RobMoSys

RobMoSys Building Blocks: Modelling Motion, Perception and World Model Stacks

Herman Bruyninckx, Enea Scioni
KU Leuven

2nd RobMoSys Brokerage Day

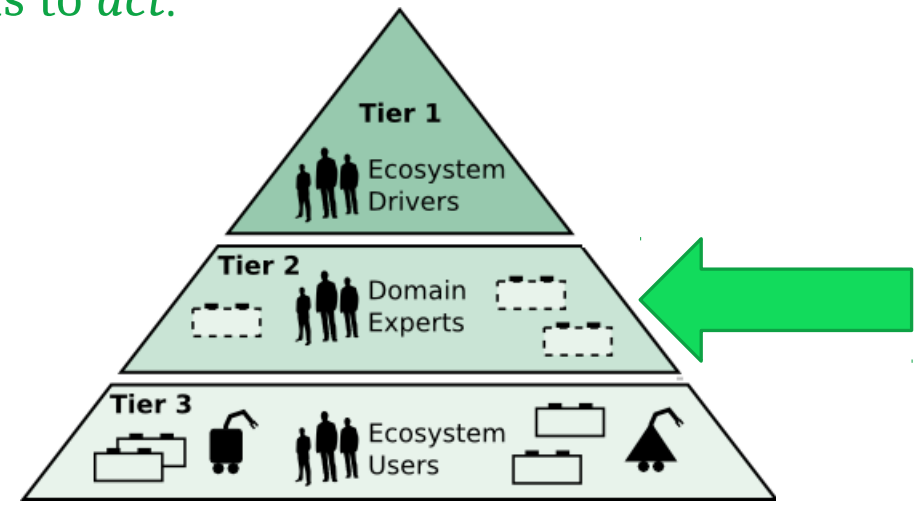
Frankfurt, August 24th, 2017

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732410

Building Blocks:
Motion, Perception and World Model Stacks

RobMoSys

Basic functionalities that allows the robot systems to *act*.



- **Digital Platform**
  - covering all possible robotic systems
  - only 100% well-consolidated functionalities
- **Composability**
  - of algorithms: data + function + control flow
  - *“composition over inheritance”*
  - sequential, concurrent, and distributed programming
- **Mechanisms and policies**
  - structure = graphs, tools = graph traversal
  - behavior: kin. & dyn. functionalities/solvers
  - no magic numbers: all structure & behaviour parameters in models

Modelling Principles

Motion, Perception and World Model stacks are about ***functionalities***

- **Composability:** new functionalities are created by composing existing ones, not by inheriting their behaviour
- **Reusability:** minimize duplication in different contexts
- **Usability:** easy to use, intuitive, user-friendly

(Who are my users?)

Models should be **designed** for **composability** and **reusability**.

Usability is provided by **tools** and **domain-specific languages**

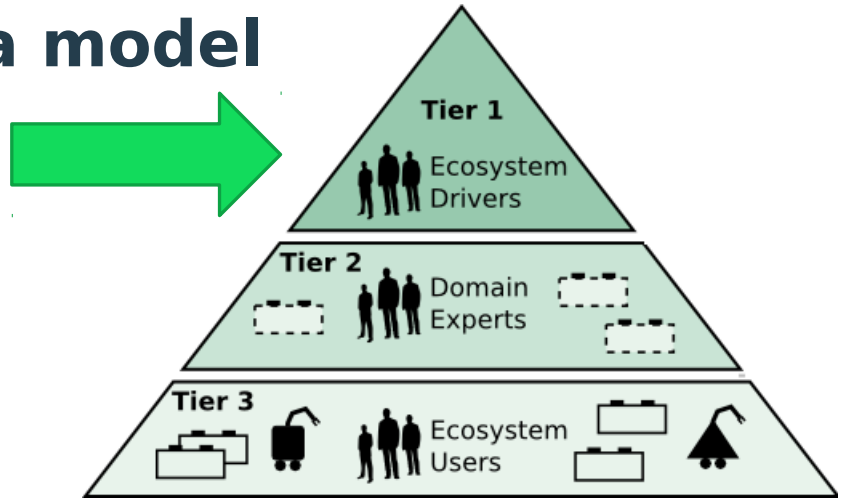
Composability at functional level is different from composability at component level:

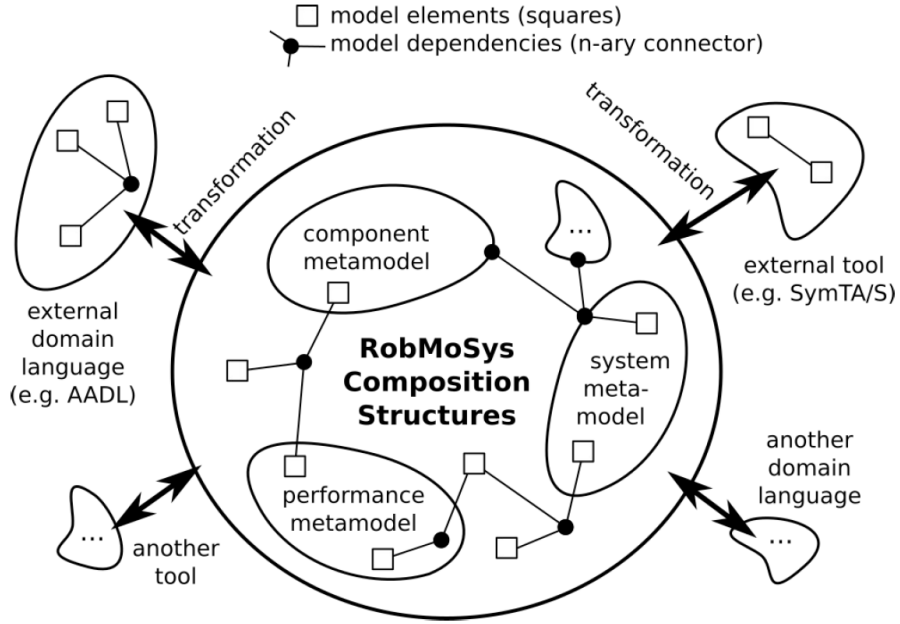
- Component-based architecture is about **activities**, **services**, etc;
- Functional architecture is about **data**, **functions**, and **control flows**;
- Concurrency is modelled as **constraints** in **information architectures**, and instantiated in function execution control flow
- Parallelism is instantiated in component-based architectures (deploying functionalities into components)

RobMoSys Composition Structures: Block-Port-Connector meta meta model (BPC)

- Entity-Relation **hypergraphs**
- Mereological information (`has-a`)
- Topological information (`contains, connects`)
- Common structure ↔ common tools!

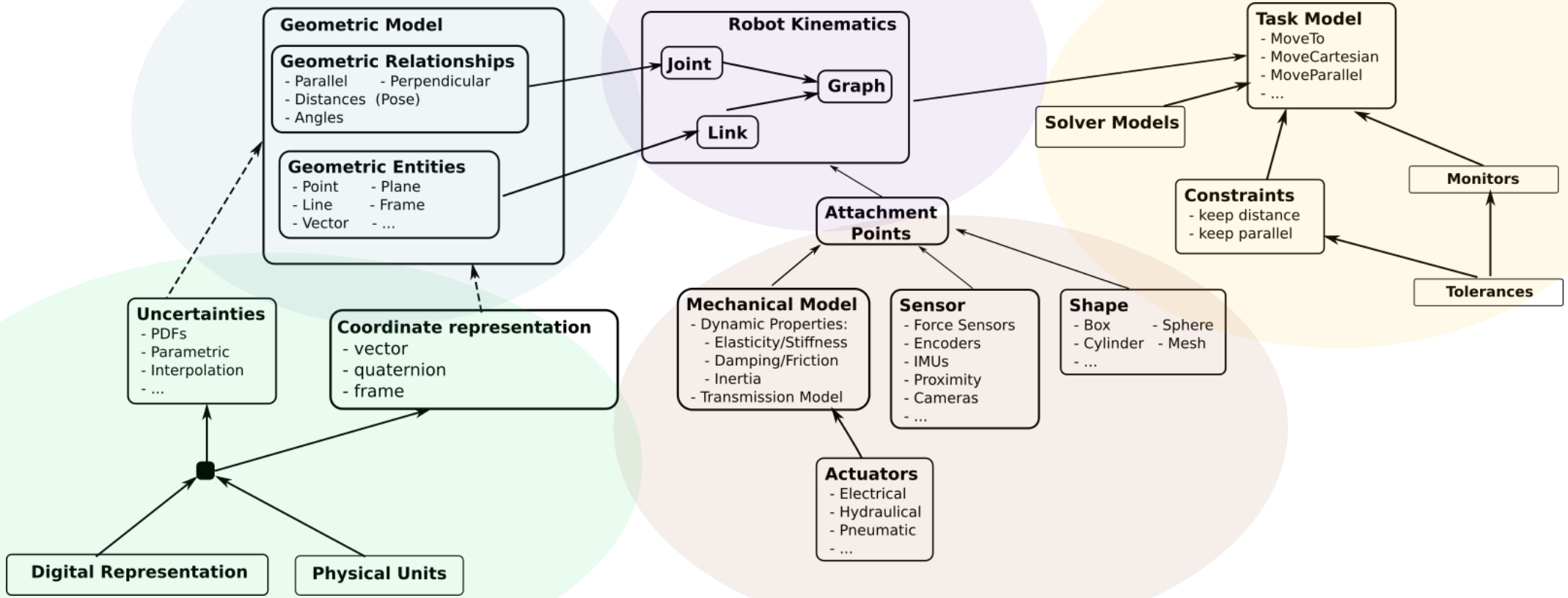
All models conform to the BPC structural meta meta model





