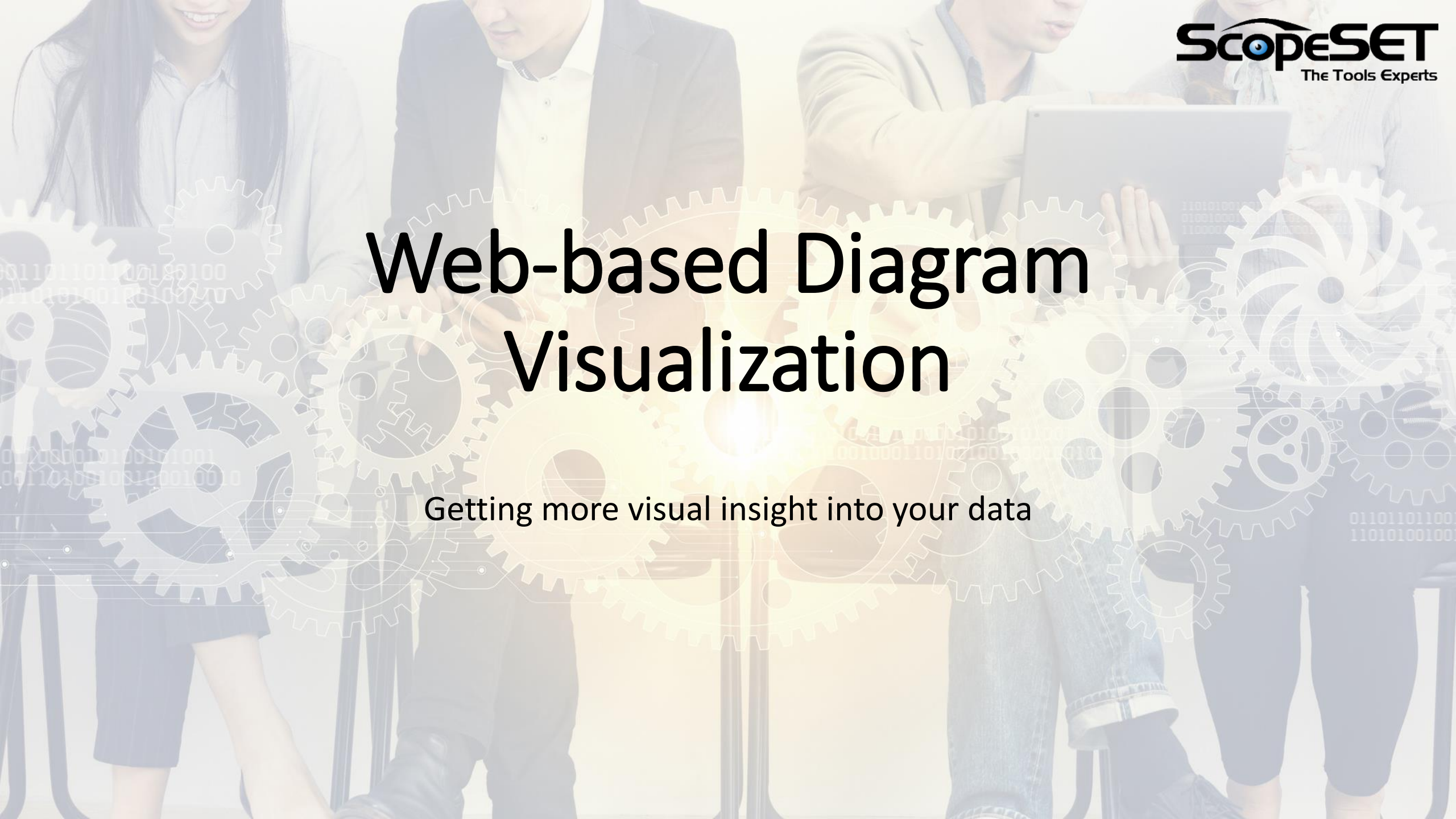


ScopeSET
The Tools Experts

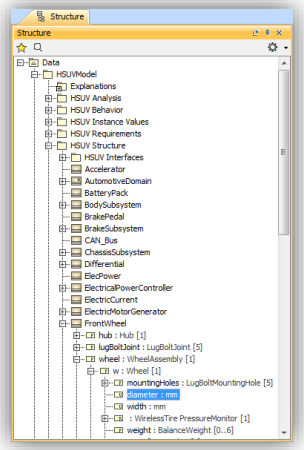
Web-based Diagram Visualization

Getting more visual insight into your data

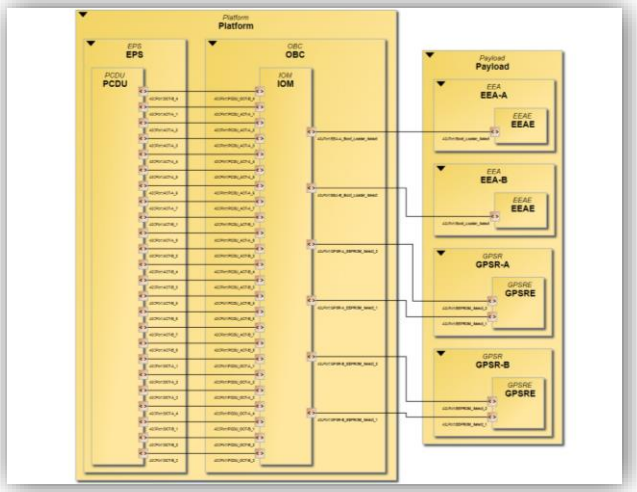
Introduction

- With the utilization of cloud services, an **increasing amount of data/information is collected and available** for analysis in every enterprise
- The information is often displayed as **trees/tables, lines/bar/pie charts** or simple graphs but this typically **lacks additional dimensions** like cross-dependencies to better understand the data
- The **benefit of diagrams** compared to other representations is the capability to provide a **multidimensional view** on the data
  - for example, a hierarchical breakdown (by nesting elements into each other) together with a relationships to other elements on the same level by a connection which can also indicate the type of the relationship
- **Diagrams** are also considered **more descriptive** than tables and trees and can provide a notation for faster recognition
- A first impression on how this can look like is given on the next slide

