Best Practices in System Design for Robot Control

Herman Bruyninckx
Eindhoven University of Technology
KU Leuven
http://people.mech.kuleuven.be/~bruyninc/

May 4, 2016

Overview of this lecture

- *objectives*:
  - to provide some **structure** in your project designs
  - using **generally applicable** design patterns
- *design* = first **modelling**, then implementation.
- systems have **Behaviour**, **Activity**, and **Structure**
  - model for *Behaviour* = *Mission-Task-Motion Skills*
  - model for *Structure* = **Composition Pattern**
  - model for *Activity* = *Port-based containers*
    (⇒ next lecture, on "Software")
- **control & sensing**:
  - discrete *and* continuous
  - must **couple** different *levels of abstraction*
  - coupling *modelled* "best" as *constrained optimization problems*

Structure, Behaviour & Activity

Behaviour: *"purposeful reaction to stimuli"*

- as seen from the "outside", with **external task goals**
- software systems: interact via communication
- hardware systems: interact mechanically, electrically,...

Activity: *"executing the code"*

- how behaviour is realised "internally"
- software: CPU + RAM + bus
- hardware: mechanical, electrical,..., impedance

Structure:

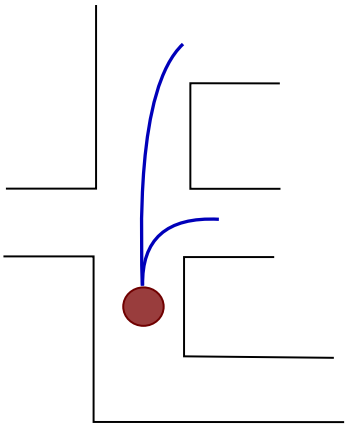
- *interconnection* of Behaviours and Activities

= *system architecture*

Quality of a system: not just externally visible behaviour, but also internal structure and diagnosis!

Maze navigation: Mission – Task – Motion

Intelligent control: system can explain, at all times, **what** it does, **why**, and **how well** it is doing it.



Mission control:

- what: which corridors to follow?
- how well: where have I already been?

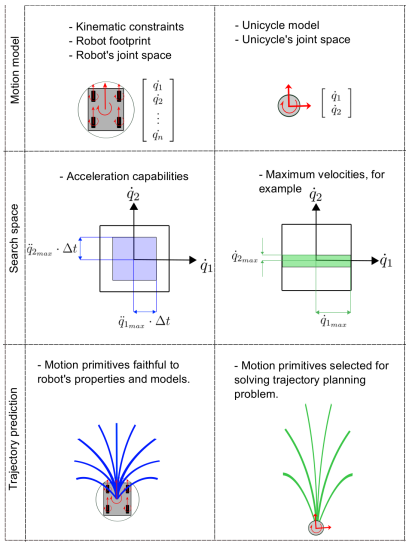
Task control:

- what: **select** a *trajectory* (don't *plan* one!)
- why: reaching selected corridor.
- how well: . . .

Motion control:

- what: select a *wheel velocity profile*.
- why: following selected trajectory
- how well: . . .

Robot motion models



You choose:

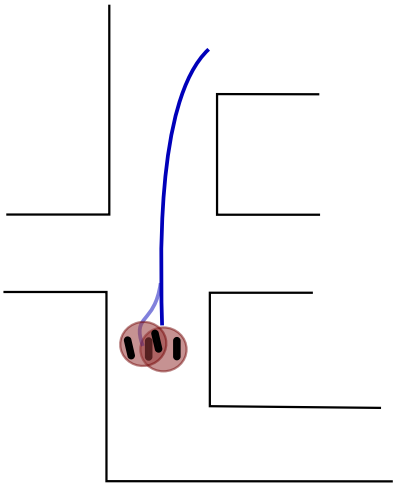
- number of motion *models*
- *parameters* in it
- *level of abstraction*: geometric, kinematic, dynamics,. . .

