First National Workshop on Control Architectures of Robots - April 6,7 2006 - Montpellier

A multi-level architecture controlling robots from autonomy to teleoperation

Cyril Novales, Gilles Mourioux, Gerard Poisson
Laboratoire Vision et Robotique, 63 av. de Lattre de Tassigny, F-18000 Bourges
Cyril.Novales@bourges.univ-orleans.fr

Abstract

This paper presents a specific architecture based on a multilevel formalism to control either autonomous or teleoperated robots that have been developed in the laboratory vision and robotics. This formalism separates precisely each robot functionalities (hardware and software) and provides a global scheme to position them and to model data exchanges among them. Our architecture was originally built from the classical control loop. Two parts can thus be defined: the Perception part which manages the processing and the models construction of incoming data (the sensor measurements), and the Action part which manages the processing of controlled outputs. These two parts are divided in several levels and depending on the robot, the control loops that have to be performed are located at different levels: from the basic one (i.e. level 0) composed by the jointed mechanical structure and level 1 which performs actuators servoing, to the highest one (i.e. level 5) which manages the various missions of the robot. The higher the level is, the shorter the loop reaction time has to be. Each level clusters, for their respective part, specific modules which perform their own goals. This general scheme permits to integrate different modules issued from various robot control theories. This architecture has been designed to model and control autonomous robots. Furthermore, a third part, called "teleoperated part", can be added and structured in levels similarly to the two other parts. Distant from the robot, this teleoperated part is managed by a human operator who controls the robot at various levels: i.e. from the first level (the basic remote control) to the upper one (where the operator only controls the robot mission).

Hence, this architecture merges two antagonist concepts of robotics, i.e. teleoperation and autonomy, and allows a sharp distribution between these two fields. Finally, this architecture can be used as a main frame to build its own control architecture, using only a few clusters with dedicated modules. Some examples and experimental results are given in this paper to validate the proposed formalism.

Introduction

When turning a robot on, the problem of its autonomy is quickly addressed. However, several type of autonomies can be considered: energetic autonomy, the behaviour autonomy or smart autonomy. The designer has to choose the way he will give autonomy to his robot. He has mainly two orientations: "reactive" capacities or "deliberative" capacities. These two capacities are complementary to let a robot perform a task autonomously. The designer must built a coherent assembly of various functions achieving these capacities. This is the role of the control architecture of the robot. To design an autonomous robot implies to design a control architecture, with its elements, its definitions and/or its rules.

One of the first author who expressed the need for a control architecture was R.A. Brooks [1]. In 1986, he presented an architecture for autonomous robots called "subsumption architecture". It was made up of various levels which fulfil separately precise function, processing data from sensors in order to control the actuators with a notion of priority. It is a reactive architecture in the sense that there is a direct link between the sensors and the actuators (Figure 1). This architecture has the advantage to be simple and thus easy to implement, nevertheless, the priorities given between the different actions to perform are fixed in time and do not allow an important flexibility.

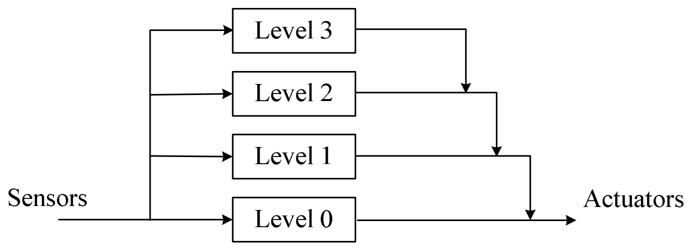

Figure 1 - « Subsumption Architecture »

Then other various architectures were developed based on different approaches, generally conditioned by the specific robot application that the architecture had to control.

The architecture 4-D/RCS developed by the Army Research Laboratory [2] has the main characteristic to be made up of multiple calculative nodes called RCS (Real time Control System). Each node contains four elements, performing the 4 following functionalities: Sensory Processing, World Modeling, Behavior Generation and Value Judgment. Some nodes contribute to the perception, others contribute to the planning and control. These nodes are structured in levels, in which one can find the influence of the reactive behaviors in the lower levels and of the deliberative behaviors in the higher levels. The general management is carried out via communications using a specific language NML (Neutral Messaging Language) and according to the decision made by the Value Judgment modules.

The Jet Propulsion Laboratory developed in collaboration with NASA its own control architecture called CLARAty [3]. Its principal characteristic is to free itself from the traditional diagram on 3 levels (Functional, Executive, Path-Planner) and to develop a solution with only 2 levels which represent the functional level and the decisional level. A specific axis integrates the concept of granularity of the architecture for compensating the difficulties of understanding due to the reduction of the number of levels (Figure 2). One of the interests of this representation is to work at the decisional level only on one model emanating from the functional level. The decomposition in objects of this functional level is described by UML formalism (Unified Modeling Language) that allows an easier realization of the decisional level.

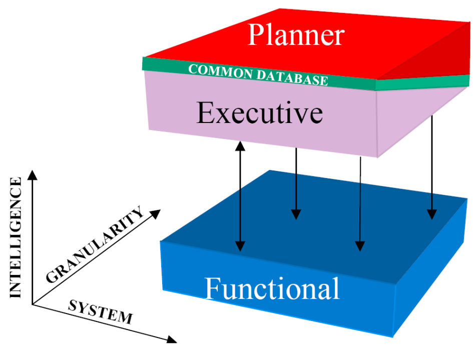

Figure 2 – Two level Architecture

The LAAS architecture (Laas Architecture for Autonomous System) [4] is made up of 3 levels: decisional, executive and functional (Figure 3). Its goal is to homogenize the whole mobile robotics developments and to be able to re-use already designed modules.

