From SMOL to Digital Twins

Einar Broch Johnsen

University of Oslo, Norway
einarj@ifi.uio.no

IFIP WG2.2
Bologna, 7 September 2023

UNIVERSITY OF OSLO

SIRIUS Center for Scalable Data Access in the Oil and Gas Domain

sfi Centre for Research-based Innovation
The Research Council of Norway

http://www.sirius-labs.no

Talk Overview

- Dichotomy between formalization of structural and behavioral knowledge
- SMOL: Semantically reflected programs
- Applications to digital twins

The Many Shapes of Semantics

Dichotomy in formalization

- **Semantics of control: Formalization of behavioral knowledge**
  SOS, transition systems, …
- **Semantics of data: Formalization of structural knowledge**
  Knowledge graphs, ontologies, semantic web, …

Formalized knowledge representation increasingly used in applications

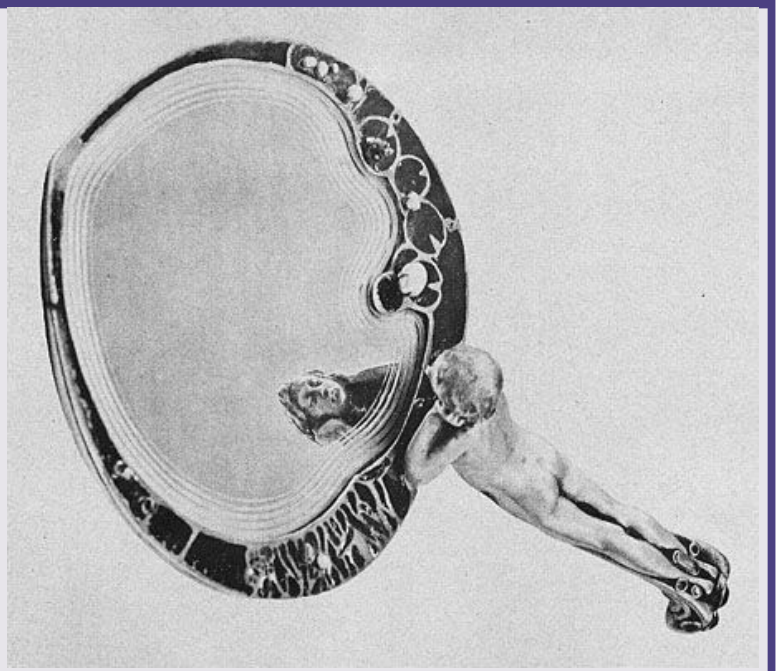
- **Industry**: Industry 4.0, digitalization, asset models, standardization, …
- **Software**: Digital twins, robotics, …

How can these formalizations meet?

- Integrate structural knowledge in behavior models
- Integrate behavior in structural models: semantically lifted programs

Reflection

What is reflection in programming?



- Reflection is the ability of a process to examine, introspect, and modify its own structure and behavior (Wikipedia)
- Lisp, Java, rewriting logic, …
- Reflection used to observe and modify program execution at runtime
- Representation of program as terms in the programming language: meta-level terms, towers of interpreters, upTerm(…), downTerm(…)
- Need a lot of machinery to inspect and manipulate these terms!
- Fun, … but easy to make mistakes!

Semantic Reflection

Can we use semantic technologies for reflection?

Semantic technologies

- **Semantic technologies:** languages and tool that support inference when querying knowledge representation: SPARQL, SHACL, etc
- Made for formalization of complex knowledge structures

Reflecting programs into a structural model

- **Semantic lifting**: Integrate meta-level terms with domain knowledge
- **Semantic reflection**: Let the programs query their own meta-level domain knowledge

Programming in SMOL

Programming with a behavioral and a structural layer

SMOL: Semantic Model Object Language [ESWC 2021, ESWC 2022]