Motion Stack Models: an overview

RobMoSys



Motion Stack

Knowledge-based system (database) conforms to BPC

RobMoSys

Natural hierarchy (physics):

- battery/mains
- power convertor
- motor
- transmission
- joint
- kinematic chain
- end effectors

Artificial hierarchy (tasks):

- sustainable lifetime, efficiency
- minimal heath and vibration
- mechanical & electrical safety
- Move, MoveTo, MoveConstrained
- MoveCoordinated, MoveOrchestrated, MoveChoreographed

Energy, force and voltage are composable,
time, motion and current are not.

Models of Kinematic Chains

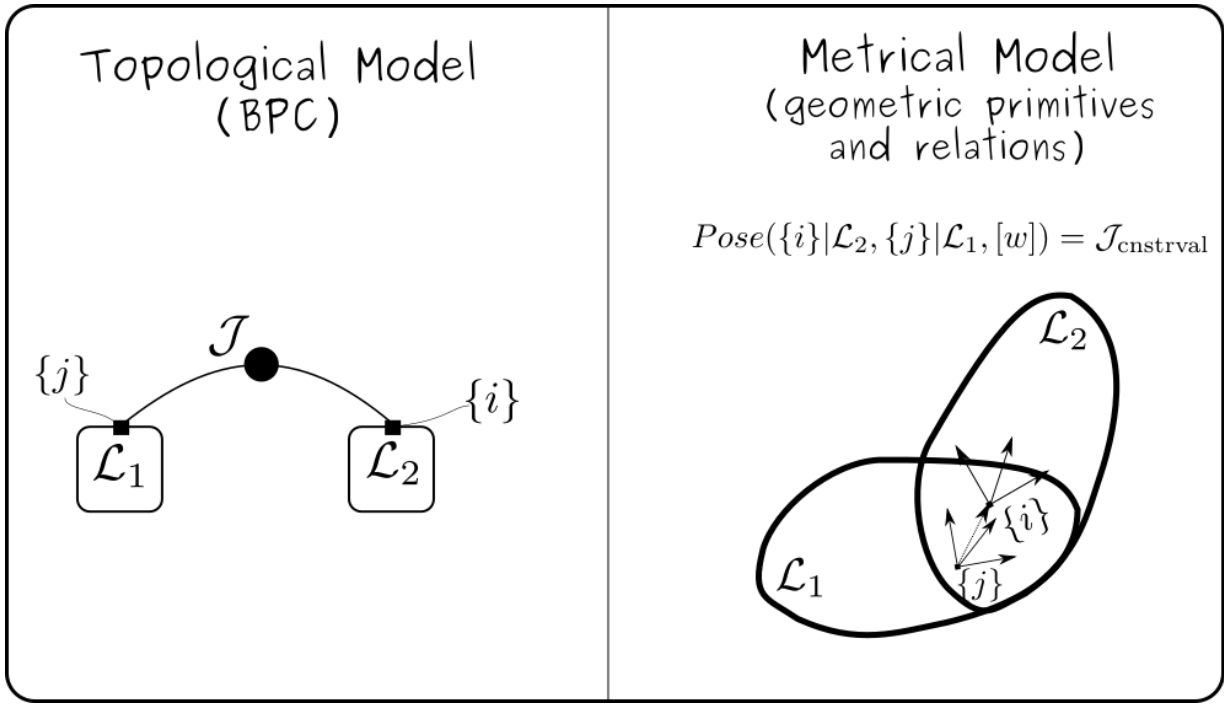
RobMoSys

Domain: geometric semantics of rigid bodies

Geometric Entities
Point, vector, line, plane, orientation frame, displacement frame, rigid bodies, ...

Geometric Relationships
Position, orientation, pose, linear and angular velocities, ...

- **Link:** a rigid body
- **Joint**: (mechanical) constraint **-** *relationship of relationships*
- **Kinematic chain:** a graph



joint constraining value

kinematic constraint

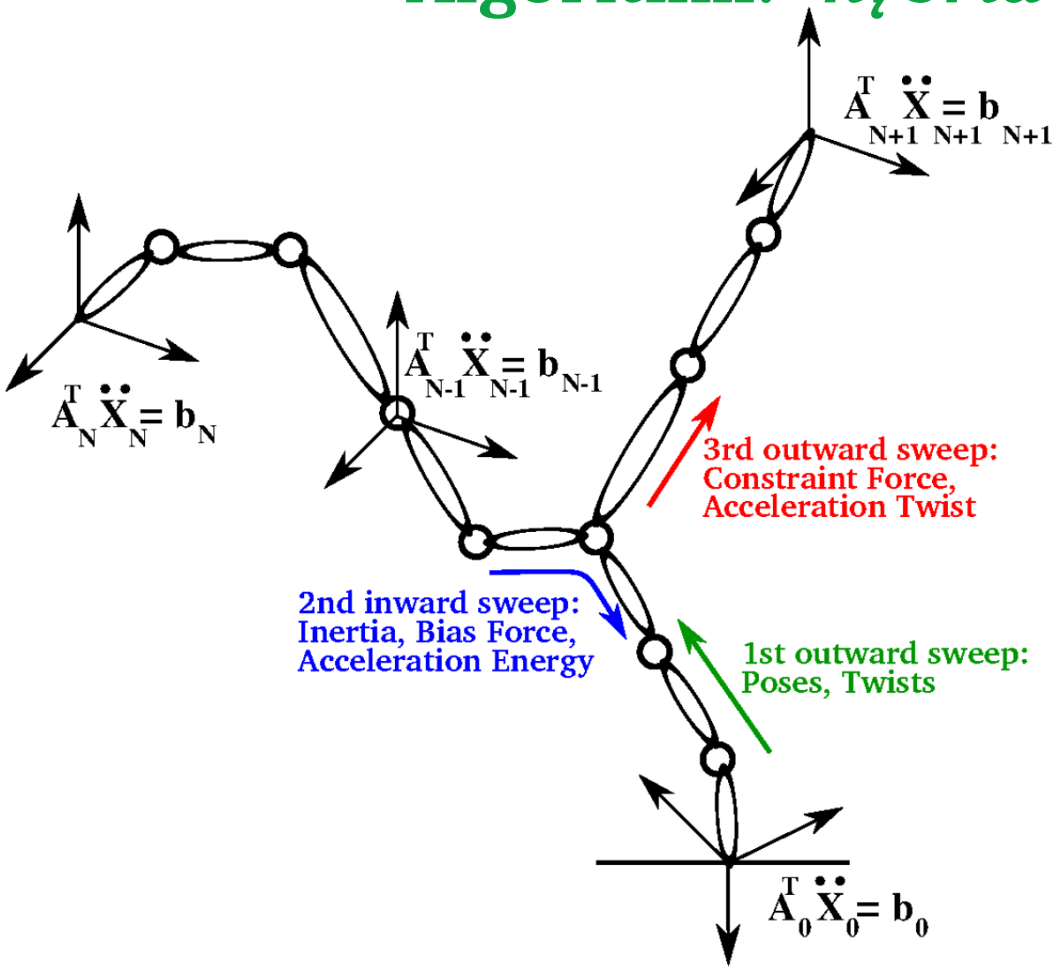
$$\text{Pose}(\{j\}|\mathcal{L}_2, \{i\}|\mathcal{L}_1) = \mathcal{J}_{\text{cstrval}}$$

