Organic Computing

Dr. rer. nat. Christophe Bobda
Prof. Dr. Rolf Wanka
Department of Computer Science 12
Hardware-Software-Co-Design

Organization and design of autonomous systems

Outline

- Organization and design of autonomous systems
  - Terminology and Concepts
- Architecture
  - Functional architecture
  - Operational architecture
  - Implementation architecture

Terminology and Concepts

Autonomous systems - terminology

- Present in any system that can reach a specified goal or perform a specified task independently
- The autonomy aspects of the behaviour are not very interesting if the goal or task is specified in terms of very detailed parameters of the system itself
  - This is the domain of control theory
- More interesting is a system that can reach a goal or perform a task that is given in terms of parameters or properties of the world around it
  - Translation of the task description into internal parameters which it can control
  - Observe corresponding relevant world parameters and compare those to internal state parameters
  - Decide on its actions on the basis of this comparisons which may involve some form of planning or reasoning

Autonomous systems - terminology

- The behaviour of such a system appears to us, observers, as much more autonomous
- The system is described as intelligent if it performs better and better every time it encounters similar circumstances
  - It seems to 'learn from experience‘
- In this section, we considerer the principles of behaviour and designing such intelligent autonomous systems,
- Autonomous system is a developing field, and one of the difficulties we face is that not all terms have the same meaning for everybody
- relevant terms such as 'perception', 'autonomy', 'action', 'behavior', 'goal-directed', 'learning' and, especially, 'intelligent' must be defined carefully to be meaningful in our descriptions

Autonomous systems - terminology

- The body of an autonomous system is the part of the world that is inside the system
- We defined the Environment as the world outside a system
- External parameters and external variables are parameters and variables that characterize the environment is in a particular representation
- Internal Parameters and internal variables characterize the body
- The internal variable and internal parameters that can be directly controlled by the system are called control variables and control parameters
- Distinction between variables and parameters is related to the level of abstraction.
    - Parameters on lower level may be variable on a higher level

Autonomous systems - terminology

- The parameters and variables need not be quantities that can be measured by particular physical sensors in the system
  - they may be more abstract, higher level concepts, which can be derived from the physical measurements.
- The perception or observation is the indirect measurement of parameters and variables at a given level of hierarchy
- An abstract sensor that do perception is called virtual sensor
  - Does not measure physical values, but some form of internal representation of those measurements
  - Ex: wall sensor, thresholding in image representation
- Virtual actuators are defined similar to virtual sensors
- An internal representation is a representation of perception data on different level of abstraction

Autonomous systems - terminology

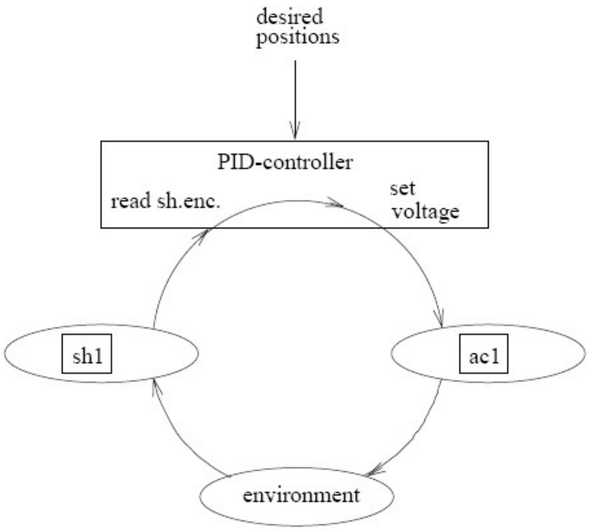
- An internal representation is a representation of perception data on different level of abstraction
- At every level of abstraction, a reasoning component is present between (virtual) sensors and (virtual) actuators
  - Implementation is the domain of artificial intelligence, machine learning
- Viable representation, language, framework and formalism is very important to represent data that flow between the sensors and the actuators
  - Representations may have an enormous effect on the capability of an autonomous system
  - Sometimes so much that a particular approach suddenly becomes feasible whereas before it was not
- The framework defines the world model
  - Instantiation of the world model defines the world parameters at a given time

Autonomous systems - example

- Example: Interleaved sensing and actuation

  A motor cart without adaptive capabilities must drive autonomous with a constant speed

  - Task specified by the desired position in shaft encoder counts for the motor
  - The output of the system is measured continuously by the shaft encoder and fed back to be compared with the desired position
  - A proportional plus integral plus derivative (PID) controller can be used in this case to computes the voltage V to be applied to the motor based on the error e using the gains Kp, Ki and Kd:
    - The proportionality constant Kp is the gain which amplifies the error e.
    - The integral constant Ki is used to decrease the steady-state error
    - The derivative constant Kd determines the rate of change of the error



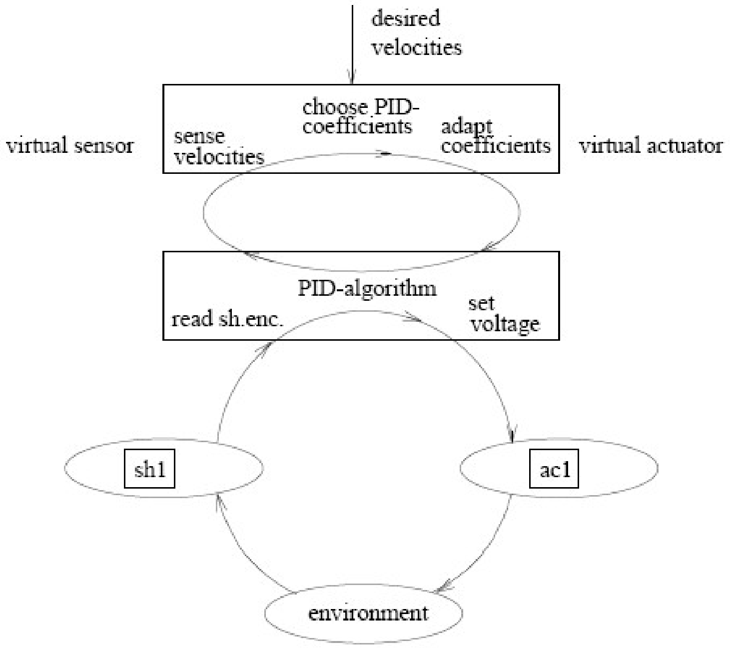
$$\mathbf{V} = K_p\mathbf{e} + K_i \int \mathbf{e}dt + K_d\dot{\mathbf{e}}$$

