Knowledge Engineering in robotics

Herman Bruyninckx
K.U.Leuven, Belgium
BRICS, Rosetta, euRobotics

Västerås, Sweden
April 8, 2011

BRICS, Rosetta, euRobotics

BRICS:

Rosetta:

euRobotics:

BRICS, Rosetta, euRobotics

BRICS:

- software engineering for complex robotic systems
- **how** to do that in (Eclipse, **MDE**) tool support
  = lots of knowledge engineering

Rosetta:

euRobotics:

BRICS, Rosetta, euRobotics

BRICS:

- software engineering for complex robotic systems
- **how** to do that in (Eclipse, **MDE**) tool support = lots of knowledge engineering

Rosetta:

- intelligent skills for force-controlled robotic assembly
- skill to be described at several **levels of abstraction** = lots of knowledge engineering

euRobotics:

BRICS, Rosetta, euRobotics

BRICS:

- software engineering for complex robotic systems
- **how** to do that in (Eclipse, **MDE**) tool support = lots of knowledge engineering

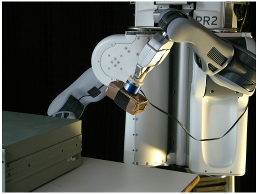
Rosetta:

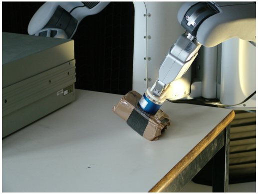
- intelligent skills for force-controlled robotic assembly
- skill to be described at several **levels of abstraction** = lots of knowledge engineering

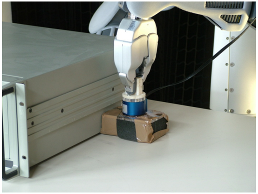
euRobotics:

- **semantic web** for robotics portal
- **need** for *open content* robotics **ontology**

Examples in my research







Knowledge representation

Knowledge needed is of various **types**:

- robot motion controllers
- geometry of objects + "scene graph"
- sensor capabilities & data interpretation
- (partial) ordering of actions in task
- common sense + physical laws
- relationships robot actions $\leftrightarrow$ effects
- ...

Knowledge representation

Knowledge needed is of various **types**:

- robot motion controllers
- geometry of objects + "scene graph"
- sensor capabilities & data interpretation
- (partial) ordering of actions in task
- common sense + physical laws
- relationships robot actions $\leftrightarrow$ effects
- ...

Representation of knowledge:

- (hyper)graphs (Topic Maps, RDF,...)
- rules (logic, OWL-x,...)

Knowledge representation

Knowledge needed is of various **types**:

- robot motion controllers
- geometry of objects + "scene graph"
- sensor capabilities & data interpretation
- (partial) ordering of actions in task
- common sense + physical laws
- relationships robot actions ↔ effects
- ...

Representation of knowledge:

- (hyper)graphs (Topic Maps, RDF,...)
- rules (logic, OWL-x,...)

How to **integrate** them...?

Types of ontologies

- **object** ontology
- **domain/system** ontology
- **profile** ontology

Types of ontologies

- **object** ontology
  - what knowledge do our robots need to become "intelligent"
- **domain/system** ontology
- **profile** ontology

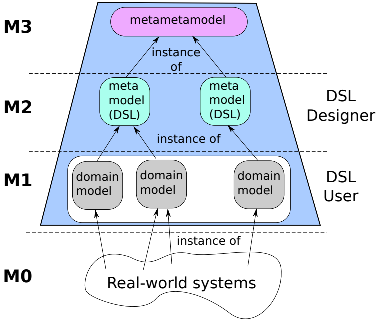
Types of ontologies

- **object** ontology
  - what knowledge do our robots need to become "intelligent"
- **domain/system** ontology
  - what is *"Field robotics"*? Or *"Assembly robotics"*?
  - Reference: Hallam & Bruyninckx, *An ontology of robotics science*, First European Robotics Symposium, 2006.
- **profile** ontology

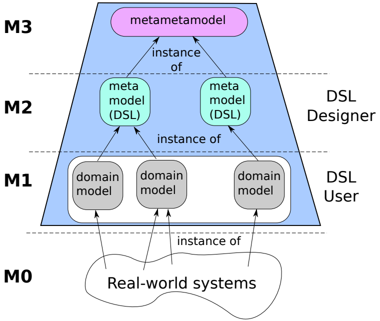
Types of ontologies

- **object** ontology
  - what knowledge do our robots need to become "intelligent"
- **domain/system** ontology
  - what is *"Field robotics"*? Or *"Assembly robotics"*?
  - Reference: Hallam & Bruyninckx, *An ontology of robotics science*, First European Robotics Symposium, 2006.
- **profile** ontology
  - what are the competences/expertise of a researcher?

