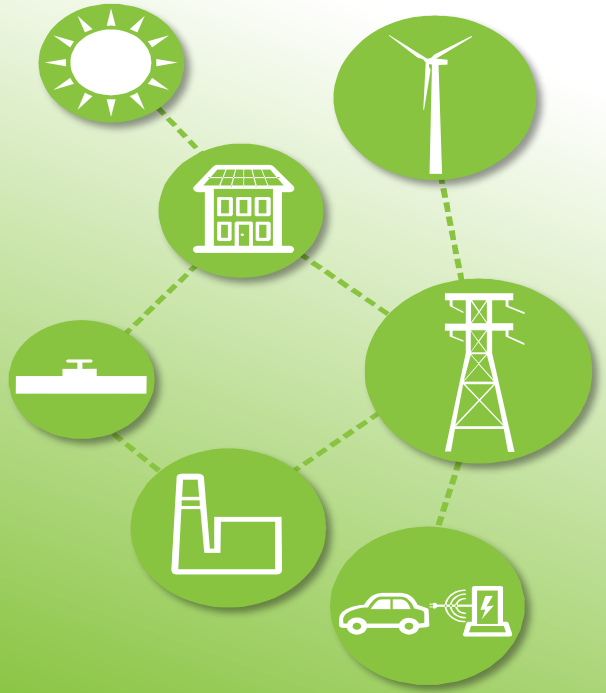
Spine



Software design choices related to data sharing and data acquisition

Juha Kiviluoma

Senior scientist, VTT Technical Research Centre of Finland

Senior energy systems researcher, University College Dublin

9th Oct. 2019, EC DG R&I

EMP-E workshop

VTT

UCD DUBLIN

KU LEUVEN

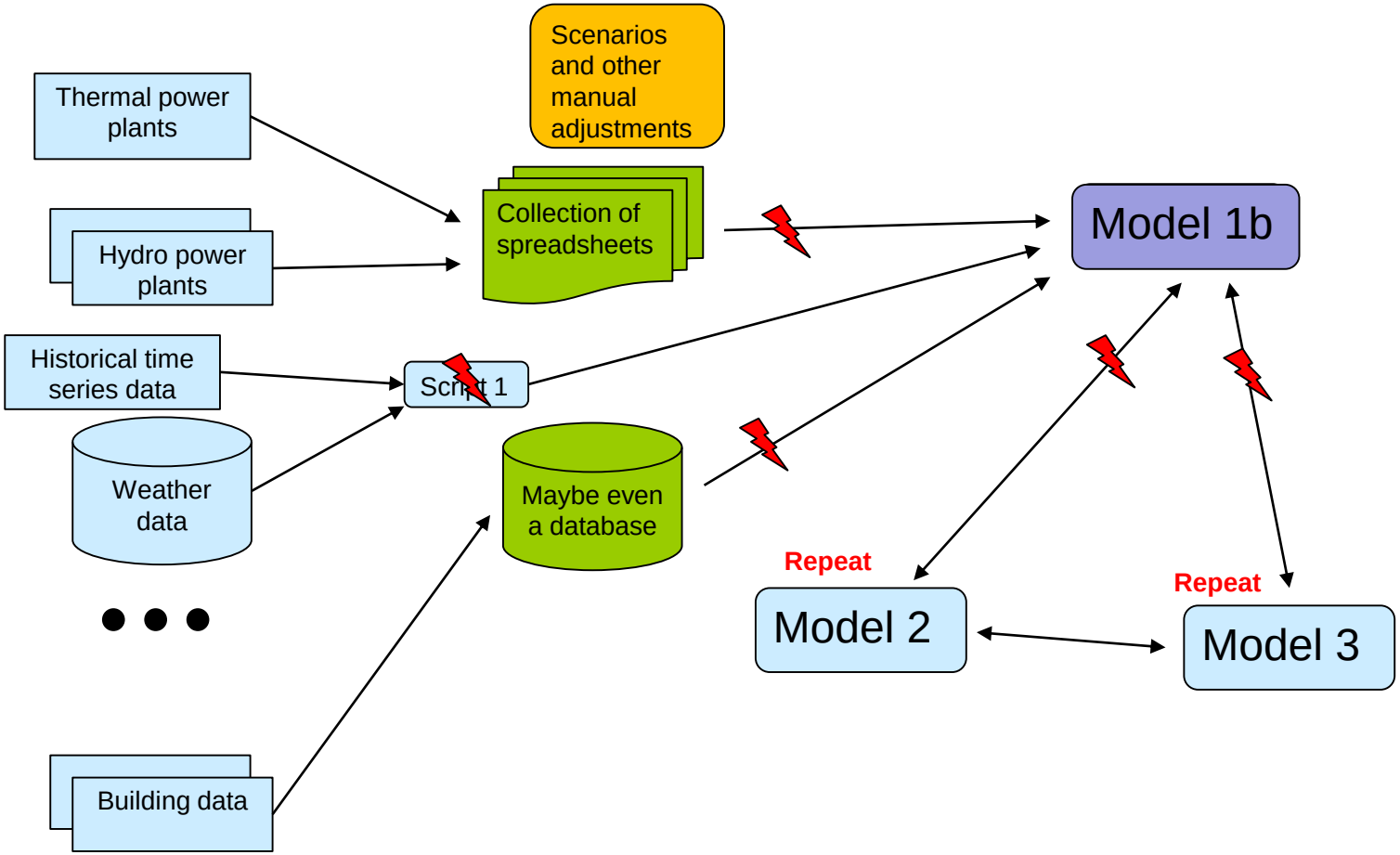
KTH VETENSKAP OCH KONST

ENERGY REFORM CONSULTING SERVICES

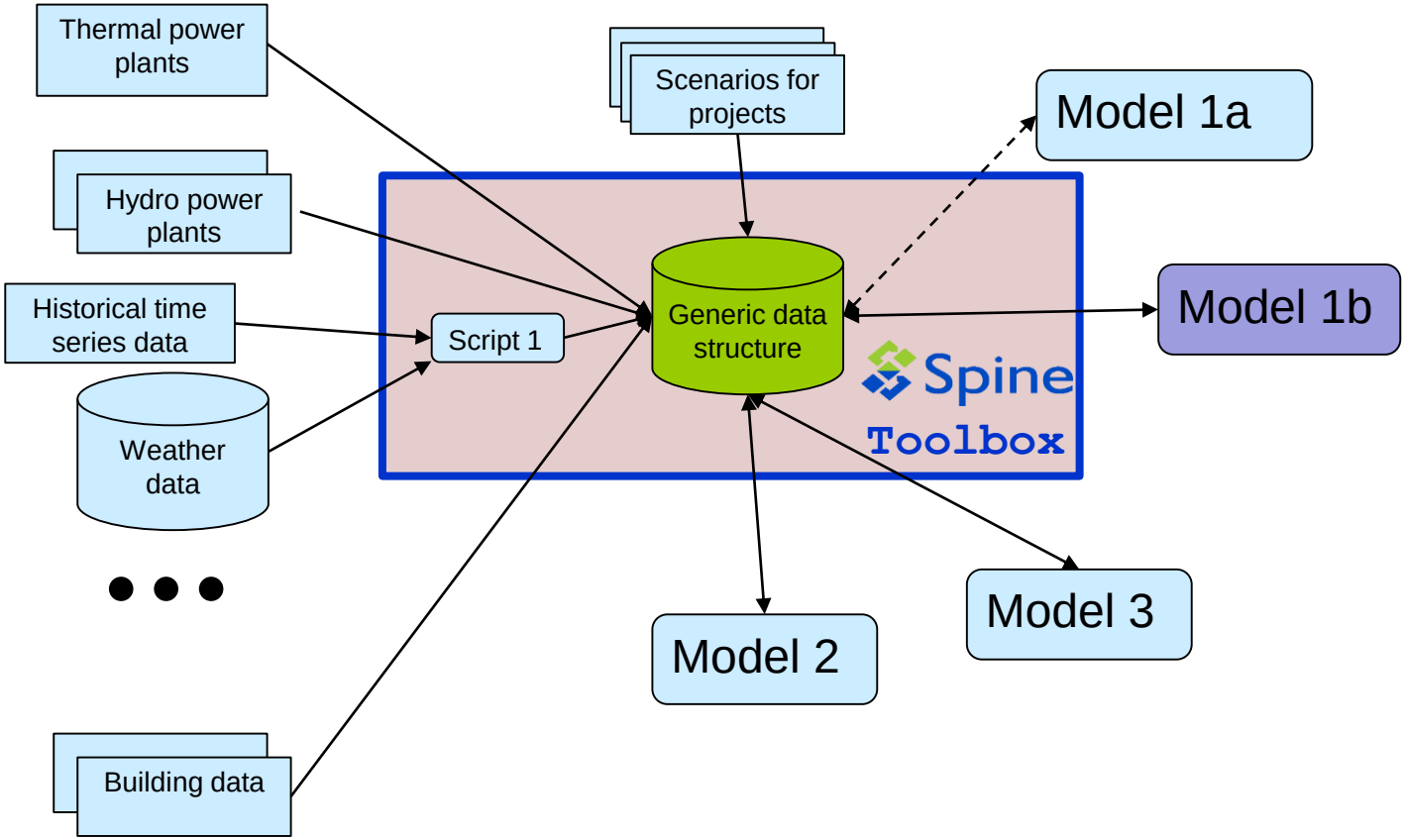