Autonomous systems - example

- Example: Interleaved sensing reasoning and actuation

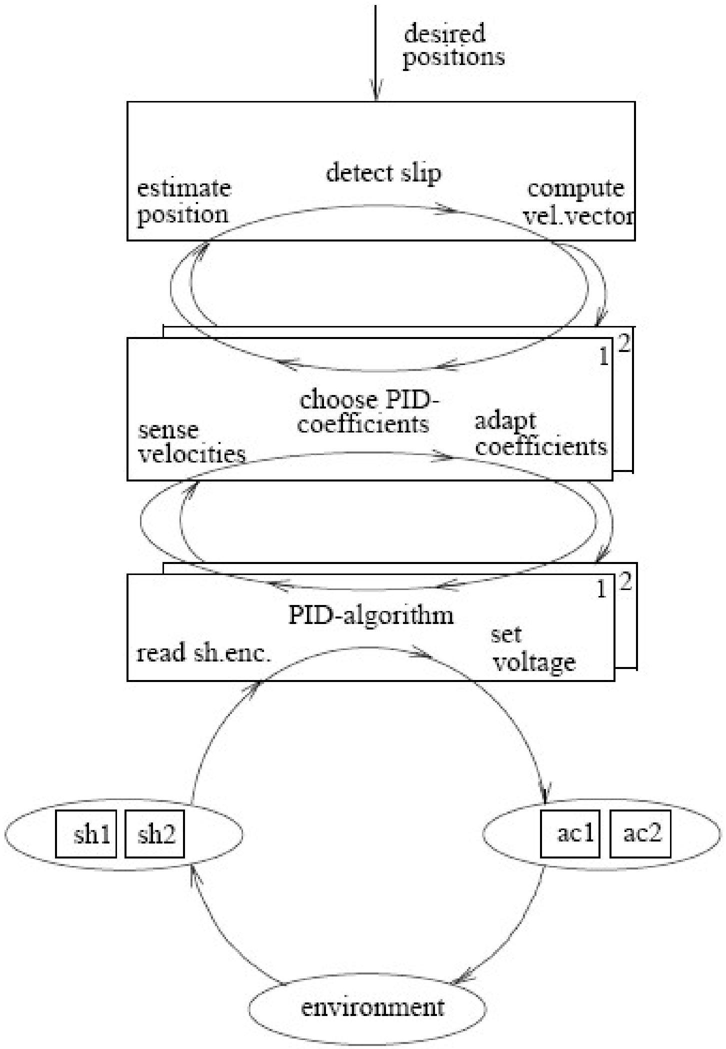
  the cart has to drive to a specified position, starting with velocity zero, reaching a certain constant velocity with a certain specified acceleration and slowing down with a certain specified deceleration

- PID-control can provide a better performance of the cart when the PID-parameters are adjusted depending on the situation
  - This means, we try to find the "best" PID-coefficients for a number velocity domains.
  - Sense the cart’s velocity and choose the optimal PID-parameters
  - Introduce a new level of hierarchy in which the PID-parameters are adjusted.
  - At the lowest level the control variable is changed according to the desired and the actual value.



Autonomous systems – example

- More complex behaviour for the cart
  - Two driving motors for each of the rear wheels
  - The steering is done by the difference of the velocities of the rear wheels.
  - The behaviour of the two motors must be combined to control the cart as whole
  - The low level control loop consists of the concurrent control loops for the two motors
  - A new level of abstraction is added in which the combination of the individual motors is realized
  - Reasoning on this level if for example the detection of slip (when the motors are expected to behave in the same way and they don't act like this)



Autonomous systems – Design approaches

- Top down vs. bottom up
  - Bottom up approach considered so far
  - The bottom-up approach makes clear what is needed in detail for the control of an autonomous system
  - New levels of abstraction must be added until we reach a level where the communication with the device is possible
  - At the inverse, top-down approach starts at the top level where general commands are specified by the human
  - The human doesn't want to communicate with a robot on a low level of abstraction
    - It is for example easier to specify a path in terms of starting position and end position, instead of a sequence of desired positions.
  - The human communicate on a higher level of abstraction using a symbolic way
    - go from A to B, or drive along a wall, or park.

Autonomous systems – Design approaches

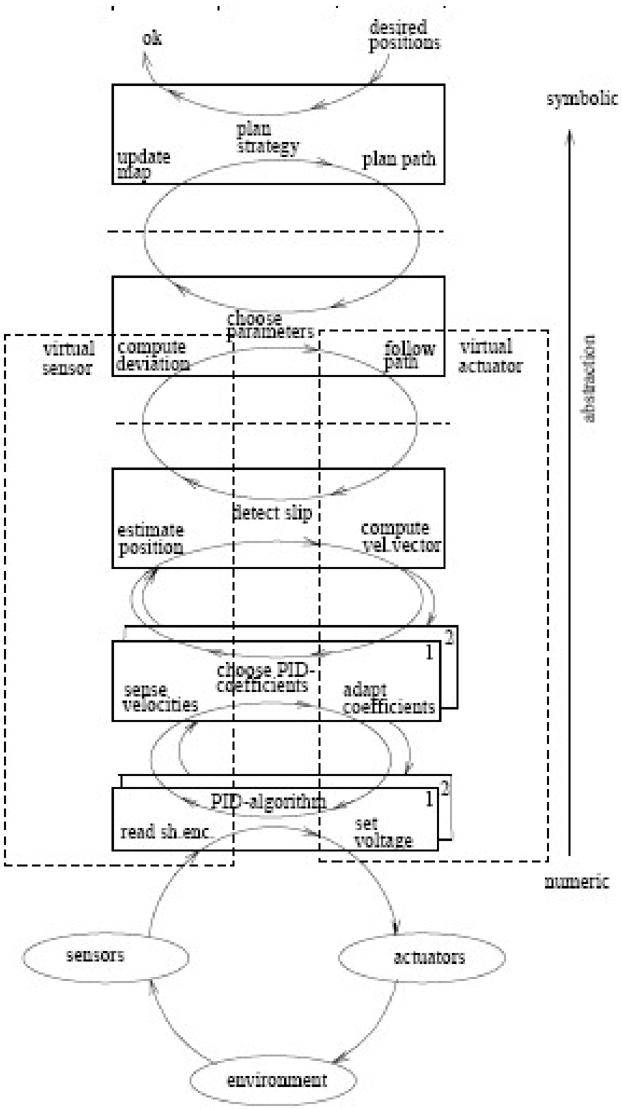
- The human communicate on a higher level of abstraction using symbolic way
  - go from A to B, or drive along a wall, or park.
- Human commands have to be translated through the successive levels until the lowest level
  - The voltages for the motors are obtained.
- Each lower level of abstraction investigates details ignored at the previous level.

