Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

# Spine Toolbox: A flexible open-source workflow management system with scenario and data management

Juha Kiviluoma [d], Fabiano Pallonetto [a,b,*], Manuel Marin [c], Pekka T. Savolainen [d], Antti Soininen [d], Per Vennström [d], Erkka Rinne [d], Jiangyi Huang [d], Iasonas Kouveliotis-Lysikatos [c], Maren Ihlemann [e], Erik Delarue [e], Ciara O'Dwyer [a], Terence O'Donnel [a], Mikael Amelin [c], Lennart Söder [c], Joseph Dillon [f]

[a] *UCD Energy Institute, Ireland*
[b] *Maynooth University, School of Business, Ireland*
[c] *KTH – Royal Institute of Technology, Sweden*
[d] *VTT – Technical Research Centre of Finland, Finland*
[e] *KU Leuven, Belgium*
[f] *Energy Reform Ltd, Ireland*

A R T I C L E I N F O

*Article history:*
Received 30 September 2021
Received in revised form 4 December 2021
Accepted 15 December 2021

*Keywords:*
Workflow management
Data management
Modelling
Optimisation
Open source
Scenario

A B S T R A C T

The Spine Toolbox is open-source software for defining, managing, simulating and optimising energy system models. It gives the user the ability to collect, create, organise, and validate model input data, execute a model with selected data and finally archive and visualise results/output data. Spine Toolbox has been designed and developed to support the creation and execution of multivector energy integration models. It conveniently facilitates the linking of models with different scopes, or spatio-temporal resolutions, through the user interface. The models can be organised as a direct acyclic graph and efficiently executed through the embedded workflow management engine. The software helps users to import and manage data, define models and scenarios and orchestrate projects. It supports a self-contained and shareable entity-relationship data structure for storing model parameter values and the associated data. The software is developed using the latest Python environment and supports the execution of plugins. It is shipped in an installation package as a desktop application for different operating systems.

## Code metadata

| | |
|---|---|
| Current code version | v0.6.1 |
| Permanent link to the repository used for this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-21-00187 |
| Spine Toolbox | https://github.com/Spine-project/Spine-Toolbox/releases/tag/0.6.5-final.0 |
| Legal Code License | LGPL 3.0 |
| Code versioning system used | Git |
| Software code languages, tools, and services used | Python, Qt, SpineOpt: Julia |
| Compilation requirements, operating environments | Linux, Mac, Windows |
| If available Link to developer documentation/manual | https://spine-toolbox.readthedocs.io/en/latest/?badge=latest |
| Support email for questions | spine_info@vtt.fi |

## Software metadata

| | |
|---|---|
| Current software version | v0.6.5 |
| Permanent link to executables of this version | https://github.com/Spine-project/Spine-Toolbox/releases/tag/0.6.5-final.0 |
| Legal Software License | LGPL 3.0 |
| Computing platforms/Operating Systems | BSD, Linux, OS X, Microsoft Windows, Unix-like |
| Installation requirements & dependencies | Python, QT/PySide |
| If available, link to user manual - if formally published include a reference to the publication in the reference list | https://spine-toolbox.readthedocs.io/en/latest/?badge=latest |
| Support email for questions | spine_info@vtt.fi |

* Corresponding author at: Maynooth University, School of Business, Ireland.
*E-mail address:* fabiano.pallonetto@mu.ie (Fabiano Pallonetto).

https://doi.org/10.1016/j.softx.2021.100967

## 1. Motivation and significance

Modelling and simulation are crucial methods for many complex scientific and engineering endeavours. Models often utilise various sources of input data, some of which requiring complex processing before the data is model compatible. Many topics are multi-faceted and multi-scale, requiring several models and processing tools to be adequately represented. Finally, uncertainties can be diverse yet critical for robust decision making. Tackling these intricacies in a repeatable and dependable fashion requires not just reliable models, but also reliable workflow management that can deal with the versatile processes and data.

There are several existing open source workflow management tools, each with their strengths and weaknesses. Some of the present authors are experienced power and energy system modellers who understand the modelling workflow challenges in their domain. One specific challenge is to manage not just the execution of the tools but also to represent manifold data and scenarios. Most generic workflow management tools do not have explicit means to represent data as part of the workflow, and this was a focal point for Spine Toolbox, which will be described further below. At the same time, workflow tools (or 'modelling frameworks') in the energy domain typically fall short in their capabilities to orchestrate complex workflows. In the next paragraphs we will highlight these aspects using a diverse sample of workflow tools sourced both from the generic domain and from the energy systems specific domain. For a more comprehensive review, Atkinson et al. [1] give an overview of workflow management tool development over the past ten years, while Ringkjøb et al. [2] provide an overview of recent energy system modelling tools.

