Overview, motivation, benefits of model-driven approaches in robotics

Herman Bruyninckx, RobMoSys project
KU Leuven – TU Eindhoven

IEEE/RSJ IROS 2019 Tutorial
Macau SAR, China, 8 November, 2019

RobMoSys

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

Motivations for this presentation

What holds for good **software design**, also holds for good **modelling design**:

- it's hard.
- it's art.

→ one designer can be an order of magnitude better than the next one...

Major challenges that RobMoSys tackles head-on:

- avoid too early/late **binding of semantics** ("model lock-in", "model legacy", "one model/tool to serve them all",...)
- **composability**
- **compositionality**

→ all three are highly interconnected!

Objectives of this presentation

- to give a **sample** of relevant models and modelling practices:
  - which we found **very effective** in **education**, coaching and consulting,
  - where they have proven to have a *good enough* trade-off between **freedom of choice** and **freedom from choice**.
- (mostly) limited to **structural** models

  `<showing-off mode>`

  (**mereo-topological** models, that is!)

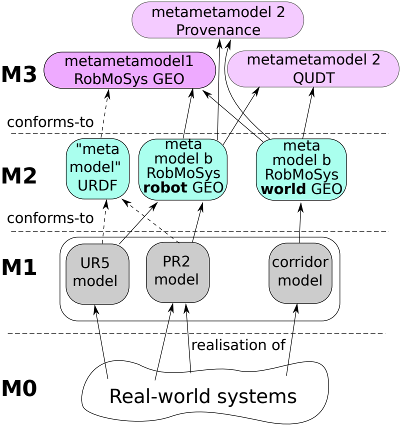
  `</showing-off mode>`
- assumption: we all know what **behaviour** should be composed **onto** those structures.

RobMoSys' five levels of modelling

1. **Abstraction**: informal models used by humans as guidance for other humans.
   → **harmonization** of terminology and interpretation of the abstractions.
2. **Reuse & Flexibility**: reuse and customization of robotics software assets
   → **formalized models** ("**data sheets**")
3. **Predictability**: composition is *correct by construction*
   → **formalized meta models**.
4. **Automation**: automate labor-intensive stuff:
   (Validation & Verification, code generation,. . .)
   → **off-line "reasoning" tools**.
5. **Autonomy**: models used by robots at run-time.
   (self-X, with X = configuration, adaptation, explanation,. . .)
   → **on-line "reasoning" tools**.

Core "meta meta meta" model :-)

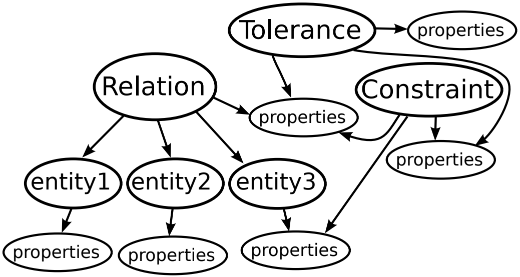
OMG's M0–M3 meta model



- OMG's M0–M3 hierarchy of modelling.
- M1–M3 relations are **relative**; "hierarchy" can be extended "upwards" indefinitely.
- objective: to support **model-to-model** and **model-to-text** transformations between (meta) models, for their **conforming** parts.
- typically, those transformations are done by **humans** using a **tool chain**.

Experience: it is **very worthwhile** to educate robot developers to grasp these **"multiple inheritance"** concepts!

Core "higher-order modelling" primitive



→ key for **composability**!

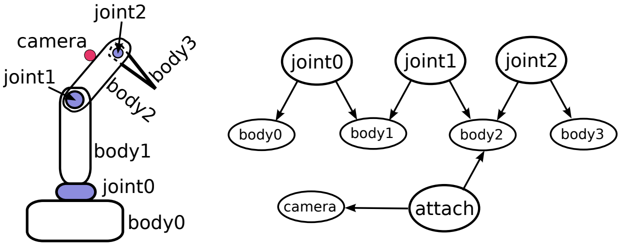
E.g., add **provenance** model, for all "magic numbers" in models (and hence, in software)

Experience: it is **very worthwhile** to educate robot developers to grasp the fundamental differences between

- **property** of an entity.
- **attribute** of an entity = property of a relation with that entity as an argument.

