WHAT?

An End-to-End Experiment Management Framework

WHY?

for (assisted design of) Experiments/Simulations

HOW?

using Replicable/Reusable Experiment Specifications, Logical Reasoning, and MDE

Rakshit Mittal, Inte Vleminckx, Hans Vangheluwe – University of Antwerp, Belgium
Diego Pires, Lucas Lima – Universidad Federale Rural de Pernambuco, Brazil
Dominique Blouin – Telecom Paris, France

Current problem

- Semantics of models, simulations, parameters, results, and their relations are not well-defined

    - ***Why*** *was a simulation performed?*
    - ***What*** *were the parameters? architectures? workflows?*
    - ***Why*** *were their values so?*
    - *Where are results stored?*
    - *What format?*

- Experimental data cannot be trusted anymore (due to lack of provenance)!!

- Represents significant loss of time and effort ...

Proposed framework

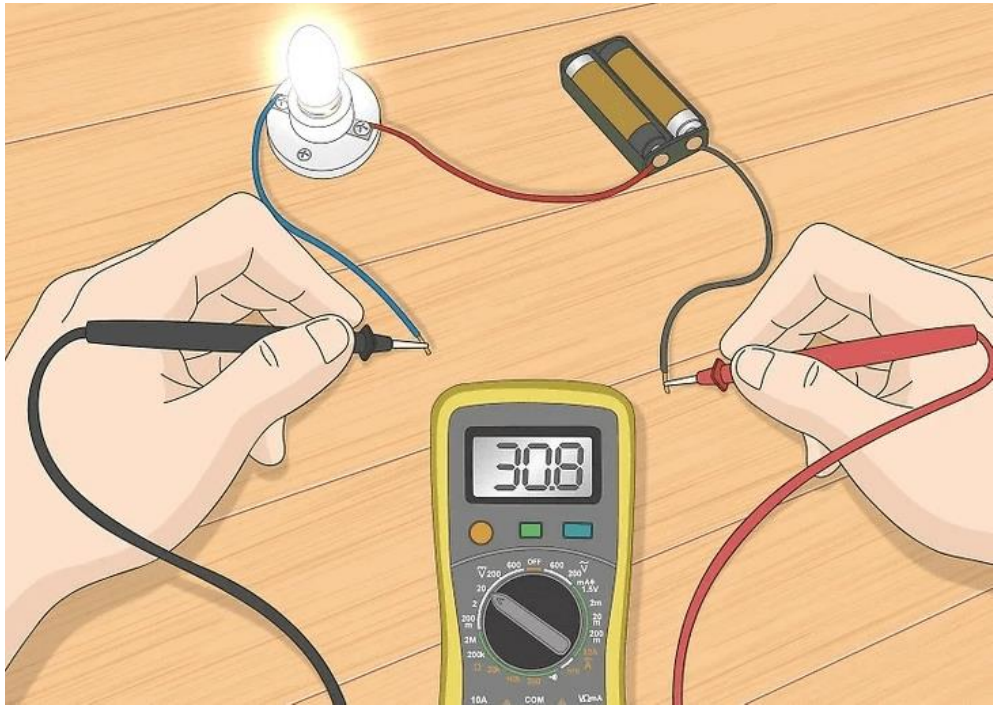
- Already tackled traceability and versoning of artifacts in engineering workflows

  Arkadiusz Ryś, Lucas Lima, Joeri Exelmans, Dennis Janssens, and Hans Vangheluwe. Model management to support systems engineering workflows using ontology-based knowledge graphs. Journal of Industrial Information Integration 42 (2024). doi:10.1016/j.jii.2024.100720

- **General, extensible framework for the explicit representation and utilization of simulation/experiment meta-information** with a focus on experiment design
  - Utilize logical reasoning to ask user relevant questions (only)
  - Generate a model of the user's intent
  - Schedule and perform model transformations to consistently design experiment workflows and architectures (iterative)
  - Enact experiment and generate final report

Case-study (Resistor)

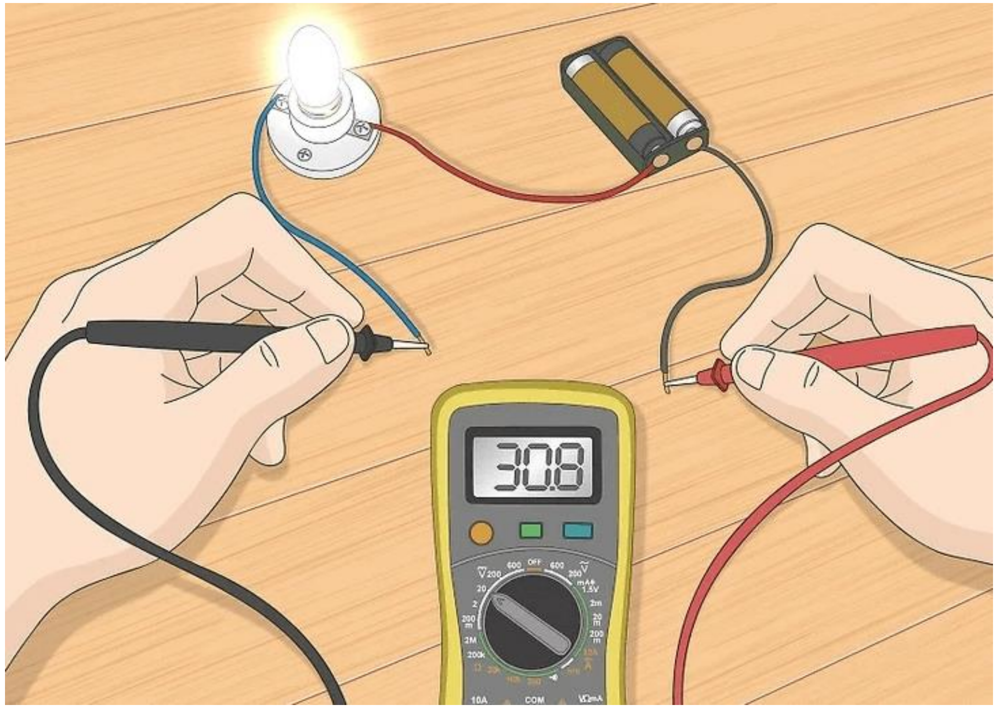
"If there is a (constant-in-time) 5 (+/- 0.1) V voltage source applied across an Ohmic model of resistor R1, I want to measure the steady- state current (with 1mA accuracy) that flows through the resistor, and considering no other effects on R1."



Case-study (Resistor)

"If there is a (constant-in-time) 5 (+/- 0.1) V voltage source applied across an Ohmic model of resistor R1, I want to measure the steady- state current (with 1mA accuracy) that flows through the resistor, and considering no other effects on R1."

HOW DO WE GO FROM THAT TO THIS?



Case-study (Resistor)

BEING EXTREMELY PEDANTIC

To demonstrate the kind of information needed to be explicit even about the simplest experiment

- to be re-usable
- to be replicable

- ***Which*** *components?*
- *What is the* ***architecture*** *of connecting the components?*
    - *Battery / Voltmeter is connected in parallel ... because ...*
    - *Battery / Voltmeter is associated with Voltage ... and ...*
    - *Voltage is an "across" variable (Bond graphs)*

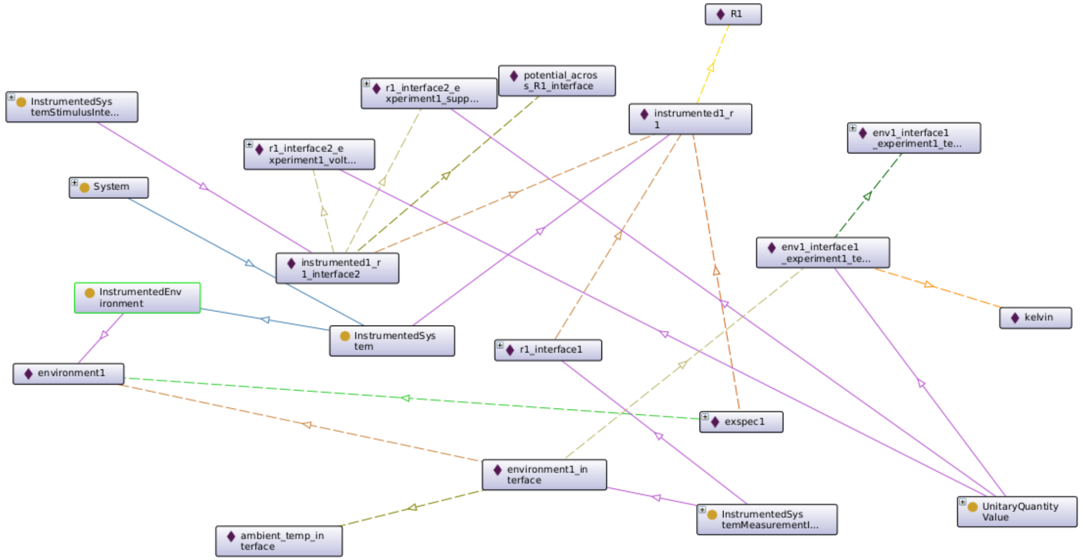
    - *Ammeter / Current Source is connected in series ... because ...*
    - *Ammeter / Current Source is associated with Current ... and ...*
    - *Current is a "through" variable (Bond graphs)*

- *Which ammeter from available stock?*
- *Which battery from available stock?*
- *Which wires to connect them?*

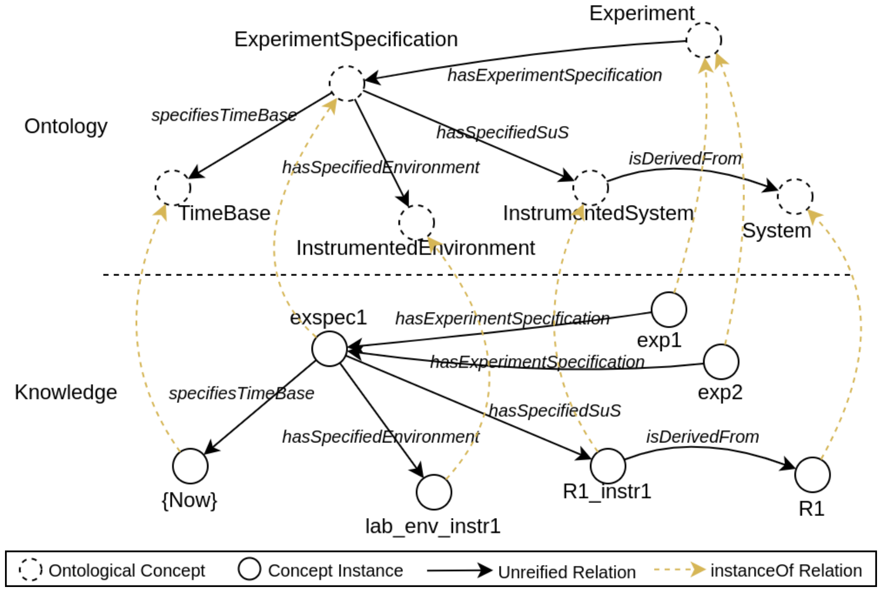
- *How to connect the wires and components? Solder? Manual Touch?*