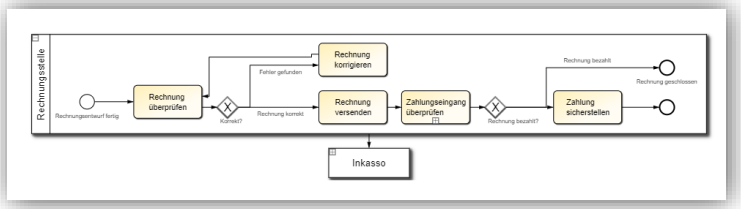
Possible Diagram Solutions

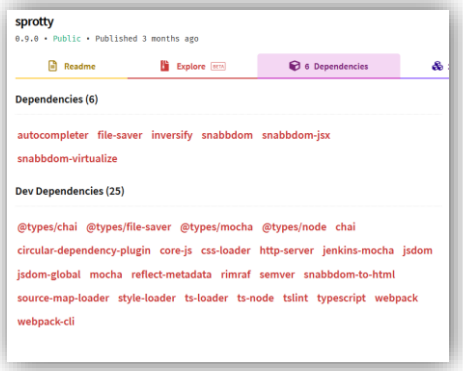


System Model

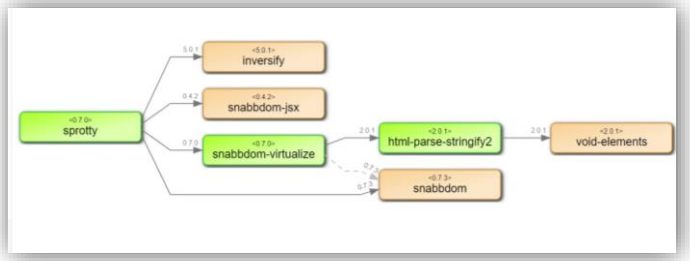


Business Process





Dependencies



Benefits of Diagram Visualization

- Allow more efficient overview and insight into different data aspects to support the overall decision-making process
  - **Web based** diagram visualization for data which is more than a number or text.
- Provide dedicated diagram-oriented functions, i.e.:
  - **Data Flow analysis** – Visualize where data comes from or where it goes to
  - **Filtering** – Reduce the view to a set of aspects which are relevant for the users work
  - **Semantic Search** – improved search accuracy by understanding the searcher's intent and the contextual meaning
- Collaboration in shared diagram views to utilize teamwork without the overhead of the synchronization effort
- Auto-layout of diagrams to provide a sophisticated view on the data which can be used for review, documentation or collaboration purpose
  - Boxes and Lines are positioned automatically to provide a nice-looking visualization
