Deliverable 7.8 Development roadmap to guide development after the project

Revision 1
Preparation date 2021-09-30 (M48)
Due date 2021-09-30 (M48)
Lead contractor VTT

Authors:
Juha Kiviluoma VTT
Maren Ihlemann KU Leuven

Dissemination level

<table>
<thead>
<tr>
<th>PU</th>
<th>Public</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
<td></td>
</tr>
</tbody>
</table>
Deliverable administration

<table>
<thead>
<tr>
<th>No &amp; name</th>
<th>D7.8 Development roadmap to guide development after the project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Due Date</td>
<td>M48</td>
</tr>
<tr>
<td>Date</td>
<td>2021-09-28</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Juha Kiviluoma VTT</td>
</tr>
<tr>
<td></td>
<td>Maren Ihlemann KU Leuven</td>
</tr>
<tr>
<td>Reviewer(s)</td>
<td>Manuel Marin KTH, Erkka Rinne VTT, Pekka Pääkkönen VTT</td>
</tr>
</tbody>
</table>

Description of the related task and the deliverable. Extract from DoW

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.

... 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.

....

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
- Place code and case studies in public servers (also conversion tools and primary data when possible)

This task will establish a non-profit organization that will oversee the future development of Spine Toolbox and Spine Model [now SpineOpt].

D7.8: Development roadmap to guide development after the project [48]

(T7.3) The roadmap will plan the next steps in the development of the Spine Toolbox after the Spine project ends. Success criteria: the partners agree on the roadmap in the report and the roadmap covers all the major tools in the Spine toolbox.

<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>1.0</td>
<td>2021-09-28</td>
<td>VTT</td>
<td>Submitted deliverable</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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1. Roadmap for Spine Toolbox</td>
<td>5</td>
</tr>
<tr>
<td>2. Roadmap for SpineOpt</td>
<td>9</td>
</tr>
</tbody>
</table>
**INTRODUCTION**

Spine project had a goal to build both a workflow, data, and scenario tool as well as a new flexible energy systems model. While both have been achieved, there have also been many lessons. This report outlines possible improvements to both of those tools.

Even though this is a roadmap, it does not try to rank different improvements into an order of priority. What will be implemented next will depend on the future developers, their needs, obligations and resources. Prioritization is therefore their task and will be discussed in future Project Management Committee meetings (see D7.6 Spine Governance).

The roadmaps for Spine Toolbox and SpineOpt have been divided into categories to make it easier to locate possible future features. For the most part, there is a link to an issue in the GitHub repositories with much more discussion on each feature. This document acts as a reference for the near-term. Over time it will grow old and the reader is better off by going directly to the repository and looking for issues using labels like ‘Roadmap’, ‘Discussion’, and ‘Feature’. The issue tracker for Spine Toolbox can be found at [https://github.com/Spine-project/Spine-Toolbox/issues](https://github.com/Spine-project/Spine-Toolbox/issues) while the same for SpineOpt is at [https://github.com/Spine-project/SpineOpt.jl/issues](https://github.com/Spine-project/SpineOpt.jl/issues).

1. **ROADMAP FOR SPINE TOOLBOX**

- Portability of Spine Toolbox projects
  - Toolbox capable of maintaining programming environments (Python, Julia etc.)
    - [https://github.com/Spine-project/Spine-Toolbox/issues/1037](https://github.com/Spine-project/Spine-Toolbox/issues/1037)
  - Which project files should be included in version control and which should not. This is difficult for the user and could be improved.
  - Projects connect to the plugins needed by the project.
  - Making a ‘label’ that defines a project resource, so that we can handle missing / broken links to files more graciously.
  - Differentiating between shared and local tool specifications:
    - [https://github.com/Spine-project/Spine-Toolbox/issues/1102](https://github.com/Spine-project/Spine-Toolbox/issues/1102)
  - Separation of project.json into functional and visual parts for easier version control of project parts.
    - [https://github.com/Spine-project/Spine-Toolbox/issues/1238](https://github.com/Spine-project/Spine-Toolbox/issues/1238)
  - Remove dependency on the location of a local Conda environment
    - [https://github.com/Spine-project/Spine-Toolbox/issues/1490](https://github.com/Spine-project/Spine-Toolbox/issues/1490)

