Ontologies & Meta meta models at KU Leuven

Herman Bruyninckx
University of Leuven
Eindhoven University of Technology

European Robotics Forum, Ljubljana
March 23, 2016

Meta models for geometry in robotics

- **Full**(?) ontology of frame + kinematic chain + dynamics
- *partial* composition with *software representations* and *physical units*.

- De Laet, T., Bellens, S., Smits, R., Aertbeliën, E., Bruyninckx, H., and De Schutter, J.
  Geometric Relations between Rigid Bodies (Part 1): Semantics for Standardization, Robotics and Automation Magazine, 2013.
- Shakhimardanov, A.
  Composable Robot Motion Stack. Implementing constrained hybrid dynamics using semantic models of kinematic chains, PhD 2015.

Meta models for skills

- ontology of **skill** as a composition of
  - task
  - object affordances
  - robot capabilities (motion, perception)
  - environment context
  - *Constrained Optimization Problem* formulation for "control"
  - solver to generate actual setpoints
- *simple* by concept, *complicated*, by nature of composition requirements

- Vanthienen, D., Klotzbücher, M.m and Bruyninckx, H. *The 5C-based architectural Composition Pattern: lessons learned from re-developing the iTaSC framework for constraint-based robot programming*, JOSER, 2014.

Meta meta model for structural composition

- *NPC4*: Node, Port, Connector; Containment, Connection, Composition
- **Full**(?) ontology of hierarchical hypergraph structures.
- Meta models we are building with it: FSMs, computational models (data, functions, schedulers).

- Scioni, E., Hübel, N., Blumenthal, S., Shakhimardanov, A., Klotzbücher, M., Garcia, H., and Bruyninckx, H.
Hierarchical Hypergraphs for Knowledge-centric Robot Systems: a Composable Structural Meta Model and its Domain Specific Language NPC4, JOSER, under review.

Meta meta model for ontologies/DSLs

Every ontology/DSL has the same structure:

- *Primitives*: the objects and concepts for which a formal knowledge model is being made.
- *Relationships*: the relationships that exist between the Primitives, in the domain that is being modelled.
- *Constraints*: the constraints that exist on the properties of the Primitives and Relationships.

- *Tolerances*: the deviations that an application in the modelled domain can allow for the Constraints it relies on.

Meta meta model for ontologies/DSLs

Every ontology/DSL has the same structure:

- *Primitives*: the objects and concepts for which a formal knowledge model is being made.
- *Relationships*: the relationships that exist between the Primitives, in the domain that is being modelled.
- *Constraints*: the constraints that exist on the properties of the Primitives and Relationships.
  By far the most difficult part of the modelling job!
- *Tolerances*: the deviations that an application in the modelled domain can allow for the Constraints it relies on.
  Only come in when **building applications**!

Skill Dependency Graph

- models execution order constraints
- hierarchical composition: the **relevant**
  - "future" constraints, and
  - "past" constraints

  come into the **context** of current one.
- Just-in-time optimization, at runtime

$\Rightarrow$ serves as our new meta meta model for all kinds of dependency ontologies.

Enea Scioni.
Online Coordination and Composition of Robotic Skills. Formal Models for Context-aware Task Scheduling, PhD KU Leuven/Università di Ferrara, 2016.

Under construction. . .

AB5C ("Algorithmic Building Blocks for 5C compositions"):

- *sensor fusion* + *sensori-motor control* (and many other very composite algorithms, i.e., with high variability in "API") requires separation and composition of *data*, *functions* and **schedulers**.
- every part has a:
  - *model*,
  - *meta model*
  - and *unique ID*

  available, introspectable at runtime from deployed binary code.
- single-threaded *event loop* implementation in C, from deeply embedded till widely distributed.

Lessons learned

- **Ontologies pay off, from day one**: formulating problems and solutions + design of software.
- **Simple over easy**;
  complicated over simplistic, but only **by composition**.
- Meta-meta-∗-modelling continues till one reaches **formal mathematics**.
- *Host languages*:
  - Semantic Web languages: poor representation capabilities for continuous space-time dynamics.
  - adding JSON-LD, JSON Schema, and "GraphQL", because of *graph relations* and *composition via context* being built-in.
- As soon as one follows a *ontology/model-driven* approach, **all** "attractive" features of the everything-and-the-kitchen-sink OO languages (C++, Java,. . .) disappears, via **semantic-level tooling**.