  - avoids time consuming manual layout performed by the User to produce high quality diagrams for marketing, documentation, review and collaboration

➢ **ScopeSET has designed and implemented various visualizations for multiple input data-sources and can assist you in developing and deploying such views and services. Some examples are shown on the next slides**

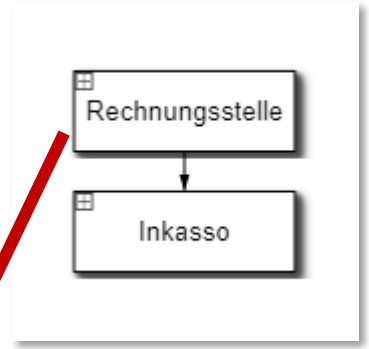
Diagram visualization Examples

- The next slides will show some example diagrams for different use cases to provide a first impression of what could be done
  - The examples are not complete and can be adapted to any business case
- ***Business Process*** (Slide 6-7)
  - *Other views with similar characteristics: Mission Phases, Activities, …*
- ***Software Dependencies*** (Slide 8-9)
  - *Other views with similar characteristics: System-, Division-Dependencies, …*
- ***System Model*** (Slide 10-11)
  - *Other views with similar characteristics: Logical-, Functional-Architecture, Networks, …*
- ***Composed Views*** (Slide 12-13)
  - *Provide different synchronized views on a complex system*

Business Process

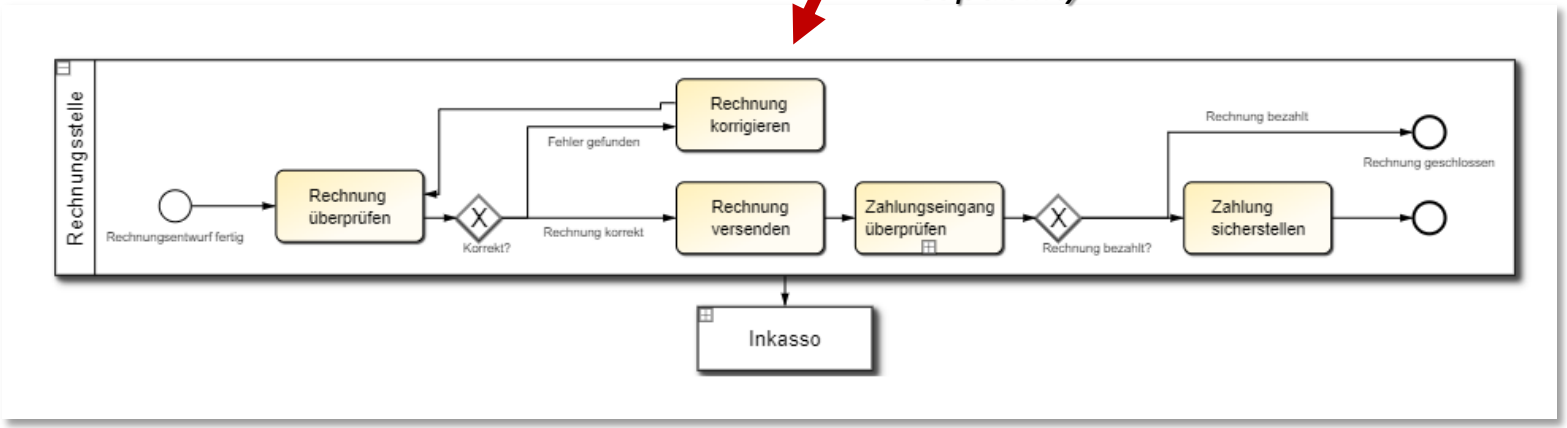
(Views with similar characteristics: Mission Phases, Activities, …)