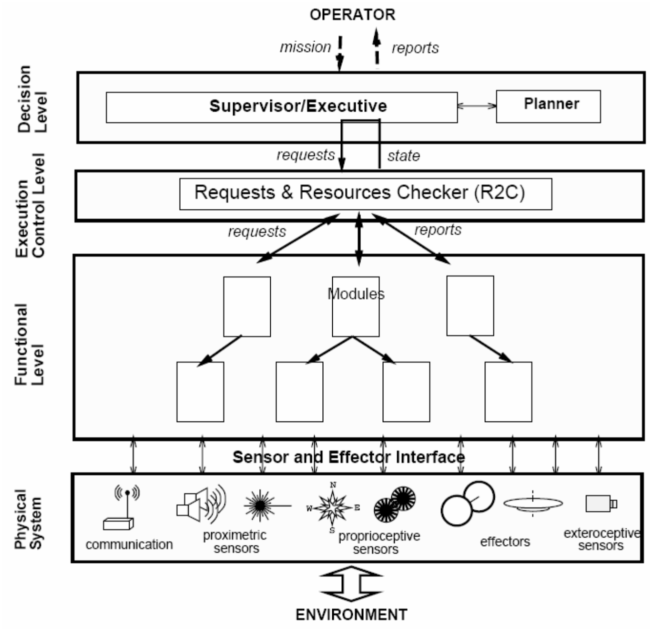

Figure 3 – LAAS Architecture

All the modules of the functional level are encapsulated in a module automatically generated by GenoM. These have to interact directly with the actuators and other modules of the functional level. The higher level is a controller of execution (Request & Ressources Checker). Its main function is to manage the various requests emitted by the functional level or the decisional level. The operator acts only at the decisional level by emitting missions which depend on the information incoming from the lower levels. This architecture has an important modularity even if the final behavior is related to the programming of the controller of execution.

R.C. Arkin describes and uses a hybrid control architecture, called AuRA for Autonomous Robot Architecture [6], including a deliberative part and a reactive part. It is made up of two parts, each using distinct method to solve problems (Figure 4). The deliberative part which uses methods of artificial intelligence contains a mission planner, a spatial reasoner and plan sequencer. The reactive part is based on sensors. A “schema controller” controls the behavioral processes in real time, before they were sent to a “low-level control system” for execution. The deliberative level stays in standby mode and is activated only if an impossibility is generated by the reactive part of the task execution. The levels are progressively activated according to the needs.

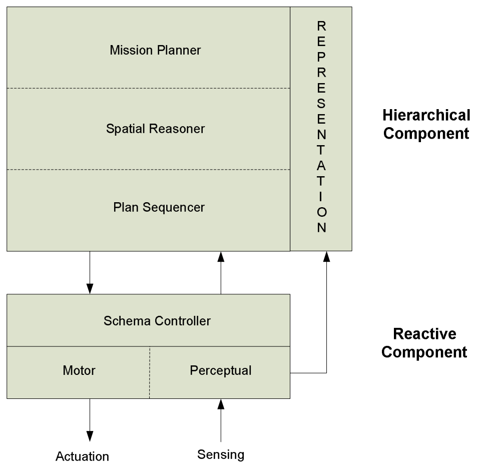

Figure 4 - AuRA Architecture

A. Dalgalarrondo [7] from the DGA/CTA proposed another architecture. It presents a hybrid control architecture including four modules: perception, action, attention manager and behavior selector (Figure 5). It is based on sensor based behaviors chosen by a behavior selector. The “perception” module carries out models using processing which are activated or inhibited by the “attention manager” module. The “action” module consists of a set of behaviors controlling the robot effectors. A loop is carried out with the information collected by the perception part. This is particularly necessary for low level actions.

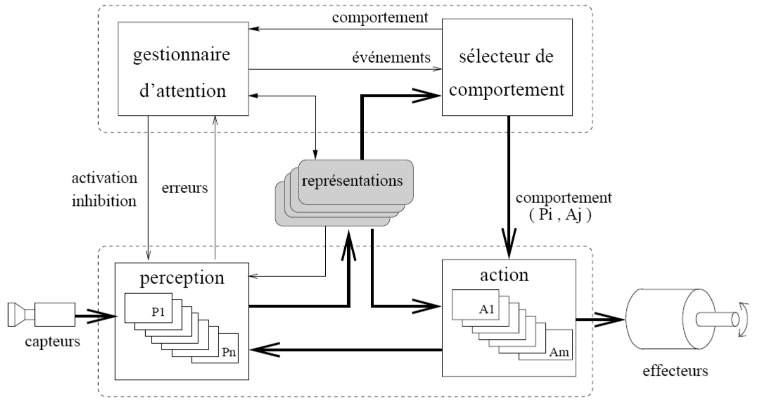

Figure 5 – DGA Architecture

The “attention manager” module is the organizer of the control architecture: it checks the validity of the models, the occurrence of new facts in the environment, the various processing in progress and finally the use of the processing resources. The “behavior selector” module must choose the robot behavior according to all information available and necessary to this choice: the fixed goal, the action in progress, representations available as well as the temporal validity of information.