MDE's M0–M3 & ontology

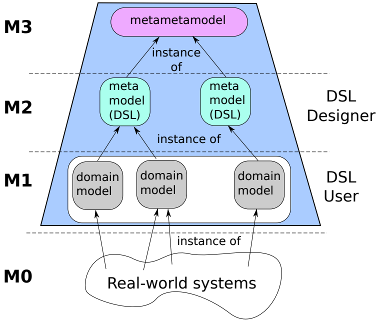


MDE's M0–M3 & ontology



M0–M3 **is** ontology (not other way around!)

MDE's M0–M3 & ontology



M0–M3 **is** ontology (not other way around!)

Claim: MDE's Domain Specific Language concept is pragmatic way to start robotics objects ontology, in particular for **action representation**

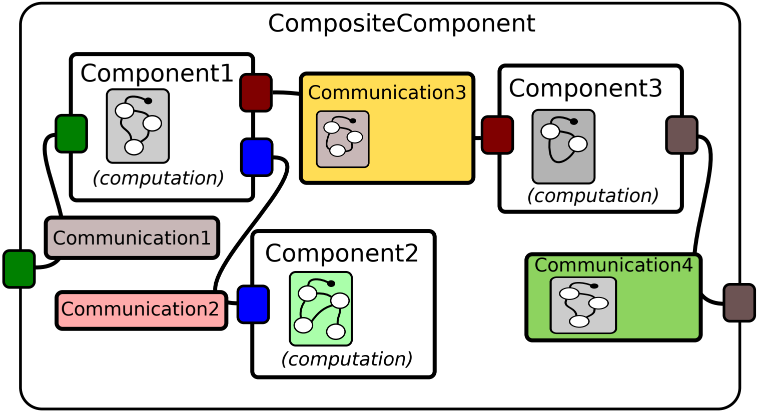
DSL for assembly case —Discrete behaviour: FSM—

```
move_up = apply(tff_motions.move_up, {zt=-0.3}) end
move_down = apply(tff_motions.move_down, {zt=0.1}) end
align = apply(tff_motions.push_down, {zt=10}) end
slide_x = apply(tff_motions.compliant_slide_x,
                {xt=0.2, zt=1}) end
trans:new{ src="initial", tgt="move_down" },
trans:new{ src="move_up", tgt="move_down",
  guard = return get_total_distance() > 0.2 end },
trans:new{ src="align", tgt="slide_x",
  guard = return get_move_duration() > 2 end },
```

DSL for assembly case —Continuous behaviour: control—

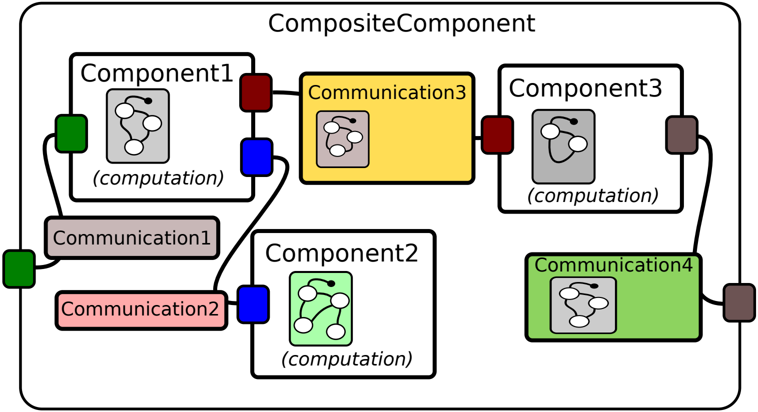
```
move_down =
 xt = tff.axis_spec:new { value=0 , type='velocity' }
 yt = tff.axis_spec:new { value=0 ,  type='velocity' }
 zt = tff.axis_spec:new { value=0.01 ,  type='velocity
})

compliant_slide =
 xt = tff.axis_spec:new { value=0,  type='force' }
 yt = tff.axis_spec:new { value=-0.03, type='velocity'

 zt = tff.axis_spec:new { value=1,  type='force' }
```