- Process data is often stored as simple Text or in Databases. To provide a more descriptive view for Documentation, Review, Communication, etc. a graphical representation is useful.
- **Creating the graphical representation can be time consuming** and needs to be update if the process change.



Example of an auto-layouted Business process derived from Json Data with expand and collapse capability

➢ **Auto generated graphical representations combined with auto-layouting simplifies this process, which will save you time and money!**

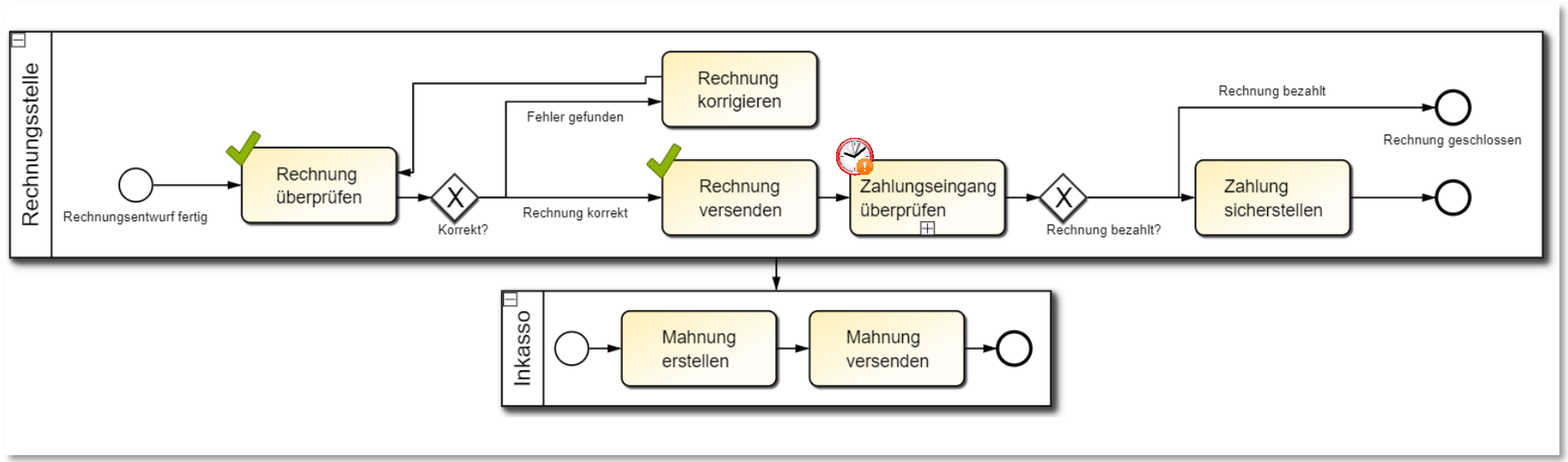


Business Process

(Views with similar characteristics: Mission Phases, Activities, …)

- Sometimes it is **hard to track** where certain **processes are blocked** or what must be done next.