- *What is the workflow of performing the experiments?*
    - *What* ***order*** *to connect them?*
    - *How to* ***start*** *the experiment?*
    - *When to* ***end*** *the experiment?*
    - *What and how to record?*

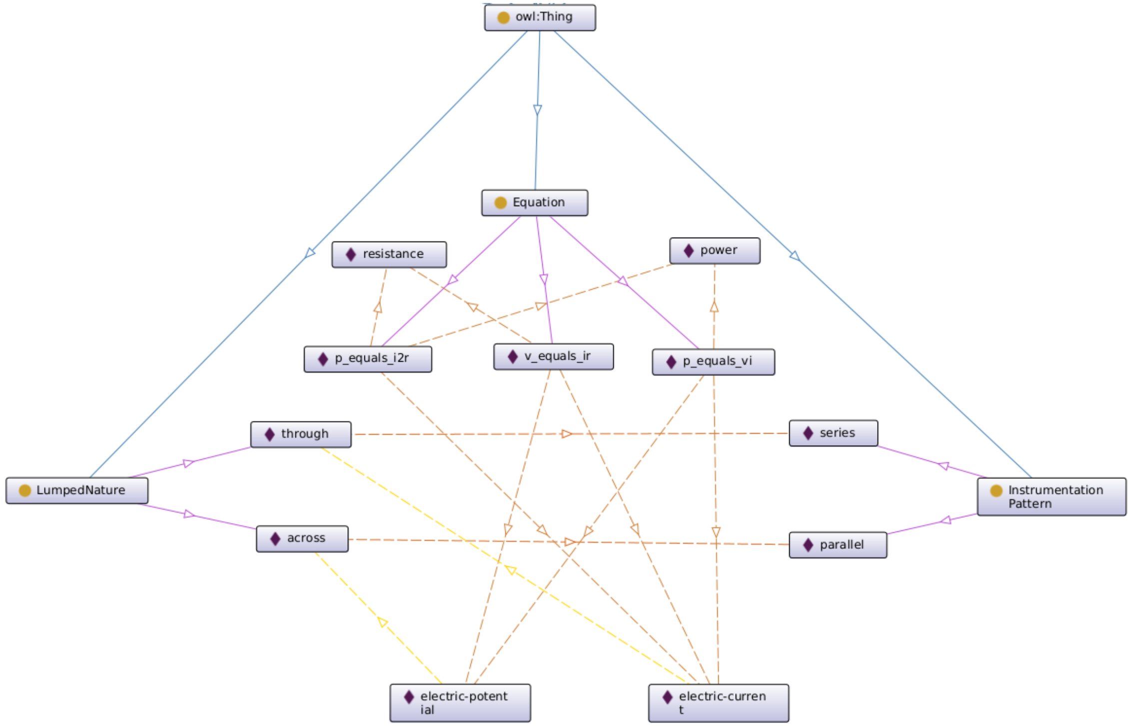
Systems Engineering: **Concepts** and **Relationships** Knowledge Representation



Ontology of **systems** and **domains**



Ontology of **experiment specifications**



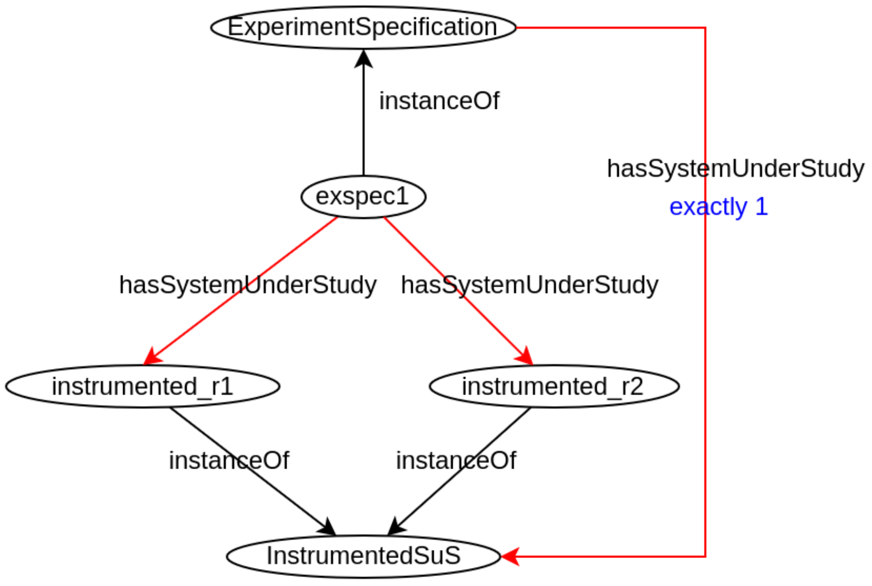
Ontology of **physical quantities** that imports:

- ISO/IEC-8000 (physical quantities)
- VIM4 (quantity types, metrology)

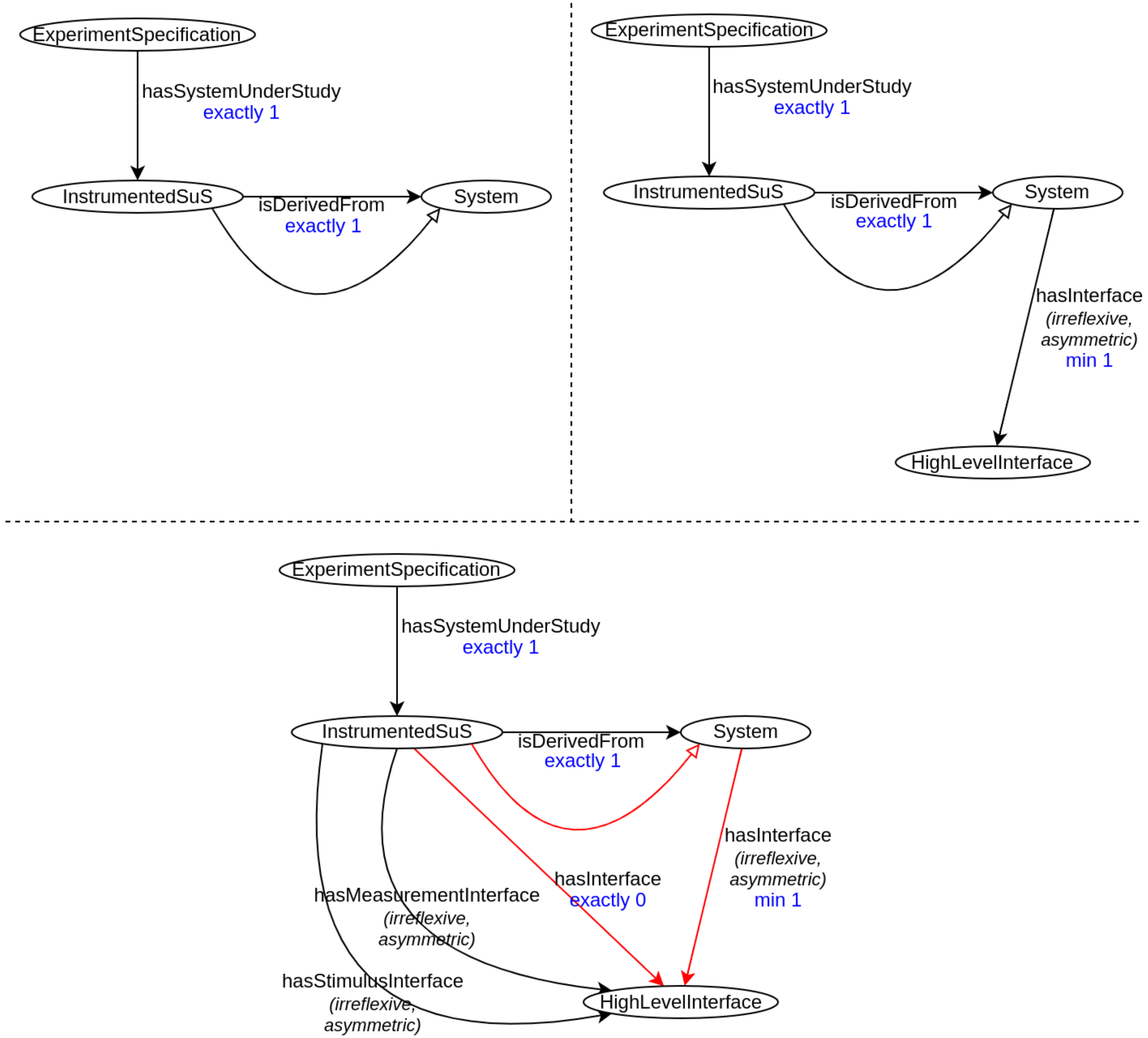
Consistency Checking

supported by

Automated Ontological **Reasoning**

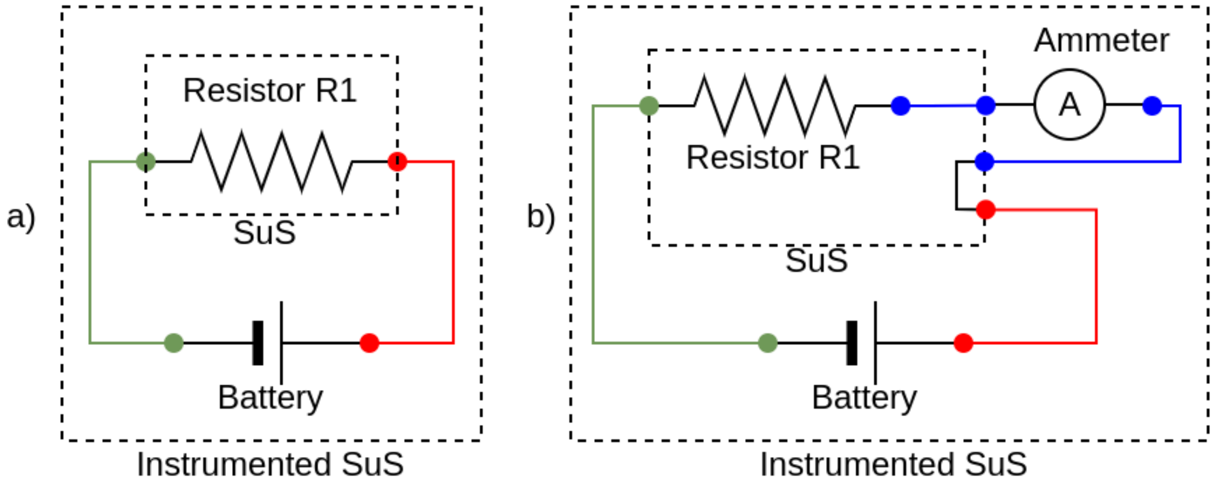


Knowledge **Inconsistency**



Ontological **Inconsistency**

Systems Engineering: Instrumentation



While in classical systems theory a system is distinguished from its environment by a clear boundary (abstract or physical),

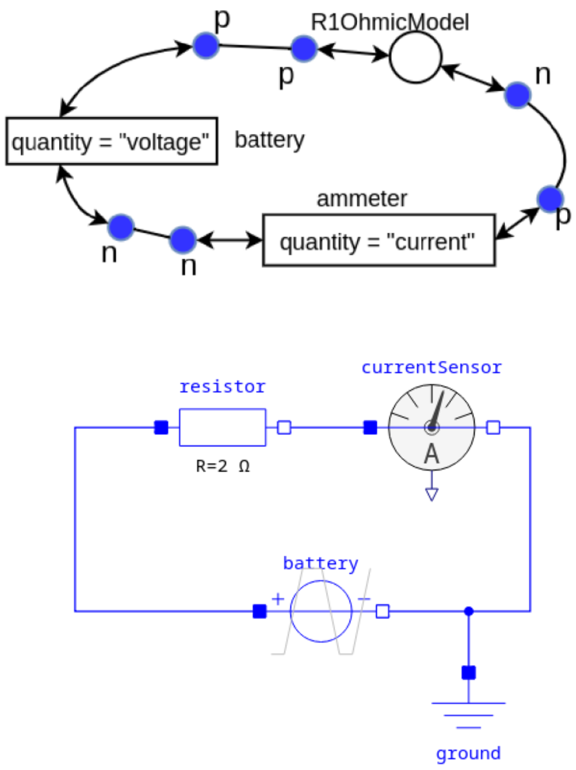
the instrumentation of a system often transcends these boundaries in the real world.