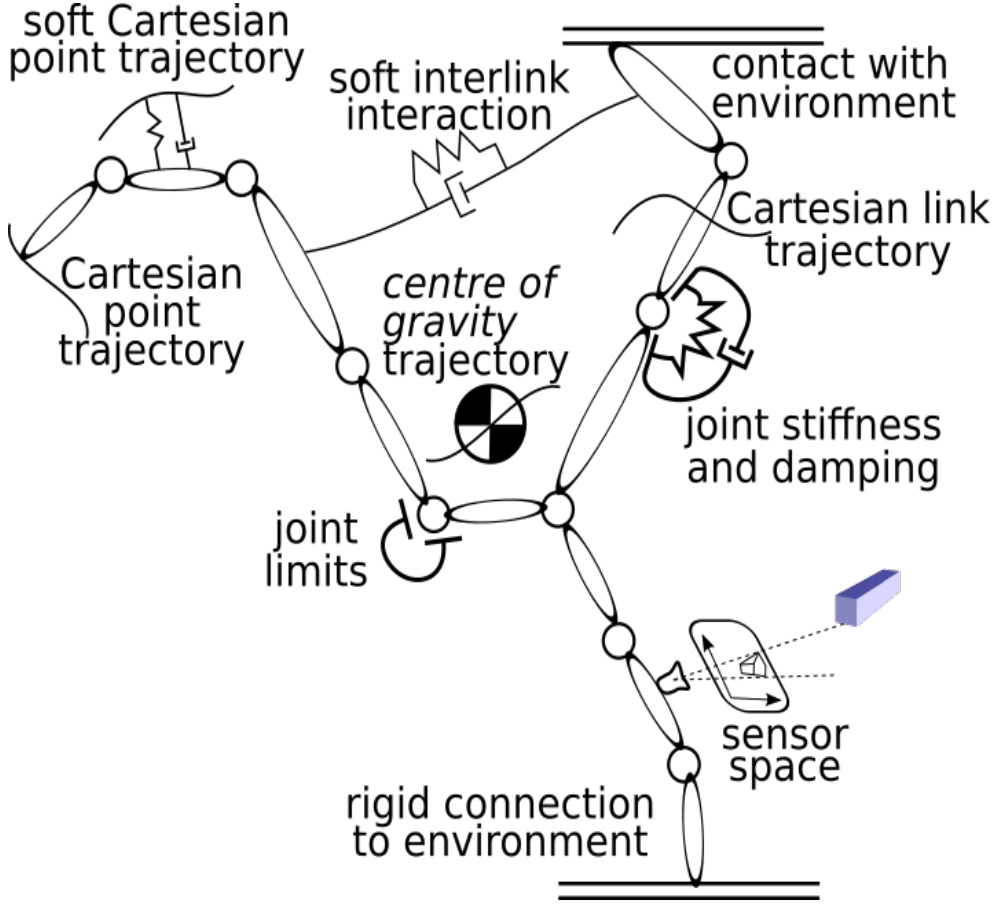
geometric relationship

Motion Stack (2)

RobMoSys

Algorithm: *"hybrid dynamics" solver*





Solver sweeps: structural model, to connect "fusion" to, with perception and world model in most efficient way.