Higher order models are **graphs**.
They carry **meaningful** structures.
→ **reasoning** = **graph traversal**.

Motion stack (level 0)
Model of a robot's kinematic chain



The kinematic **model** represents:

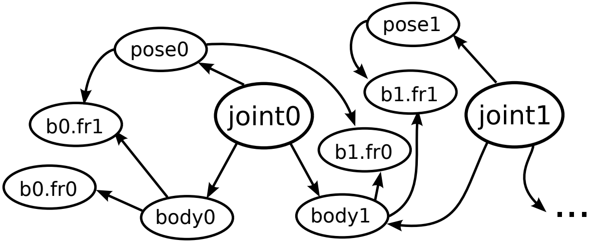
- **parts** in the model,
- **connections** between those parts,
- **attachment points** for further composition

Notes:

- URDF is a **poorly composable** meta model.
- KDL is a **poorly composable** software library.

Motion stack (level 1)

Compose kinematic chain with relative pose

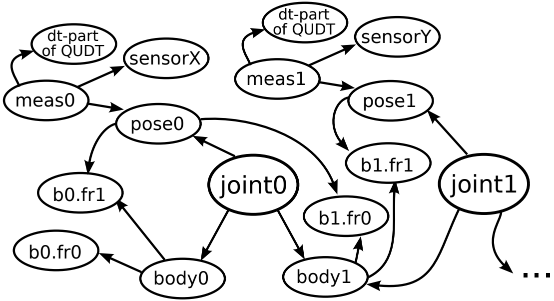


The "higher-order" **model** represents:

- joint is a **motion constraint** between robot's links
- at every moment in time, two links have a relative **pose** whose properties depend on the type of the joint constraint

→ mathematical constraints between positions on connected body points.

Motion stack (level 2)
Compose pose model with measurement model

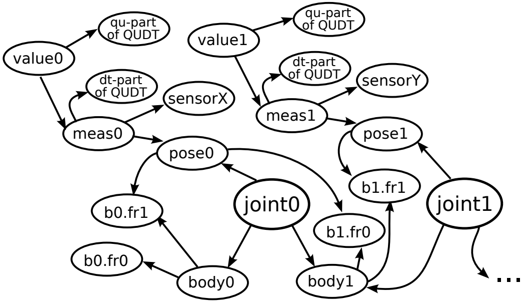


The extra higher-order **model** represents:

- the pose is **measured** by sensors
- it has a **d**imension and **t**ype
- **QUDT** is a standard meta model for this purpose

Motion stack (level 3)

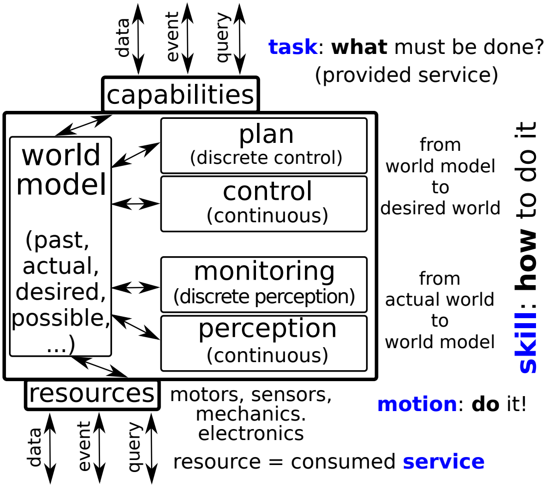
Compose measurement model with coordinates model



The higher-order **model** represents:

- measurement of pose gives **numerical values**.
- those **q**uantities have physical **u**nits
- **QUDT** is a standard meta model for this purpose