Workflow management tools (or systems/software) can be used to facilitate the design and execution of workflows. Many of them are geared towards business processes. However, there are also some open source tools that focus on research and analysis workflows. Pegasus Workflow Management System (WMS) [3] is an execution framework built with Java and Python that allows the users to define workflows as processing plans built from a selection of components. Pegasus, or other similar frameworks, can be enriched with semantic workflow structures like Workflow INstance Generation and Selection (WINGS) [4]. This facilitates the incorporation of more metadata and guidance for the user, and allows the developers to chain components into ready-made template workflows. However, Pegasus only transfers data 'replicas' from the data 'catalogue'. It does not offer a unified interface to the data, and the capability for the user to create scenarios based on alternative parameter values, which was one of the design criteria for our purposes. Similarly, Apache Airflow [5], which executes Directed Acyclic Graphs (DAG) in pure Python, does not offer data interfaces. It does not support passing data between tools in the DAGs, which is relevant for validating the data exchange across different energy systems models. There are also tools with a data analytics focus like the Java-based KNIME [6] and the Scala-based Apache Spark [7]. These also lack the capability to easily build scenarios that can be executed in optimisation and simulation models. However, they have powerful data cleaning and processing capabilities and could be incorporated in a Spine Toolbox -based modelling chain providing the first steps.

In the energy system domain, many modelling frameworks have their own system of managing data, which is often quite rigid and limited to the immediate inputs and outputs of the tool (e.g. TIMES [8], OSeMOSYS [9] and Calliope [10]). In some cases, the modelling community has adopted lower level script chaining tools - like Snakemake [11] in the case of PyPSA [12]. This allows for much more freedom(providing an API), but requires that the user understands the code if workflow changes are needed.

The design criteria for Spine Toolbox was largely drawn from the requirements of modelling which to supports decision making under uncertainty. As a consequence, Spine Toolbox offers not just workflow management, but also versatile data structures and scenario management. The users can have multiple models and tools using the same database or databases, which allows groups of modellers to perform integrated scenario analysis using a suite of specialised tools. As illustrated in Table 1, the characteristics of Spine Toolbox are similar to data analytics workflow management systems such as KNIME [6] or Alteryx [13]. However, in comparison with these widely adopted tools, the novelty of Spine resides in a set of features specialised for decision making under uncertainty. Furthermore, Spine Toolbox is written in Python, which facilitates the easy integration of Python based tools that are widespread in the research community. On the other hand, Spine Toolbox is a new entrant and lacks many specific data processing capabilities present in the more mature tools. However, Spine Toolbox workflows can easily incorporate other data processing tools available in the open source community.

## 2. Software description

Spine Toolbox is an open source Python package to manage data, scenarios and workflows for modelling and simulation. Users can have local workflows, but work as a team through version control and SQL DB. The major features of Spine Toolbox are the following:

- It implements an application which enables and orchestrates the data acquisition from multiple (and diverse) data sources and provides the mechanisms to validate and associate those data.
- It provides a generic data model, the Spine Data Structure, using a generic Entity–Attribute–Value approach with Classes and Relationships implemented as SQL databases through SQLAlchemy [17]. The abstract classes and the relationships between them enable the formulation of diverse models in an object-oriented manner.
- Entities can hold parameters that can be constants, time series, arrays and multi-dimensional maps. The interface facilitates the viewing and editing of data.
- Scenarios can be built from alternative parameter values. It provides all the required interfaces for performing calculations on the data.
- There are importer, exporter and data manipulation tools that allow conversion of data between different data structures and formats (e.g. csv, xlsx, gdf, sql). This facilitates a wide range of possible analyses.
- A Python based API is available for querying the Spine data. A similar interface is available for Julia [18] with additional capabilities for direct model building in Julia/JuMP [19].
- Tool specifications are used to define the requirements for the execution of various tools.
- Tool specifications can be turned into plug-ins. An example plug-in is the *SpineOpt.jl* Julia package, which operates on a Spine dataset, by generating and simulating the optimisation model.
- Execution has been separated from the interface, which enables parallelisation of the workflow and scenarios as well as remote execution using the inbuilt server–client system.