Starting points

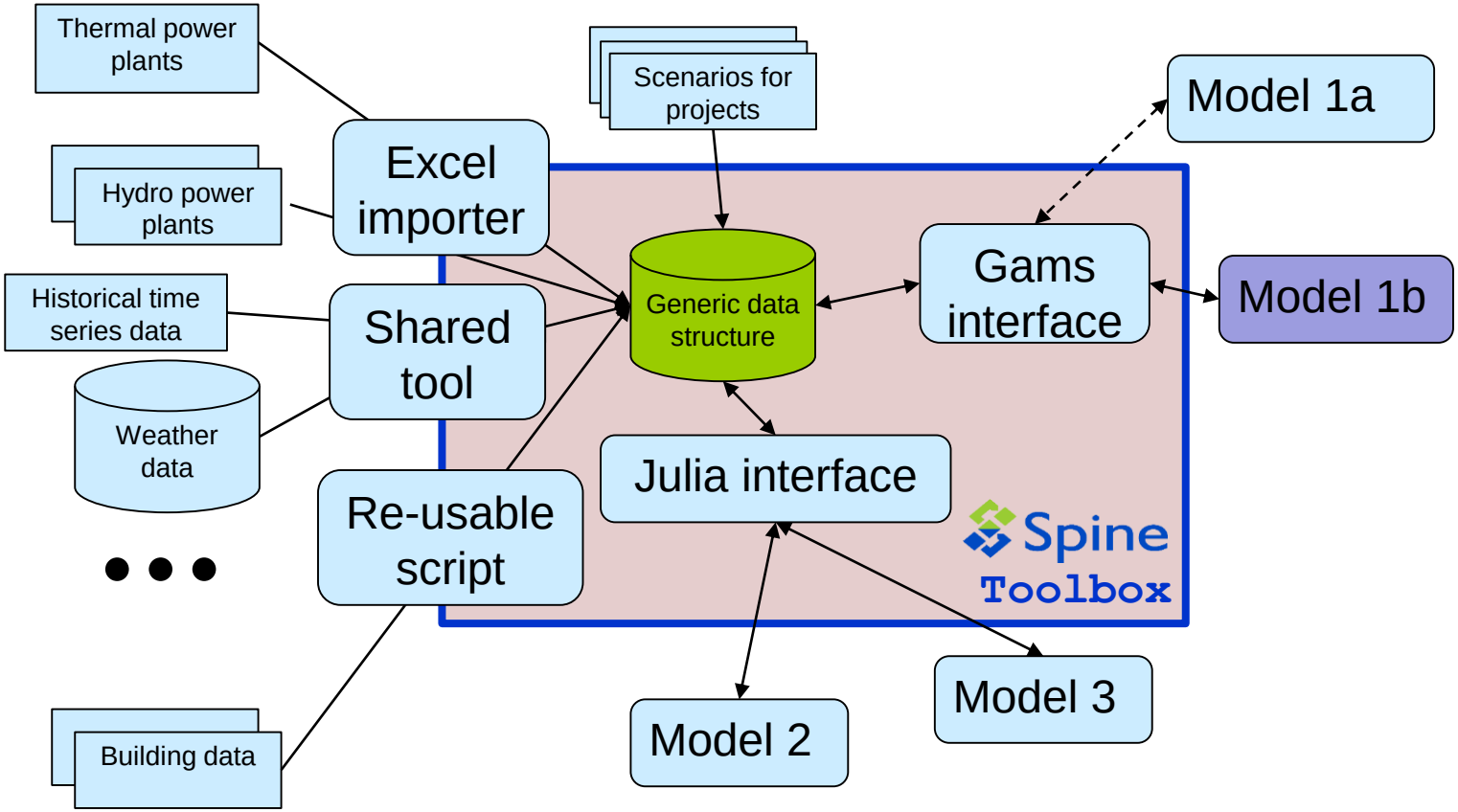
- Some aims
  - Co-operation instead of competition
  - Build tools together
  - Replicable results → scientific progress and more reliable policy support
- Some means
  - Open source
  - Python based Spine Toolbox

- Spine Model in Julia (https://github.com/Spine-project)
- Backbone in GAMS (https://gitlab.vtt.fi/backbone)
- IRENA FlexTool in Excel/Mathprog (https://www.irena.org/energytransition/Energy-System-Models-and-Data/IRENA-FlexTool)


Thermal power plants
Scenarios and other manual adjustments
Hydro power plants
Collection of spreadsheets
Model 1b
Historical time series data
Script 1
Weather data
Maybe even a database
Repeat
Repeat
Model 2
Model 3
Building data

Spine


Thermal power plants
Hydro power plants
Historical time series data
Weather data
Building data
Scenarios for projects
Script 1
Generic data structure
Spine Toolbox
Model 1a
Model 1b
Model 2
Model 3