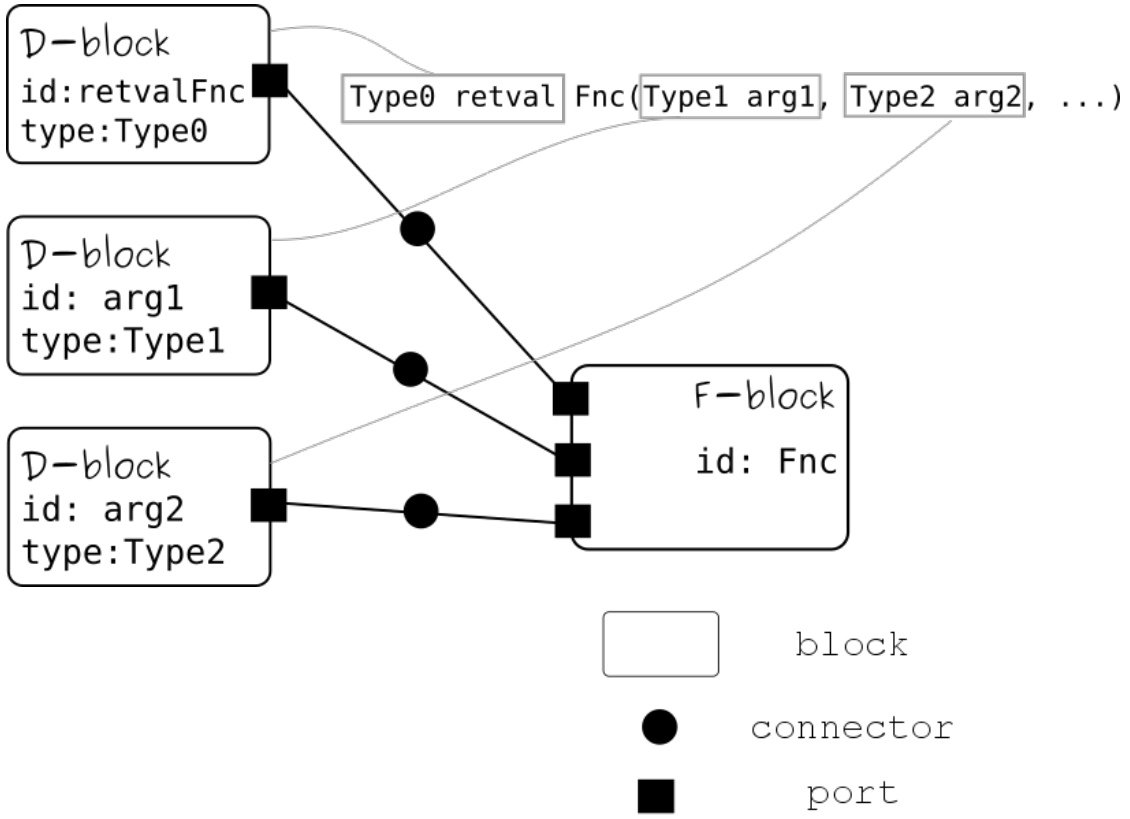
Models of Algorithms

RobMoSys

Entities and Relationships:

- D-block: (digital) **data** (representation)
- F-block: pure **function** (no side effects)
- S-block: scheduling (or **control flow**)
- C-block: **composition** of the above

Note: this is not robotics specific (hence, “meta meta”), but useful in motion, perception, and world model



Integration algorithms-components

RobMoSys

Algorithm:

- **Synchronous, no side effects**
- Processes motion, perception, world and task data, at **all** levels of abstraction
- Composes different levels of abstraction: mereology, topology, geometry, dynamics, …

Component:

- **Asynchronous, side effects**
- Responsive to real-time queries
- Composability: view and levels of abstraction in data streams
- Distributed
- Life Cycle, history, logs

Synchronous, concurrent, distributed programming is tough

Concurrency relevant for algorithms *and* components

Integration algorithms-components (2)

Composition pattern: *"event loop"*

RobMoSys

```
when triggered    // = scheduling by OS, i.e., asynchronous side effect
 do {             // = the Composition structure of the event loop
  communicate()   // get "message" with events & data that have been filled
                  // in by other asynchronous activities since last execution
  coordinate()    // handle the events, and decide which ones to react to
  configure()     // realise reconfiguration of local Computations
                  // whenever this is required by the events

  schedule()      // execute your algorithms' side effect-free Computations

  coordinate()    // this execution could trigger new events,
                  // both internally and externally
  communicate()   // the execution could generate data that other
                  // asynchronous activities must know about
  log()           // remember what happened
 }
```

`communicate()`: takes place via "Ports" of containing Component.
`coordinate()`, `configure()`, `schedule()`, `compute()`: take place via "Ports" of Algorithms