### 2.1. Software architecture

Spine Toolbox can be used to design and execute workflows consisting of multiple tools and models [20]. It supports problem independent data and scenarios with user interfaces that

**Table 1**
Analysis of the offered functionality of commercial/open source packages versus the Spine solution.

| | Input model as data | Parallel execution | Workflow visual editing | Data conversion capabilities | Integration with external API | Open source community led | Remote workflow execution infrastructure | Support for optimisation engines | Support for open data format | IoT/Real time data stream integration | Extendable modelling framework | Multi language support | Components marketplace | Access to 3rd party components |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Spine Toolbox | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | (✓) | ✓ |
| KNIME [6] | ◇ | ✓ | × | ✓ | × | (✓) | (✓) | × | (✓) | × | × | ✓ | ✓ | ✓ |
| Alteryx [13] | ◇ | ✓ | ✓ | ✓ | ✓ | × | × | × | (✓) | (✓) | × | ✓ | ✓ | × |
| Airflow [14] | × | ✓ | (✓) | × | ✓ | (✓) | × | × | (✓) | × | ✓ | × | × | ✓ |
| Apache Spark [15] | ◇ | ✓ | × | ◇ | (✓) | ✓ | (✓) | × | (✓) | × | × | ◇ | ◇ | ✓ |
| OSeMOSYS [9] | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | × | ✓ | × | ✓ | × | × | × |
| Calliope [10] | × | × | ✓ | ✓ | × | ✓ | ✓ | × | ✓ | × | ✓ | (✓) | × | × |
| PyPSA [16] | ✓ | ✓ | ✓ | ✓ | (✓) | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | × | × |

*Legend:* ✓ Fully supported, (✓) Partially supported, ◇ Supported with additional software or a plugin, × Not supported.

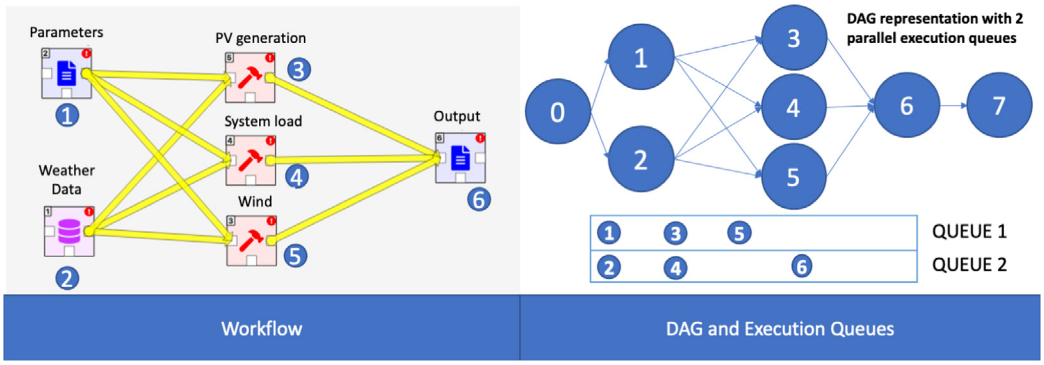

**Fig. 1.** The Spine Toolbox workflow is composed of 3 optimisation models (red icons with the hammers) and 1 Spine Data tool, and two input/output nodes. The goal of the workflow is calculating system energy demand and electricity generation from two renewable energy generators using a weather file. The results of the models are stored in an output node (a database or CSV file), and the workflow can be executed in parallel with two execution queues (on the right).

automatically support different data structures as long as they conform with the entity–attribute–value with relationships and classes data model. The user interface is built using the Model-View-Controller architectural pattern. The high level structure strives for modularity, both inside the Spine Toolbox, but also for the workflows it can execute. For the workflows, an equivalence between a composite model and a DAG computational workflow is assumed, where each node has four main elements: an input from the previous node, an output to the successor, some internal operations (workflow step) and an access to external data sources. As illustrated in Fig. 1, a computational node needs to receive the input from the predecessor node, or an empty value from the root (node 0) and during the execution interact with the external data source, both in reading and writing mode. At the end of the execution, the node will push the output data to the successor node. The composition of various nodes will result in a computational workflow equivalent to a DAG. The software architecture of Spine follows the workflow control architectural pattern, using a tight integration [21] between the toolbox and the other components (described in the following sections).

Fig. 2 illustrates the interaction between the Spine Toolbox components and the plugins developed during the project. All the Graphical User Interface (GUI) tools are embedded and available through the Spine Toolbox application. The Spine Data Structure is accessible through the toolbox that also exchanges data with the Spine Engine at the execution level. Spine Interface links the SpineOpt optimisation model with the Spine data structure. The following subsections describe each of the main components of the software in more detail.