Task-skill-service stack (level 0)

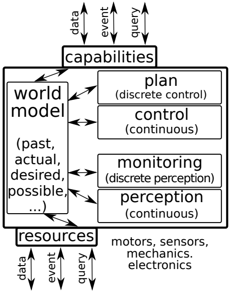


Experience: it is **very worthwhile** to educate robot developers to grasp the fundamental role of the **various** types of **state of the "world"**:

- **behavioural** state of an activity
- **continuous** state of "motion" + "perception"
- **discrete** state of "task plan"
- **logical** state of "constraints"

Task-skill-service stack (level 1)

Compose with "guarded optimization" behaviour model



$+$

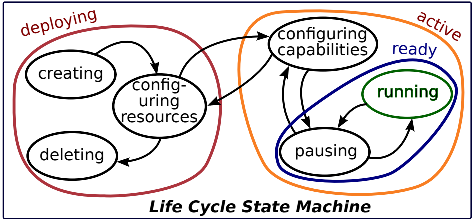
task state & domain	$X \in \mathcal{D}$
desired state	X_d
robot state & domain	$q \in \mathcal{Q}$
objective function	$\min_q f(X)$
equality constraints	$g(X) = 0$
inequality constraints	$h(X) \leq 0$
tolerances	$d(X, X_d) \leq A$
solver	algorithm computes q
monitors	decide on switching

Experience: it is **very worthwhile** to educate robot developers to grasp the fundamental role of

- **modelling** the **origins** + **causes** of all behaviour

→ **composable** + **explainable**.

Coordination via Finite State Machines (level 0)

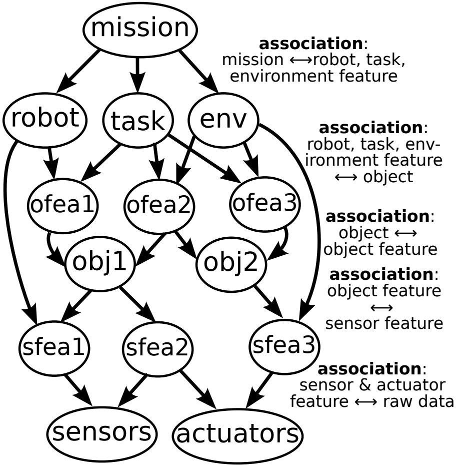


Example: **Life Cycle** State Machine (LCSM) with *hierarchical* states.

Experience: **need** to decouple:

- **structure** = `states` + `transitions`
- **behaviour**: `event` → `transition`
- **behaviour**: `transition` → `events`
- **distribution** of `event` handling:
  - **Coordination**: `event` = `flag` inside **one** activity
  - **Orchestration**: one activity coordinates many, via **communicated** events
  - **Choreography**: activities coordinate themselves, generating events from **observation**

Perception stack (level 0)



Experience: it is **very worthwhile** to educate robot developers to grasp the large amount of **association** challenges:

- each association = **relation** + **constraints** + **tolerances**
- **complementary** and **hierarchically inter-related**

→ "higher" and "lower" level are **sources of magic numbers** in **task-skill-services** model of a given level

Conclusions

- robotics has **to integrate**, and within **a lot of context**, hence ("higher-order") modelling is a must:
  - "magic numbers" in code come from **somewhere**, **every time**.
  - systems must often be **(re-)composed**, statically and **dynamically**.
- doing the modelling effort, even 100% informally, leads to better software designs, *because* of better informed software *designers*.
- modelling remains an **art**.
  - especially for designing the "right" **higher-order** models and abstractions,
  - connected to **system-wide** integration/**composability** dependencies.
  - **freedom of choice** ↔ **freedom from choice**.

→ those are where **the money** is!

→ those are where **the education** needs to be!

Further reading

- **consolidated**: https://robmosys.eu/wiki/
- **not yet consolidated**: *living* project Deliverable: https://robmosys.pages.mech.kuleuven.be/

Thank you for your attention