Autonomous systems – Design approaches

- Example: a cart which has the goal to drive from position A to B, where A and B are in different rooms connected by a corridor
- We assume the map of the environment is known and that the cart uses shaft encoders and ultrasonic sensors to sense the environment
- First, decompose the task in a number of subtasks based on the knowledge in the map
- This reasoning on the highest level is called the strategy planner
- Strategy:
  - drive to the corridor, drive through the corridor until the door of the second room is reached, drive to B.
  - Virtual actuators are needed for corridor and room at this level, and modules like "drive through corridor" as possible actions.
  - For each of the subtasks a path has to be planned by a path planner.

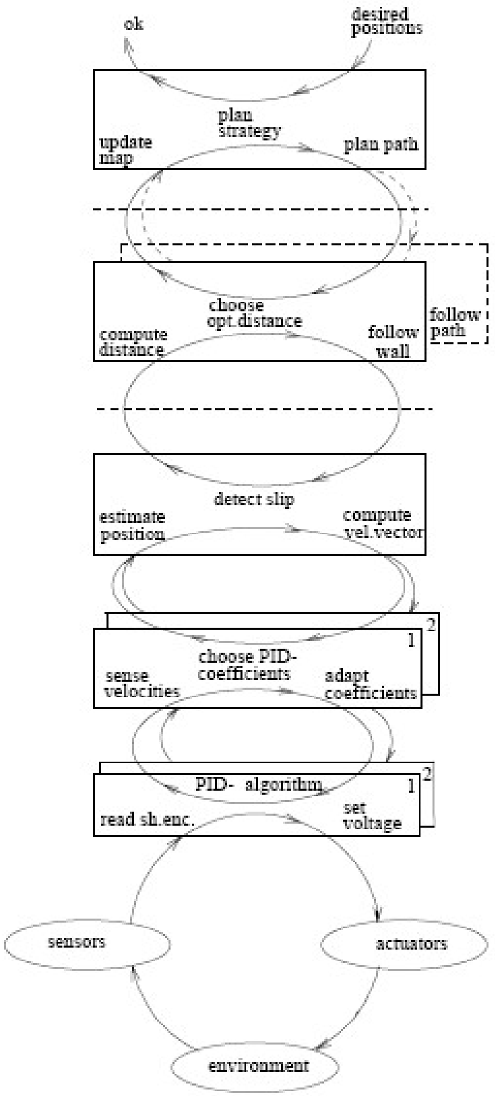
Autonomous systems – Design approaches

- A "Path follower" is necessary to follow the specified path
  - Input for the path follower comes from the virtual sensor "compute deviation" which computes the deviation from the desired path based on the outcomes of the shaft encoders
  - The corresponding control variable "deviation from path" is controlled by the virtual actuator "follow path", which tries to follow the path as close as possible
  - The output of the path follower has to be translated into lower-level commands until finally the setpoints for the PID-loops are obtained.



Autonomous systems – Design approaches

- Representing alternatives
  - Alternatives behaviour, strategies and algorithms can be represented by adding dashed planed to diagrams
  - An alternative results from the combination of available components or strategies on the a given level of hierarchy
  - In the cart example we could use a wall follower to follow the wall in the corridor at a certain distance
  - The wall follower is an alternative for the path follower and should be activated when the cart reaches the corridor
  - Also an alternative sensing module for the wall follower, namely a module which computes the distance to the wall using the ultrasonic sensors is needed

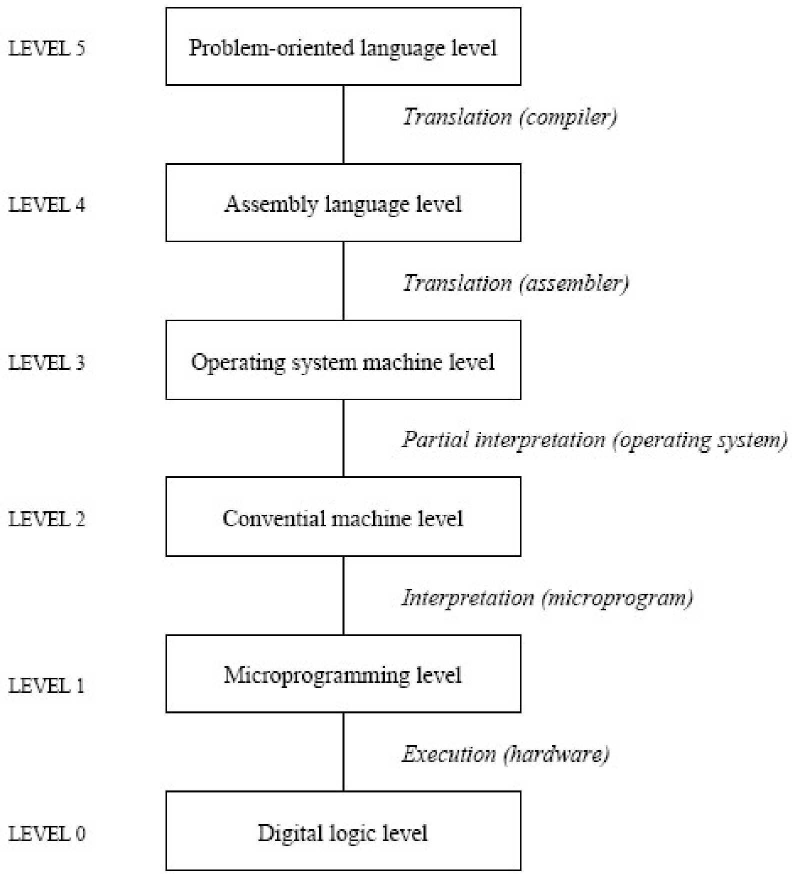


Autonomous systems – Abstraction

- The essence of abstraction is to extract essential properties while omitting inessential details
  - Abstraction separates concept from implementation details
- The successive decomposition of a system in hierarchy levels shows abstraction in its most pure form
- Each level of decomposition shows an abstract view of the lower levels purely in the sense that details are designated to the lower levels
- The decomposition of a system into components is highly context dependent
- The result is not only the components, but also the relationships between those components, to create the whole again
- Abstraction is the key principle that is used for decomposition