### 2.1.1. *Spine data structure*

The Spine data structure is an entity–attribute–value with classes and relationships data model for the structured yet flexible storage of data. The classes and relationships define the structure between different data elements with a strong resemblance to graphs. Graph-like structures are prevalent in modelling and optimisation, which makes the data structure well suited to the intended purpose. The data structure is further augmented with the capability to contain multiple alternative values for the same parameter (i.e. attribute). Sets of alternatives can then be used to create scenarios. The data structure is an integral part of Spine Toolbox because it enables the users to work with a single dataset serving multiple models, thus enabling an efficient, repeatable and more reliable way of working.

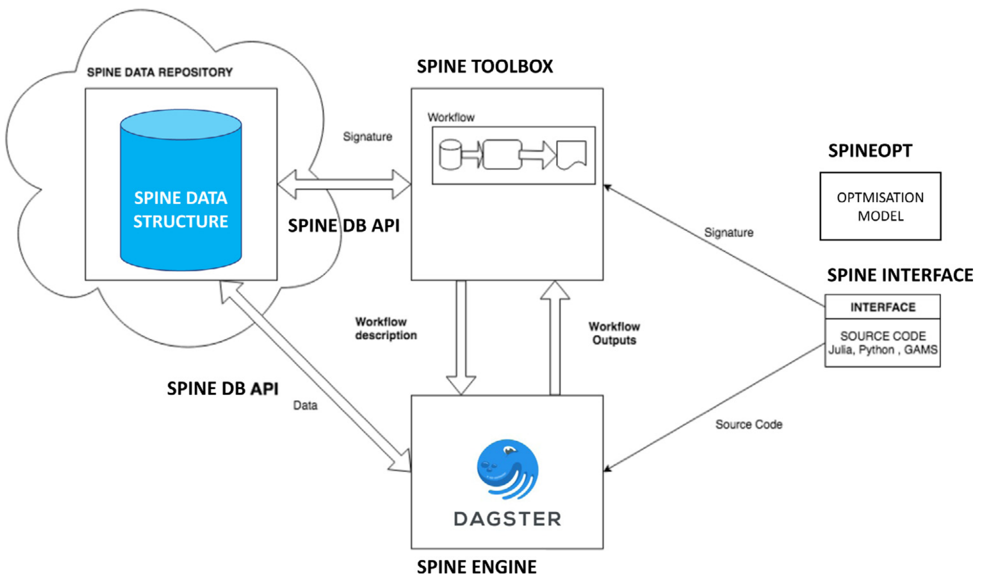

**Fig. 2.** Overall Spine Software architecture with the different components interacting for the composition and execution of workflows.

## 2.2. Spine DB API

Spine DB API holds the low-level Python functions that can access and modify all the different parts of the Entity–Attribute–Value data structure while maintaining structural integrity. Spine's inbuilt importer, exporter, data manipulators and SpineInterface.jl all use Spine DB API. Users can create their own direct connections to the Spine database using the Spine DB API.

### 2.2.1. Spine toolbox user interface

Spine Toolbox provides the GUI application that enables the definition, management, and execution of energy system models. It gives the user the ability to collect, create, organise, and validate model input data, execute a model with selected data and finally archive and visualise results/output data. Spine Toolbox is designed to support the creation and execution of optimisation and simulation models as well as data processing tools. It can also be used for other applications that can take advantage of the inbuilt data structure.

## 2.3. Spine DB editor

Part of the GUI of Spine Toolbox is the Spine DB editor. The editor allows users to view and modify data, add alternative data values, build scenarios and view metadata. It has an undo functionality and data changes are submitted through a commit that asks for a commit message. The editor consists of four main components:

**Entity tree** shows entities organised according to the object and relationship classes and allows filtering in the following data views.

**List view** represents data as a list of entities and their parameter values. It is the most straightforward interface for viewing and manipulating both the entities and the parameter values. A version of the list view also shows the available parameter definitions for the classes and facilitates their editing.

**Tabular view** provides a table which can be used to view and manipulate parameter values of a given entity class. The user controls the data that the two axes contain, which can also be filtered. The axes can take structural dimensions (classes), index dimensions from the parameter values as well as alternative and scenario dimensions.

**Graph view** provides a visual presentation of the data structure. The various entities of a Spine dataset are represented as nodes, with vertices signifying the relationships between them. The end user can choose classes and entities to be displayed. Parameter data for selected entities can be seen and manipulated in a separate list view below the graph.