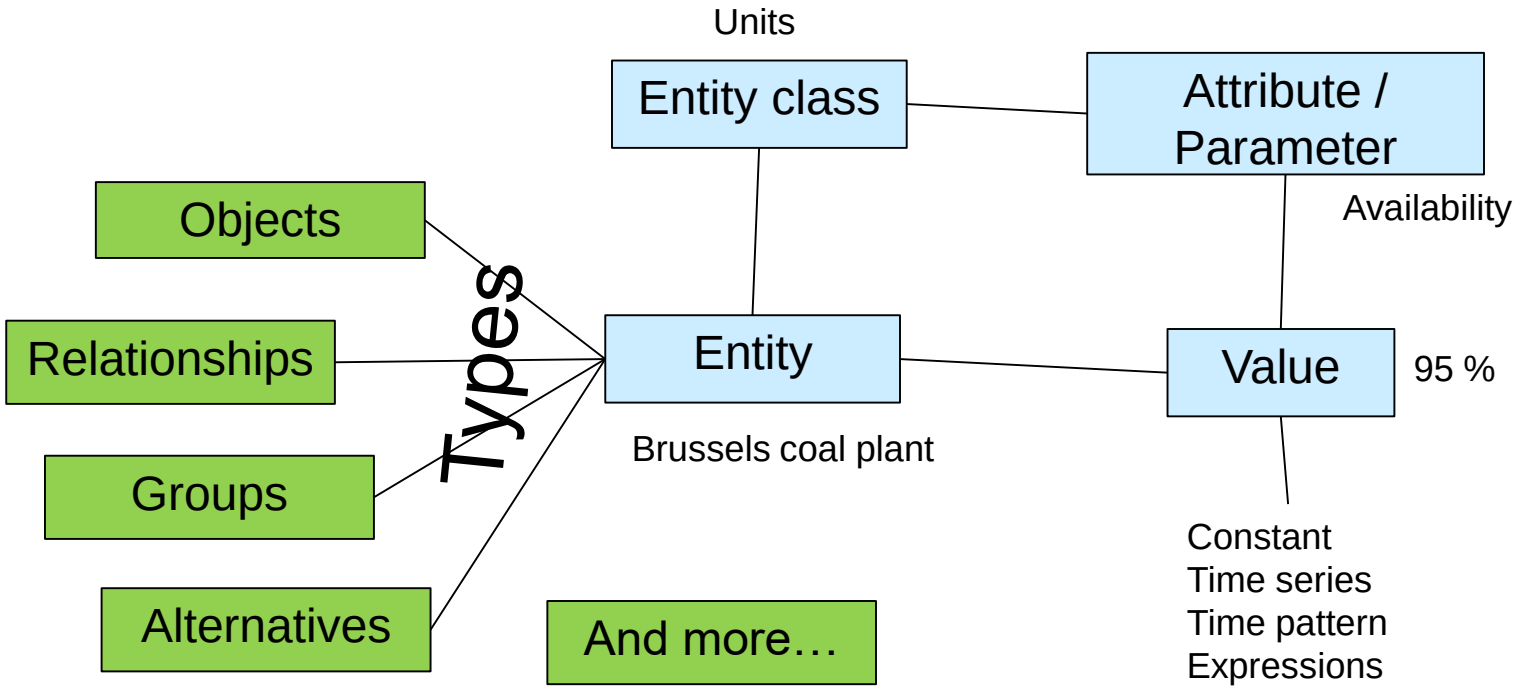

Thermal power plants
Hydro power plants
Historical time series data
Weather data
Building data
Excel importer
Shared tool
Re-usable script
Scenarios for projects
Generic data structure
Julia interface
Gams interface
Model 1a
Model 1b
Model 2
Model 3
Spine Toolbox

