- Why would this course be useful?
- What comes next?

- We built several robots including ant robot that I designed the mechanical part and implemented the control in Labview and also a surgical robot where I was responsible for designing driver for its DC motors and controlling them. I also designed an exoskeleton robot for upper limb rehabilitation for my final project.
- I fly an SRD 280 v3 racing drone, and I like to prototype cool things with my Arduino.
- When I was 12 years old I was the senior pilot of a toy Spider Man helicopter. I'm pretty sure this counts as robotics experience.
- Really all I have is experience with Legos and imagining how I would want them to work if they were robots.

ME, EE, BioChem

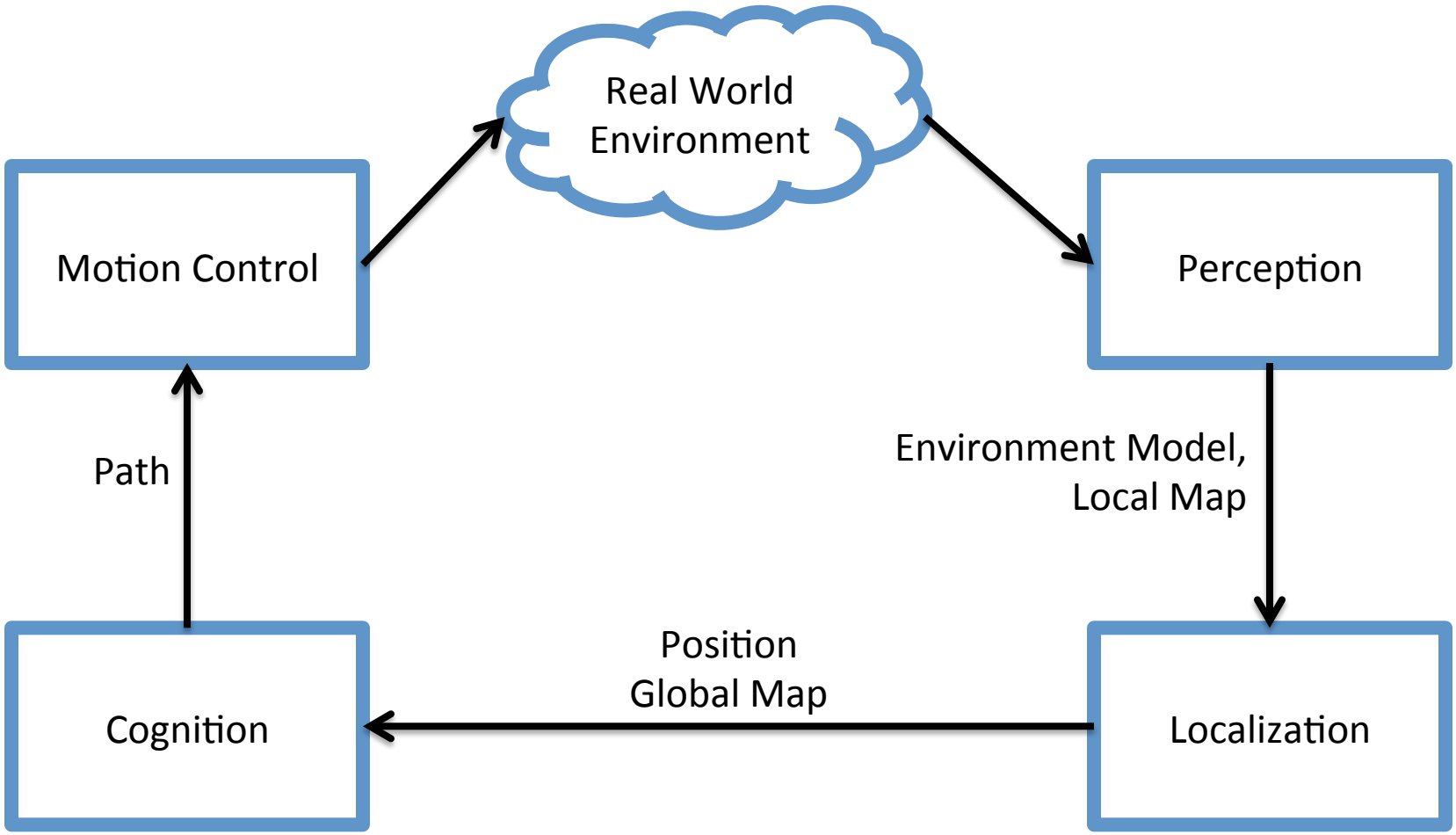
- Python 2 vs. Python 3

- http://play.elevatorsaga.com/

Lab 1?

Computers?