### 2.3.1. Spine engine

Spine Engine provides the functionality for workflow execution. The objective of this component is to execute a part of the workflow or the whole workflow either locally or using a client–server setup.

During the workflow execution, each of the items in the workflow are executed and their outputs passed to the successor node(s). The main inputs of the Spine Engine are items, specifications, successors, and execution permits. These inputs are parsed from a project file which describes the workflow to be executed. The project file is parsed before the Spine Engine object is instantiated, allowing for the relevant inputs to be made available to the Spine Engine. The inputs for the Engine are described as:

An **Item** in the context of a Spine workflow is an object of computation which can be connected with other items to form a workflow. The Spine Toolbox provides the interface where the user can choose the relevant items to construct the desired workflow. Each item available in the Toolbox interface should provide a distinct but complimentary functionality to other available items. The functionality of these items includes the execution of program files (with direct support for Python, Julia, and General Algebraic Modeling System (GAMS) code while other tools can be run through shell executables), importing data, exporting data, storing data, executing Jupyter notebooks [22], and more.

**Tool specifications** describe item attributes such as input files, output files, executable locations, etc. They are instantiated as items in the workflow.

**Successors** are mappings from an item name to a list of successor item names which describe the dependencies between items in the workflow.

**Execution Permits** are mappings of an item name to a Boolean value, describing which items in the workflow should be executed. If the Boolean is false then the item will not execute but its resources will be collected. The interface allows the selection of any number of items for execution.

Based on the aforementioned inputs, the engine will construct a set of objects called solids. Solids are defined by the items belonging to the workflow. From the set of solids, a pipeline is constructed. The use of “solids” and “pipelines” in describing the inner workings of the Spine Engine is derived from the Dagster [23] library, which the engine utilises to efficiently execute a workflow. Once the pipeline is constructed it is available for execution.

Workflow execution is performed in parallel with items in the workflow being executed concurrently where item dependencies allow. Parallelisation of workflow execution is widely used by established workflow engines such as openDIEL [24] and Chiron [25] to decrease the execution time of a workflow. However, Spine Engine can also parallelise the execution of scenarios that have been built in the Spine DB manager. With the growing availability of multi-core CPU’s for desktops and laptops, most user hardware should be capable of taking advantage of this Spine engine feature.

The engine also offers two modes of execution, the first of which is performed from within the Spine Toolbox interface. When the engine is executed from the toolbox the user is provided with a log of item executions along with animations for items, which provides the user with a visual representation of item execution progress. The second mode is performed in a headless manner where the execution is performed from the command line, completely separately from the user interface. When executed in the headless mode, the user is provided with the execution log from within the terminal window.

Spine Engine also has client–server capability for remote execution. A Spine Toolbox instance on the client side can send a workflow to a Spine Engine instance on the server side. Messaging is based on Zero-MQ [26] and the messaging can include the data required by the workflow. However, when possible, it is better to use server-based SQL which the Spine Engine server can access directly given the instructions from the client. This remote execution feature is relatively new and is likely to evolve.

#### *2.3.2. Spine interface*

*SpineInterface.jl* is a Julia package for interfacing with the Spine Data Structure within a Julia session. It relies on the Spine DB API and, given the URL of a Spine database, it creates a series of convenience functions to retrieve the contents of that database in the Julia module or session where it is called. It allows users to rapidly build Julia tools that use Spine databases. It is especially useful for building optimisation models in Julia JuMP. SpineOpt.jl energy system model is an example of this.

#### *2.3.3. SpineOpt*

*SpineOpt.jl* is an open-source energy system modelling framework that uses Spine Toolbox data structures directly through SpineInterface.jl. It is a Spine Toolbox plugin and can be efficiently integrated into Spine Toolbox workflows. Through a commodity-agnostic and problem-independent formulation, SpineOpt facilitates the modelling of integrated energy systems and can also incorporate other phenomena that can be represented with conversions and transfers between nodes. The data-driven approach of SpineOpt enables user-flexibility to add additional parameters and constraints. Ihlelmann et al. [27] provide a detailed overview of SpineOpt.

## 3. Illustrative examples

The Spine project performed thirteen case studies to validate, expand and demonstrate the capabilities of Spine Toolbox and SpineOpt.[1] A workflow has been created for each case study, linking all the required data and tools necessary for its completion. The workflows are available as Spine projects in the official git repository.[2] Note that while Spine Project has tried to use open access data where possible, in some instances, some of the original data was not publicly available and has been omitted or replaced with dummy data. In the following we will highlight two case studies.