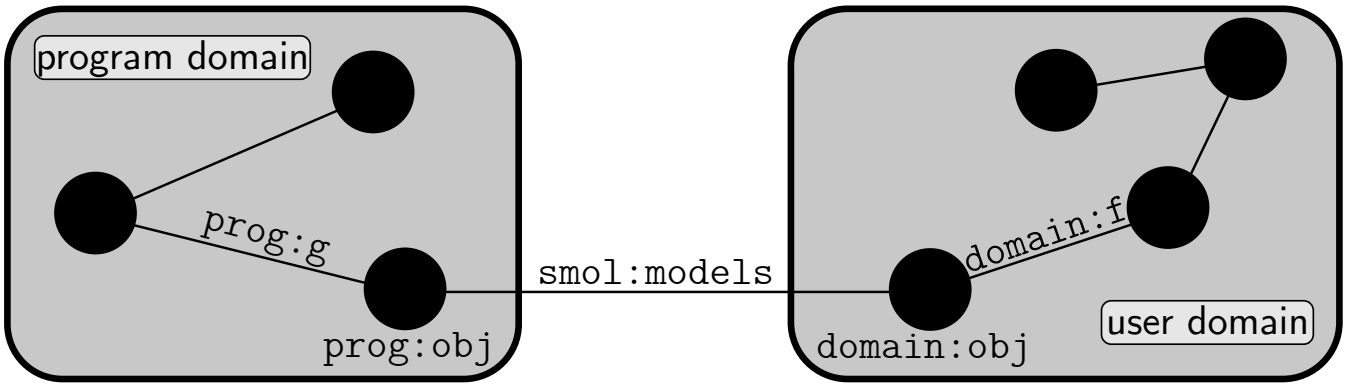
- SMOL is a small OO programming system which supports reflection into knowledge bases
- Runtime states in SMOL are automatically lifted into a KB, and integrated with domain knowledge formalised using ontologies
- Ontology reasoners allow querying the KB
- SMOL programs can use reasoners to query the KB about themselves

- Source code + documentation: https://smolang.org/
- Formalization almost done: SOS, simple type system for queries to KB
- Examples of use, inspired by digital twins

SMOL Syntax

Prog	::= $\overline{\text{Class}}$ **main** Stmt **end**	Programs
Class	::= **class** c [**extends** c] ($\overline{\text{Field}}$) [Models] $\overline{\text{Met}}$ **end**	Classes
Type	::= t | c | List<c> | List<t>	Types
Field	::= [**hidden** | **domain**] Type f	Fields
Models	::= $\overline{\textbf{models}(\text{Exp})\ \text{be};}$ **models** be;	Modeling bridges
Met	::= Type m($\overline{\text{Type v}}$) Stmt **end**	Methods
Stmt	::= Loc:=RHS; | **if** ... | **while** ... | **return** Exp;	Statements
	| Exp.m($\overline{\text{Exp}}$); | **skip**; | Stmt Stmt	
RHS	::= **new** c [<$\overline{\text{Type}}$>] ($\overline{\text{Exp}}$) [Models] | Exp.m($\overline{\text{Exp}}$)	RHS expressions
	| **access**(sparql, $\overline{\text{Expr}}$) | **member**(owl)	
	| **validate**(shacl) | Exp	
Exp	::= **this** | **null** | Loc | Exp *op* Exp	Expressions
Loc	::= Exp.f | v	Locations

Interacting with Formalized Domain Knowledge

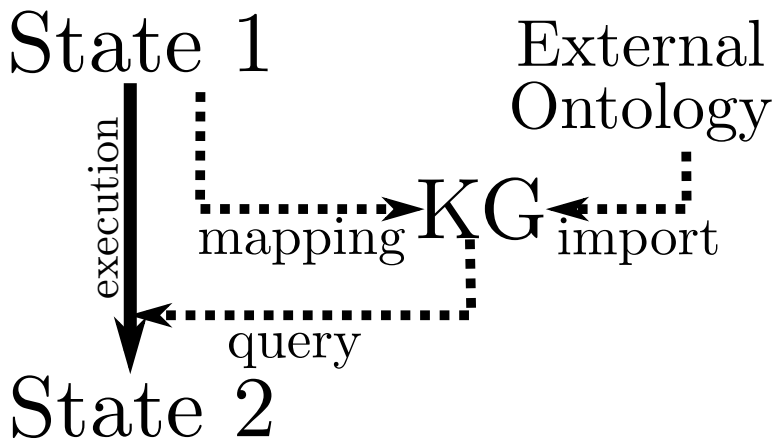


class D() **models** (exp) be
class C(Int g, **domain** Int f) **end**

Modeling bridge

- **Program domain**: KB includes an axiomatization of the runtime system:
- KB may contain *more* than just knowledge of the runtime system
- **User domain**: Formalized knowledge about the application domain
- **Modeling bridge** dynamically connects runtime entities with user domain