- Humans Need Not Apply
  - https://www.youtube.com/watch?v=7Pq-S557XQU


Real World
Environment
Motion Control
Perception
Path
Environment Model,
Local Map
Position
Global Map
Cognition
Localization

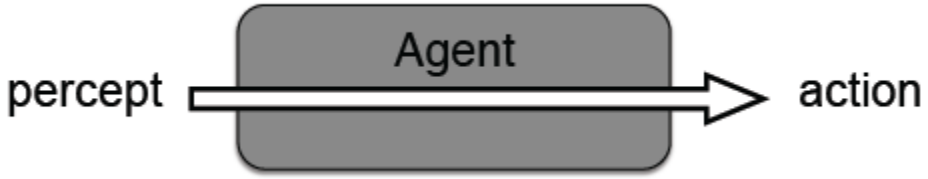
- How would you design the “architecture” for a robot?
  - Inputs/outputs?
  - Considerations?
  - Software Engineering ideas?

Robot Software Architectures

- Principled design for software modules that control a mobile robot system.

- Advantages (Goals):
  - Modularity
    - code reuse and sharing
  - Control localization within the architecture
    - Individual component testing
    - Optimization through learning

1) Reactive Architecture



- *Actions* are directly triggered by *Sensors*
  - no representations of the environment
  - predefined, fixed response to a situation
  - fast response to changes in the environment

Limitations of a Reactive Robot

- Knowledge of the world is limited by the range of its sensors
- Unable to count (how could you get around this?)
- Unable to recover from actions which fail silently
- Can not “undo” an incorrect action
- Not possible to “plan ahead”
- Can not coordinate with other robots in a reasonable way
- many others ...

2) Deliberative Architectures

- Organized by decomposing the required system functionality into concurrent modules or components.
  - Map building
  - Path planning
  - Navigation
  - …
- Problems:
  - overall complexity of the system may grow
  - hard to offer real-time guarantees on performance:
    - solving any given problem takes longer than an equivalent reactive implementation
    - solving different problems takes different amounts of time

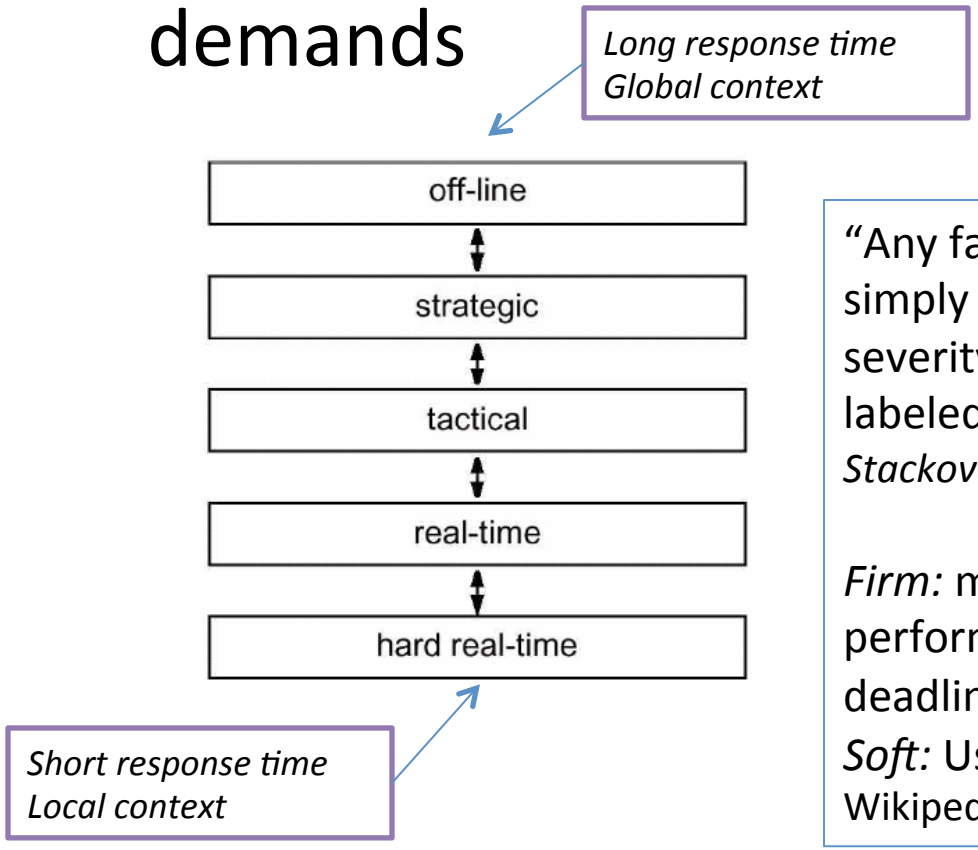
Aside: Architecture Decomposition

- Decomposition allows us to modularize our control system based on different axes:

- Temporal Decomposition
  - Facilitates varying degrees of real-time processes
- Control Decomposition
  - Defines how modules should interact: serial or parallel?