➢ **An automatically generated and updated graphical representation assist your analysis of the active processes and helps you to identify possible issues.**

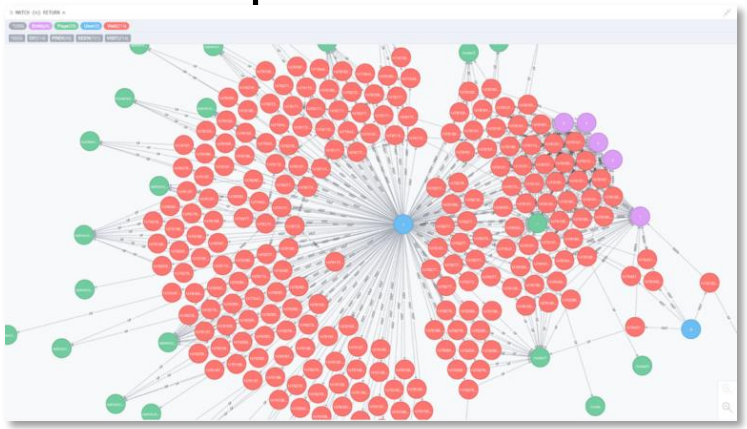


Example of an automatically generated business process derived from Json Data with indicator for the task states. (this could also include roles and persons based on your privacy regulation)

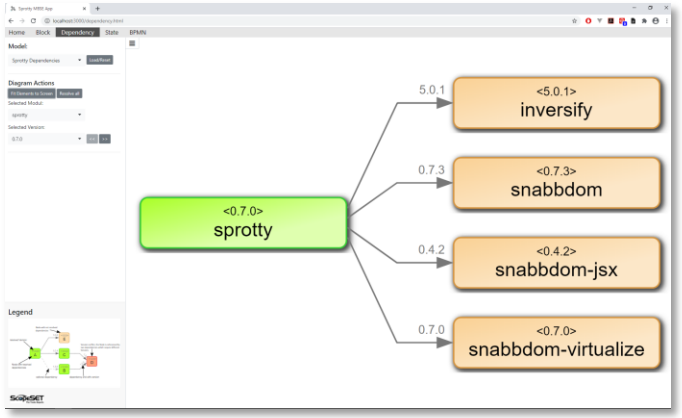
Software Dependencies

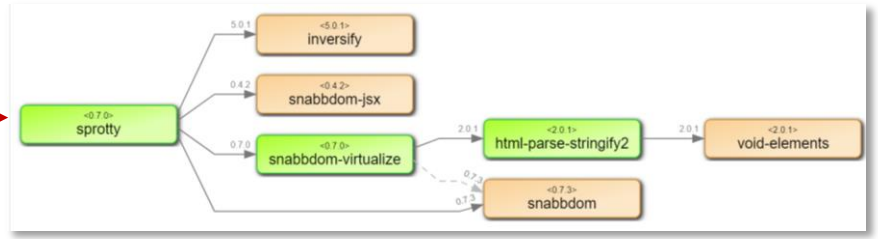
(Views with similar characteristics: System-, Division-dependencies, …)

- **Dependencies** between different artifacts like datasets or system components are often **implicit or stored in meta files/databases**.
- **Simple graph views** can **overwhelm the user** with the sheer amount of information and bad layout (see right side).



➢**Auto-layout and expanding dependencies on demand can reduce the cluttering for the user and provides the capability to focus on the important path.**