Executing semantically reflected programs




```
main
  List<Int> results
   = access( "SELECT ?obj {?a a asset:Room. ?a asset:id ?obj}");
  while results != null do
    Int current = results.content;
    results = results.next;
    print(current);
  end
end
```

Formalization of SMOL

Meta-theory

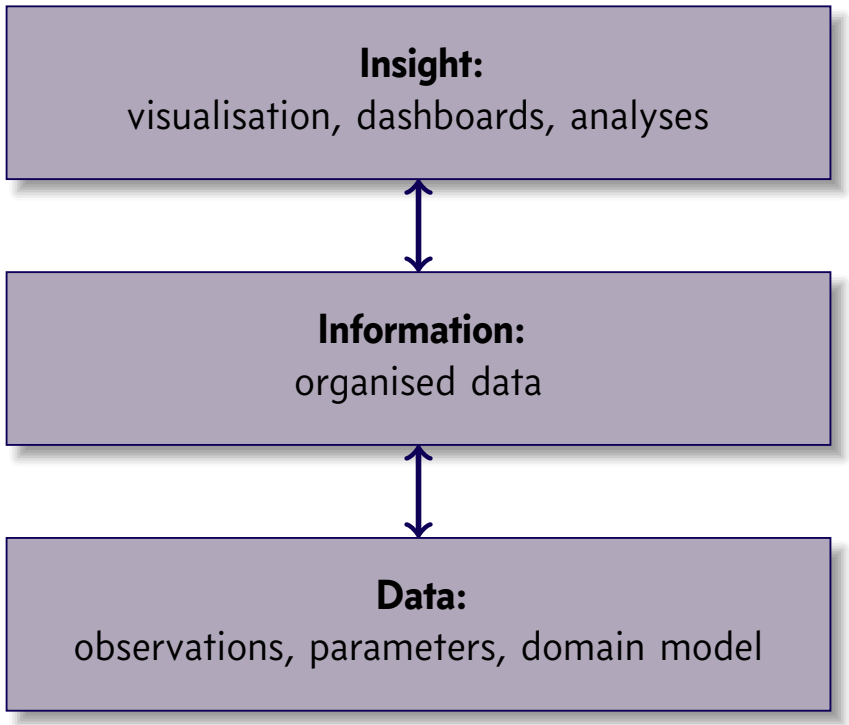
- Standard SOS rules, with KB as a component of runtime states
- Formalize the lifting of runtime states
- Answering queries first lifts the runtime configuration

Types for semantic reflection

- **Representation Failure**: Return values from **access** should have runtime representations
- **Location Failure**: Return values must have the expected types
- **Inconsistency**: Query answering only defined for consistent KBs