High-Level Interface (HLI)

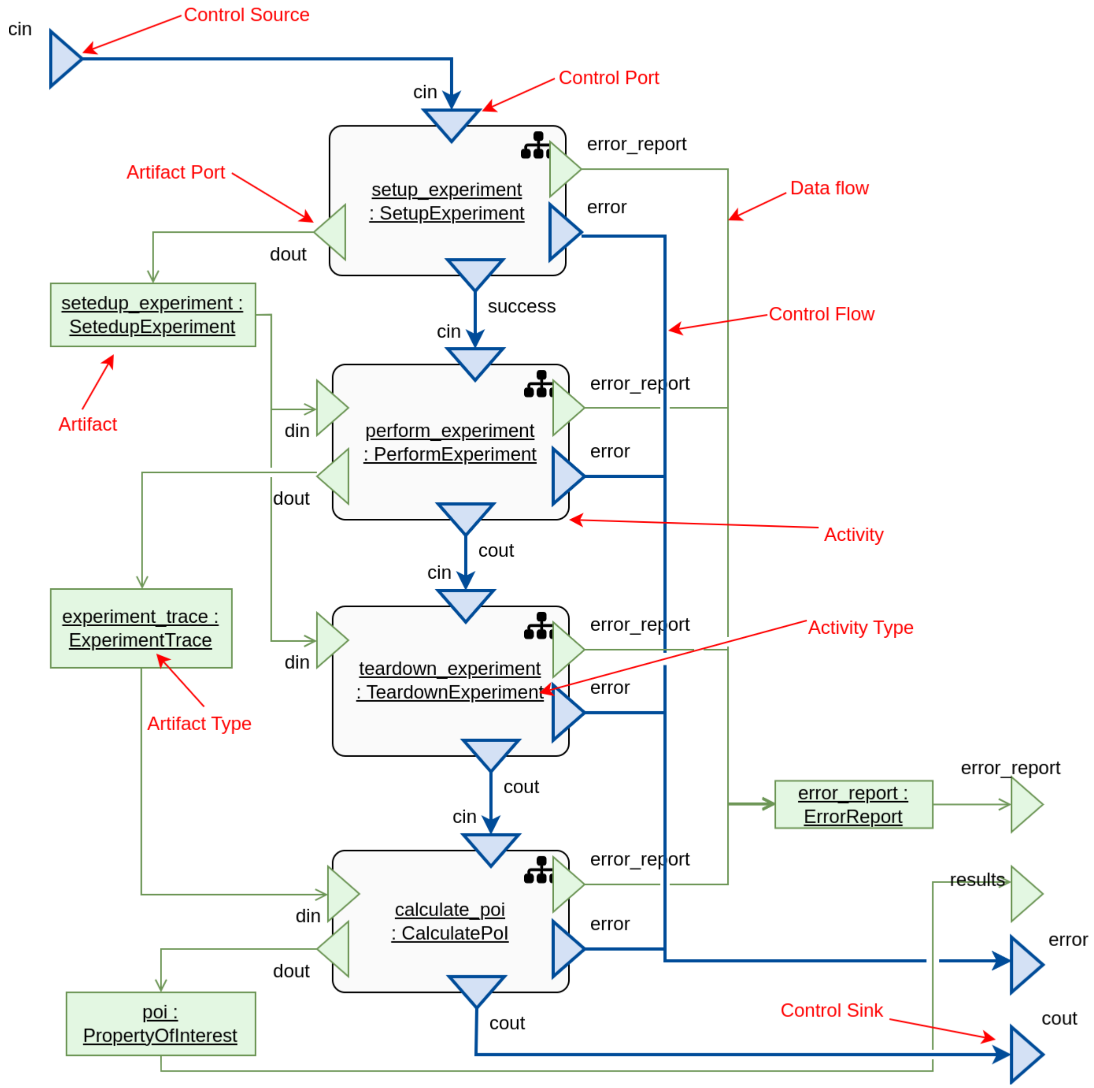
A possible avenue of interacting with the system, possibly by ‘opening’ the black-box and modifying the original system.

Each HLI for a system corresponds to RBGT rules on the architecture and workflow models of the simulation

Workflows and Architectures



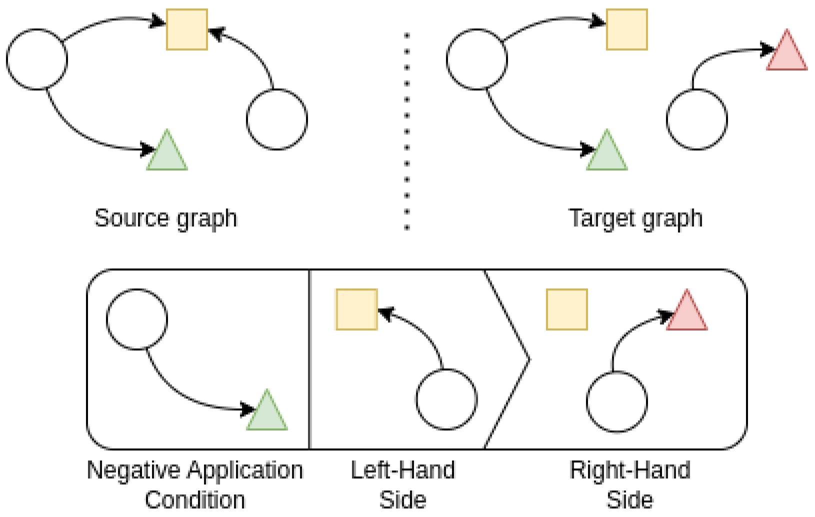
Architectures



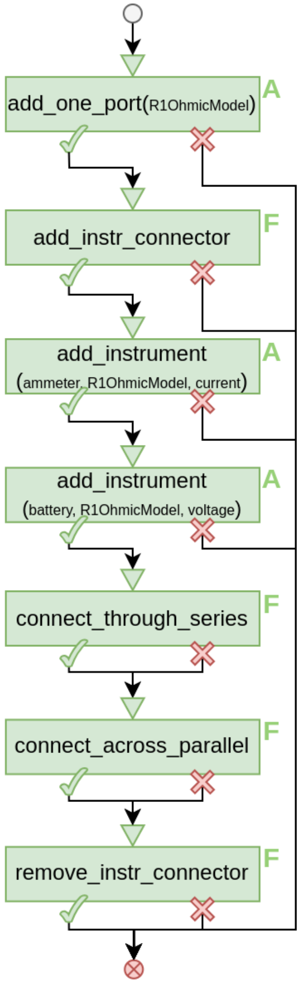
Workflows
(primarily FTG+PM++)

RBGTs and M2T transformations

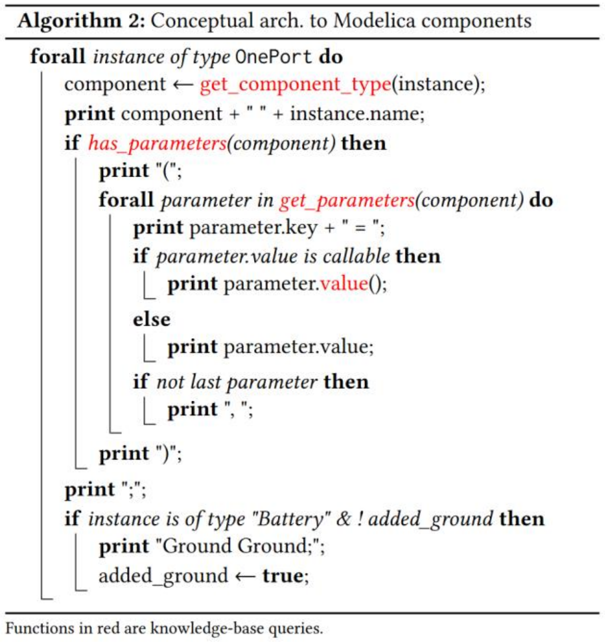
The means to MDE



Exemplary **RBGT**



Exemplary **Schedule**



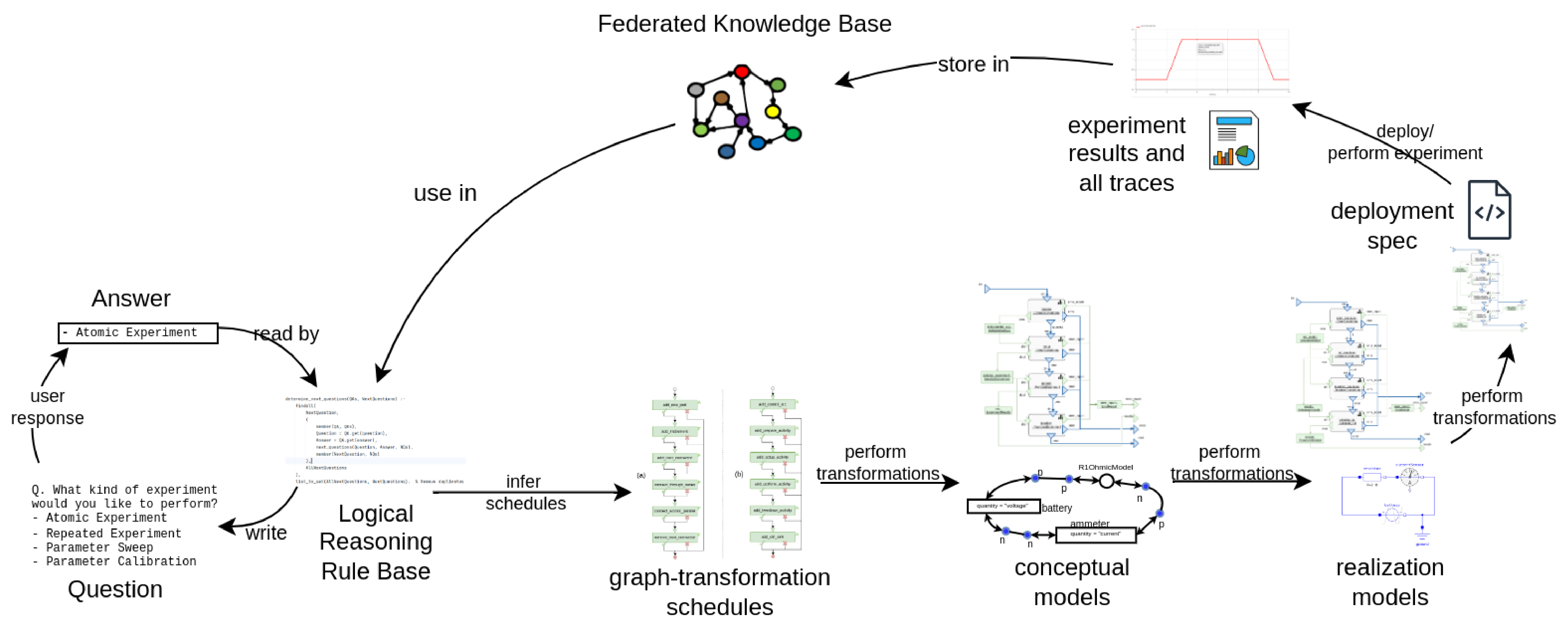
Algorithm 2: Conceptual arch. to Modelica components

```
forall instance of type OnePort do
    component ← get_component_type(instance);
    print component + " " + instance.name;
    if has_parameters(component) then
        print "(";
        forall parameter in get_parameters(component) do
            print parameter.key + " = ";
            if parameter.value is callable then
                print parameter.value();
            else
                print parameter.value;
            if not last parameter then
                print ", ";
        print ")";
    print ";";
    if instance is of type "Battery" & ! added_ground then
        print "Ground Ground;";
        added_ground ← true;
```