Robot systems: M2–M3 model



Structural model + *Communication* + *Coordination*

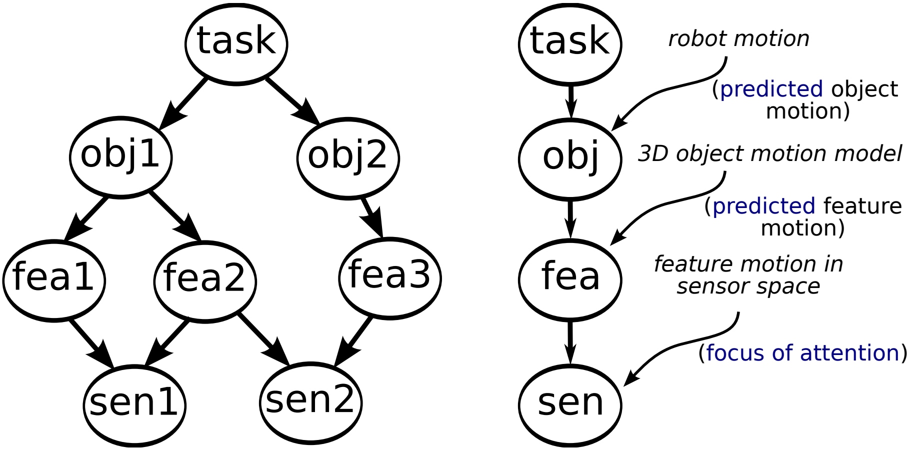
Robot systems: M2–M3 model



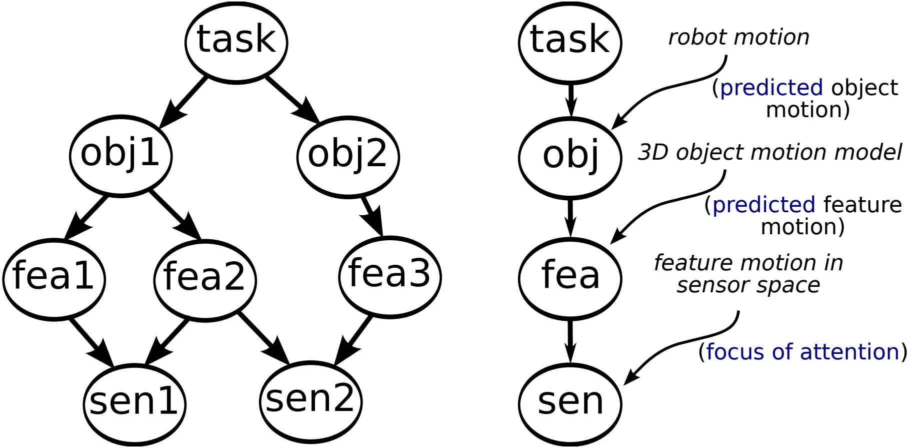
Structural model + *Communication* + *Coordination*

- Components: control, learning, planning,. . .
- M0–M1 framework DSLs: **Orocos** + **ROS**

3D perception stack: M1–M2 model

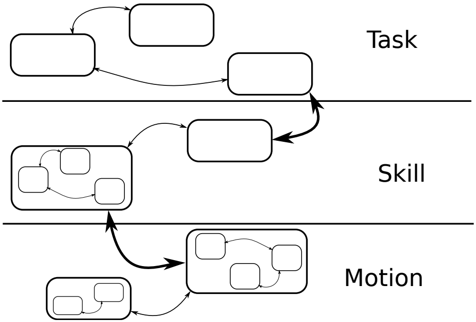


3D perception stack: M1–M2 model

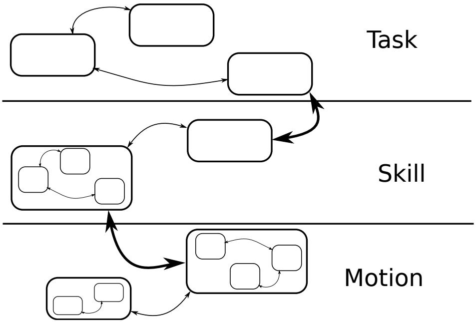


Bayesian probability excellent candidate for DSL!