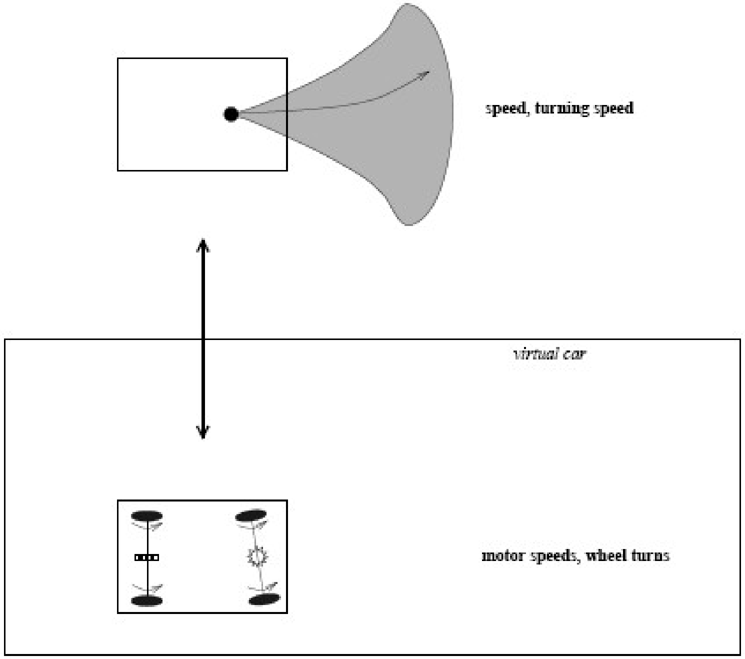
Autonomous systems – the virtual machine

- The term virtual refers to a characteristic whose existence is simulated by software rather than actually existing within hardware.
- A virtual machine is a hypothetical computer, whose characteristics are defined by its machine language, or instruction set
- A computer can then be viewed as series of virtual machine layers, on top of each other
  - The simplest is the bottom-most machine language and the
  - The highest language or level is the most sophisticated.



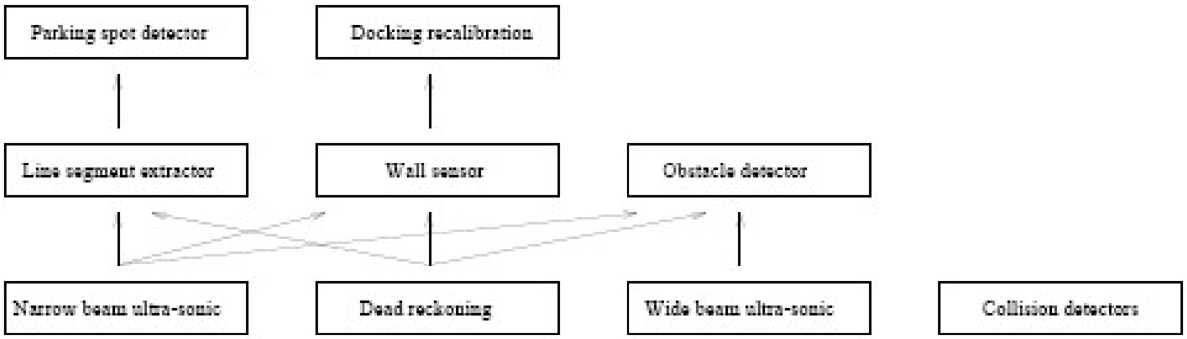
Autonomous systems – the virtual robot

- The translation of virtual machine concept to autonomous control results in the model of a stack of control levels
  - Each level is represented by a language
  - The lower level is the level of the robot electronic
  - The next level provides an interface to a more general robot, independent from the underlying hardware
  - A virtual robot layers depends on the lower layers, but can work independently from the higher layers



Autonomous systems – the virtual sensor

- Used to bridge the gap between the complex symbolic models needed for symbolic reasoning, and the numeric data available from physical sensors
- In every virtual layer the detailed data from the lower layer is combined into data for the higher layer
  - Sensor data fusion
- Virtual sensors maintain the world model
  - Can be central or distributed



Architecture of autonomous computing Systems

Functional Architecture

Autonomous systems - Functional Architecture

- A well-designed architecture shows the desired functionality, without the intention to pin the implementation to certain solutions
- A designer of a functional architecture concerns himself with the functional behaviour that the system should exhibit
- Many applications need the same sort of functionalities, and only differ in the importance of the different functionalities
- A functional architecture should be so general, that it can be (re-)used for different applications
- Appropriate medium to compare different systems
- This chapter indicates the functionalities generally needed for autonomous systems
- Two general ways exist to describe autonomous systems