Subject reduction: Reachable states lift into consistent KBs

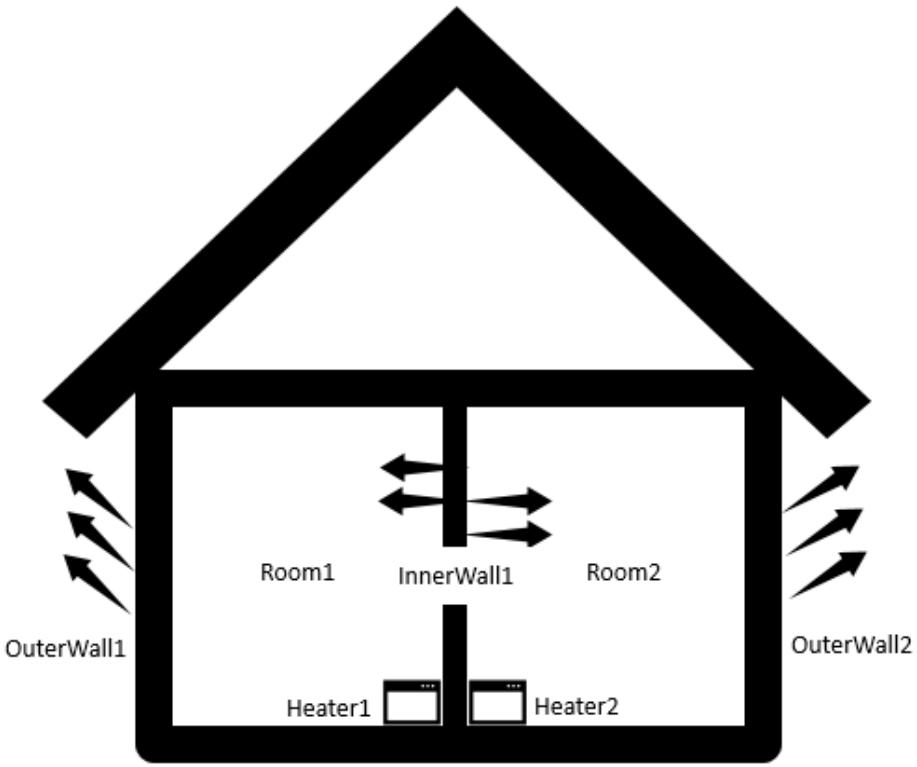
Digital Twins: Conceptual Layers



Behavioral twins in SMOL [AnnSim 2022]

- SMOL can encapsulate simulation units based on the FMI standard
- Using semantic reflection in SMOL, the runtime configuration of the behavioral twin is automatically lifted into the KB

Compositionality & Co-Simulation



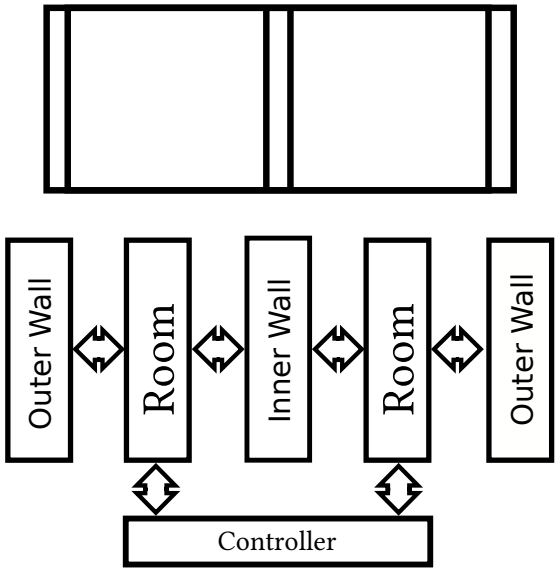
Structural twin

- **Domain knowledge:** connects the rooms, heaters, walls into a "house", with corresponding simulators, etc
- **Asset model:** instance of the domain knowledge for a particular house
- **Analyser model:** instance of domain knowledge for the analyser configuration

Behavioral twin

- **Analyser:** orchestrated simulators corresponding to the asset
- **Reasoner** configures orchestration of simulators