Temporal Decomposition

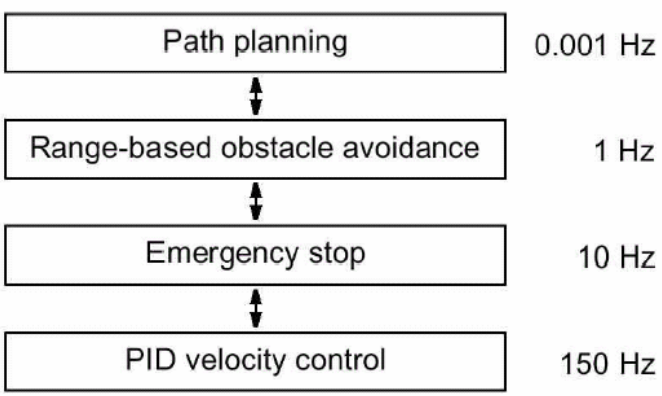
- Distinguishes between processes that have varying real-time and non-real-time demands



"Any failure to meet a *hard real-time* constraint simply means that the system is broken. The severity of the outcome when something is labeled 'broken' isn't material to the definition."
Stackoverflow.com

Firm: missing is tolerable, but may degrade performance. Usefulness of result is zero after deadline.
Soft: Usefulness of result degrades after deadline
Wikipedia.org

Temporal Decomposition

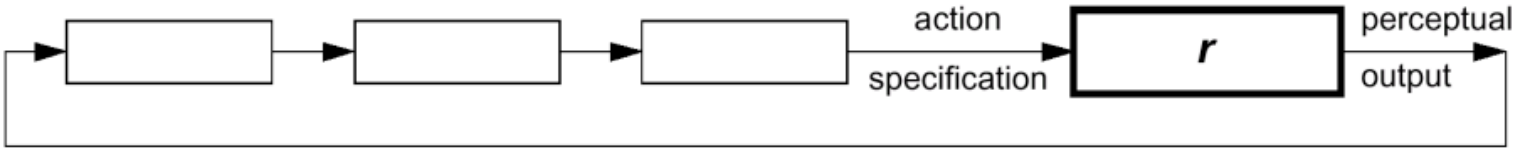


Example: Watchdog process

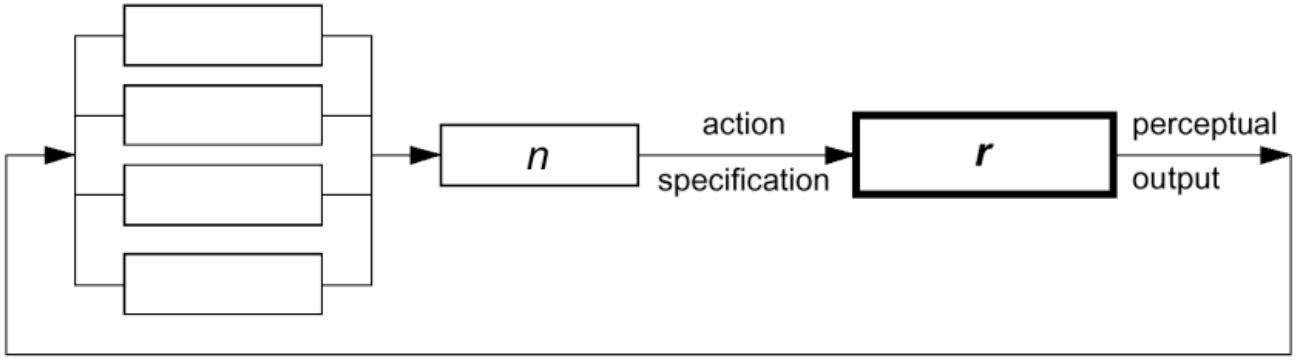
Control Decomposition

- Models the way in which each module's output contributes to the overall robot control outputs.

- Pure serial decomposition:

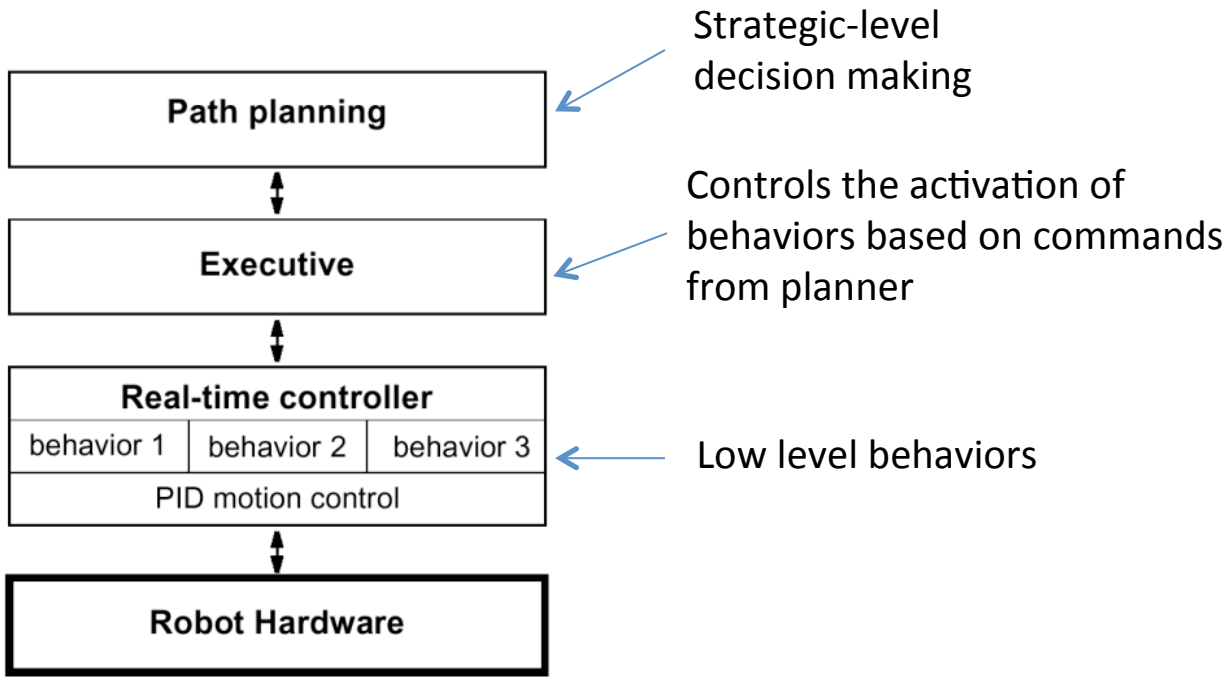


- Pure parallel decomposition:



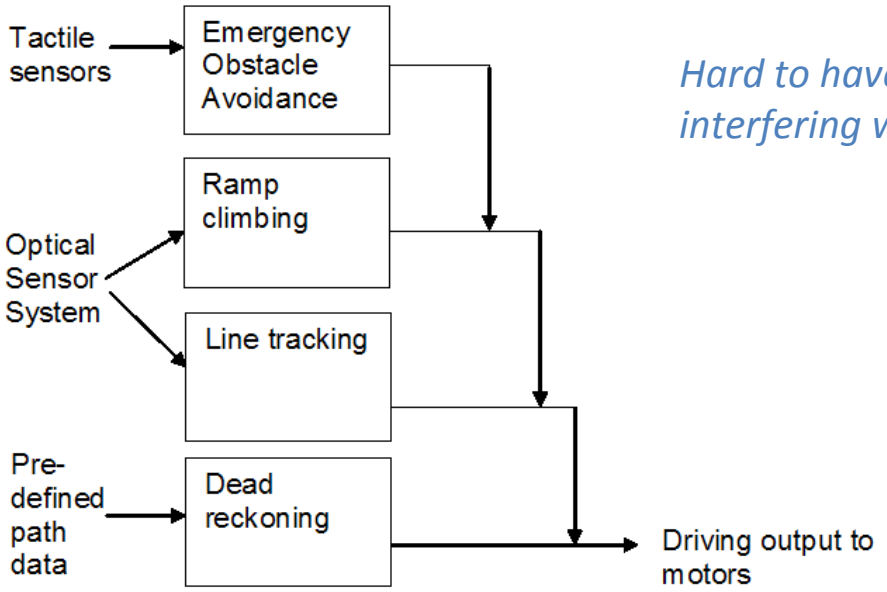
Example

- Tiered mobile robot architecture based on a temporal decomposition



3) Subsumption Architecture

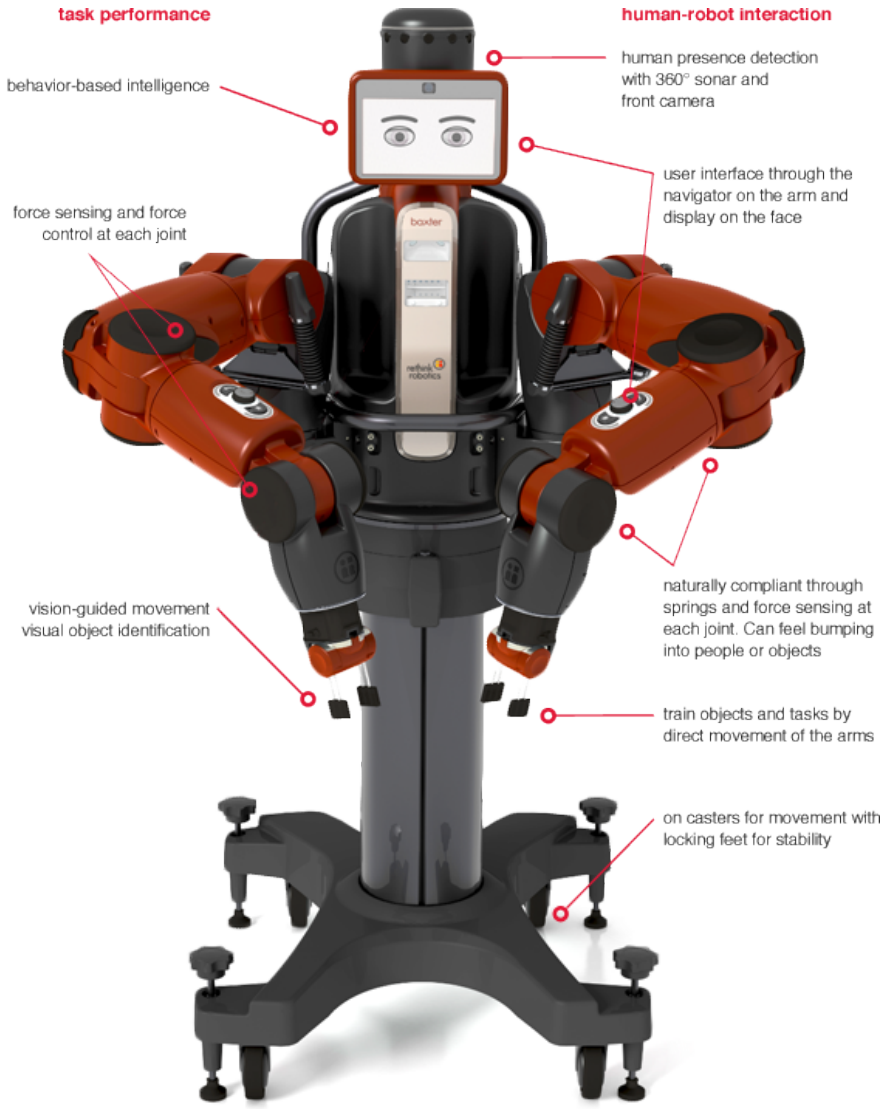
- Formed using a collection of concurrent behaviors placed in layers
- The higher-level behaviors always, if triggered, subsume the output of lower behaviors and therefore have overall control



Hard to have many layers, goals begin interfering with each other

Rodney Brooks

- Subsumption Architecture
- iRobot (1990)
- Rethink Robotics
  - Baxter: 2012



Robot Components

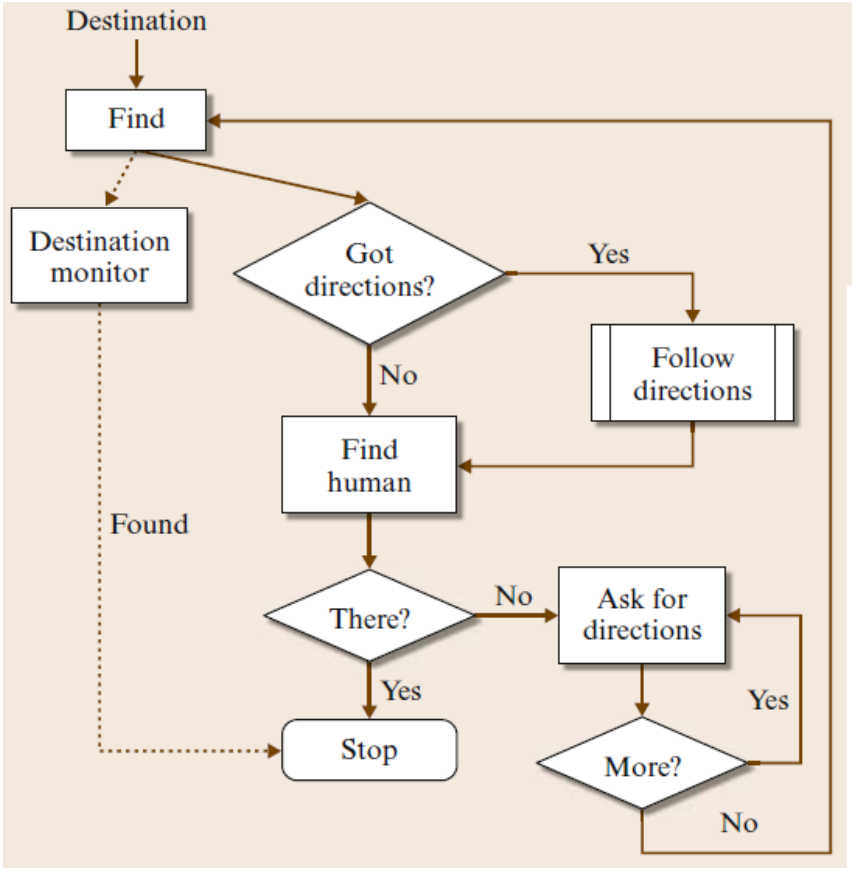
Architecture Components

- Perception
- Planning
- Obstacle avoidance
- Stability control
- Learning
- Human-robot interaction
- Short-term and long-term memory