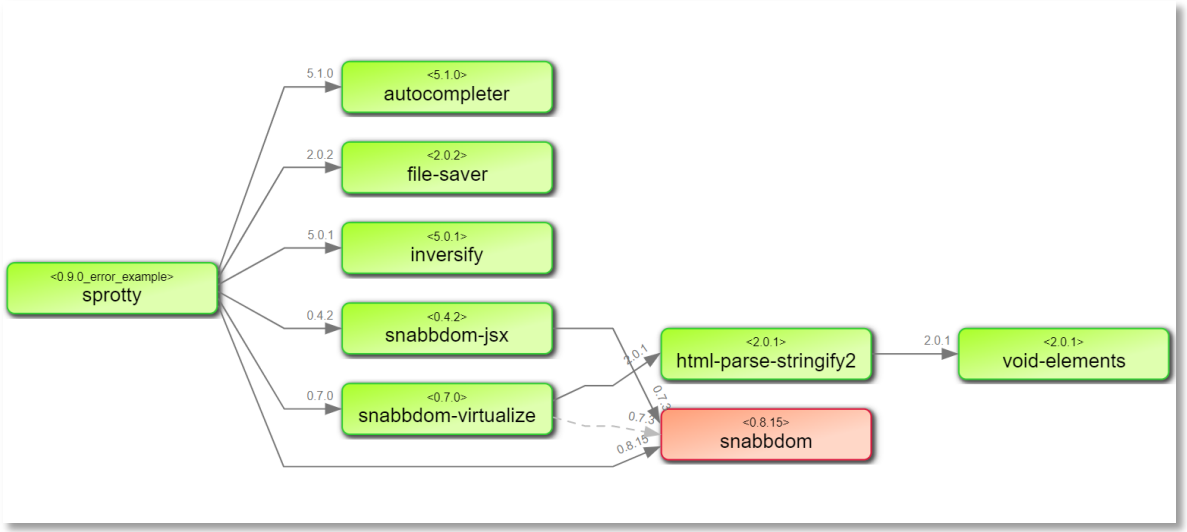
Example of auto-layouted dependencies with the capability to expand certain paths

Software Dependencies

(Views with similar characteristics: System-, Division-dependencies, …)

- It can be the case that certain artifacts have **dependencies** which produce a **conflict** with dependencies of other artifacts.
  - A common example are software components which are reused by several other components but in different versions. Some versions might be compliant will others might not be.

➢ **A proper notation can highlight such issues and makes it fast and easy to identify the problem.**



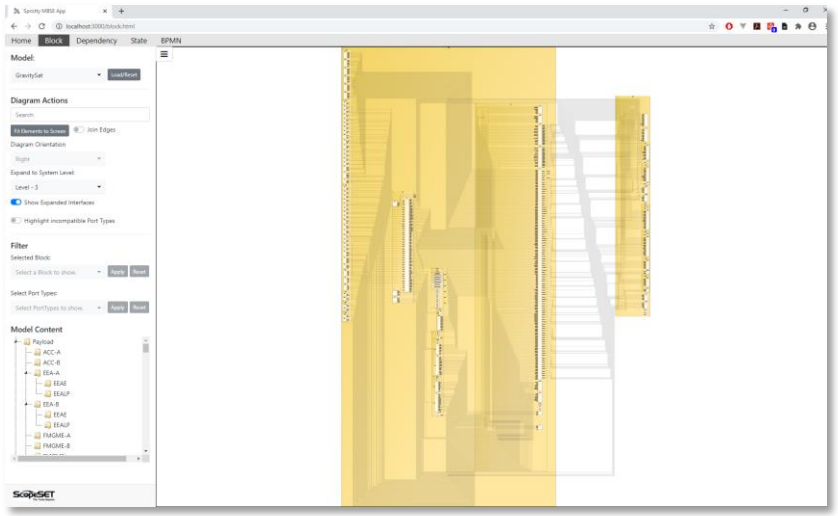
Example of auto-layouted dependencies with highlighted conflict (snabbdom version mismatch)