The DAMN architecture (Distributed Architecture for Mobile Navigation) results from work undertaken at the Carnegie Mellon University (Figure 6). Its development was a response to navigation problems. The principle is as follows: multiple modules share simultaneously the robot control by sending votes which are combined according to a system of weight attribution. Then, the architecture makes a choice of controls to send the robot, by a fusion of the possible solutions [20].

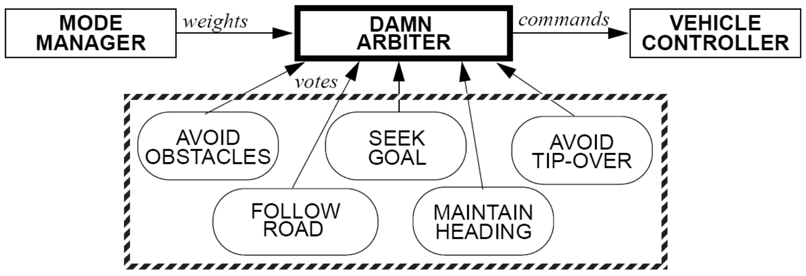

Figure 6 – DAMN Architecture

This architecture proposes the dominating presence of only one module which decides the procedure to follow. This forces to concentrate all the evolution capabilities of the robot. This mode of control does not make it possible to understand or anticipate the probable behavior of the robot with respect to a unexpected situation.

Kim Hong-Ryeol & Al, have suggested a five hierarchical level for an architecture that he patented in 2005 [8]. The physical level represents the robot, the higher level is the function level; it is an assembly of software components. The “Executive level” is level 3 and is composed of “local Agents” which are managed by the “real time manager”. The upper level is the “planning level” which includes 2 on-line and off-line modes. On the upper level of the architecture is the “design level” which represents the possibility for the designer to carrying out modifications of the architecture interactively.

Principle of the proposed architecture

The architecture, propose here, is based on the same architectures principles that have been suggested since the Nineties. It relies on the concept of levels initially developed by R. Brooks and which appear in architectures proposed by AuRA or LAAS. Similarly to the latter one, we have an upstream flow of information from the robot corresponding to its perception, and a downstream flow going down towards (to) the robot and corresponding to the control part. The specificity is to structure this robot control architecture in levels similarly to communication architectures such as the OSI/ISO (Figure 7). Each level can communicate with the higher or lower level by data transfers which follow a predefined format and processing according to the given robot application.

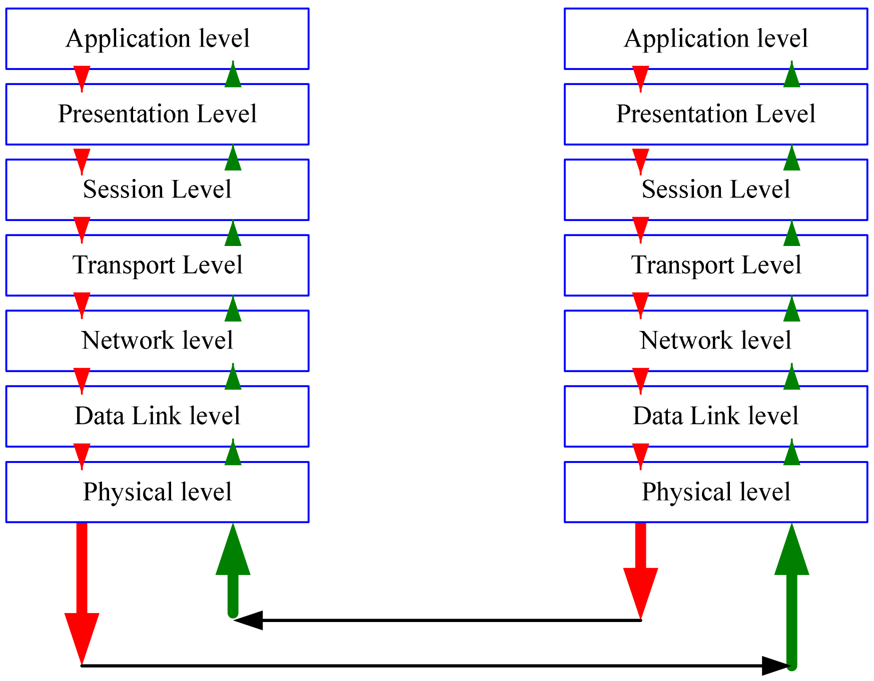

Figure 7 – Architecture of Open System Interconnection (ISO)

However, contrary to communication architectures where data must imperatively flow through all levels, in the proposed architecture, data will be able to either pass through the levels to be treated, or to be transmitted directly to the same level from the upstream part to the downstream part. Thus, data do not have to follow a unique path. They can be routed via multiple paths to perform various control loops. Even when affecting different levels, all these control loops have a common path through the articulated mechanical system (AMS) (Figure 8). The Articulated Mechanical System corresponds to the physical part of a robot.

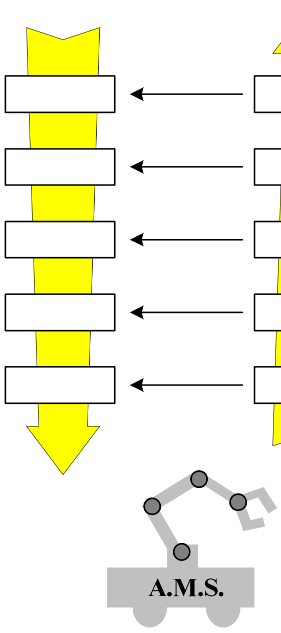

Figure 8 – Data flow inside a robot

The control loops are interwoven and pass through a more or less great number of levels. The loops of a low level are faster than the loops of a higher level, because data are processed successively by a lesser number of levels. We then find the concepts of "deliberative" levels in the higher levels and "reactive" levels in the lower levels.