Skills: M2–M3 model



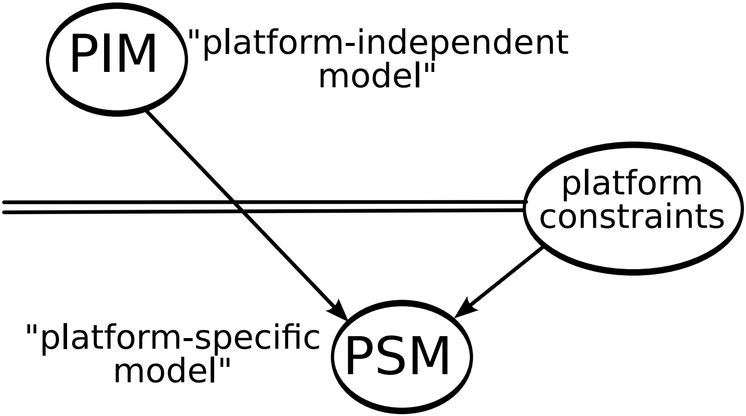
Skills: M2–M3 model



The Skill is a **probabilistic state machine**:

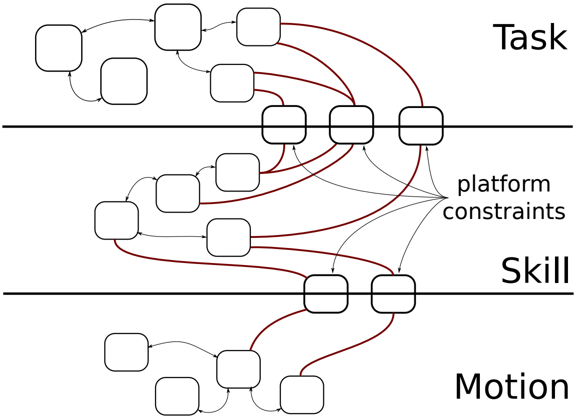
- state machine encodes causality/(partial) ordering
- events couple the symbolic and continuous domains.

Skills: M2–M3 model (2)



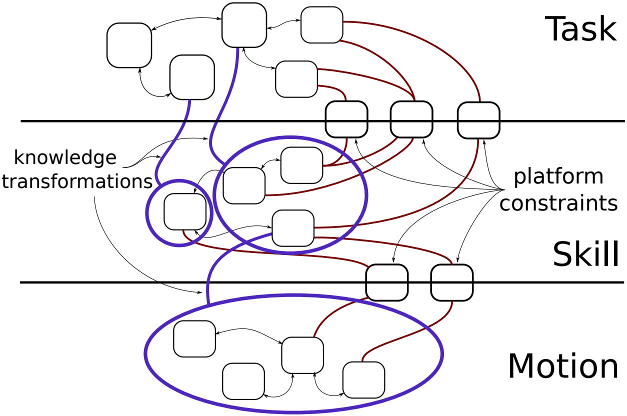
To add knowledge on: robot, controller, sensor, learning algorithm,. . .

Skills: M2–M3 model (3)



The platform constraints define parameters in the FSM behaviour.

Skills: M2–M3 model (4)



The Skill states **are** instantiations of logic symbols, and **run** continuous time/space control & sensing algorithms.

Conclusions

- major challenge: not so much the **amount** but the **variation** of different **types** of knowledge

Conclusions

- major challenge: not so much the **amount** but the **variation** of different **types** of knowledge
- our research: proposes **probabilistic finite state machine(s)** as key for **integration**:

Conclusions

- major challenge: not so much the **amount** but the **variation** of different **types** of knowledge
- our research: proposes **probabilistic finite state machine(s)** as key for **integration**:
  - focus about what knowledge and learning to use, at each moment in a robot's task
  - grounding & closing the world: "obvious"
  - lends itself very well for **DSL** representation
- Need for **open content** publicly available **ontology server**!

Conclusions

- major challenge: not so much the **amount** but the **variation** of different **types** of knowledge
- our research: proposes **probabilistic finite state machine(s)** as key for **integration**:
  - focus about what knowledge and learning to use, at each moment in a robot's task
  - grounding & closing the world: "obvious"
  - lends itself very well for **DSL** representation
- Need for **open content** publicly available **ontology server**!

⇒ multi-project cooperation can start **now**!

Conclusions

- major challenge: not so much the **amount** but the **variation** of different **types** of knowledge
- our research: proposes **probabilistic finite state machine(s)** as key for **integration**:
  - focus about what knowledge and learning to use, at each moment in a robot's task
  - grounding & closing the world: "obvious"
  - lends itself very well for **DSL** representation
- Need for **open content** publicly available **ontology server**!

⇒ multi-project cooperation can start **now**!

⇒ what license shall we use. . . ? (Creative Commons–Share alike!?)