Spine

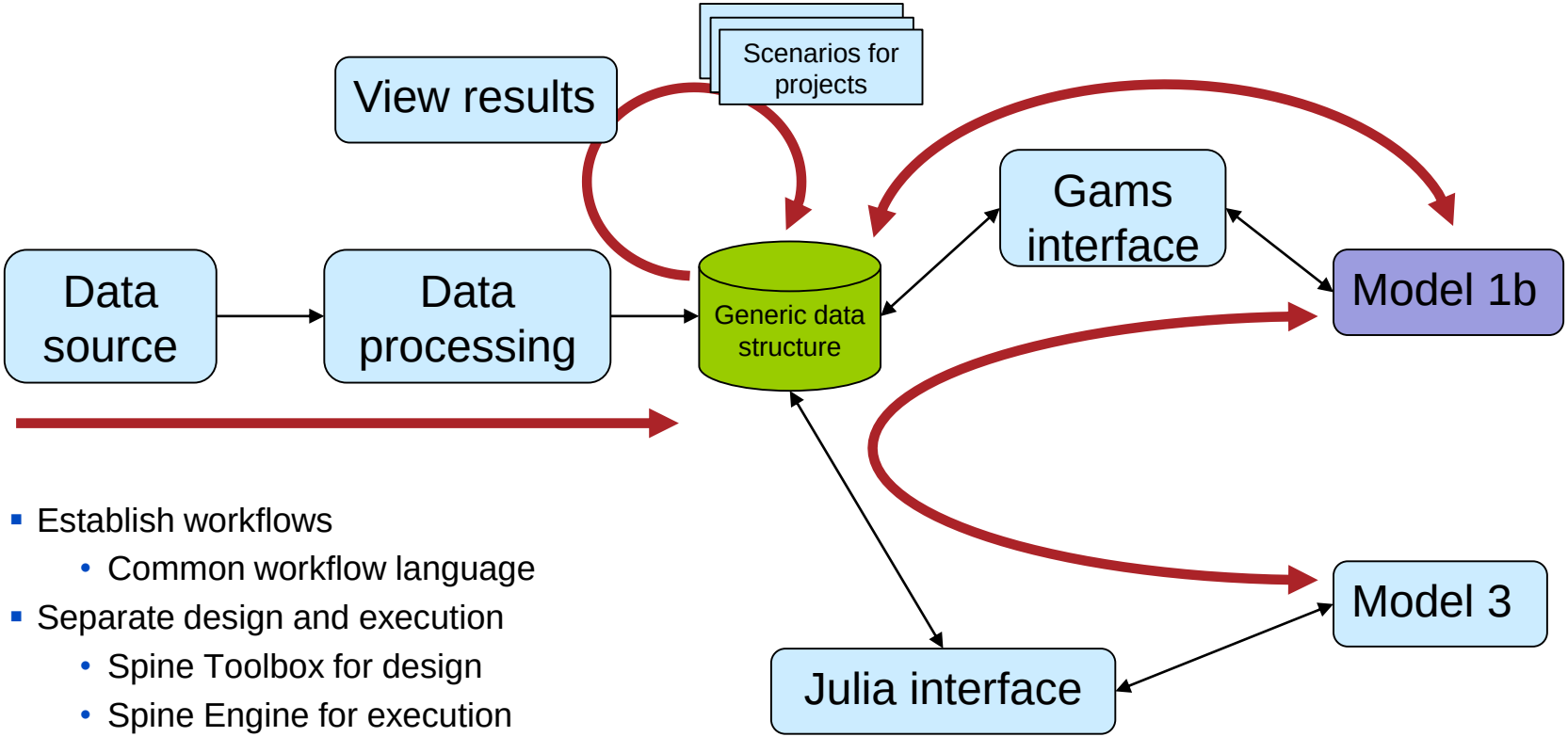
Design choices

- Interface!
  - Connect through a shared, generic, data structure
  - Each tool needs to maintain only one connection
  - Easy to add new tools and swap old tools
- Non-documented spreadsheets → Re-usable scripts and tools
- Local files → Server based databases (one version)
- Version control (know what happened) and open source (share the effort)
  - Data acquisition and data processing
  - Models
- Project based workflow
  - Keep project specific modifications separate
- Separate design and execution
  - Work on your laptop, execute in cloud

Generic data structure (EAV with classes and entity types)

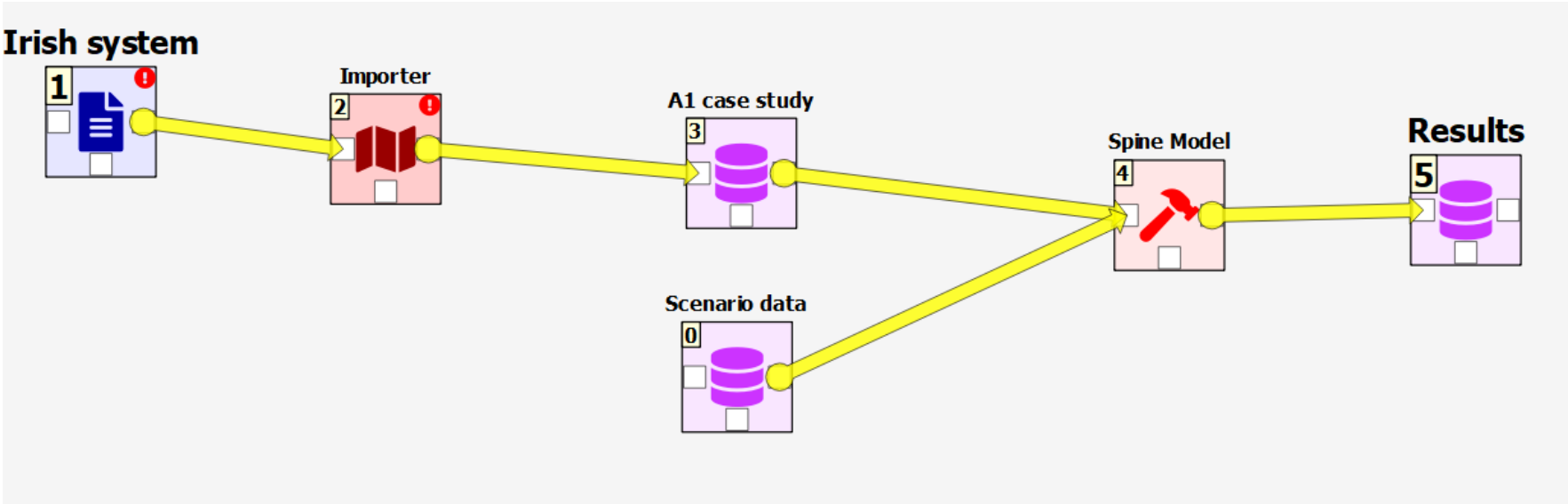


Spine



- Establish workflows
  - Common workflow language
- Separate design and execution
  - Spine Toolbox for design
  - Spine Engine for execution
    - Direct acyclic graphs
    - Scheduling, workers,…