Resources to be Controlled

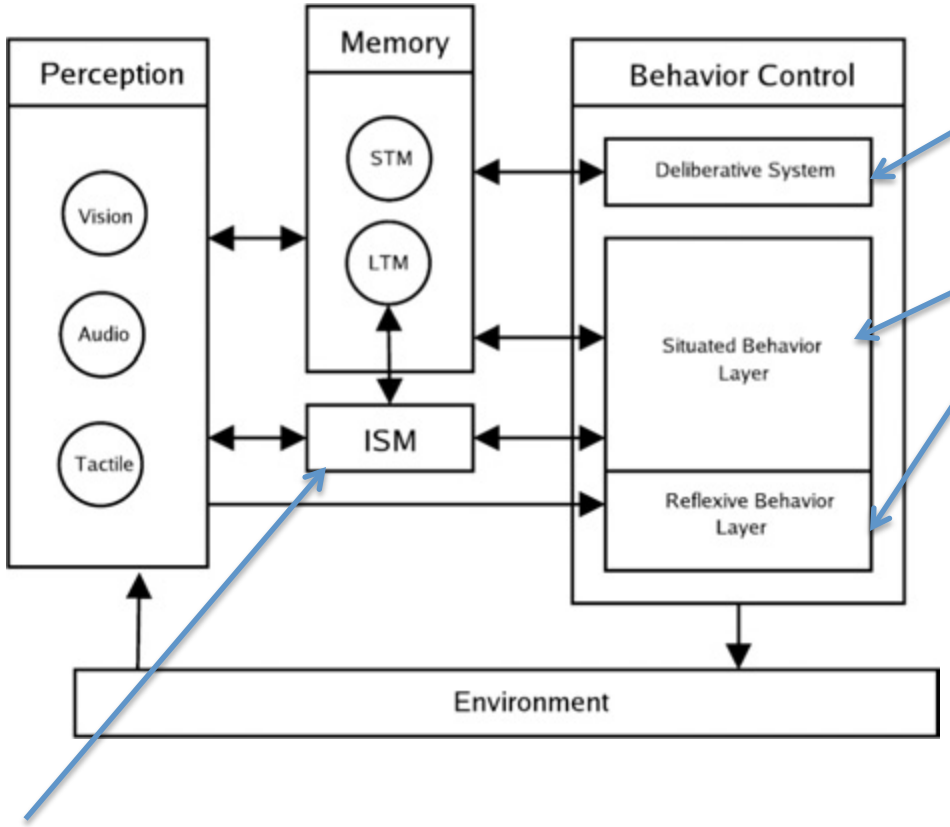
- Actuators
- Communications
- Chassis
- Processor
- Power
- Payload

The Robot GRACE



Grace is a six-foot tall robot with a digitally animated face on a flat computer screen. She autonomously registered for a national meeting on artificial intelligence, found her way to a conference room where she gave a PowerPoint presentation about herself and then answered questions.