Twinning the House



Twinning the house

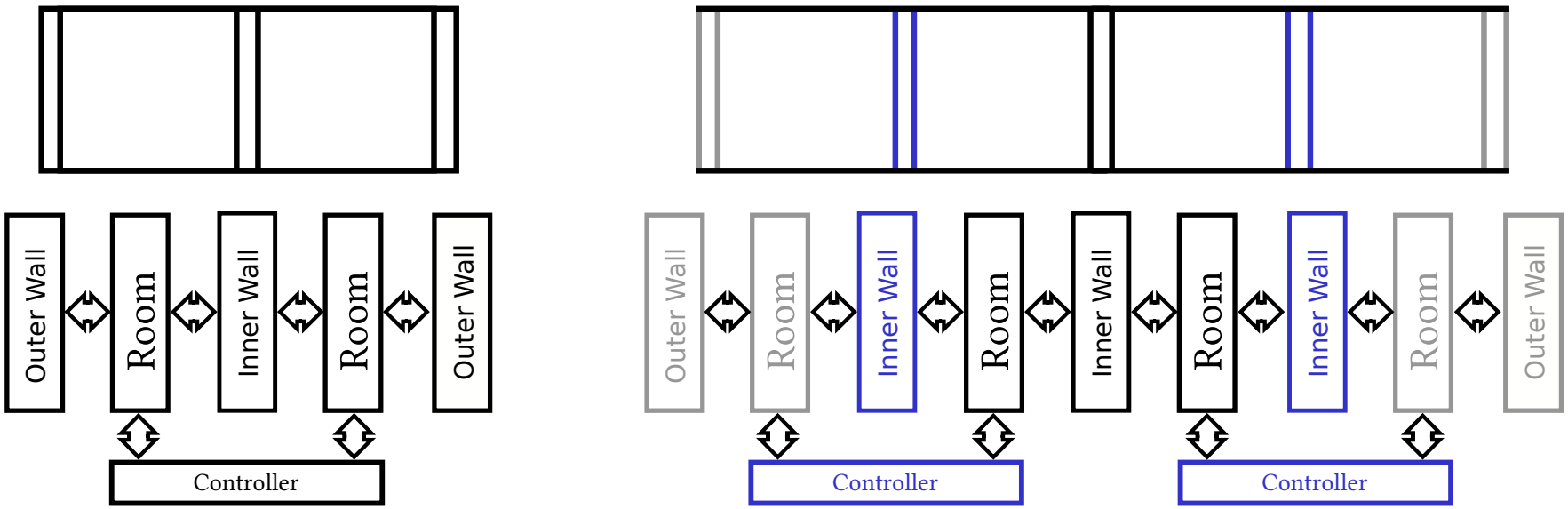
1. The asset model specifies simulators for the different physical components
2. The behavioral twin adds a controller to adjust heaters of adjacent rooms

Correctness of the behavioral twin

We can relate the structure of the asset to the structure of the behavioral twin:

- The components and structure of the asset are exactly mirrored by the twin

Structural Evolution of the Asset



Extending the house

1. New rooms are added to the house
2. Twin needs to reconfigure the simulation model and replace the controller

Structural evolution of behavioral twins [ISoLA 2022]

Idea: Use the structural twin to detect the structural drift between asset and twin as a basis for model repair of the behavioral twin

Detecting Changes in the Asset

Interacting with the structural twin

- Query to detect changes between the asset and the simulation model

```
class Room(String room, String wallLeft, String wallRight) end
  ....
  List<Room> newRooms =
   construct("
   SELECT ?room ?wallLeft ?wallRight WHERE
   { ?x a asset:Room;
       asset:right [asset:Wall_id ?wallRight];
       asset:left [asset:Wall_id ?wallLeft]; asset:Room_id ?room.
     FILTER NOT EXISTS {?y a prog:Room; prog:Room_id ?room.} }");
  if newRooms != nil then // if newRooms == nil then no update is needed
   if newRooms.length() != 2 then /* report error */
   else
    Room n1 = newRooms.content;
    Room n2 = newRooms.next.content;
   end end
```

Evolving the Behavioral Twin

Identifying structural drift

- Both rooms to the left of the old house (see example below)
- Both rooms to the right
- A room on either side

```
if  n1.wallLeft == n2.wallRight
    & n1.wallRight == house.firstRoom.wallLeft.id
then house.addTwoRoomsRight(n1.wallLeft, n1.room, n2.wallLeft, n2.room);
```

Reconfiguring the behavioral twin

1. **Create the new simulation elements** and insert them into the structure.
2. **Repair virtual elements** that are not reflecting elements in the asset
3. **Validate result:** Using reflection, we can check that the behavioral twin now mirrors the asset by a SHACL query into the structural twin

Twinning a "Production Plant" in the Lab

Digital Twin for Chilli Plants

1. **Information Level:** data from sensors (humidity, light) and actuator (water pump)
2. **Insight level:** control system for water pump

From Twinned Plant to Twinned Greenhouse

Digital Twin of a Greenhouse

1. **Information Level:** greenhouse asset model, plant knowledge
2. **Sensors:** humidity, light, temperature,...
   Actuators: water pumps, lamps, ...
3. **Insight level:** control system to optimise plant health & growth