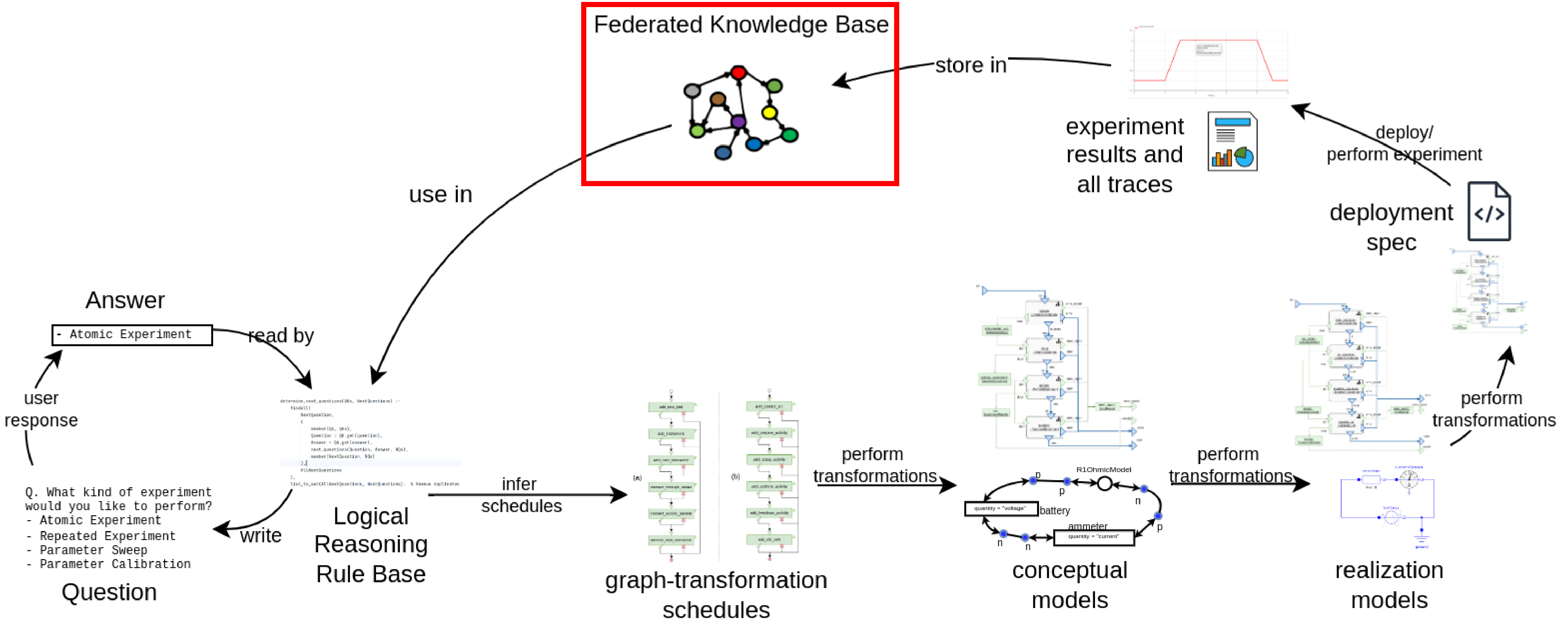
Functions in red are knowledge-base queries.

Exemplary **M2T Transformation** Template

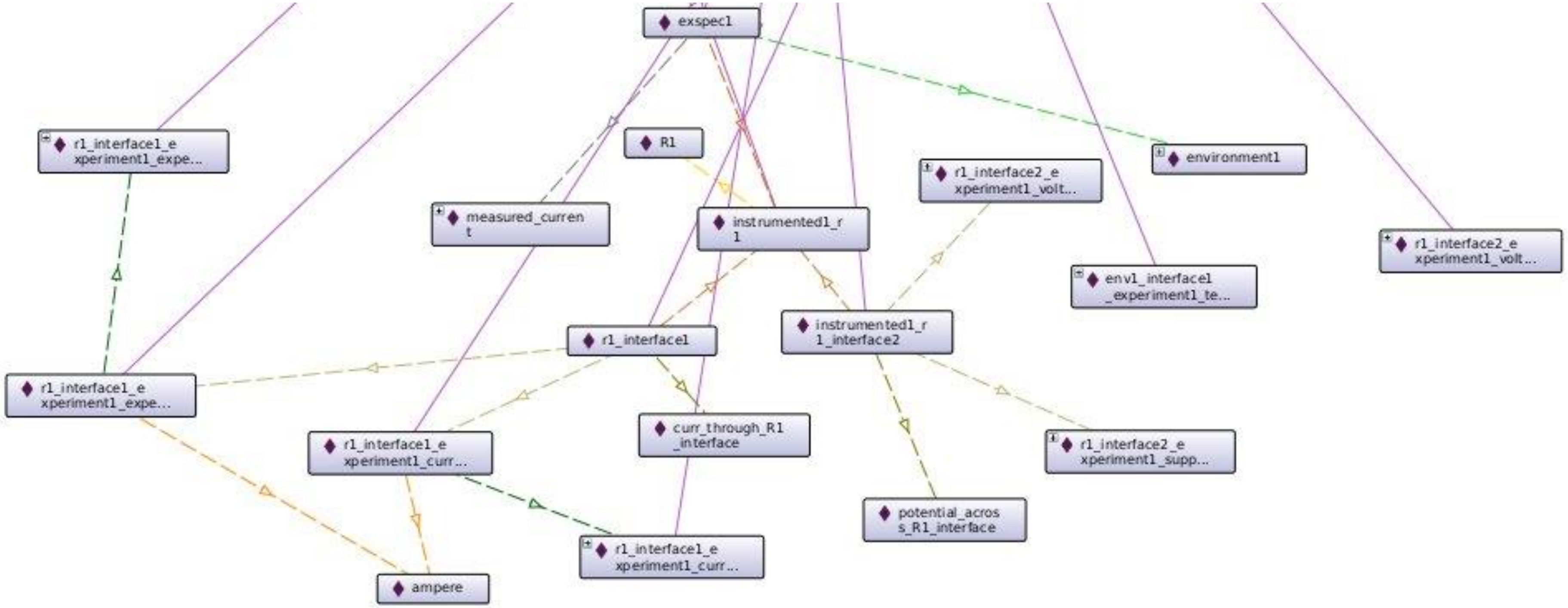
Proposed **Experiment** Management Framework



Proposed **Experiment** Management Framework

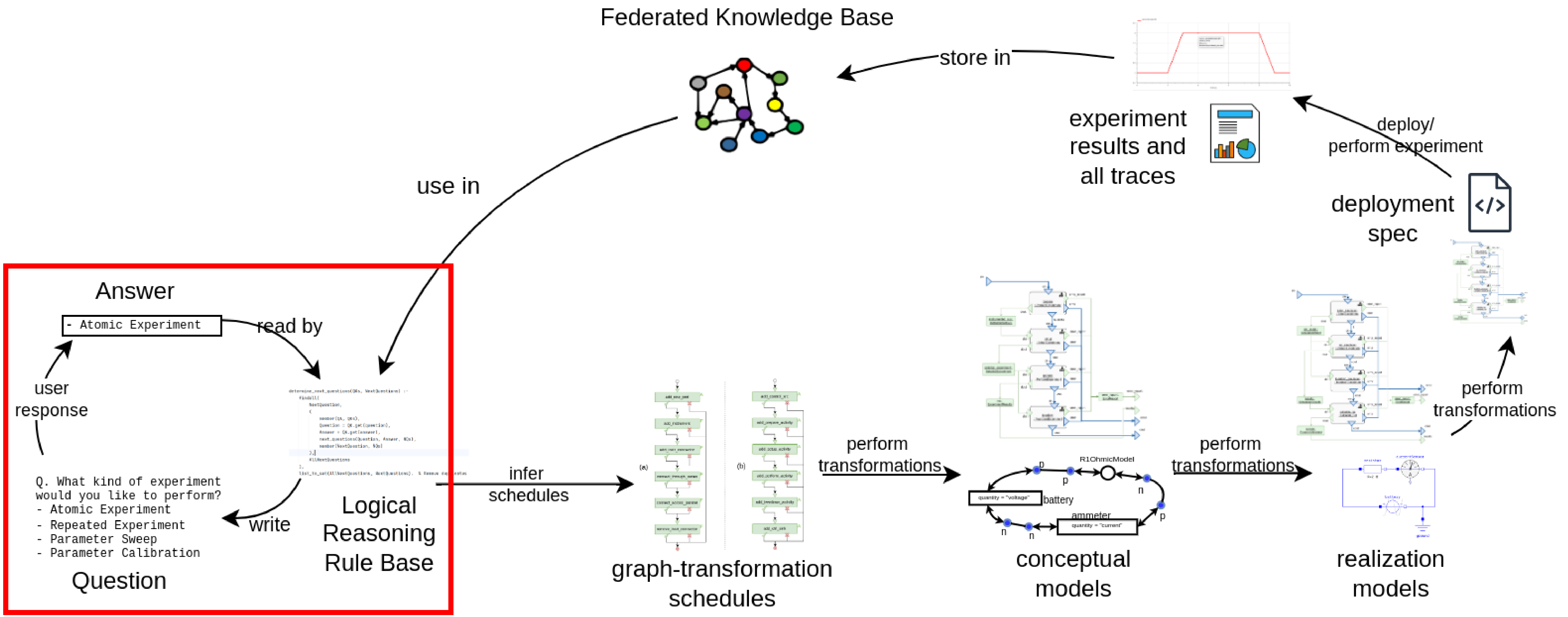


Systems Engineering: Concepts and **Relationships** Knowledge Representation



Small part of an instantiated **experiment specification** in the knowledge graph

