votes for new voting members 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.

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 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.

The PMC chooses a **maintainer** for each project component repository. The maintainer takes care of code quality and ensures fluent development work. Detailed rights and responsibilities of the project maintainers are decided by the PMC.

**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.

**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 discussion forum and issue tracker or might supply patches. The various ways of contributing are described in more detail in a separate document.

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

**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 discussion board. Occasionally, sensitive discussion occurs on the private discussion board of the PMC.
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.

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 a message to the project discussion board, file an issue to the issue tracker or submit a patch (pull request) implementing the idea. 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.

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. Every member of the community is encouraged to express their opinions in all discussion and all votes. However, only project voting members have binding votes for the purposes of decision making. See Appendix B for detailed rules on voting.
Appendix B Voting

Adapted from *Voting In Meritocratic Governance Projects* by Gabriel Hanganu [5].

If a formal vote on a proposal is called (signalled by sending a message with ‘[VOTE]’ in the title line), all project contributors may express an opinion and vote. They do this by sending a message in reply to the original ‘[VOTE]’, 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. 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 voting 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 not all members may have a binding vote, a well-justified ‘–1’ from a non-voting member 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.
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.

Table 1: Types of approval

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazy consensus</td>
<td>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.</td>
<td>N/A</td>
</tr>
<tr>
<td>Lazy majority</td>
<td>A lazy majority vote requires more binding +1 votes than binding –1 votes.</td>
<td>72 hours</td>
</tr>
<tr>
<td>Consensus approval</td>
<td>Consensus approval requires three binding +1 votes and no binding –1 votes.</td>
<td>72 hours</td>
</tr>
<tr>
<td>Unanimous consensus</td>
<td>All of the binding votes that are cast are to be +1 and there can be no binding vetoes (–1).</td>
<td>120 hours</td>
</tr>
<tr>
<td>2/3 majority</td>
<td>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).</td>
<td>120 hours</td>
</tr>
</tbody>
</table>

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.

Table 2: Action requiring a vote

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
<th>Approval type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release plan</td>
<td>Defines the timetable and actions for a release. A release plan cannot be vetoed (hence lazy majority).</td>
<td>Lazy majority</td>
</tr>
<tr>
<td>Release</td>
<td>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).</td>
<td>Lazy majority</td>
</tr>
<tr>
<td>New voting member</td>
<td>A new committer has been proposed.</td>
<td>Consensus approval of the PMC</td>
</tr>
<tr>
<td>New PMC member</td>
<td>A new PMC member has been proposed.</td>
<td>Consensus approval of the community</td>
</tr>
<tr>
<td>Voting member removal</td>
<td>When removal of voting membership is sought.</td>
<td>Unanimous consensus of the PMC</td>
</tr>
<tr>
<td>PMC member removal</td>
<td>When removal of PMC membership is sought.</td>
<td>Unanimous consensus of the community (excluding the PMC member to be removed)</td>
</tr>
</tbody>
</table>