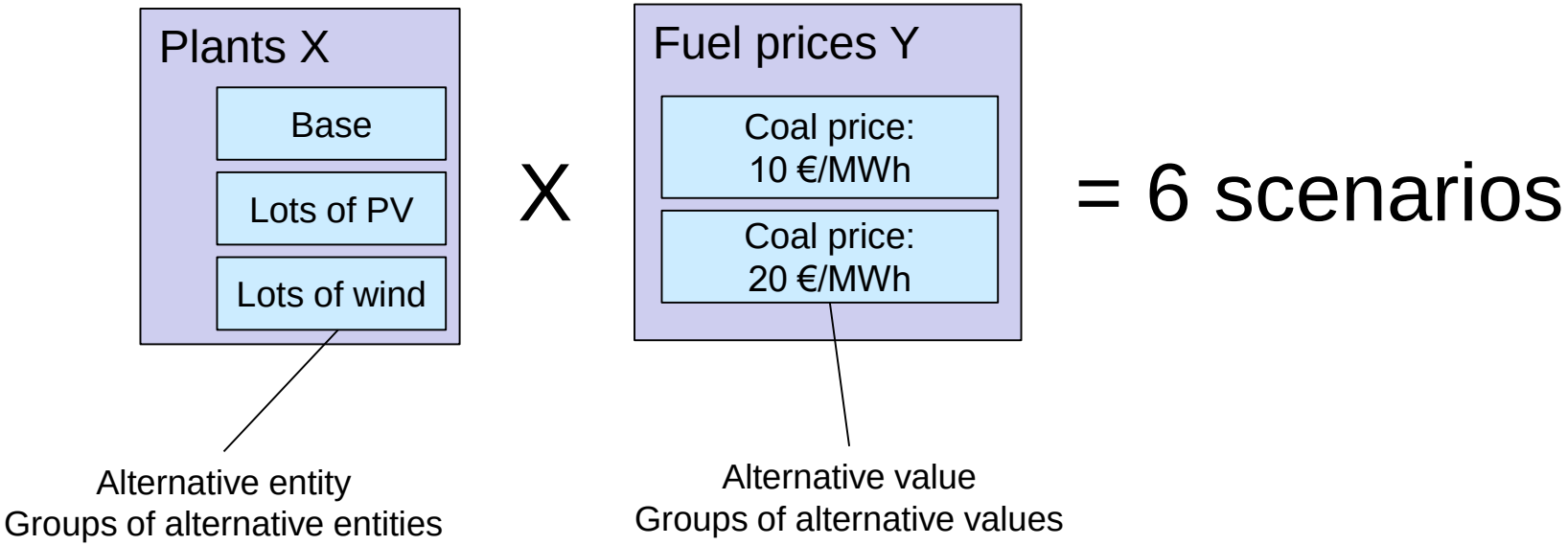
Project based workflow management



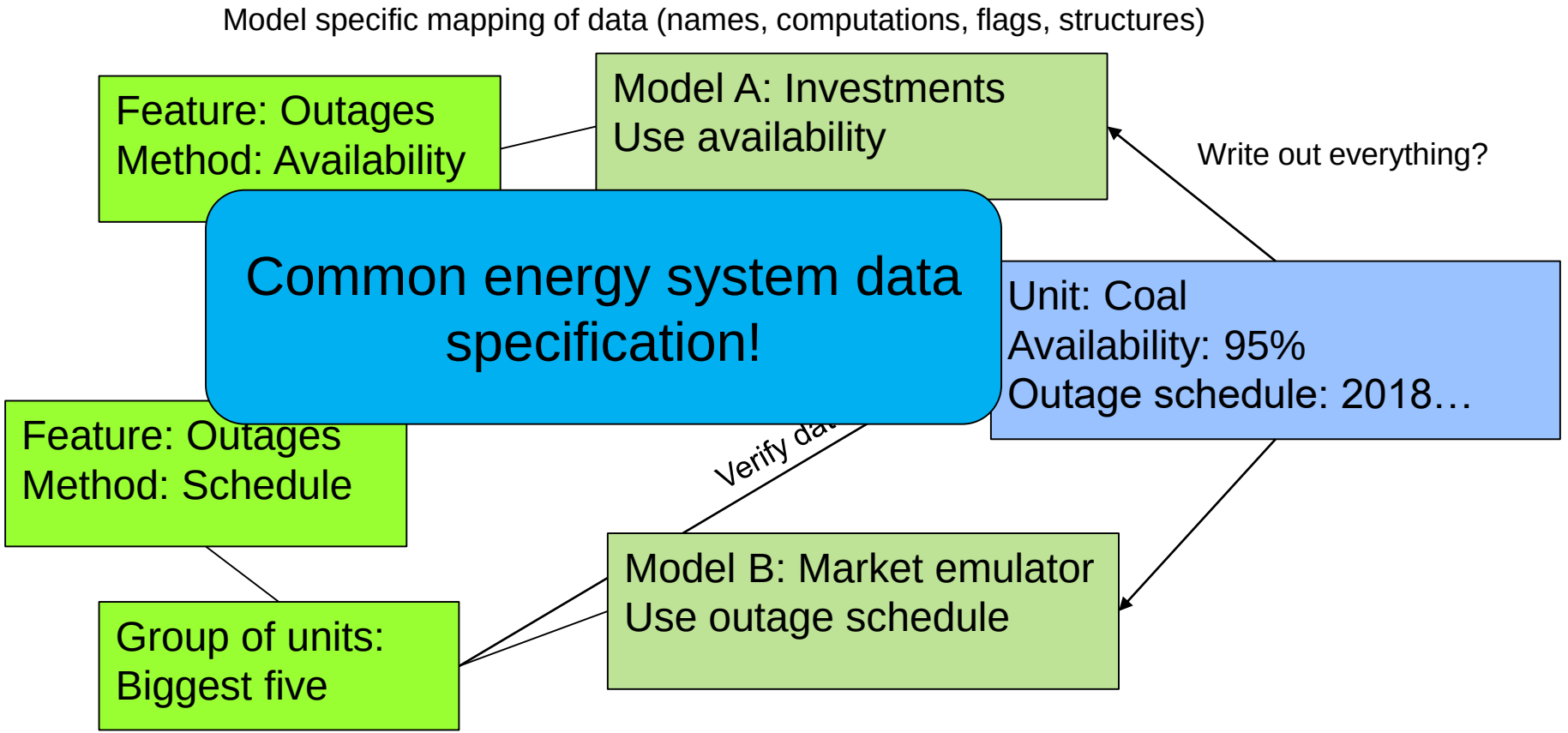
Metadata and commit messages

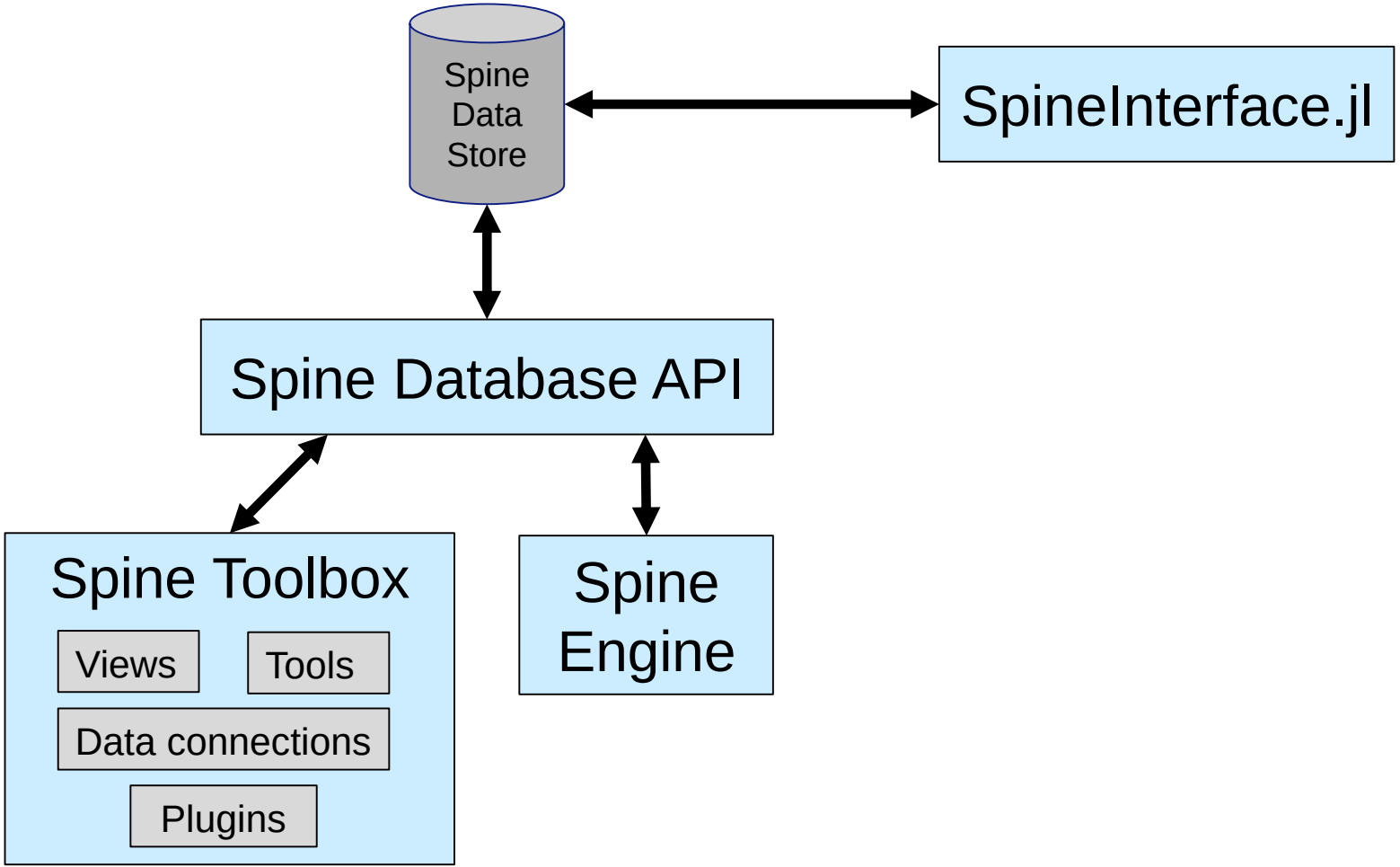
- Automatic metadata from tools/models that process data
- User can also input metadata
- Commits
  - Push users to improve documentation
  - Changes to the database will be accompanied by a commit message
  - Logs time and user, but hopefully also a helpful message about the data edits in that session