Proposed **Experiment** Management Framework



Capture User-Intent with Rule-based Reasoning

SWI Prolog

```
query_systems :-
    sparql_query(
      'SELECT ?X WHERE { ?X a <http://msdl.uantwerpen.be/experiment/vocabulary/core/systems#System> }',
      Row, [ scheme(http), host('localhost'), port(3020), path('/sparql/') ]
    ),
    process_system_result(Row).

ingest_systems(row(X)) :-
    ( X \= '$null$' ->
        assertz(system(X))
    ; true
    ),
    fail.
process_system_result(_).
```

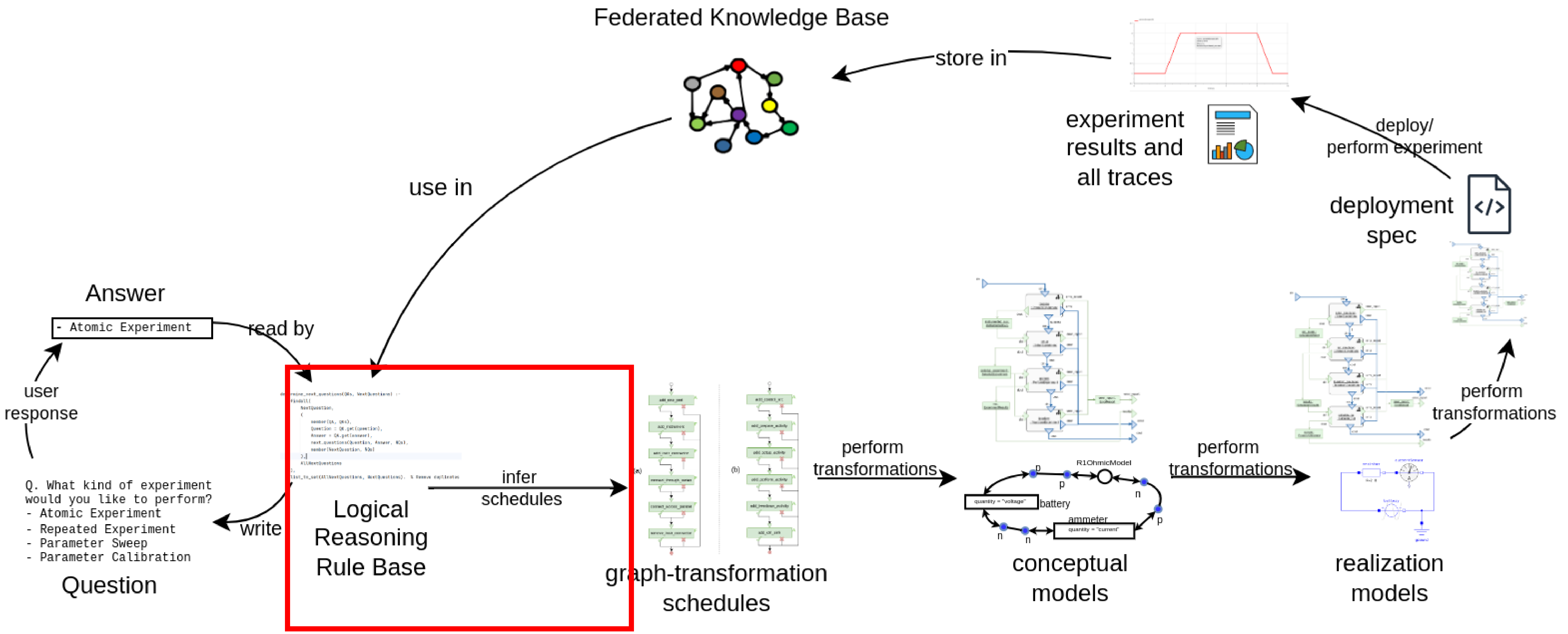
Prolog facts from knowledge graph

```
experiment(atomic_experiment) :- systemUnderStudy(_System), kindOfExperiment(atomic), environment(_Env).
experiment(composed_experiment) :- systemUnderStudy(_System), kindOfExperiment(composed), environment(_Env).
experiment(param_sweep_experiment) :- systemUnderStudy(_System), kindOfExperiment(parameter_sweep),
  environment(_Env).

systemUnderStudy(System) :-
    menuask(systemUnderStudy, System,
            [sysA, sysB, sysC, sysD],
            'Which system do you want to study?').


kindOfExperiment(Kind) :-
    menuask(kindOfExperiment, Kind,
            [atomic, composed, parameter_sweep],
            'Which type of experiment do you want to design?').


environment(Env) :-
    menuask(environment, Env,
            [lab, outdoors, room],
            'What environment will you use?').
```

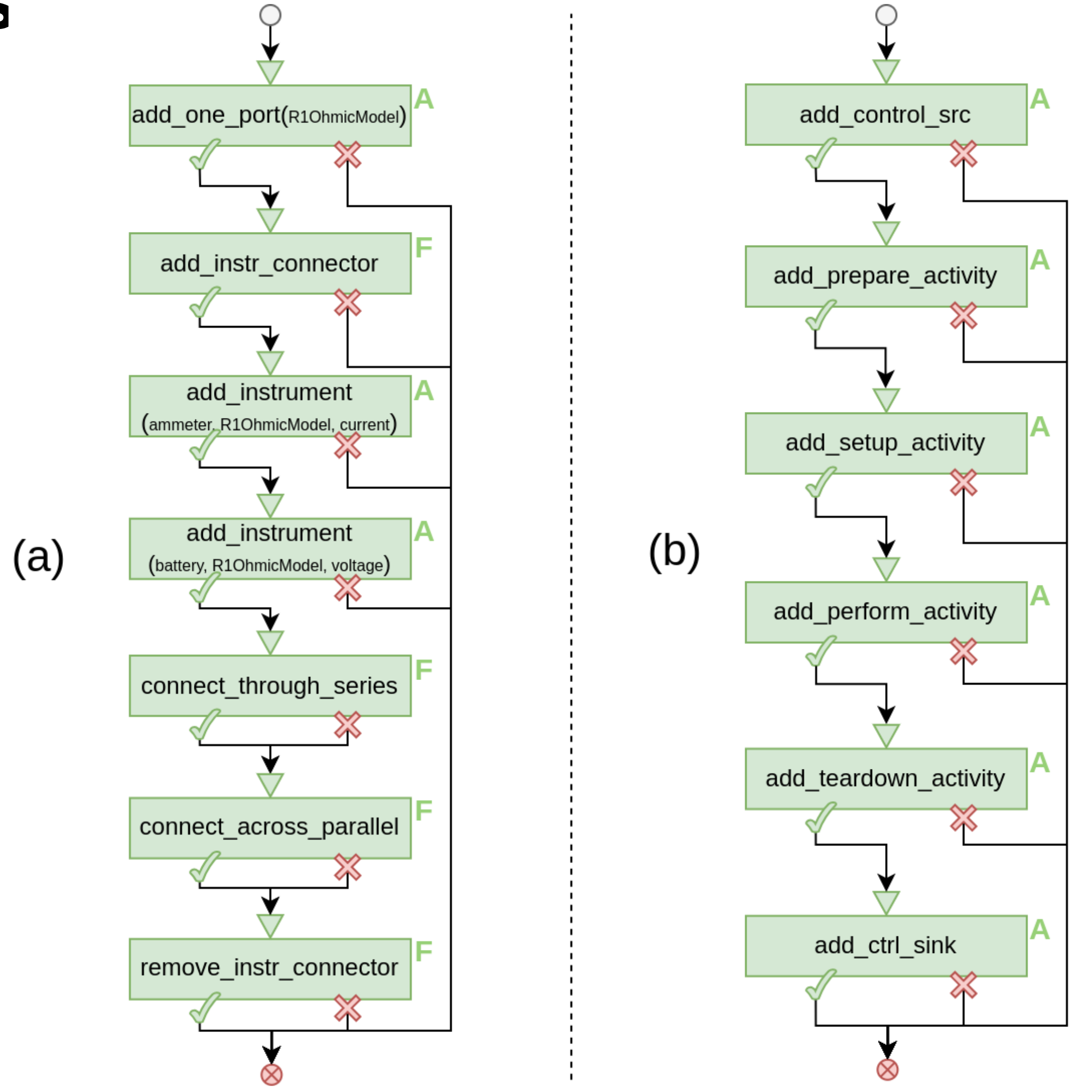
Prolog facts about questions

```
Q. What kind of experiment would you like to perform?
- Atomic Experiment        - Repeated Experiment
- Parameter Sweep          - Parameter Calibration
- Validation Experiment    - ...
Q. What is the system-under-study?
- R1                           - R1OhmicModel
- NotchFilter                  - NotchFilterModel
- NotchFilterModelParasitic    - ...
Q. What is the environment?
- LabEnvironment  - ...
```