EGO Architecture for the Sony QRIO

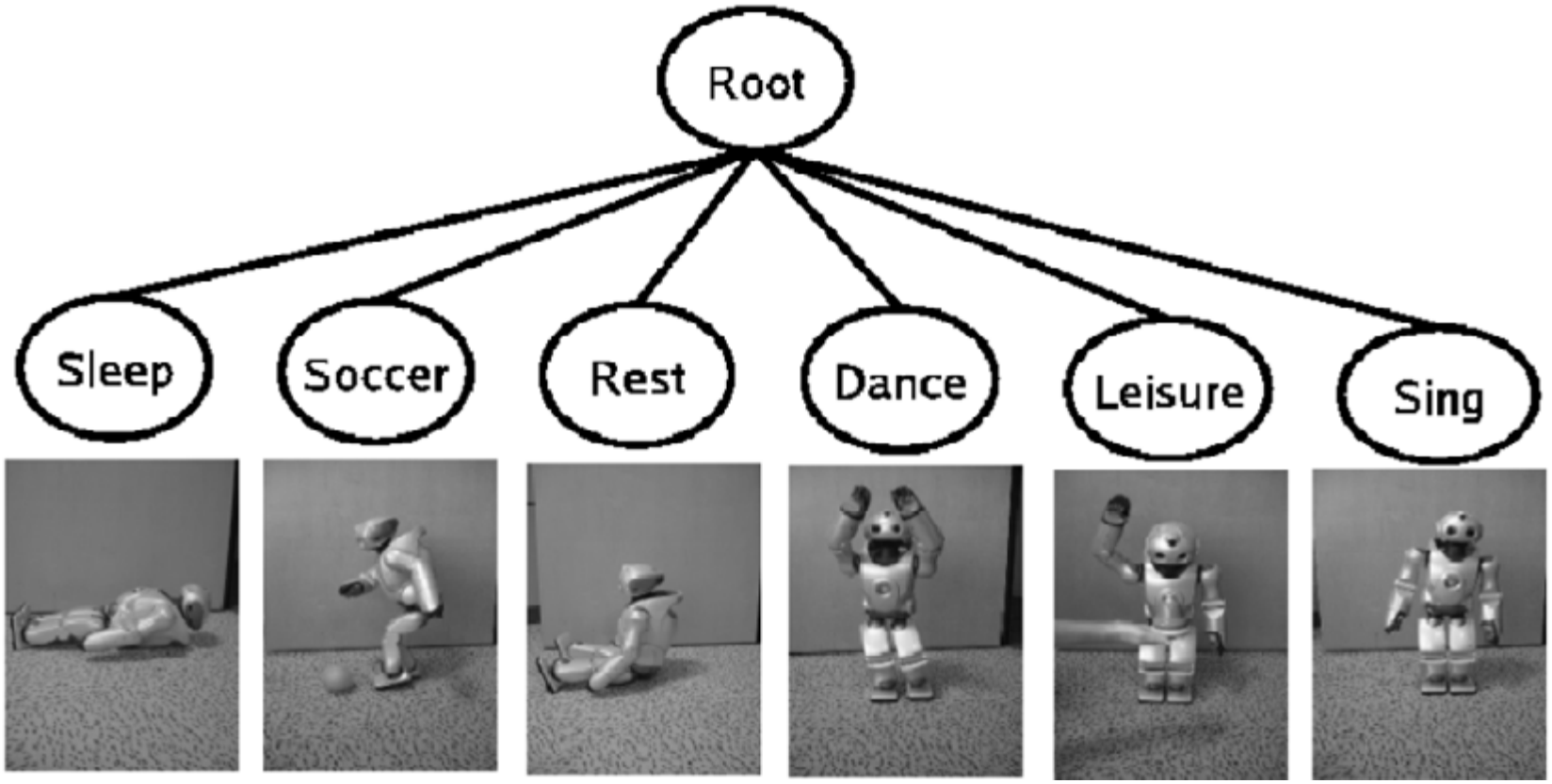


Deliberative planning, control of sequential behaviors

Homeostatic behaviors (sleep, rest)

Reactive behaviors

Internal State Module: nourishment, sleep, fatigue, vitality, etc.



https://www.youtube.com/watch?v=qcJO3NY1C2I

Behavior Selection

- Behavior cycle rate: 2 Hz

- On each cycle
  - every behavior calculates an activation level
  - indicates the relevance of that behavior in the current situation
  - Calculated via:
    - external stimuli
    - internal state of the robot
    - intentional values provided by the Deliberative System

- Behavior selection occurs is greedy:
  - behavior with highest AL selected first
  - other behaviors, from highest to lowest AL value, then selected for concurrent execution *iff* resource demands do not conflict with those already chosen

Designing a Robotics Architecture

- What tasks required?
  - long-term vs. short-term
  - User-initiated vs. Robot-initiated
  - Repetitive or different across time?

- Actions necessary to perform tasks?
  - Representation of actions
  - Coordination of actions
  - Speed of actions / how often changed
  - Necessary speed for safety of robot

Designing a Robotics Architecture

- Data
  - What is required?
  - How obtained?
  - What sensors?
  - What representation? How abstracted from sensors?
  - Data update rate
    - minimum required
    - maximum possible

Designing a Robotics Architecture

- What computational capabilities will the robot have?
- What data will these computational capabilities produce?
- What data will they consume?
- How will the computational capabilities of a robot be divided, structured, and interconnected?
- What is the best decomposition/granularity of computational capabilities?
- How much does each computational capability have to know about the other capabilities?
- Are there legacy computational capabilities (from other robots, other robot projects, etc.) that will be used?
- Where will the different computational capabilities reside (e.g., on-board or off-board)?

Designing a Robotics Architecture

- Who are the robot's users?
- What will they command the robot to do?
- What information will they want to see from the robot?
- What understanding do they need of the robot's computational capabilities?
- How will the user know what the robot is doing?
- Is the user interaction peer to peer, supervisory, or as a bystander?

Designing a Robotics Architecture

- How will the robot be evaluated?
  - What are the success criteria? What are the failure modes? What is the mitigation for those failure modes?