Utility Function: Plant Health Analysis

Normalised Difference Vegetation Index

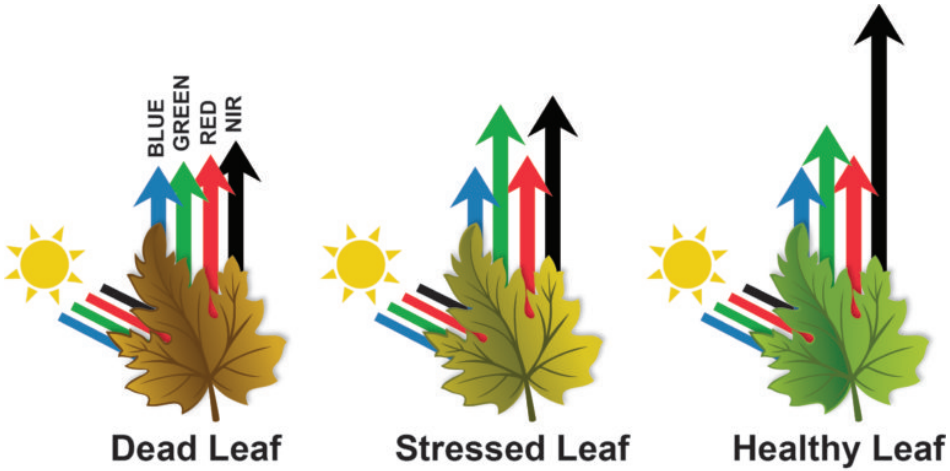
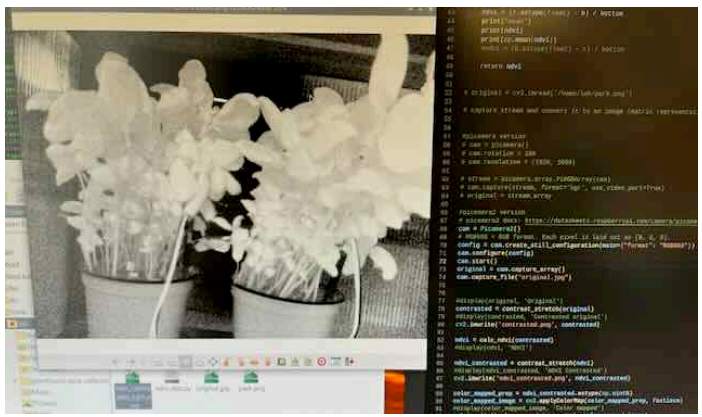


Illustration:

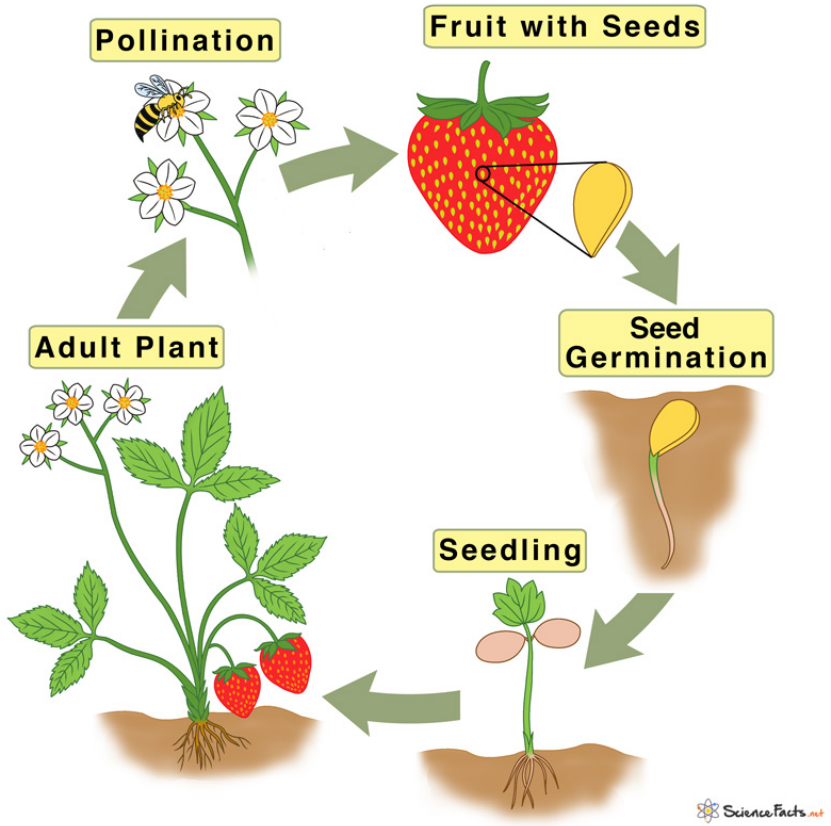
Raspberry Pi Foundation



"Monitoring" the Plant

- Use sensor analysis to "understand" the plant
- Can we measure the healthiness of the plant?