Autonomous systems - Functional Architecture

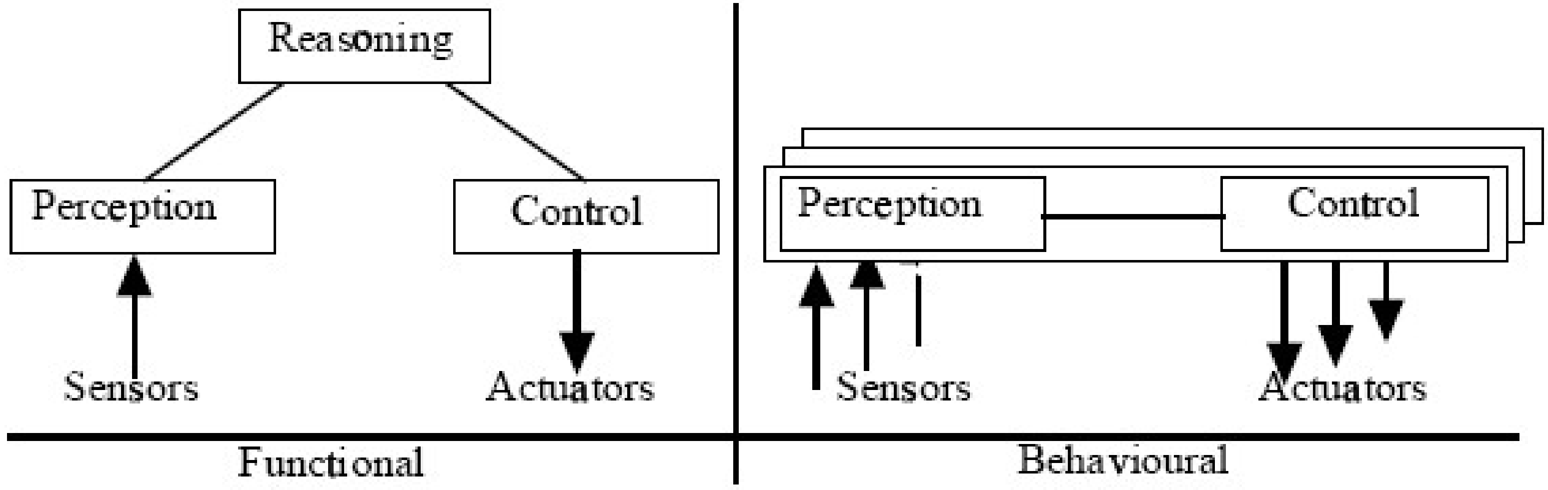
- Hierarchical approach: the assumption is made that on the highest level an abstract model of the world exists
  - Decisions are made based on this model, which are translated into commands for the actuators via several layers
  - The sensor processing branch is in this view responsible for the initialization and maintenance of the model by combination and integration of the information from different sources
- The power of this approach is the transparent control structure of the system
  - decisions are made at highest level, translated in commands, which are executed by lower levels
- The drawback of this approach is the overhead which is needed to maintain such an abstract world model
  - the system tends to be as slow as its slowest sensing process, a troublesome property for a real time system as an autonomous robot

Autonomous systems - Functional Architecture

- Behavioural approach: the assumption is made that on the lowest-level algorithms (arbiters) can be found that are able to combine and integrate the steering commands from different sources
- The main idea is to break up the control problem into goals that should be achieved instead of stages of information flow
  - Several controllers can be active at the same time
- No central intelligence.
  - Complex behaviour is the result of a number of competing simple behaviours.
- Multiple parallel data-flows paths are exploited
- The power of this approach is its controller independency
  - this makes this approach robust (one of the controllers can break down with only minor degradation of the overall system capabilities)
  - and easy to extend (the addition of a controller will only influence the arbiter)
- The drawbacks of this approach are its inefficiency and unpredictability
  - A lot of processing and computation work is done in several modules, and it is not clear in advance how the different control signals will be combined

Autonomous systems - Functional Architecture

- Hierarchical vs. Behavioural approach



Autonomous systems - Functional Architecture

- One of the goals of this course is to show the architectural concepts behind autonomous systems
- Mobile robots are a good case study for autonomous systems
  - the environment can not be ignored in a successful mobile system, unlike many industrial manipulation robots
- The majority of industrial robotarms are successfully controlled as an open loop
  - an operator instructs the robot by explicitly teaching it a sequence of motions. The environment is fixed.
- For mobile robots it is nearly impossible to structure their environment
  - This environment is in most cases too large due to the robot's mobility
  - A mobile robot has to adapt itself to its environment, not vice versa

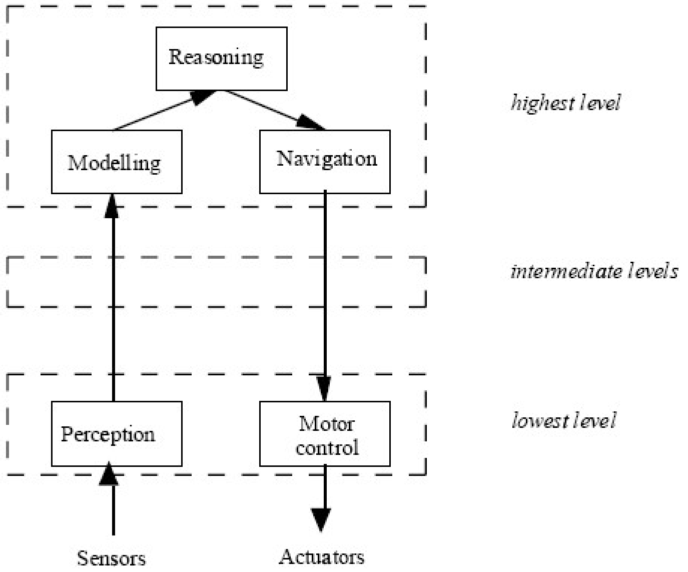
Autonomous systems - Functional Architecture

- Example of hierarchical decomposition
  - The system is divided along functional lines into progressive levels of abstraction
  - The flow of information is used as the main guideline for the decomposition of the system.
  - The design is based on the intuitive decomposition of a complex system in smaller subsystems, that are easier to design
  - the control system is decomposed into levels of abstraction
  - The interconnection between the subsystems connects adjacent layers together
  - Information flows
    - from the sensors to a series of perception and modelling processes,
    - via a reasoning or decision making process,
    - through a series of forward control processes, such as navigation and motor control,

    to the actuators.

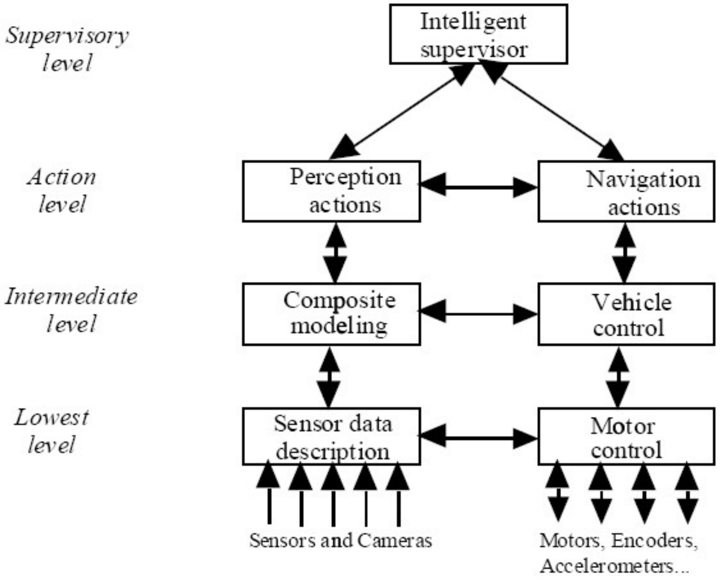
Autonomous systems - Functional Architecture