Must be connected to:

- *capabilities* of robot
- *requirements* of tasks

→ **models**

Motion control



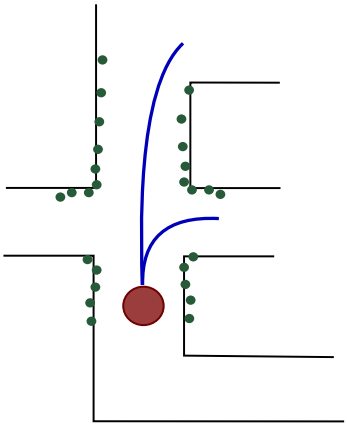
Control is needed because

- robot is not on "the right place"
- but it knows "where it should be"

How to make robot wheels move to be on task-level trajectory *as good as possible*:

- "move wheels": velocity profile? minimum-jerk?
- what is "good"?
- what is time/space/speed *horizon* of optimization?
- . . .

Link sensing to Motion and Task



Advice: *first*

- model *maze primitives*
- together with *trajectory primitives*

and only *then*

- model what information to extract from raw sensor data

→ "best fitting" maze primitives?

- how many of them?

→ define horizon on sensing *compatible* with
(i) horizon on *control*, and
(ii) horizon on *trajectory*

Workhorse "algorithm" for behaviour: Constrained Optimization Problem solvers

Constrained Optimization Problem *FSM, configuration, monitoring, events*

objective functions	$\sum_i w_i f_i(X,y)$
optimization variables	$\min_X$
constraints	$g(X,y) \leqslant 0$
tolerances	$\text{dist}(X,X_{min}) \leqslant \delta$

Advantages of a *constrained-based* design:

- composable!
- **monitorable**!
- explicit horizon!
- tolerant!
- configurable!
- **discrete + continuous**!
- **quality is explicit**!

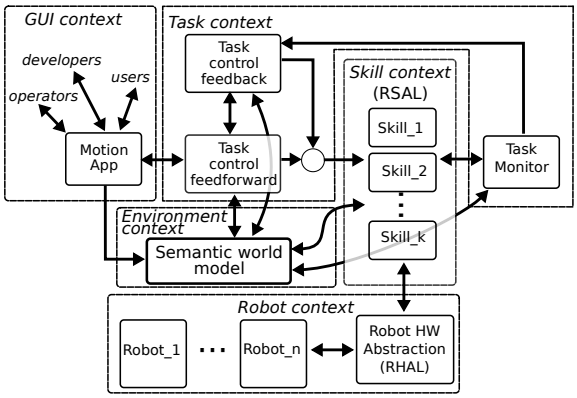
Link "control" between layers

- layers provide "constraints", "objective functions" and "tolerances" to each other;
- each layer solves (discrete & linked) set of COPs:
  - some of its "monitors" produce "events" that a layer below or above must react to
  - hierarchical design has advantage of "abstraction" but not danger of "information hiding"
- each layer has **state** in its discrete control
  = which COP is it solving at this moment?

→ difficult to keep consistent, over several layers!

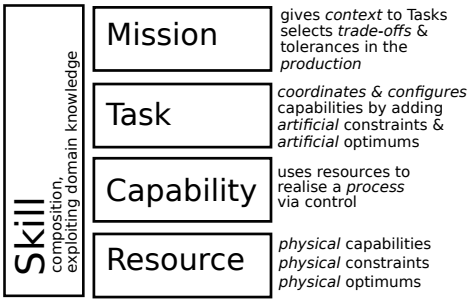
→ **system design pattern** needed!

Generic behaviour system model: Mission-Task-Motion Skills



Note: Skills share central place with *semantic world model*. . . !

Skills to integrate Mission, Tasks and Motions



Skill: adds the **knowledge how to realise** a Mission.

Capability of robots = Sensing + Motion

Semantic world model

- represents actual, desired, and previous "worlds"
- contains the geometric/topological parameters that *couple* Mission, Task, Motion, and Skills
- very poor "state of the art". . . !

→ should be in the focus of your design efforts, from Day 1!

Conclusions

- control applications have **a lot of structure**

$\Rightarrow$ exploit it, for *efficiency*, *readability* and *composability*

- control applications have **a lot of parameters**

$\Rightarrow$ relate them via *models*

- control applications have **several control layers**

$\Rightarrow$ relate them via *constrained optimization*

- **monitor everything you expect to happen**. . .
  . . . **and be ready to react if it doesn't!**
- **Skills = monitoring & coordination & configuration**