Plant Life Cycle



"Monitoring" the Plant

- Use image analysis to "understand" the plant
- Can we determine the stage of the plant in the plant life cycle?

Programming with Semantic Reflection

Dichotomy in formalization

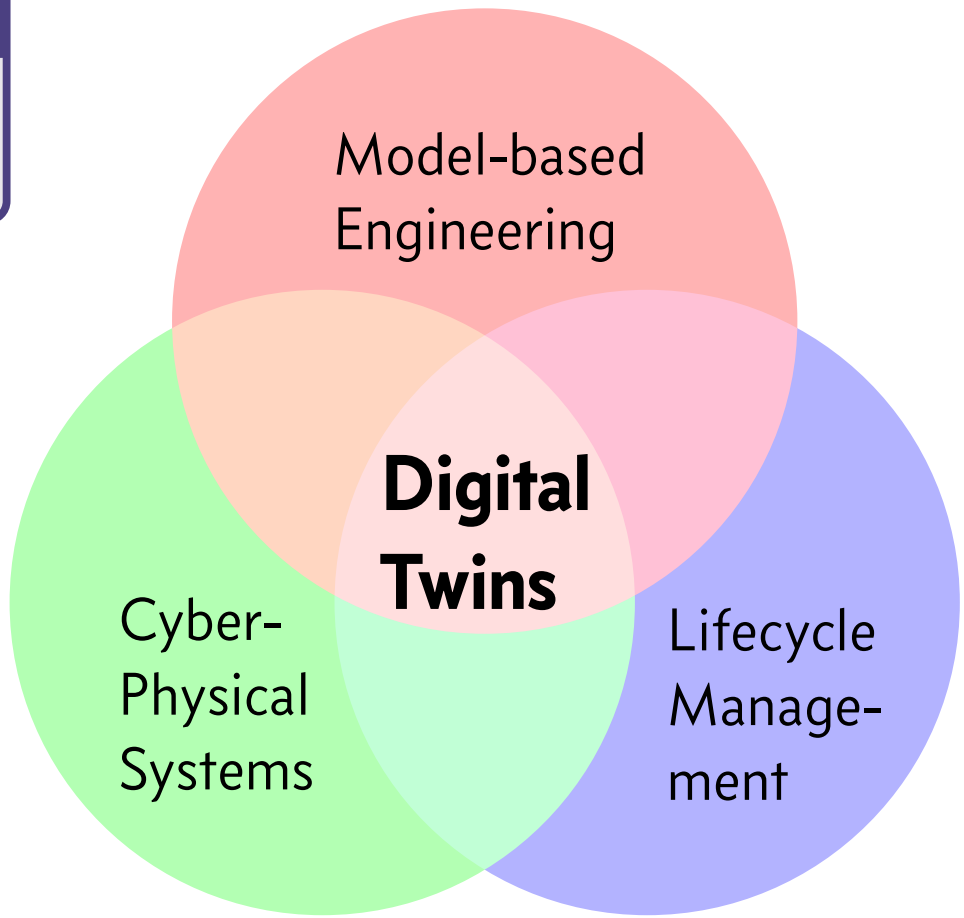
- Formalization of behavioral knowledge
- Formalization of structural knowledge

SMOL

- Underlying idea: integrate domain knowledge through reflection
- Enables reasoning about behavior in domain vocabulary
- Application to digital twin systems

Beyond SMOL

- Semantically Reflected Java
- Semantic debugging
- **Abstraction, concurrency**



Contributors: Eduard Kamburjan, Lizeth Tapia, Rudi Schlatte, Martin Giese, Egor Kostylev, David Cameron, ...