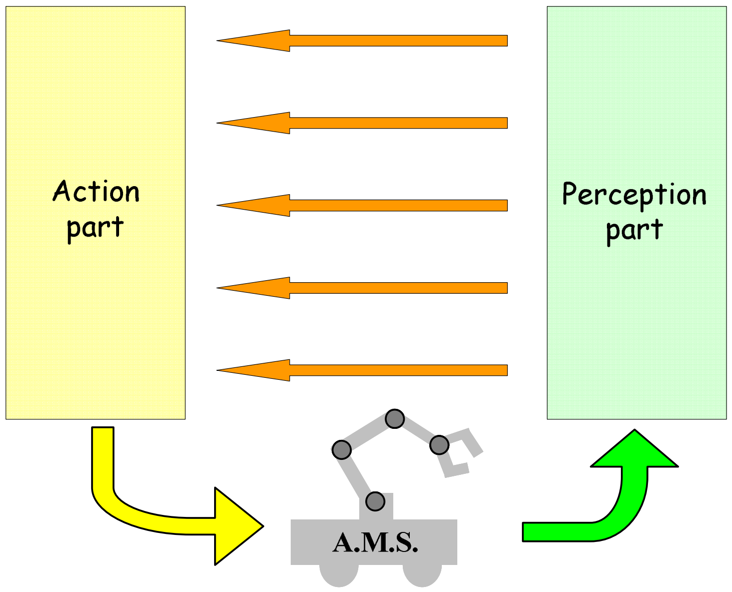

Figure 9 – Robot = AMS part + Perception part + Action part

An autonomous robot as a whole is then modelled by an Articulated Mechanical Structure surmounted by two parts: an upstream part corresponding to the perception of this AMS, and a downstream part corresponding to the action on this AMS. These two parts are divided into levels along which the various control loops are closed (Figure 9). Each level of each part must be clearly specified to allow the designer of the robot to place the respective control loops (articular control, Cartesian control, visual control...). This architecture is embedded in the robot: it defines the autonomous architecture of the robot.

The autonomous parts

Basic levels

Let us start from a basic control loop of a robot: a reference signal, compared with the sensor measurements, is sent in a correction module before being applied to the entry of the robot actuators (Figure 10a). When analysing the electromechanical part (the Articulated Mechanical Structure), two other distinct parts in this control loop are identified: the perception part composed with sensors and their controls, and the action part which gathers all processing steps (i.e. typically articular controls - PID) applied to data that are then sent to the AMS (Figure 10b).

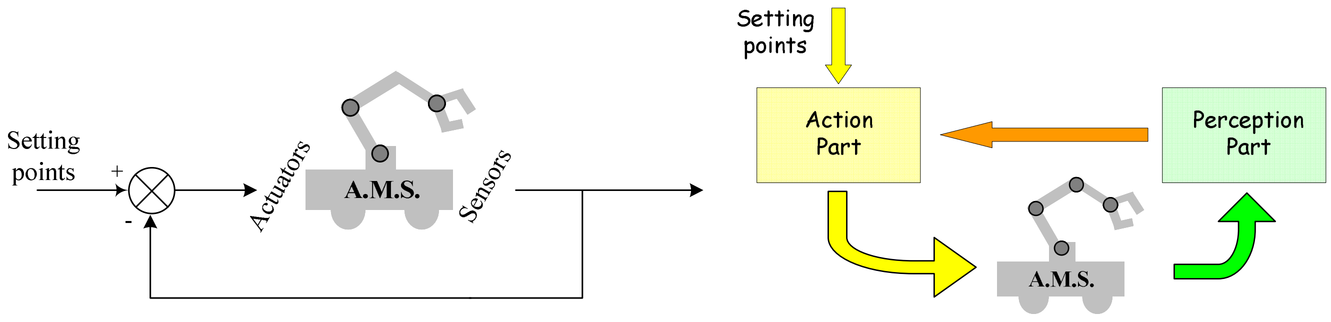

Figure 10 a- classic representation of a servoing loop b- our representation

We call level 0 the articulated mechanical structure. Servoings and sensors are located at the same level, immediately above the articulated mechanical structure; they constitute the level 1 of our architecture. This level corresponds to the shortest – and the fastest – loop of the robot.

At this level, we will find various loops functioning in parallel; for example articular servoings of each robot articulation. We thus define one module for each servoings/sensor system and modules are clustered in each level of each part (action and perception part).

The basic levels correspond to levels 0 and 1. The cluster of the level 1 of the perception part gathers the modules sensors, their respective controls and filters. The cluster of the level 1 of the action part gathers the modules of articular servoings (Figure 11). The setting points of the servoings represent the inputs of the cluster of the level 1 of the action part. The outputs of the cluster of level 1 of the perception part are all the filtered sensors measures. These measures are also transmitted to the modules of the cluster of level 1 of the action part in order to carry out the servoings. Typically, data from the exteroceptive sensors are simply processed and transmitted to the upper level of perception, and data from the proprioceptive sensors are transmitted to servoings (cluster action of the same level).