Proposed **Experiment** Management Framework



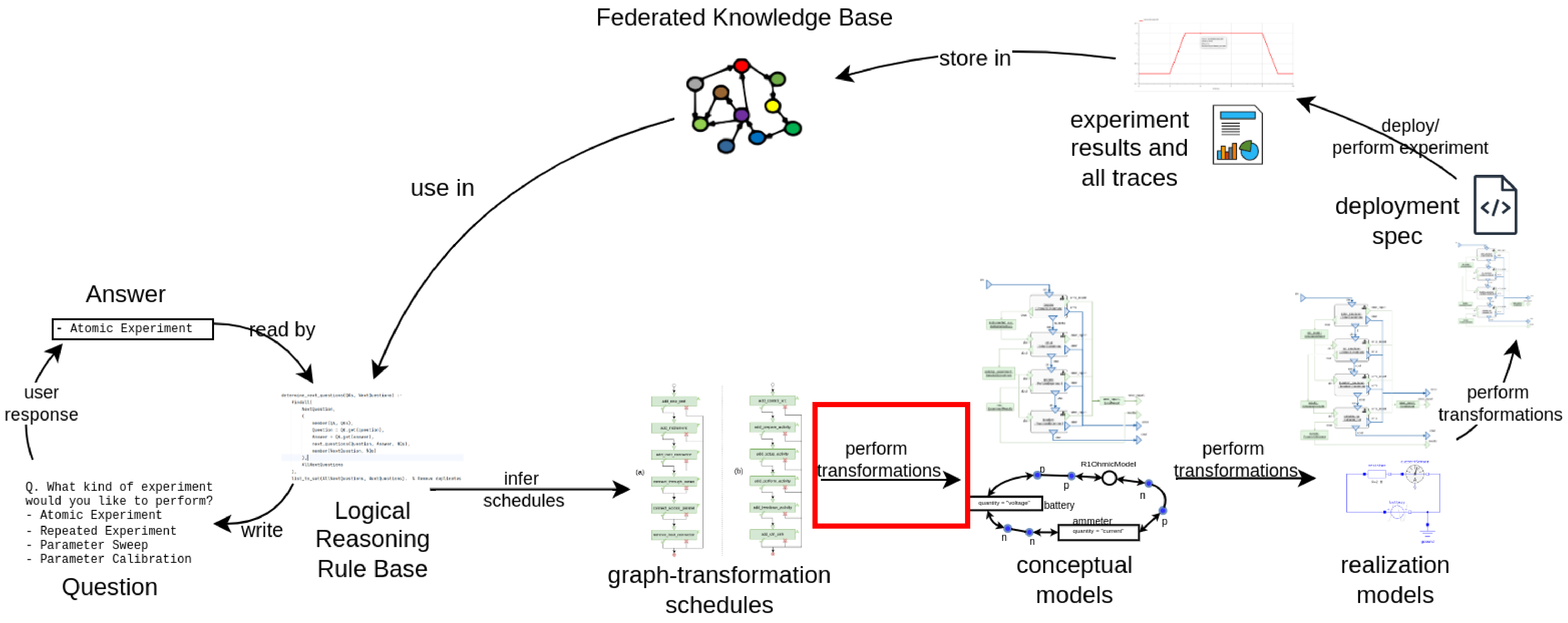
Inferred RBGT Schedules



Schedule to generate conceptual architecture

Schedule to generate conceptual workflow

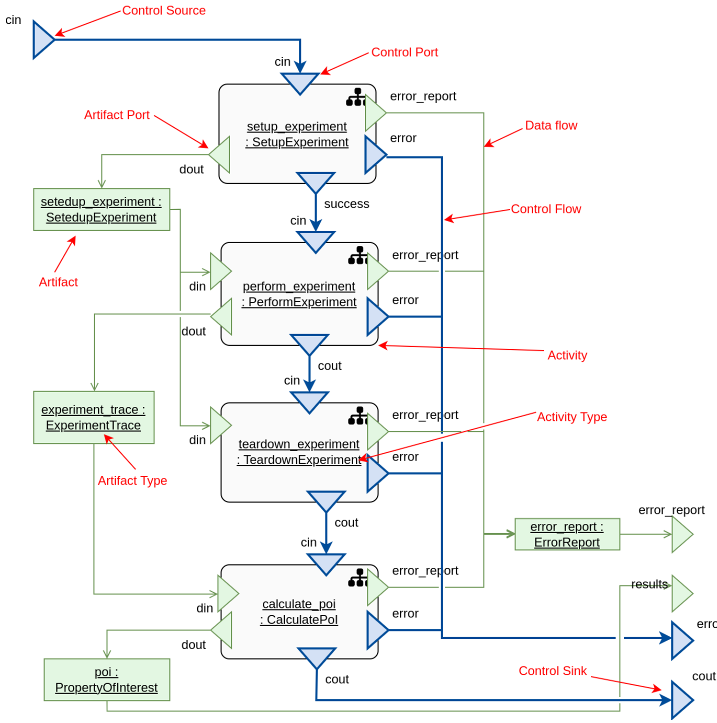
Proposed **Experiment** Management Framework

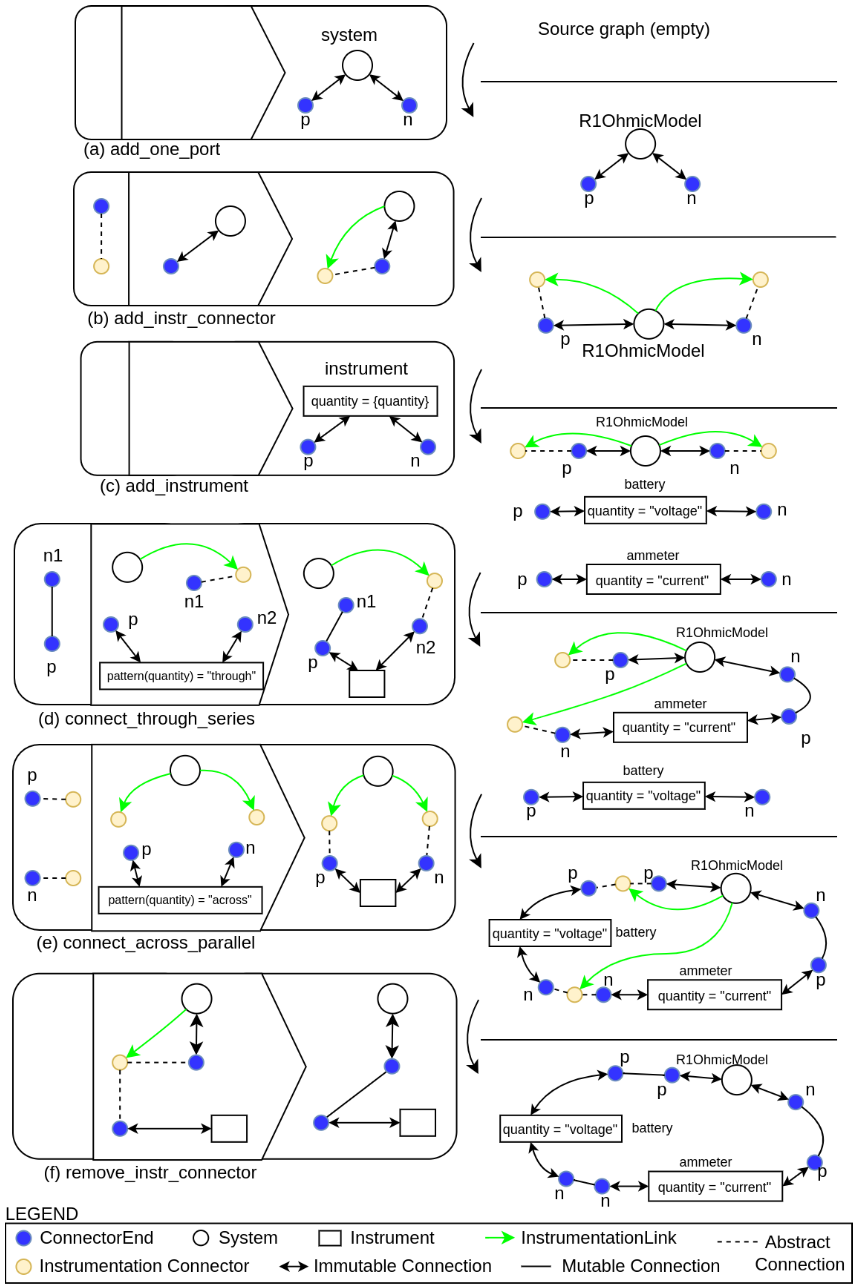


Generate Conceptual Models

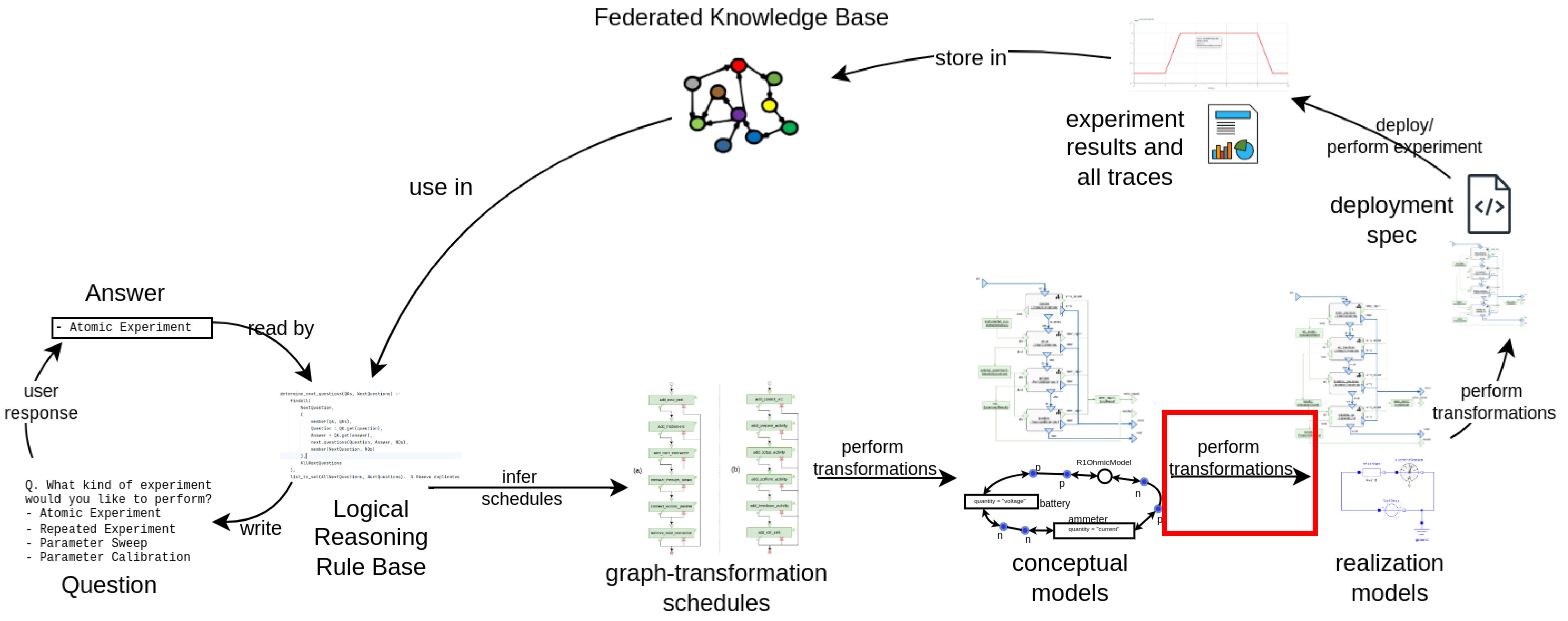
```
Q. What kind of experiment would you like to perform?
- Atomic Experiment          - Repeated Experiment
- Parameter Sweep            - Parameter Calibration
- Validation Experiment      - ...
```