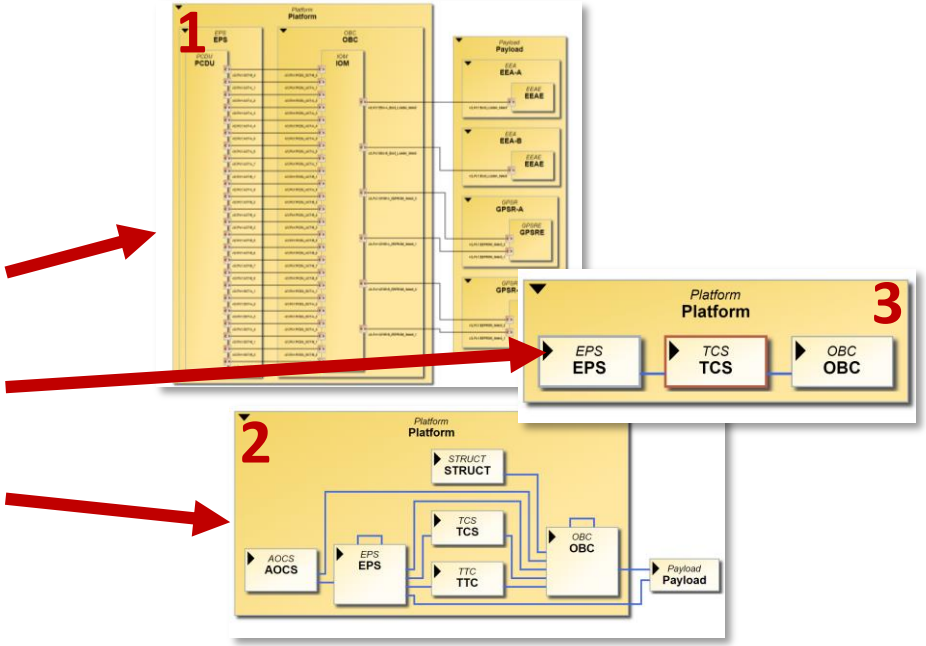
System Model

(Views with similar characteristics: Logical-, Functional-Architecture, Networks, …)

- System Models are often modeled in SysML which is a modelling language based on UML. In most cases **Block, Port and Connection** oriented Views are created which **describe the system**. In several cases also **multiple views** are created based on the same data **just to show different granularity** or **different dependencies**.
- **Creating, updating and polishing such views** (move the blocks/connections/ports to provide a nice-looking view) is a **tedious work** and **waste of time** which could be spend more productive.

➢ **Auto-generated and auto-layouted views solve this issue by providing different views on the System with "one click"**





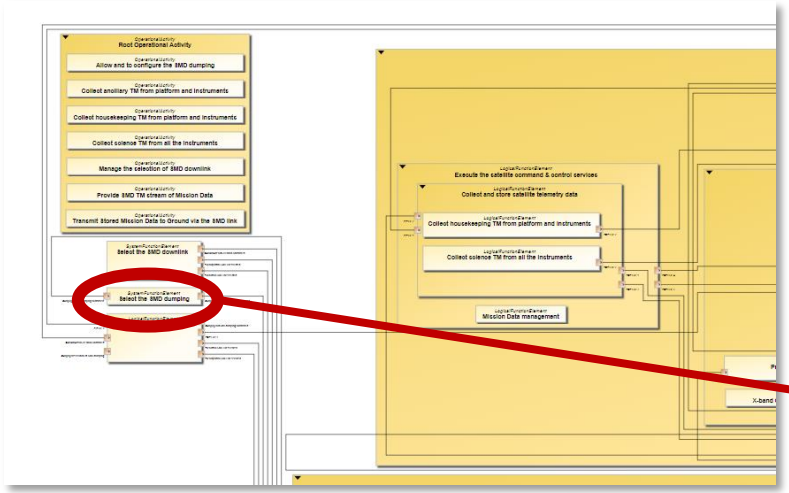
Example of a complete System Model (left) and filtered views (right).
Filtered Views:
1 - Shows only System artifacts connected by a certain port type
2 - partial expanded (hierarchy level) system with hidden ports and aggregated connections
3 – one block is set as context and only direct connected blocks are shown

System Model

(Views with similar characteristics: Logical-, Functional- Architecture, Networks, …)

- To **identify** possible **system elements** (Blocks) which are **responsible for wrong data** received by a system element a flow analysis can be useful.

➢ **A flow analysis implementation can easily be combined with a human friendly way to visualize the result by using auto-layouting and view generation.**



Example of auto-layouted flow analysis results based on a selected input block (highlighted green below).
The possible input providing blocks can be easily identified (“Select the SMD downlink” and “Decode, authenticate, validate, process and distribute TC”)