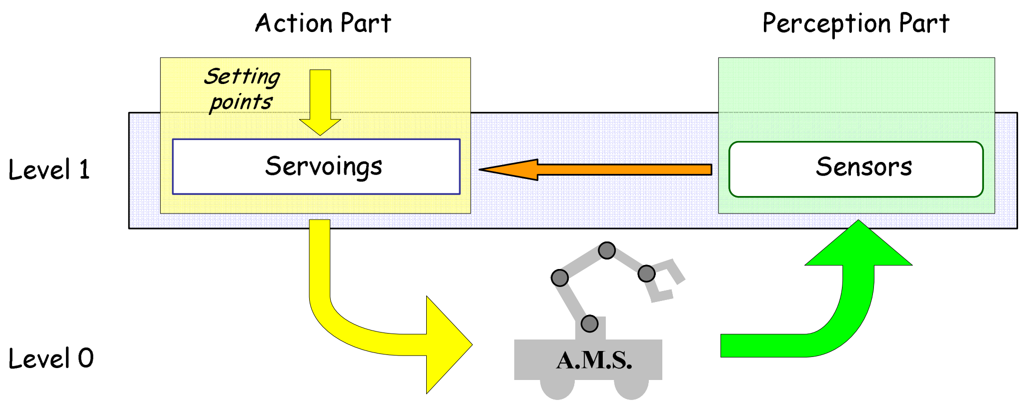

Figure 11 – The level 0 and 1 of the architecture

Level 2: the pilot

To feed the input of level 1 of the action part, the servoings setting points should be continuously provided. This is the role of the 'Pilot' cluster: it generates these setting points (e.g. articular) based on a trajectory provided as an input. This trajectory is expressed in a space (e.g. Cartesian space) different from that of the setting points. It is also a "setting point" but from a higher level, so we do not called thus. This trajectory describes, in time, the position parameters, kinematic parameters and/or dynamic parameters of the robot in its workspace. The function of the pilot is to convert these trajectories into setting points to be performed by the servoings. Typically, the pilot contains the Inverse Geometrical/Kinematics/Dynamic Models of the robot (generally, only one of them is present, according to the robot application). In our architecture, this pilot is positioned on level 2 in the action part.

However, this 'Pilot' cluster does not only contain one IKM module; it can also contain modules which give the robot the possibility to take into account information of its environment. According to the concept of our architecture, this information comes from this level 2 and from the perception part: this is the cluster of the 'Proximity Model' of the robot environment. This 'Proximity Model' cluster contains various modules which transform filtered measurements (coming from the 'Sensor' cluster of level 1 perception) into a model in the same space as that of the trajectory (e.g. Cartesian space). This transformation is performed on line using sensors measures; no temporal memorizing is carried out. The 'Proximity Model' obtained is thus limited to the horizon of sensor measures. Typically, this 'Proximity Model' cluster contains modules which apply the Direct Geometrical/Kinematics/Dynamic Model of the robot to the proprioceptive data, and which express the exteroceptive data in a local frame centred on the robot (Figure 12).

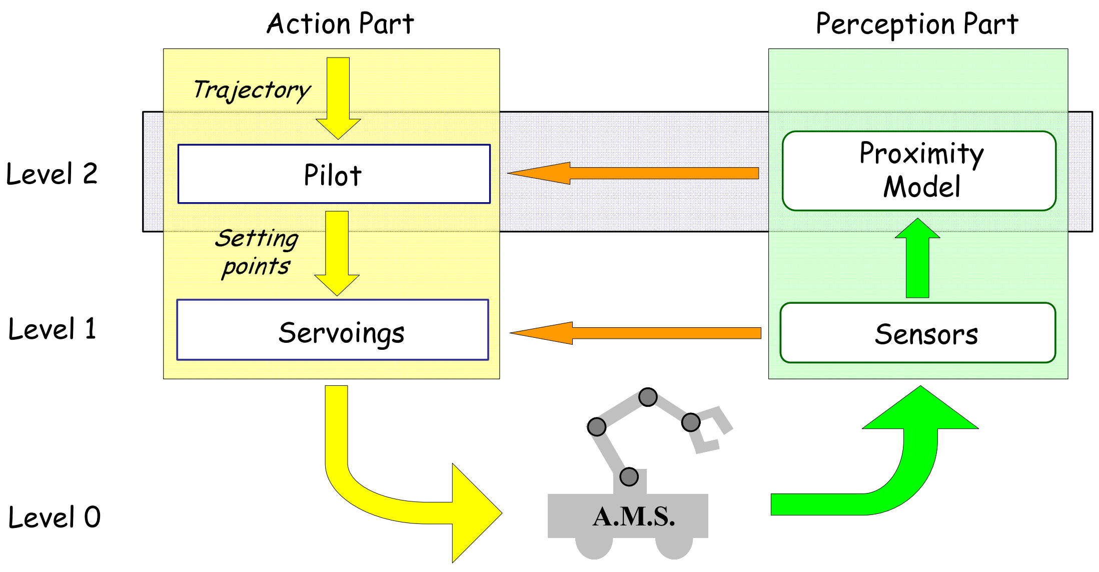

Figure 12 – The second level provide the second control loop

Depending on the robot application, the 'Pilot' cluster uses data resulting from the cluster 'Proximity Model' to carry out – or not – corrections on the reference trajectory provided to it as input. Typically, for a mobile robot, that consists in reflex avoidances of obstacles detected on the trajectory. For a manipulator robot, this consists in a loop of servoings in Cartesian space.