RBGT-based templates for each kind of experiment

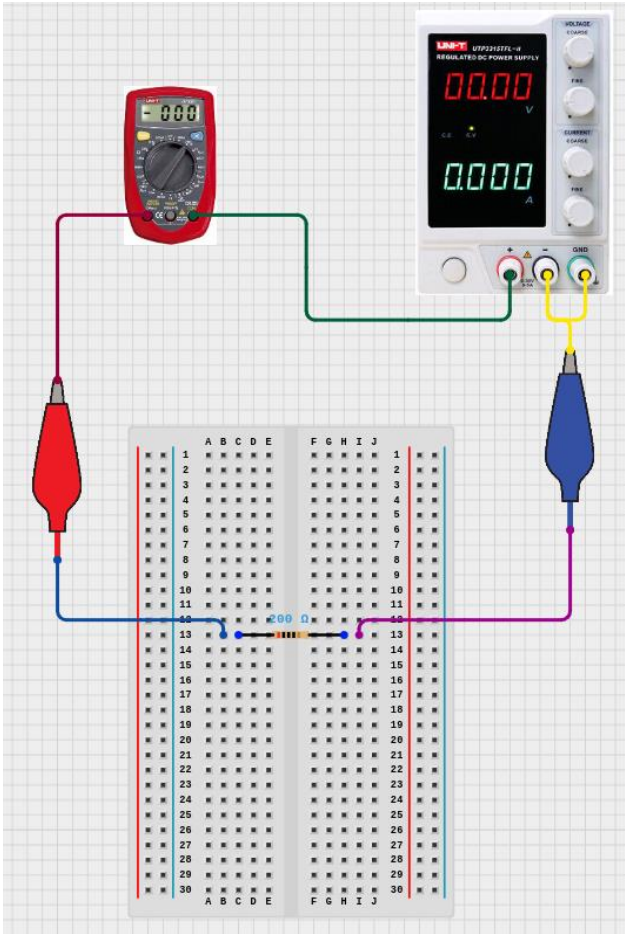


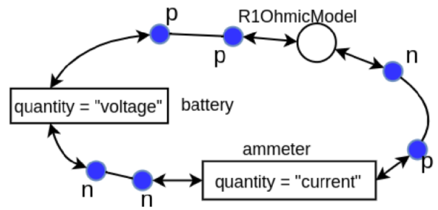


Proposed **Experiment** Management Framework

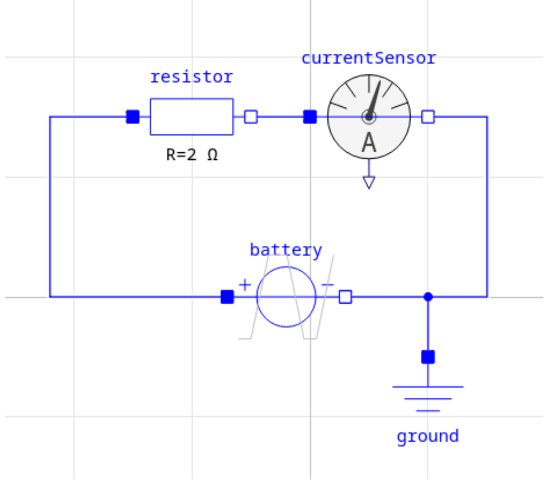


Realizing the Architecture (RBGT + M2T transformation)





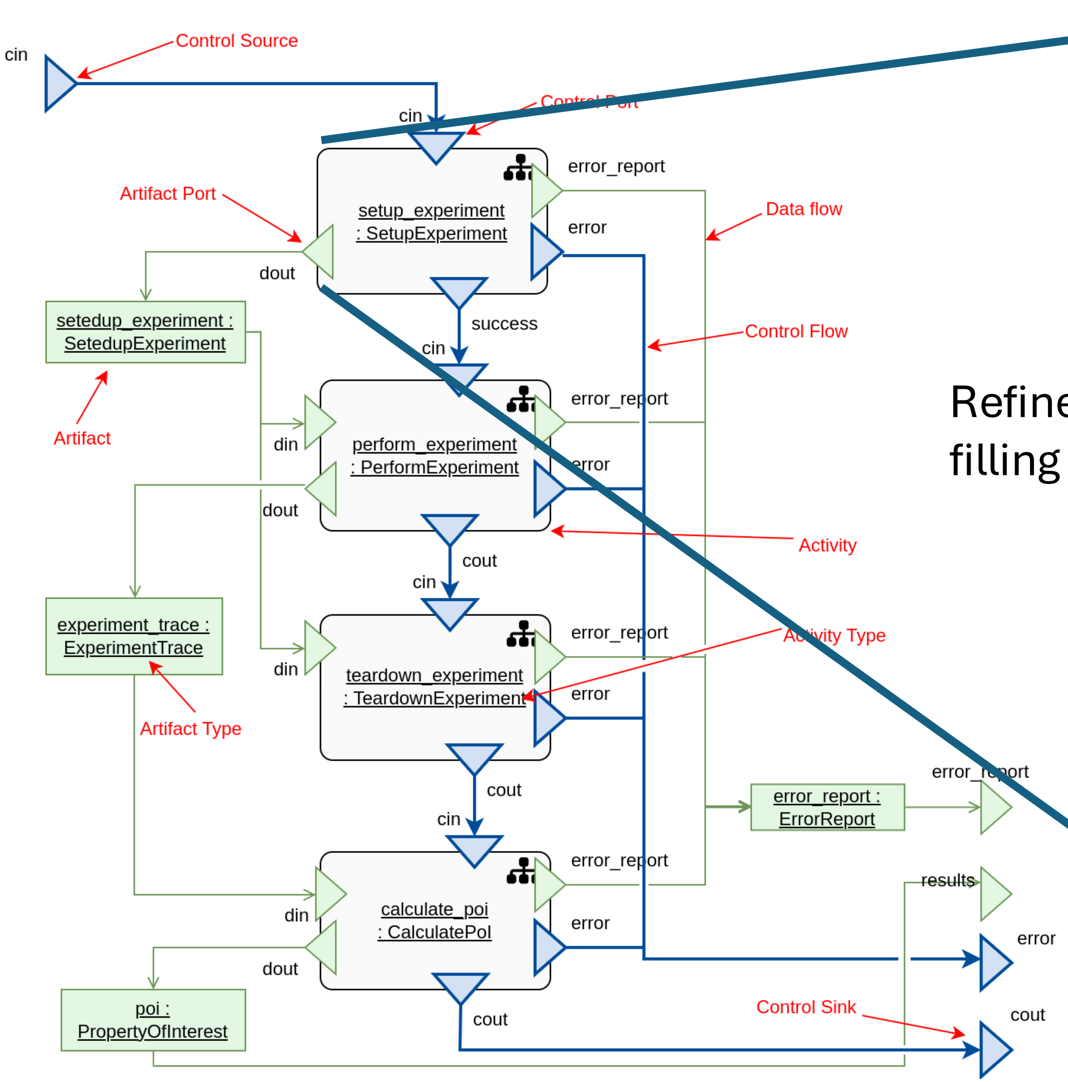
Conceptual Architecture Model



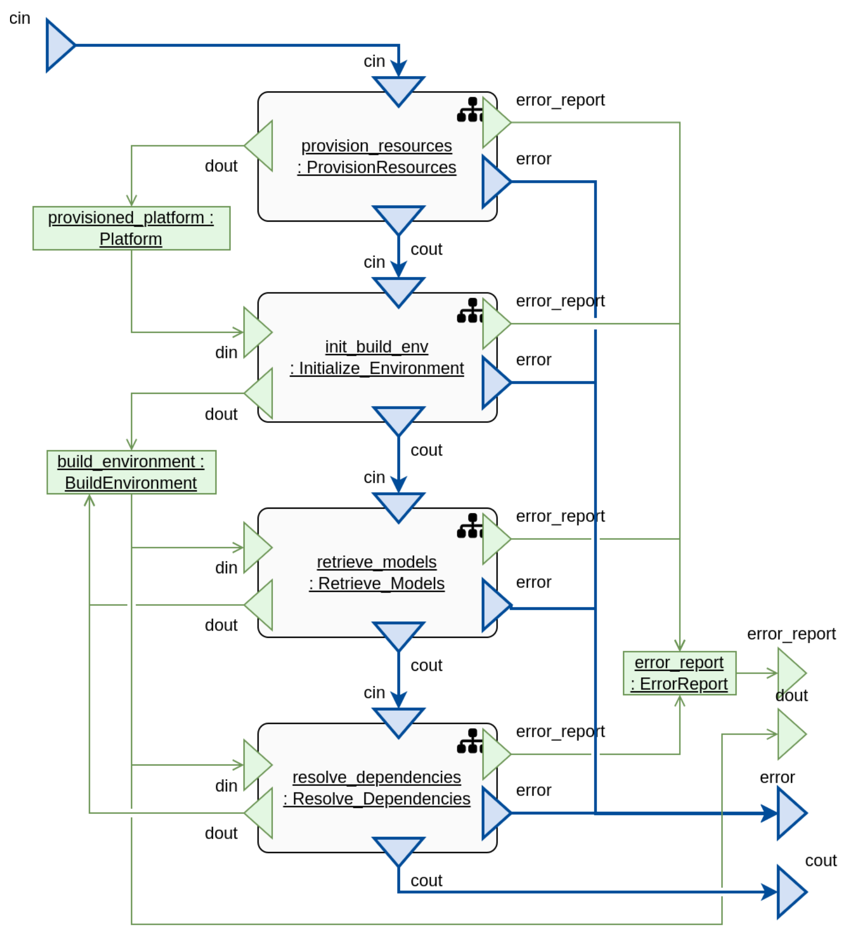
Real-World Realized Architecture Model (schematic with real-world components)

Virtual-World Realized Architecture Model (Modelica)

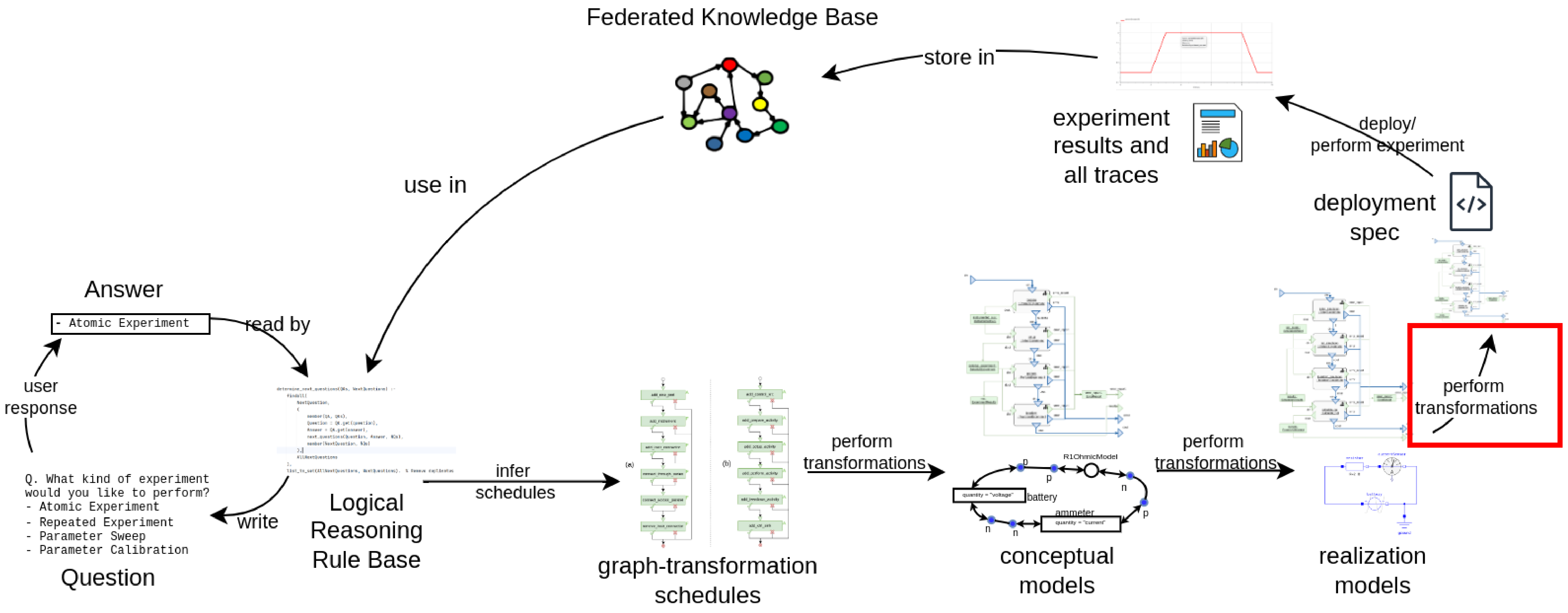
Realizing the Workflow (RBGT + M2T transformation)