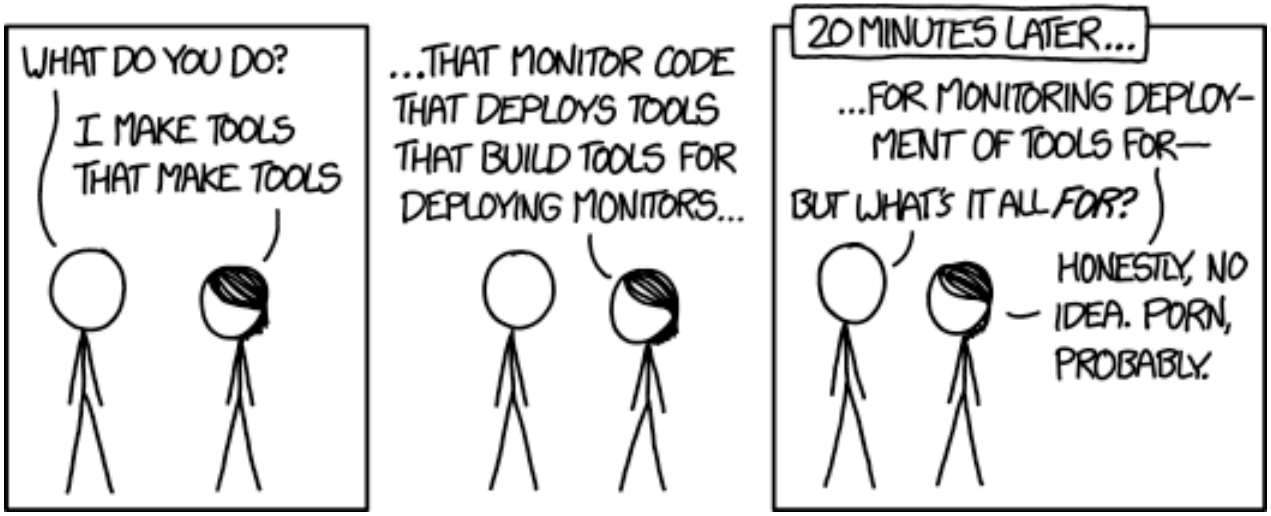
- Will the robot architecture be used for more than one set of tasks? For more than one kind of robot? By more than one team of developers?

References

- Buede, Dennis M. The Engineering Design of Systems: Models and Methods, Second Edition. John Wiley & Sons. © 2009. Books24x7. http://common.books24x7.com/book/id_31904/book.asp
- James Goodwin and Alan Winfield, “A Unified Design Framework for Mobile Robot Systems”, Workshop Proceedings of SIMPAR2008, Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, Venice(Italy), 2008, November 3-4.
- Siciliano, Khatib, Springer Handbook of Robotics, 2008.

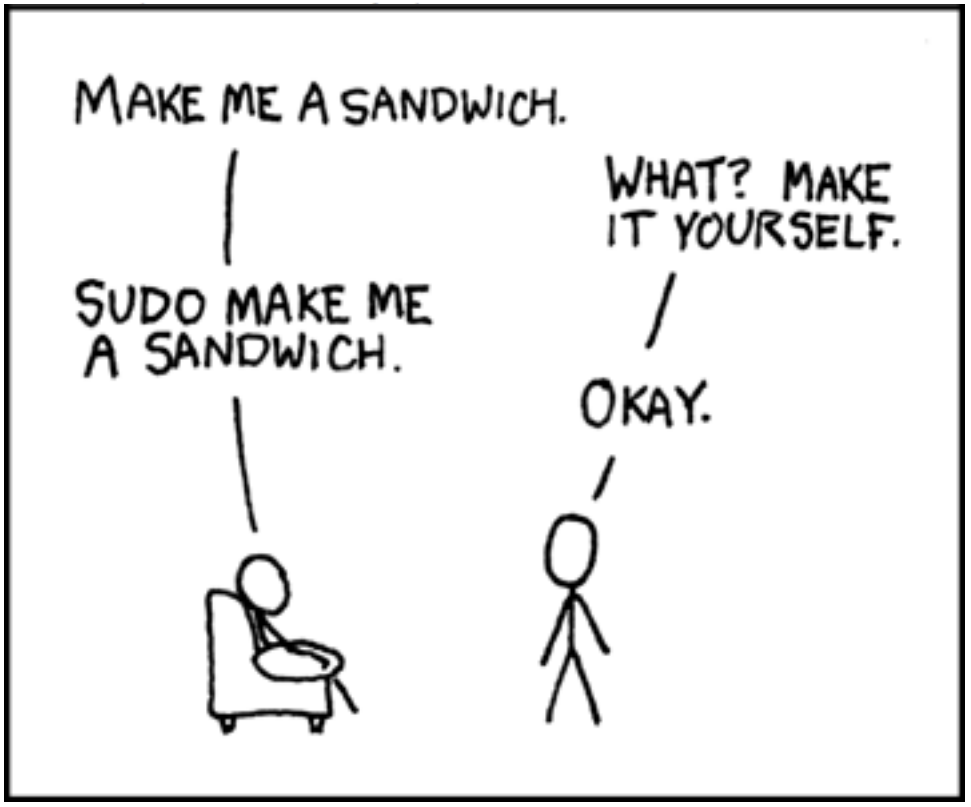
ROS

- P2P: message passing
- Tools-based



```
user@hostname$ sudo sh -c \
  'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > \
  /etc/apt/sources.list.d/ros-latest.list'
user@hostname$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
user@hostname$ sudo apt-get update
user@hostname$ sudo apt-get install ros-indigo-desktop-full python-rosinstall
user@hostname$ sudo rosdep init
user@hostname$ rosdep update

user@hostname$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
user@hostname$ source ~/.bashrc
```



Simple Vision

- How would I find the red ball?

Simple Vision

- How would I find the red ball?
- What if it's moving?