In the first example (Fig. 3, Case study A3[3]), the goal is to simulate one year of operation of a subset of Stockholm’s district heating system. The considered subset includes seven generating units: one extraction condensing steam turbine, two back-pressure turbines, one gas turbine and three heat boilers. The system can get electricity from a wind power facility and import electricity from an external source. It can also shed load and curtail wind as needed. The workflow has the input data pre-processed with two Python scripts to perform specific calculations (two red hammer icons on the left side).. This is useful when the capabilities of the Spine importer are not sufficient for the required manipulations. The scripts save data as CSV files, which are then imported to a Spine data store using the Spine importer, which takes tabular data and converts it into the Spine data store format using the user-made specifications in the importer interface. SpineOpt uses SpineInterface to interact with the input and output databases based on the resource URLs that the workflow passes to SpineOpt. Finally, the ’Convert Results’ Python script gets data both from input and output databases in order to show results that can relate the inputs to the outputs.

The second example (Case study A5[4]) aims to simulate one week of operation of the Skellefte river in the Swedish hydropower system, which includes fifteen power stations. The graph view of the database editor displays the structure of the data as a graph (Fig. 4). All entities and parameters can be edited.

## 4. Impact

Workflow management tools have had an enormous impact on the scientific process and they are also extensively used in businesses and public administration. Supporting decision making through modelling creates specific requirements for the workflow management tool, and Spine Toolbox strives to address those. Such decision making is widespread — models are used to support pandemic responses, city planning, process design, as well as energy system planning and operation to name a few examples. The common denominator in these tasks is the need to consider uncertainty — not all factors are known and therefore it is important to vary the input parameters through the use of scenarios. Spine Toolbox fills a niche in the workflow management tool landscape by providing an easy-to-use interface for the combined needs of workflow management, data manipulation and scenario building. Developers can incorporate manifold tools into complex workflows that in turn can be operated by less experienced users. Since Spine Toolbox supports server-based SQL databases, it facilitates integrated workflows for groups where individuals have dedicated roles. This can be important when modelling and analysing complex phenomena like the ongoing transition of energy systems to meet the requirements of climate change mitigation.

Spine Toolbox has the potential to have a large impact, but as the tool is still new, the current known user base consists

[1] Case studies summary document: http://www.spine-model.org/pdf/D6.1%20Summary%20of%20the%20case%20studies.pdf
[2] Case studies repository: https://github.com/orgs/Spine-project/repositories
[3] A3 Case study source code: https://github.com/Spine-project/spine-cs-a3
[4] A5 Case study source code: https://github.com/Spine-project/spine-cs-a5

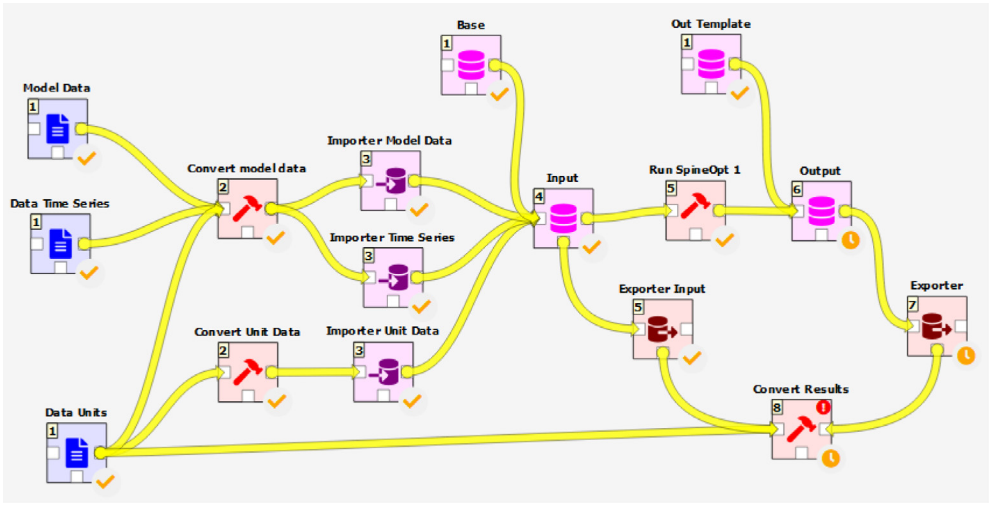

**Fig. 3.** Spine Toolbox workflow for an energy system model of a district heating grid. On the left, there are input data processing scripts and Spine importers that serve the main Input database. On the right, SpineOpt model feeds results to a processing script, which also takes input data into account.