Because of its path in an additional layer, this loop of level 2 is *a fortiori* longer – and thus slower – than the loop of the level 1. Moreover, this loop does not exist on all robotic applications. There are manipulator robotics applications which use IKM and DKM modules for the 2 clusters of level 2, without any connection between them: there is no Cartesian servoings and the processing of Cartesian position/velocity is carried out in an open loop. The level 1 - the articular servoings loop - carries out the motion alone.

In a dual way, there are robot applications where the control is carried out only in Cartesian space; that is represented in our architecture by the absence of loop on level 1. The control loop is performed only after the DKM/IKM model in the level 2. Mixed modes of articular and Cartesian servoings can also be represented in this architecture.

Level 3: the navigator

The 'Pilot' cluster must receive its trajectory from the upper level of the action part. We call 'Navigator' the cluster positioned on this level 3. It must generate the trajectories for the 'Pilot' cluster based on data received from the upper level. These input data are of a geometrical type, still in a Cartesian space, but not necessarily in the robot frame. Moreover, these data do not integer dynamics or kinematics aspects; contrary to the trajectory, there is not a strict definition of the velocity, the acceleration or the force during the time for the AMS. These input data are called path – continuous or discontinuous – in Cartesian space. We allow to set on this path temporal constraints such as indicative time of route or indicative minimal/maximal speed on specific point of the path.

The 'Navigator' cluster must translate a path into a trajectory. The path does not take into account physical constraints of the AMS, but the trajectory that it delivers must integer them. Indeed, the path received by the navigator is closer to the task to be performed by the robot than to the mechanical constraints of the AMS. The navigator is situated at the interface between the "request" and the "executable": it is the most noticeable level of the proposed architecture. According to the robot applications, the modules gathered in this cluster are based on various theoretical methods.

On the top of the mechanical constraints of the AMS, this 'Navigator' cluster must generate a trajectory in concordance with the robot environment. Therefore, it needs information modelling its environment and coming from the same level, i.e. the cluster of level 3 of perception part. This perception cluster models the local environment of the robot beyond the range of its sensors in order for the 'Navigator' to test the validity of the trajectories before to deliver them to the 'Pilot'. This cluster is called the "Local Model" of the environment. It uses the displacement of the robot to locally model the environment around the robot (Figure 13).

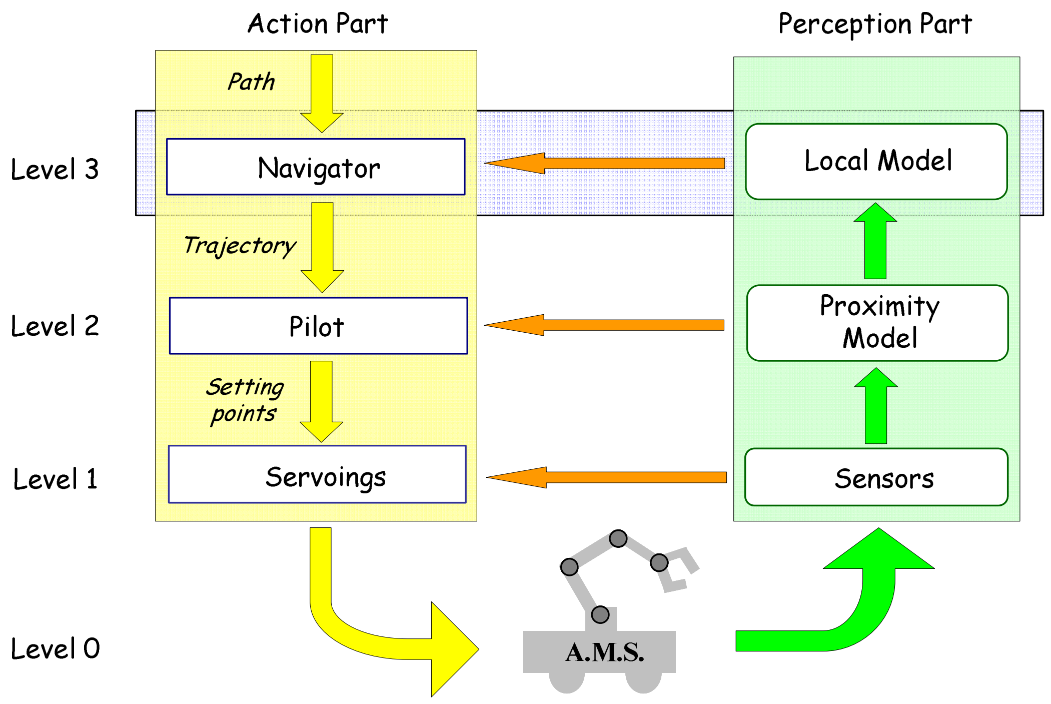

Figure 13 – The third control loop in the level 3

This level 3 makes it possible to integrate various modules based on various theories in the 'Navigator' cluster as well as in the 'Local model' cluster. Depending on the robot application, the control loop at this level may exist or not. When it exists, it crosses 6 clusters and it is longer – and slower – than the control loops of lower levels.

Level 4: the path-planner

The 'Navigator' receives as input a path resulting from the cluster of the level 4 of the action part, the 'Path Planner'. This cluster generates the path using as input a goal, a task or a mission. This functionality is performed in a long run in order to project the path on a large temporal horizon. We are located in a high control loop of the architecture which corresponds to the "deliberative" levels of similar architectures. To be valid, this path must imperatively be placed in a known environment. The path-planner use information resulting from the perception part of the same level: this cluster named 'Global Model' of the environment must provide to the path-planner an *a priori* map. Hence, the path-planner can validate its path on this pre-established map. This map does not need to be accurate with the metric direction (absence of obstacle, errors of dimension or orientation, presence of icons out of the scale...) but must be correct topologically (closed area, connexity, possible way-out, inaccessible areas...). This model can be either built on-line, by using data resulting from the lower level ("Local Model" cluster), or pre-established and updated by the lower level (Figure 14).