- Data store views
  - Remember layouts in the different views
  - Find and replace functionality across the data store
More keyboard functionality for power users
https://github.com/Spine-project/Spine-Toolbox/issues/224

Reorder entities in the Spine DB editor
https://github.com/Spine-project/Spine-Toolbox/issues/691

Greying out entities that are not in the selected scenario or alternatives

Showing the commit history and metadata better
https://github.com/Spine-project/Spine-Toolbox/issues/123

Graph view specific

- Some ideas how to improve:
  https://github.com/Spine-project/Spine-Toolbox/issues/180
- Allow to choose the depth of the graph network in the graph view.
  https://github.com/Spine-project/Spine-Toolbox/issues/1493
- Having defaults in classes that affect how the entities of that class are treated in the graph views could also facilitate more useful graph views. For example, report entities would not be included unless specifically selected.
- Allow cloning entities in the graph view
  https://github.com/Spine-project/Spine-Toolbox/issues/1492

Pivot (and plotting) specific

- Enhancements for organizing the axes in the pivot view and allowing more selection options.
  https://github.com/Spine-project/Spine-Toolbox/issues/1126

List view specific

- Showing entities with no parameter data in the list view.
  https://github.com/Spine-project/Spine-Toolbox/issues/1211

Database editor

- We have a working database editor, but it is still somewhat insufficient when editing multi-dimensional maps. The editor could move towards pivot table like capabilities, but there are many complications with multi-dimensional data that need to be designed and implemented. Some ideas in:
  https://github.com/Spine-project/Spine-Toolbox/issues/943
- Case insensitivity could be used when it makes sense
  https://github.com/Spine-project/Spine-Toolbox/issues/776

Workflow view

- Making the use of workflow resources more obvious to the user
  https://github.com/Spine-project/Spine-Toolbox/issues/1519

Visualization

- Using graph view for showing some results
  https://github.com/Spine-project/Spine-Toolbox/issues/180
- Improve our Matplotlib-based plotting capabilities
- Link with visualization packages like Bokeh or Seaborn or Plotly
- GIS data for entities. It would enable geographic plotting, using e.g. geoplotlib.
- Fix inconsistencies in UIs across Toolbox (improve also the UI guidelines)
- Legends need improvements (can fill the whole canvas)

Multiple executions

- Control how many executions are made at once
- Individual execution logs work fine, but the master Event Log provides very little information on the different executions taking place and the phase of the execution (could have links that would take to the individual executions)
Keep sub-processes running to make re-executing a tool faster

Remote executions

We have remote execution based on a server–client model. However, there are difficulties in setting this up and there could be improvements in the way a project and its data are transferred and kept up-to-date. At the moment, the whole project needs to be duplicated, but it may be that only part of the workflow is actually executed remotely. Also the portability of the projects is discussed separately above. Part of this is that the server would need to be able to install plug-ins.

The server side does not need Spine Toolbox – only executable parts of Spine Items, SpineDB API and Spine Engine.

There should be more testing on how Spine Engine can and should access resources in different locations like MySQL server, SharePoint or other cloud file systems.

Scenarios, alternatives, recipes

We now have a scenario generation tool, but there is more to the idea of recipes that can be implemented.

Alternatives could have a history. Additional dimension in alternative to keep history for results and also would allow to keep history of changes in the input data. It could be a user choice to have saving history on or off. There should also be an easy way to purge history. The implementation could use the existing parameter_value table (consider commit_id) or a new table.

Selection of alternatives for the runs (can be done in the workflow as it is, but when there is lots of scenarios, it becomes not very functional)

Scenarios from multiple databases. At the moment it is safest to work from a single database with scenarios, but users could have multiple databases with different alternatives and it is not clear how that would work. See and the discussion that follows.

Create new alternative names automatically when importing.