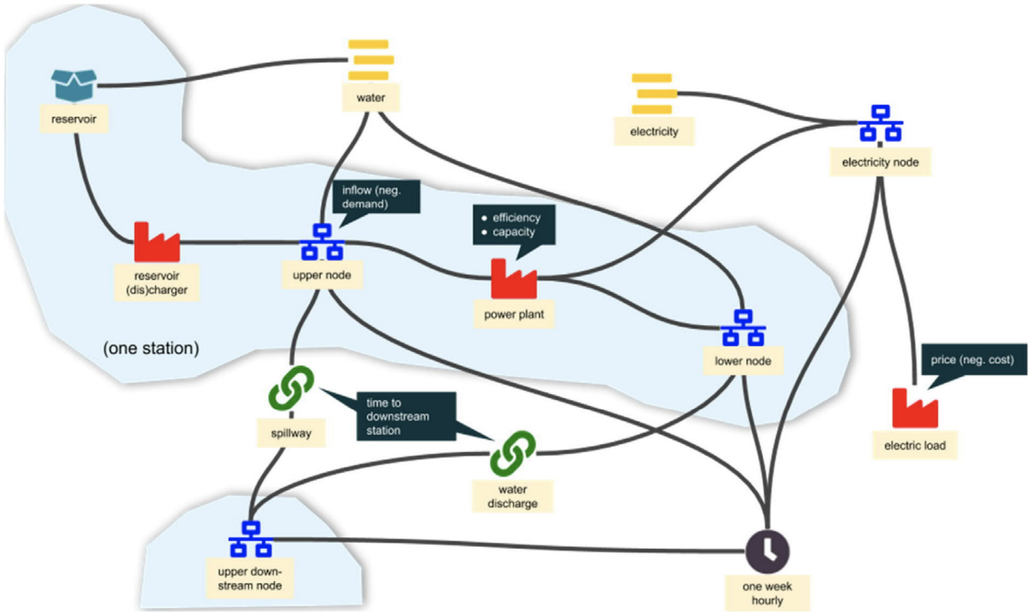

**Fig. 4.** SpineOpt model of *Skellefteälven*. For space limitations, only the *Rebnis* power station is presented, but the same applies to all stations.

of approximately ten different research groups, all in the energy domain. Spine Toolbox has a GNU Lesser General Public License (LPGL) license, enabling the commercial use of the tool and it allowing e.g. commercial plugins even though the core tool must remain open source. Customising workflows and building ad-hoc models and modelling workflows are potential further avenues for commercial activities.

Spine Toolbox is problem agnostic and allows for rapid development of new ad-hoc optimisation and simulation models in Python, GAMS, and other languages, but especially in Julia [18] through the SpineInterface package. Research groups can build tools and workflows to support their needs and also to integrate available tools in the wider community. Consequently, Spine Toolbox could facilitate an efficient way of answering a wide variety of research and practical questions. The software can be applied for modelling and answering a wider range of research questions concerning integrating electrified transport and energy systems or simulating a complex energy system with very high penetration of variable renewable generation, including a gas network converted to hydrogen (Case Study C3). It also supports modelling capabilities for hydroelectric systems (Case Study A1), flexibility assessment using building physics and long-term power grid investment models.

## 5. Conclusions

Spine Toolbox provides a workflow, data, and scenario management framework that can combine multiple data sources and tools while giving the user a full view of the workflow from sources to outcomes. It allows groups of users to collaborate on large-scale problems that require data curation as well as multiple tools and models. The toolbox and the data structure features are capable of performing a wide range of data processing and modelling tasks with a specific focus on data consolidation and scenario analysis.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nr. 774629. The authors would thank all other community members who have contributed by taking part in discussions, filed bug reports etc. Special thanks go to Toni Lastusilta for the data import design and Luc Cluitmans for data structure design, both at VTT, as well as Hanspeter Höschle (@hhoeschle) for filing and commenting a number of issues with various parts of the software.