Refinement/
filling the gaps

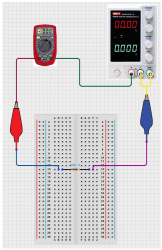


Proposed **Experiment** Management Framework



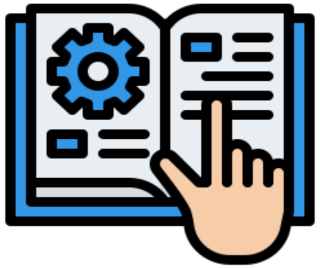
Generate Deployment Specification

Real-World



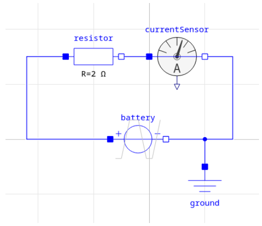
+

Bill of Materials



Work instructions / orders

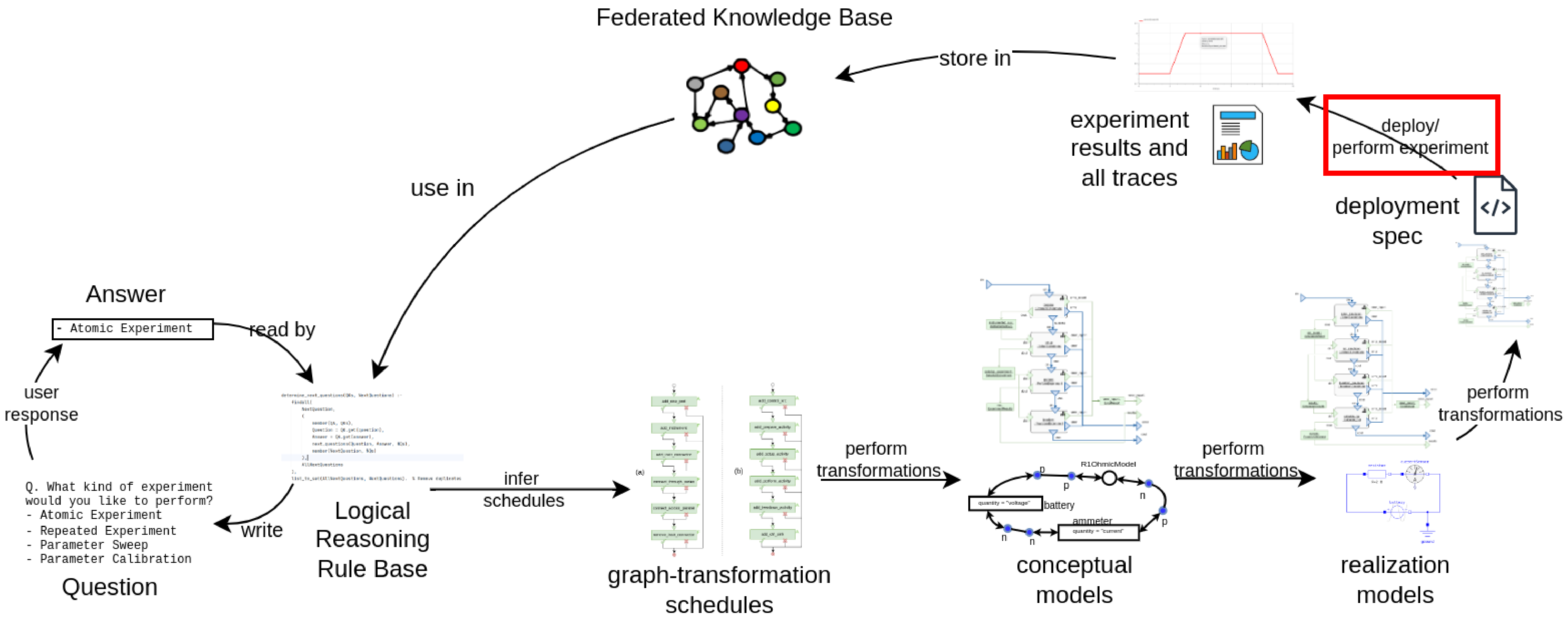
Virtual-World



Modelica
Compiled C-code

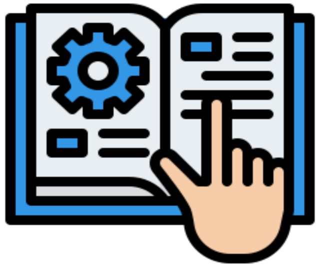
Action code for
experiment
orchestration

Proposed **Experiment** Management Framework

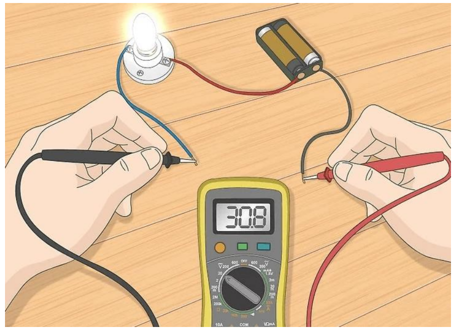


Deployment of Experiments (architecture + workflow)

Bill of Materials



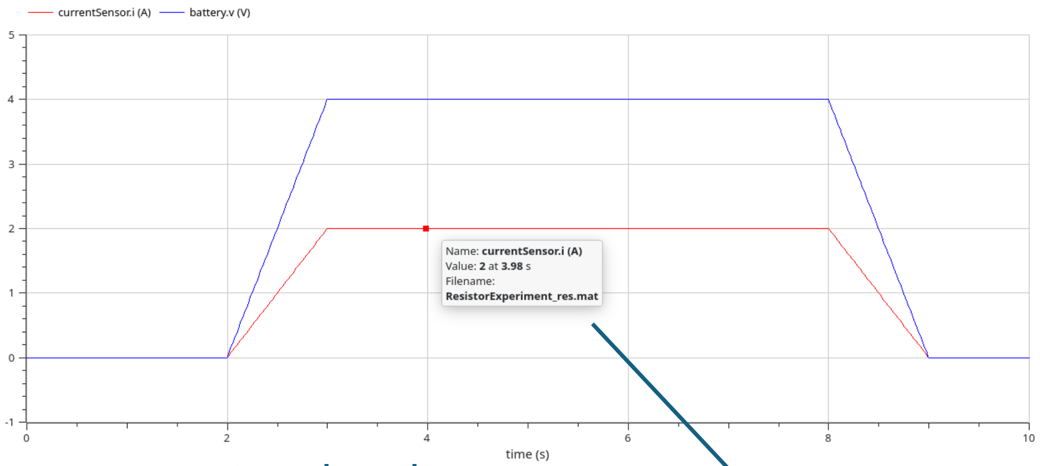
Work instructions / orders



The current is 2 Amperes

Engineer / Robot executes work orders

Simulation / Co-simulation / Orchestration



Ramp-up time   Settling time

The current is 2 Amperes

enactment of workflow

Real-World Deployment ← (and combinations) → **Virtual-World** Deployment

Generated Report (end-to-end)

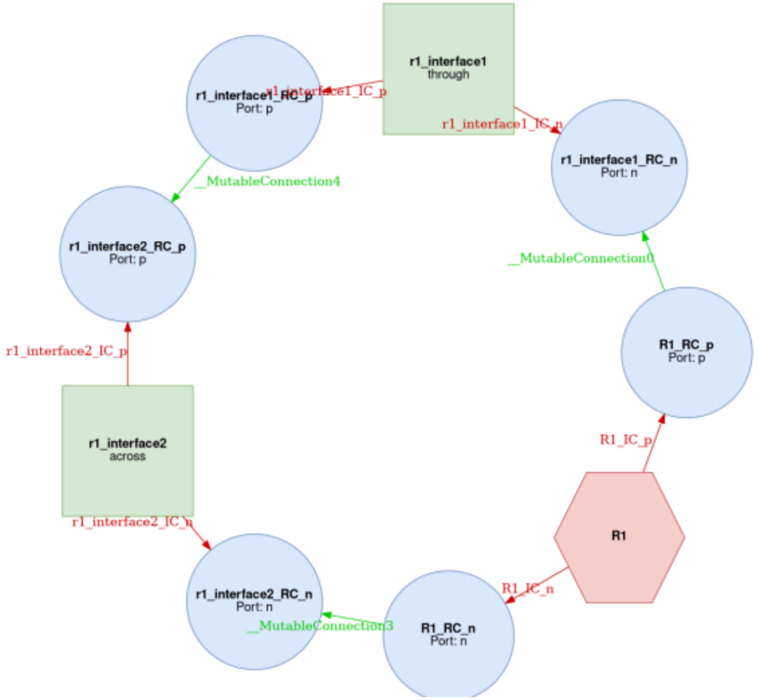
Report

February 12, 2025

Experiment URIs:

- Experiment URI:
  - http://msdl.uantwerpen.be/experiment/description/experiments/experiment1
- Experiment Specification URI:
  - http://msdl.uantwerpen.be/experiment/description/experiments/experiment1#exspec1

Conceptual Model:



Realized Model:

```
model ExperimentModel
    Modelica.Electrical.Analog.Sensors.CurrentSensor r1_interface1;
    Modelica.Electrical.Analog.Basic.Resistor R1(R = 10);
    Modelica.Electrical.Analog.Sources.TrapezoidVoltage r1_interface2(nperiod = 1, V =
        5.0, rising = 1, width = 5, falling = 1, period = 7);
    Modelica.Electrical.Analog.Basic.Ground ground;

equation
    connect(R1.p, r1_interface1.n);
    connect(R1.n, r1_interface2.n);
    connect(ground.p, r1_interface2.n);
    connect(r1_interface1.p, r1_interface2.p);
end ExperimentModel;
```

Listing 1: Realized model

Deployment Specification:

- /tmp/tmp495wpomm

Simulation Trace:

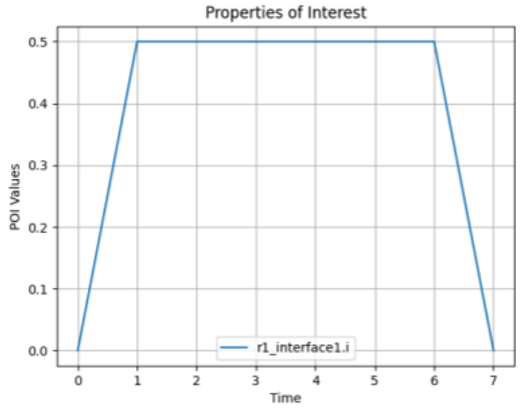


Figure 2: Simulation Trace

Discussion

- Enables truly re-usable experiments
  - Separate ongoing work (formal treatment of experiment re-use logic)

- Experiment specifications serve as basis for validity frames
  - Validity frames are sets of experiment frames that are abstracted from experiment specifications

- Morphism between corresponding real and virtual experiments via a shared conceptual model – validity of validity (higher-order)

- Semantically enriched data can be mined