Choosing entities for scenarios

Currently we can leave entities (objects or relationships) out from a scenario using the feature/tool functionality. It can be used to create a parameter like ‘is_active’ to filter the entities. This is still a bit insufficient and alternative implementations should be considered. The current system could be improved through interface changes. The membership in a scenario could be made into a more fundamental feature of the data structure.

Data processing speed

Adding metadata about array and map sizes, shapes and data types could help to improve showing data. Currently there are cases where data needs to be parsed in order to find out its structure and it can make the views slower.

Consider using more SQL capabilities in fetching data (currently we create a cache of the data structure by fetching all queries fully). Indexed queries and SQL caches could be faster than caching in Python.
○ Consider re-writing some of the bottleneck functions with a compiled lower level language like C++ or Rust.
○ In SpineInterface.jl a pure Julia approach is also possible (to avoid PyCall), but would require re-making SpineDB functionality. If most functions were written in lower level language, this should also become more feasible.
○ Consider filtering by object class when fetching objects.  
https://github.com/Spine-project/Spine-Toolbox/issues/1516
○ Database manager code structure.  
https://github.com/Spine-project/Spine-Toolbox/issues/1512
○ Consider moving data integrity checks to SQL.  
https://github.com/Spine-project/Spine-Toolbox/issues/1515

**Importer**
○ Choosing what the importer does when there is existing data (currently overwrites)  
https://github.com/Spine-project/Spine-Toolbox/issues/1091
○ Clarifying how alternatives are treated in the import process  
https://github.com/Spine-project/Spine-Toolbox/issues/1141
○ Filtering what is imported  
https://github.com/Spine-project/Spine-Toolbox/issues/1323
○ Improve importing of empty cells  
https://github.com/Spine-project/Spine-Toolbox/issues/1325
○ Improved support for importing JSON data  
https://github.com/Spine-project/Spine-Toolbox/issues/468

**Exporter**
○ The regex filtering works nicely, but it is not very intuitive to use. We could add some user friendliness.  
https://github.com/Spine-project/Spine-Toolbox/issues/1367

**Plugins**
○ Installing plugins with Git.  
https://github.com/Spine-project/Spine-Toolbox/issues/1397
○ Make more plugins.  
https://github.com/Spine-project/Spine-Toolbox/issues/1256

**Data**
○ Time patterns were suggested by the Advisory Board. We do have time patterns, but there is more to be done in order to improve their support in the APIs.
○ GIS data could be used by tools that would create e.g. possible network pathways for the planning models.

**Metadata**
○ We have a working metadata system, but it has limitations. The database structure allows metadata items to be freely connected with entities and parameter values. There is a link with commit as well. However, metadata can only be shown in Spine Toolbox interface and not edited. Furthermore, importing metadata has limitations that makes it difficult to take full use of the available metadata structure.
○ Metadata could be automatically generated for commits (who did, using what tool, etc.).
○ Editing metadata during imports:  
https://github.com/Spine-project/Spine-Toolbox/issues/673
○ Capability to add metadata links to commits:  
https://github.com/Spine-project/Spine-Toolbox/issues/670
To utilize metadata from Data Packages: https://github.com/Spine-project/Spine-Toolbox/issues/645

Archetype
- SpineOpt has implemented a light-weight Archetype through the use of templates (see below in the SpineOpt roadmap). However, archetypes, which pre-define functionality for a new instance of a particular entity, could be a Toolbox level functionality. This has been extensively discussed during the project, but lacks implementation. https://github.com/Spine-project/Spine-Toolbox/issues/134 https://github.com/Spine-project/Spine-Toolbox/issues/53

Hierarchical classes and entities
- Along the way, we have considered hierarchical classes that would allow inheritance of parameter definitions and hierarchical entities that would allow inheritance of parameter values, but we haven’t implemented either. There are question marks whether they would be beneficial, since the system already has many degrees of freedom and hierarchies could make it too confusing. https://github.com/Spine-project/Spine-Toolbox/issues/186

Documentation
- Add tutorials on how to use also other tools besides SpineOpt.