- Perception interprets the sensor data, and builds abstract representations for it
- The representations used at the intermediate levels are often geometrical primitives like lines, circles, or polynomial objects
- The modelling process uses the perception data to build high-level models of the world
- Symbolic representations are used by the reasoning process to make decisions
- Symbolic data and task instructions as supplied by a human operator
- The navigation module converts the symbolic activities into geometrical primitives
- The motor control module uses the geometric primitives to generate path descriptions for a low-level controller
- This activates the actuators so that the robot vehicle moves in its environment



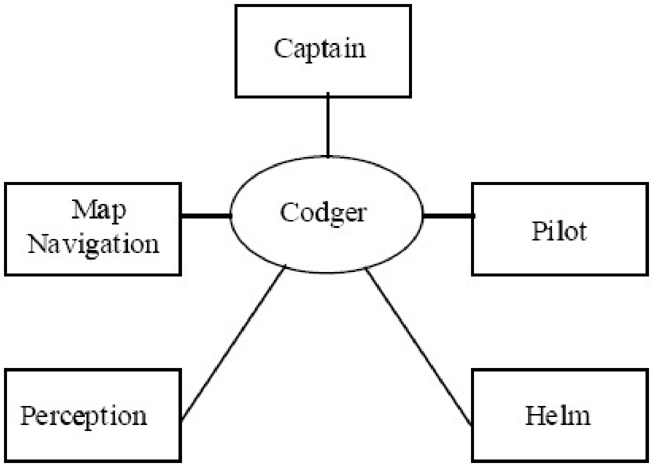
Autonomous systems - Functional Architecture

- Crowley's (1989) surveillance robot
- The architecture is based on a twin hierarchy of perception and control
- An "Intelligent Supervisor“ manages the whole system
  - monitors the execution of each task, and dynamically generates the actions required to accomplish the goals
  - Rule based with a procedural orientated lower-level
- The action level executes actions as required by the supervisor
- Model of the vehicle and environmen is present at intermediate level
- The motor level is the lowest level



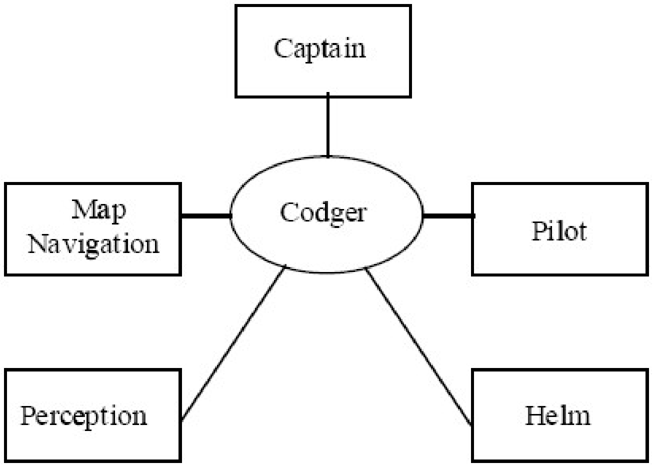
Autonomous systems - Functional Architecture

- Example: CMU's database approach
  - Carnegie Mellon University
- Autonomous outdoor Robot
  - follow the street autonomously, in various environments and under various conditions
- The control system consists of
  - several modules, each dedicated to a special subtask
  - a communication database (Codger) linking the modules together
  - The Captain executes user mission commands and sends each mission's destination and constraints to the Map Navigator
  - The map navigator selects the best route from the database, and sends it to the Helm
  - The helm co-ordinates local navigation continuously within each route segment



Autonomous systems - Functional Architecture

- The Pilot coordinates the activities of Perception and Helm,
  - performing local navigation continuously within a single route segment
- Perception uses sensors, i.e. a color video and a laser range finder, to
  - find objects predicted to be within the vehicle field of view and
  - estimates the vehicle position when possible
- The modules are interconnected by a central database system called Codger.
  - It supports parallel asynchronous execution and communication between modules
  - also handles sensor data fusion



Autonomous systems - Functional Architecture

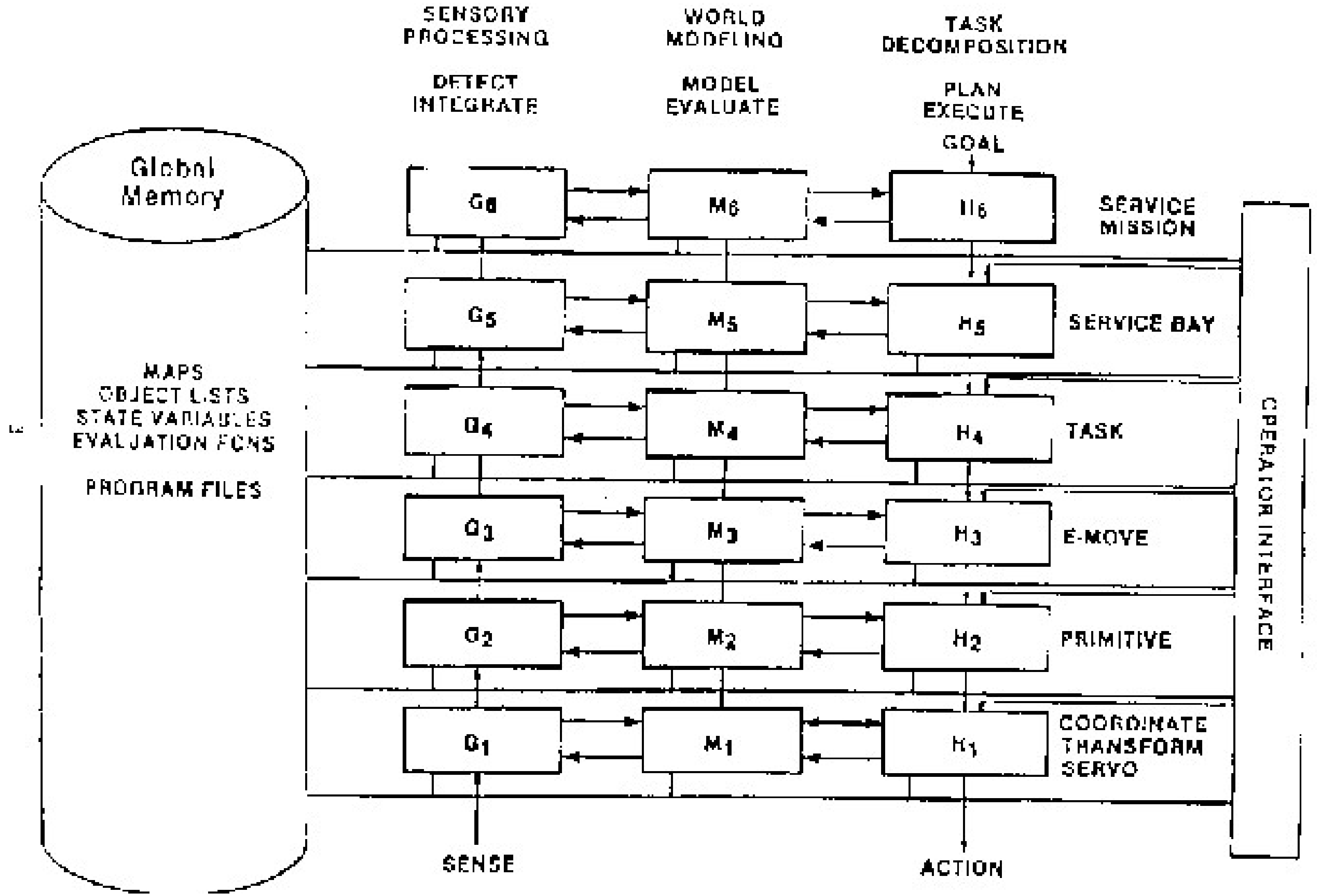
- Example: NASREM
  - In 1987, NASA needed a reference model for the control system of their largest space robot project at that time
    - a long manipulator arm for the space station "Freedom“
  - A teleoperated arm performs services at the space station