Alternatives, scenarios and recipes



Many tools and models, but one data set?




Spine
Data
Store
SpineInterface.jl
Spine Database API
Spine Toolbox
Views
Tools
Data connections
Plugins
Spine
Engine

More design choices

- Try to use the generic data structure as far as reasonable
- Don’t allow screw ups
  - Data validation
  - Explicit selection
  - Archetypes – features – methods
- Open interfaces – can grow and accommodate new things
  - Databases through SQLAlchemy (db agnostic)
  - Direct connection to Julia/JuMP (Spine Model)
  - Plug-in infrastructure

Spine: Open source toolbox for modelling ~~integrated energy systems~~

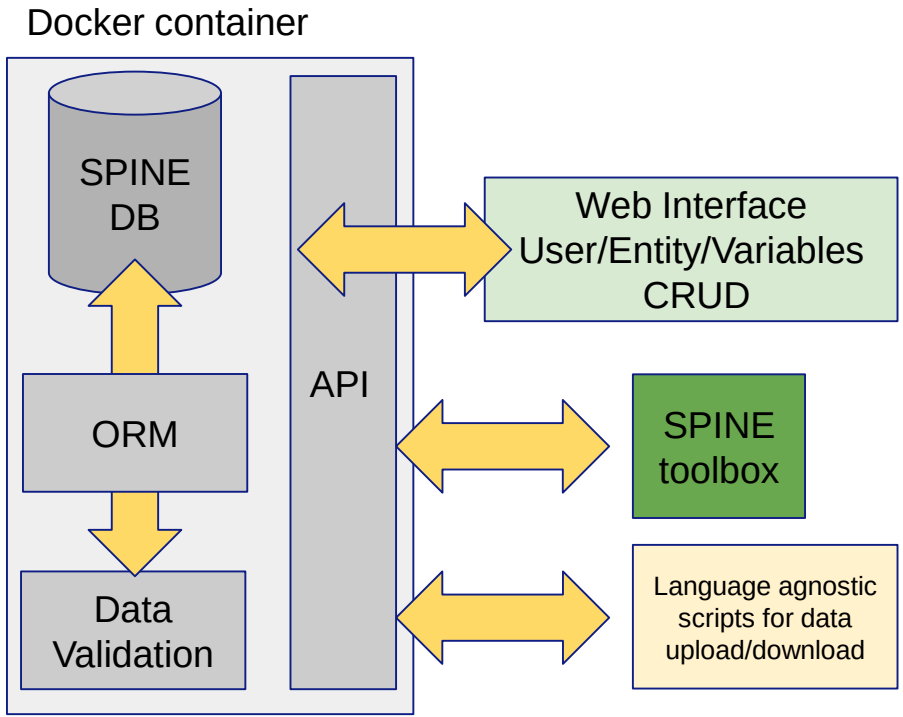
- Project part funded by the Horizon 2020 program of the European Union
- LCE-05-2017 - Tools and technologies for coordination and integration of the European energy system
- 4 year project commenced October 2017 with a €3.7m budget
- 5 Partners, collaboration with NREL

- https://spine-toolbox.readthedocs.io/en/latest/
- https://github.com/Spine-project
- http://www.spine-model.org
- **spine_info@vtt.fi**

Spine

Modularity

- Interfaces between components
- Spine data store has an API that accepts specific types of data
  - We use EAV-CT
    - Entities with attributes, values, classes and types
    - I.e. sets, set members, multi-dimensional sets, and parameters
  - You can build your own interface for your entity type
- New views on data use the API
- New models use the API
- However, also separate direct API from Julia to the DBs (speed and usability)
- Spine Engine has an interface for CWL



- Modelling is too complicated
- Generic data structure
  - Entity, Attribute, Value (with classes and entity types)
  - View and edit
  - Import and export any data
  - Connect tools and model
- Entities of different types: objects, relationships, groups, model setup,…
- Allow timeseries and time patterns
- Alternative values, alternative sets, scenarios, recipes
- One model sees parameter value and does something
- Another model uses flags
- But common database cannot rely just on these
- Features and methods