Benchmarks and architectures for autonomy and robustness

Herman Bruyninckx

Department of Mechanical Engineering
K.U.Leuven, Belgium

GEMBENCH Forum 2008
March 25–26, 2008

Goals of this presentation

"Political" goal:

- *Call 3* emphasises robustness and autonomy, so we should have an idea what our proposals should be talking about...

Scientific goals:

- To identify different semantic contexts (algorithm, agent, system, ...)
- "To standardize" meaning of terminology:
  - Robustness (in different contexts)
  - Autonomy
- To discuss possible benchmarks

Components vs System

Components (from "simple" to complex):

1. Hardware & Device drivers
2. "Input-Output" algorithms (planning, control, perception)
3. Middleware (= glue between "all the rest")
4. "Side effect" algorithms (**agents**, threads,...).

Systems:

- Specific **architecture** between components.
- Choice of **policies** on inter-component communication

Robotic systems —Three complexity levels—

CC Centralized hardware/Centralized software
E.g.: industrial robot.

DC Decentralized hardware/Centralized software
E.g.: autonomous car.

DD Decentralized hardware/Decentralized software
E.g.: "system-of-systems" RoboCup team.

Each has different robustness issues!

Robustness

"Ability to cope with disturbances."

- Property of *components* as well as *systems*.
- Of **major importance** in robotics, due to **high dynamics** of changes in task & environment.

↔ Aerospace/automotive/telecom:
dependability!
(uptime, predictable performance, predictable failure modes, safety,...)

I/O robustness

- Applies to algorithms.
- Degree to which **non-nominal** inputs result in **acceptable** outputs.

Specify what "non-nominal", "acceptable" mean!

Does any HW/SW provider already do this?

Device driver robustness

- ??

Middleware robustness

- Degree to which the middleware's **Quality of Service** degrades with **scaling** (number of interconnected components; amount of interaction;...).
  Specify: "QoS", "scaling"!

Agent robustness

- Degree to which the agent can cope with disturbances in the **availability of resources** (RAM, CPU, sensors,...)

  Specify: "availability", "resource"!

Robotic components need standardized. . .

- **requirement specifications**: hidden assumptions, nominal working conditions, foreseen exceptions,. . .
- **failure mode descriptions**: *warning*, *functional error*, *fatal error*, . . .
  What QoS, functionalities are delivered in which failure mode?

What Good Experimental Methodologies exist already?

System robustness

For example: RoboCup team

- **Internal** robustness: $\sim$ component robustness.
- **External** robustness: wrt "disturbances" in **behaviour** of **other systems** ("agents", task, environment).
- "Disturbances" come from the fact that one system can most often only **observe** other systems' behaviour; sometimes **intentions** can be communicated.

System robustness —From "simple" to difficult—

- **Perception robustness**: degree to which perception of other systems' behaviour is correct.
- **Interpretation robustness**: degree to which interpretation is correct.
- **Action robustness**: degree to which actions are correct.

Autonomy

"Degree of decision making during lifetime"

- Is only a **system** property.
- Link with robustness:
  - decision making must remain "correct" under "disturbances".
  - decision making relies on *perception* and *interpretation* (which are not automatically robust).

How do we measure autonomy??

Architectures for robustness and autonomy

1. Middleware $= \sim$ connecting components into a system $\longrightarrow$

- robustness wrt **data flow**, **event flow** and **decision flow** in the system.
- **communication policies** have large influence on perception and interpretation.
  (What information is propagated when, to whom, with what QoS,...?)

Architectures for robustness and autonomy (2)

2. Agents $= \sim$ *understanding* other systems $\longrightarrow$

The following successful(?) approach can be seen already in existing systems...

- keep internal reflection of the world
- allow *trust measure* on other systems' communications
- propagate perceived intentions of other systems within this model
- make decision on that simulation

Is this a GEM? Or do we need more?