- 6 levels of responsibility
  - Service Mission Level (Level 6)
    - decomposition of the servicing plans into service bay action commands
  - Service Bay Level (Level 5)
    - Decomposition of service bay action commands into sequences of object task commands (action to be performed)
  - Task Level (Level 4)
    - decomposition of each object task command into sequences of "elementary move" (E-move) commands
  - E-Move Level (Level 3)
    - E-move commands are decomposed into strings of intermediate (primitive) poses which defines motion pathways that are clear of obstacles and singularities

Autonomous systems - Functional Architecture

- Primitive Level (Level 2)
  - the primitive pose is attained by the generation of a dynamical smooth path expressed by evenly spaced trajectory points
- Servo Level (Level 1)
  - the trajectory points are transformed into joint co-ordinates and joint positions, velocities and forces are servoed to actually drive the equipment.
- Every level in itself is partitioned into three sections:
  - task decomposition, world modelling and sensory processing.
- World modelling is done on geometrical and topological maps, lists of objects with their features and attributes, and tables of system and environmental state variables
- Sensory processing includes signal processing, detection of patterns, recognition of features

Autonomous systems - Functional Architecture

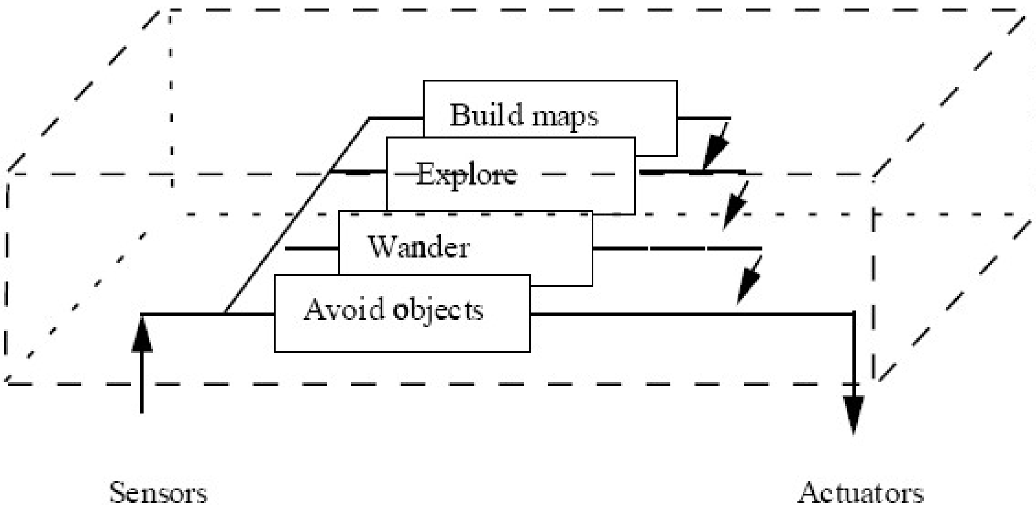


Autonomous systems - Functional Architecture

- Example of behavioural decomposition
  - Brooks' subsumption architecture, 1986
- Decomposes a control system into a set of behaviours
- Each behaviour is a complete control system going from sensory inputs to motor outputs
- Each behaviour is a level of competence, responsible to achieve and maintain a certain goal
- Lower levels represent simple goals, while higher levels perform more complex and situation specific tasks
- Hierarchical layering of behaviours
  - Behaviours use priorities to gain complete control over the actuators

Autonomous systems - Functional Architecture

- If a higher-level behaviour fails, the lower-level behaviours are still active, and no longer inhibited
  - The performance of the system degrades gradually when behaviours fail
  - On the other hand, its performance can get progressively better as more and more levels are added
  - Provide nice possibility for run-time service composition



Autonomous systems - Functional Architecture

- Example of behavioural decomposition
  - Arkin's Robot Schemas Behaviour, 1989
- Behaviour cannot only be generated by chains of modules, but can also be produced by a network of schema instances
- The building block of this approach are the schema instances
- Each schema instance (SI) is a distinct process, applying the knowledge and procedure that is contained in a store: a schema.
- This approach encourages the spawning of multiple schema instances, each instantiated with its own parameters
- The interaction mechanism defines the activity level associated with each schema instance
  - If the activity level is below a certain threshold the instance does not produce output
  - Cooperating instances increase one another's activity level
  - Competing instances lower another's activity level