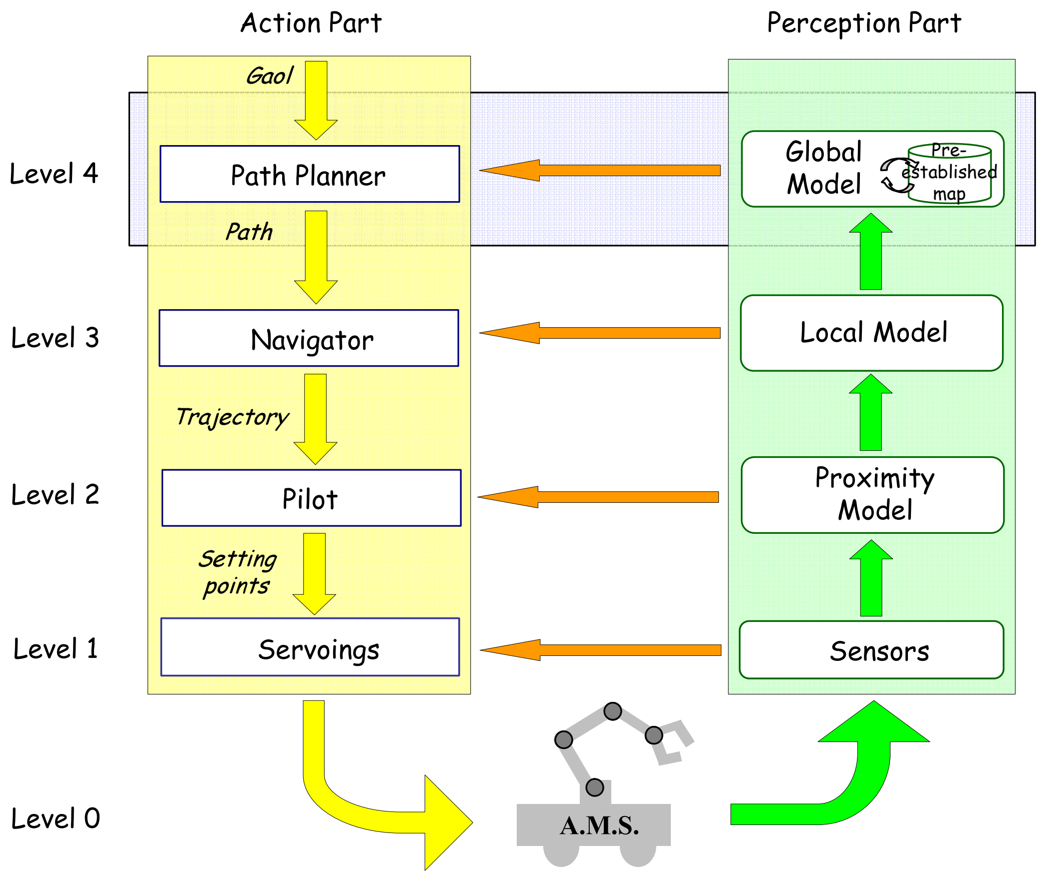

Figure 14 – The level 4

Level 5: the mission generator

The 'Mission Generator' is the cluster of the highest level of the action part. It must generate a succession of goals, tasks or missions for the 'Path-planner', according to the general mission of the robot. It is the "ultimate" robot autonomy concept: the robot generates itself its own attitudes and its own actions by using its own decisions. The 'Mission Generator' cluster does not really have any input. It only needs general information on its environment, its state and its possible attitudes. This is provided by the cluster of the perception part of the same level. This cluster is a 'General Model' of the robot and its environment and could be based on a data base (Figure 15).