2. ROADMAP FOR SPINEOPT

Archetypes
- While we have implemented an initial support for archetypes in the form of JSON templates (see folder /templates/archetypes in the repository), more advanced ideas have been heavily discussed throughout the project, considering for example a validation layer that enforces and/or validates the existence of certain (method) parameters, graphical user interface support for an archetype template, and inheritance. This also in parts relates to the relevant Spine Toolbox issues (see above)
  - Archetype methods as parameters or relationship classes: https://github.com/Spine-project/SpineOpt.jl/issues/302

Investments
- SpineOpt supports investments in all its fundamental (physical) entities, i.e. nodes, connections and units. Theoretically this would also allow investments in pressure-driven gas networks and (nodal) DC power flow networks, but is still experimental and needs some more testing. Further, we want to include technology degradation over time (e.g. decreasing efficiencies over the
lifetime). Another step in the roadmap is the representation of long-term storages in the decomposed investment problems.

- Investments in pressure-driven gas pipelines
  [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/358)

- Investments in lossless node-based DC power flow networks
  [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/359)

- Investments with technology degradation
  [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/325)

- Optimisation of long-term storages in decomposed problems:
  [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/211)

### Model linking
- Before we implemented Benders decomposition, the infrastructure for model linking and decomposition was implemented. It is possible to have two (or more) model objects within one database, as exemplified by the (decomposition) algorithm in SpineOpt. However, to this day, SpineOpt only supports Benders decomposition for investment problems. We want to extend this in the future to also support decomposition techniques such as Dantzig–Wolfe and progressive hedging. Therefore, we will also revisit the code structure to facilitate the addition of new algorithms
  - Multi-model architecture/ linking of related optimization
    [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/318)
  - Code requirements/place-holder Dantzig–Wolfe decomposition
    [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/183)

### Style and performance
- An ongoing goal is to improve compilation times for the SpineOpt module, also to ensure competitiveness of SpineOpt
  [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/178)

### Validate
- During the project we have developed a large set of so-called ‘unit tests’ to validate code and ensure that code additions do not cause conflict with the existing functionality. In order to also capture more interactions between different functionalities, we want to use some of our publicly available case studies for continuous testing and validation. This means that our case study repositories would be kept up to date, facilitating the entrance for new users.
  - Validation through publicly available case studies:
    [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/383)

### Output
- We can currently report outputs for each objective function term separately and of course for each decision variable. In the future, we want to enable more options for the output of the objective function, e.g. the option to choose which parts of the rolling window should be reported at which resolution. Writing results back in the form of a Map could facilitate this process.
  - Adding objective output functionality:
    [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/313)
  - Results as Map parameter
    [GitHub](https://github.com/Spine-project/SpineOpt.jl/issues/195)

### Temporals
- The temporal structure of SpineOpt is very flexible, but we could implement a so-called static temporal block in the future (co-existing with rolling temporal blocks). Furthermore, we want to give the user more convenience in terms of defining the temporal structure database side.
Implementation of static temporal blocks:

Model start/end & temporal block convenience:

Commodity physics networks

- Power Transfer Distribution Factors (PTDF) and Line Outage Distribution Factors (LODF) network representations have been implemented and tested in SpineOpt. This functionality is currently limited to one PTDF/LODF network per model, but should be generalized to support possibly a manifold of PTDF networks side by side
  - Multiple commodity physics networks side-by-side

User constraints/input expression

- We support a so-called ‘user constraint’ in SpineOpt, which allows the user to add new constraints based on data. Here are some ideas to make their use more convenient and also to take their capabilities further.
  - Modernize model extension
  - Specify variables in db; autocode
  - Symbols readout
  - Advanced user constraints

SpineInterface

- SpineInterface was developed as a Julia wrapper over Spine database API and SpineOpt. Ideas for improvements relate to advanced functionalities for time series arithmetics
  - Time dependent data arithmetics:

SpinePeriods

- SpinePeriods is a supplementary package to SpineOpt, developed to provide (optionally ordered) representative days to SpineOpt. An idea for future improvements is that SpinePeriods could automatically determine a suitable temporal resolution
  - Finding optimal resolution for representative days:
    https://github.com/Spine-project/SpinePeriods.jl/issues/8