## References

[1] Atkinson M, Gesing S, Montagnat J, Taylor I. Scientific workflows: Past, present and future. Future Gener Comput Syst 2017;75:216–27. http://dx.doi.org/10.1016/j.future.2017.05.041.
[2] Ringkjøb H-K, Haugan PM, Solbrekke IM. A review of modelling tools for energy and electricity systems with large shares of variable renewables. Renew Sustain Energy Rev 2018;96:440–59.
[3] Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, et al. Pegasus, a workflow management system for science automation. Future Gener Comput Syst 2015;46:17–35. http://dx.doi.org/10.1016/j.future.2014.10.008.
[4] Gil Y, Ratnakar V, Kim J, Gonzalez-Calero P, Groth P, Moody J, et al. Wings: Intelligent workflow-based design of computational experiments. IEEE Intell Syst 2010;26(1):62–72.
[5] Shubha B, Prasad A. Airflow directed acyclic graph. J Signal Process 5(2).
[6] Warr WA. Scientific workflow systems: Pipeline pilot and KNIME. J Comput Aided Mol Des 2012;26(7):801–4.
[7] Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, et al. Apache spark: a unified engine for big data processing. Commun ACM 2016;59(11):56–65.
[8] Zhang H, Chen W, Huang W. TIMES modelling of transport sector in China and USA: Comparisons from a decarbonization perspective. Appl Energy 2016;162:1505–14.
[9] Gardumi F, Shivakumar A, Morrison R, Taliotis C, Broad O, Beltramo A, et al. From the development of an open-source energy modelling tool to its application and the creation of communities of practice: The example of OSeMOSYS. Energy Strategy Rev 2018;20:209–28.
[10] Pfenninger S, Pickering B. Calliope: a multi-scale energy systems modelling framework. J Open Source Softw 2018;3(29):825.
[11] Köster J, Rahmann S. Snakemake—a scalable bioinformatics workflow engine. Bioinformatics 2012;28(19):2520–2.
[12] Brown T, Hörsch J, Schlachtberger D. PyPSA: Python for power system analysis. 2017, arXiv preprint arXiv:1707.09913.
[13] Idoine C, Krensky P, Brethenoux E, Hare J, Sicular S, Vashisth S. Magic quadrant for data science and machine-learning platforms. Gartner, Inc; 2018, p. 13.
[14] Foundation Apache. Apache airflow. 2020, URL https://airflow.apache.org/.
[15] Foundation Apache. Apache spark. 2020, URL https://spark.apache.org/.
[16] Brown T, Hörsch J, Schlachtberger D. PYpsa: Python for power system analysis. J Open Res Softw 2018;6. http://dx.doi.org/10.5334/jors.188.
[17] Copeland R. Essential sqlalchemy. " O'Reilly Media, Inc." 2008.
[18] Bezanson J, Karpinski S, Shah VB, Edelman A. Julia: A fast dynamic language for technical computing. 2012, arXiv preprint arXiv:1209.5145.
[19] Dunning I, Huchette J, Lubin M. JuMP: A modeling language for mathematical optimization. SIAM Rev 2017;59(2):295–320.
[20] Savolainen P, Kiviluoma J, Rinne E, Soininen A, Marin M, Dillon J. Spine deliverable 2.1 software design document. 2021, http://www.spine-model.org/pdf/D2.1SoftwareDesignDocument.pdf.
[21] Glatard T, Rousseau M-E, Camarasu-Pop S, Adalat R, Beck N, Das S, et al. Software architectures to integrate workflow engines in science gateways. Future Gener Comput Syst 2017;75:239–55. http://dx.doi.org/10.1016/j.future.2017.01.005.
[22] Randles BM, Pasquetto IV, Golshan MS, Borgman CL. Using the jupyter notebook as a tool for open science: An empirical study. In: 2017 ACM/IEEE joint conference on digital libraries. IEEE; 2017, p. 1–2.
[23] Dagster: a python library for data orchestration. 2021, URL https://dagster.io/.
[24] Betancourt F, Wong K, Asemota E, Marshall Q, Nichols D, Tomov S. openDIEL: a parallel workflow engine and data analytics framework, in: Proceedings of the practice and experience in advanced research computing on rise of the machines (learning). 2019. pp. 1–7.
[25] Ogasawara E, Dias J, Sousa V, Chirigati F, de Oliveira D, Porto F, et al. Chiron: A parallel engine for algebraic scientific workflows. Concurr Comput 2013;25:2327–41. http://dx.doi.org/10.1002/cpe.3032.
[26] Zero-MQ Open Source Community. Zero-MQ. 2021, URL https://zeromq.org/.
[27] Ihlemann M, Kouveliotis-Lysikatos I, Huang J, Dillon J, O'Dwyer C, Rasku T, et al. Spineopt: A flexible open-source energy system modelling framework. 2021, Submitted for Review.