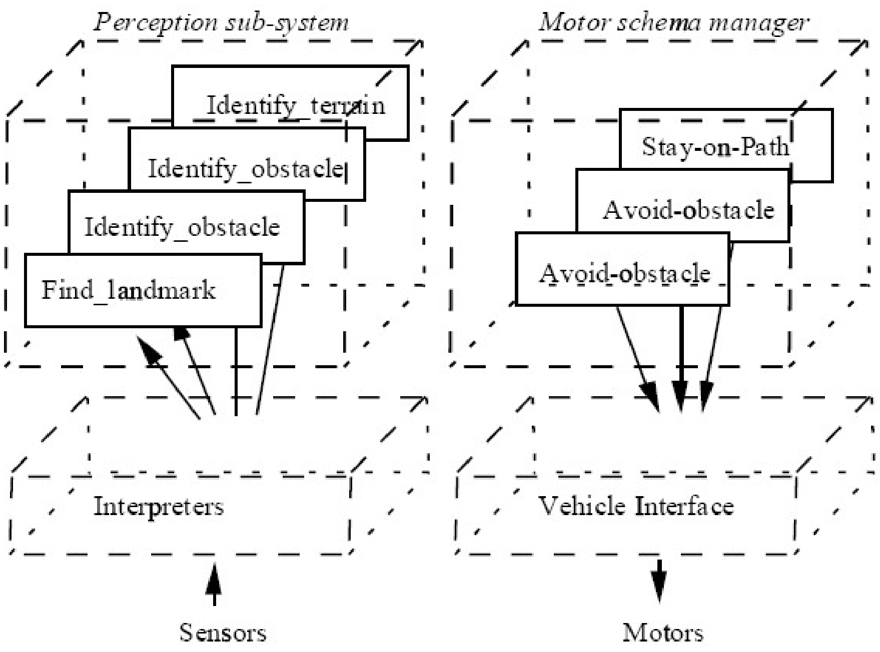
Autonomous systems - Functional Architecture

- As an example of the application is a mobile robot controller for the HARV robot
- Arkin uses different types of schemas
  - Perception schemas are meant to produce sensor independent scene interpretations
  - The activity level of a perceptual schema instance can be interpreted as the confidence in this interpretation
    - Examples of interesting interpretations are landmarks, pathways, and obstacles
    - The corresponding perceptual schemas are find landmark, identify terrain, and identify obstacle
  - Those schemas make use of different interpreters, which have preprocessed the raw data on several layers, before the results are presented to the high-level perception schemas
    - Motor schemas must drive the robot while taking in account the feedback from the environment

Autonomous systems - Functional Architecture

- Arkin's motor schemas do this by producing a velocity vector as output
- The vehicle interface collects all velocity vectors from all concurrent schema instances, sums them up and converts the result into commands for the different motors
- Examples of motor schemas are avoid obstacle and stay on path

- More than one instance of the same schema can be active, for instance if more than one obstacle is in the neighbourhood of the robot

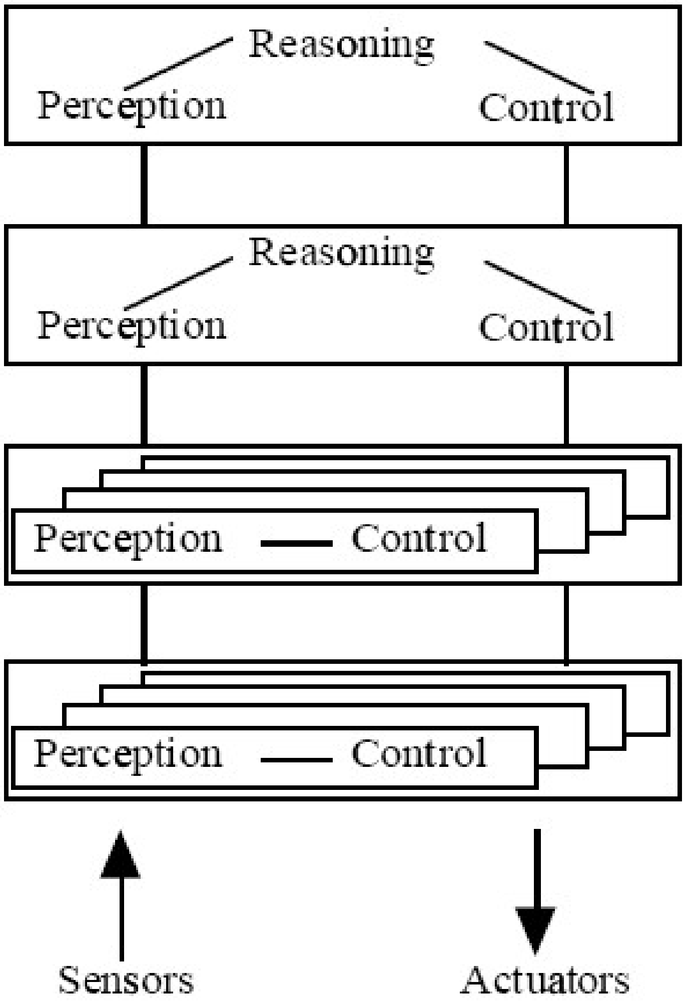


Autonomous systems - Functional Architecture

- Hybrid decomposition
- Incorporates the characteristics of both functional and behavioural systems
  - high-level reasoning is a sequential process
  - real-time robot control involves mostly parallel processing
- Demand for highly abstracted knowledge about the state of the environment
  - This suggests a functional decomposition
- This requires parallel execution of both perception and actuator control
  - Especially when execution takes place in a dynamic environment, real-time sensor information is mandatory to guide the actuator control process

Autonomous systems - Functional Architecture

- Organization of the structure in a number of hierarchical layers
- The internal structure of the levels is functional at the higher levels and behavioural at the lower levels
- Task achieving behaviors are exploited at the lowest level,
- Perception-reasoning-control loops are used at the higher levels



Autonomous systems - Functional Architecture

- Example of hybrid decomposition
  - Payton's hybrid architecture , 1986, 1990, 1991
- Used for reflexive control of an autonomous land vehicle (ALV)
- Allow abstract symbolic plans to modify the performance of low-level behaviours in accordance with changes in goals and environmental context
- The control system is divided into separate perception and planning units
- System divided in four layers
  - The higher levels operate on assimilated data that pertain to long-term decisions
  - The lower layers use more immediate, numerical data
- A number of virtual sensors produce partial world models
  - Aimed at detecting very specialized environmental features

Autonomous systems - Functional Architecture

Payton's hybrid architecture