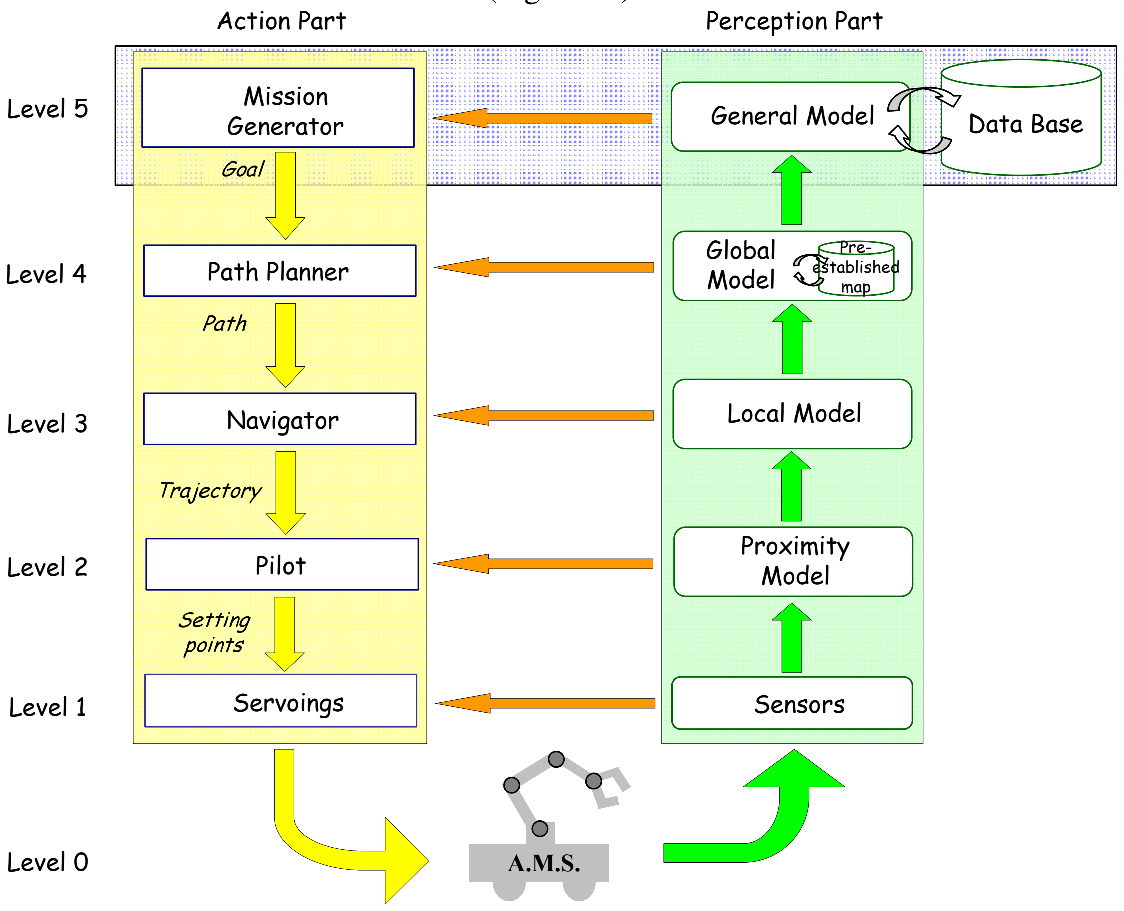

Figure 15 – The level 5

This level is the higher control loop of the proposed architecture and does not have any entry, as it corresponds to the highest decisional autonomy level. According to the robot application, this level is based on modules related to artificial intelligence. Although it must project its missions on a very large temporal horizon, reaction time of this level is slow (this loop has multiple levels to cross). However, this level of autonomy corresponds to the "smart" or "clever" attitude of the robot. The remaining part of the robot autonomy is dispatched on the lower levels of the architecture: the projection of the motion according to the obstacles is in the navigator, the reflex action is in the pilot level, the servoings level ensures the accuracy of the gesture...

The autonomous part of the architecture

The whole architecture, made up of 2 parts divided into 5 levels, represents the autonomy of the robot. In fact, these autonomous parts of the architecture are embedded in the robot: the two parts – 'Action' and 'Perception' – and the Articulated Mechanical Structure (level 0) constitute the robot itself. The 'Perception' part corresponds to the upstream data flow from the AMS and the 'Action' part corresponds to the downstream data flow to the AMS. Connections on each level ensure the feedback loops of variable length allowing the control of the robot.

This architecture makes it possible to model any autonomous robot whatever its application or its degree of autonomy. Its functioning is ensured by modules that are located in the different clusters. Depending on the applications, all modules are not required for all clusters and data flows may vary. Reversely, to produce a specific robot for a dedicated application, this architecture can be used in order to specify each module in each cluster, before carrying out the programming and the implementation.

The teleoperated part

This architecture makes it possible to model and control a robot which has various degrees of autonomy. But it does not make it possible to take into account the remote control of a robot. Indeed, the tele-operation is considered as antagonist to autonomy. A robotized system is regarded either as autonomous or as tele-operated. In the proposed architecture, we have leveled the autonomy of a robot in several degrees of autonomy. In a similar way, we propose to level the tele-operation of a robot. This leveled tele-operation complements the levels of autonomy of a robot, substituting itself to the missing degrees of autonomy.

Hence, we define a third part, called ‘Tele-operation’, distant to the robot (Figure 16). In order to modulate the degree of the tele-operation, this part is also organised in levels similarly to the one defined for the ‘Action’ and ‘Perception’ parts. Each level of the ‘Tele-operation’ part receives data resulting from the corresponding levels of the ‘Perception’ part and can replace the corresponding level of the ‘Action’ part by generating in its place data necessary to the lower level of the ‘Action’ part.

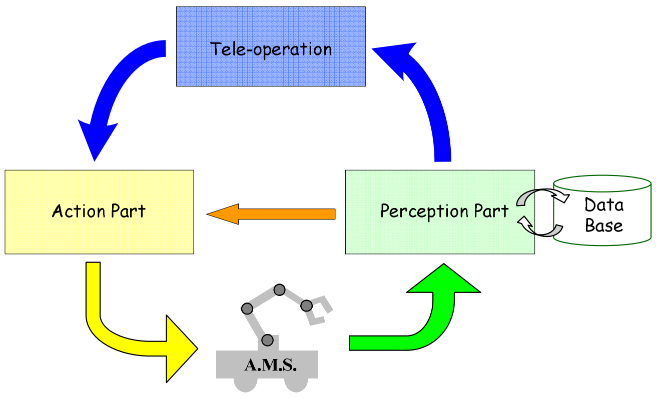

Figure 16 – The third ‘distant’ part: the tele-operation

Thus, several possible levels of tele-operation can be identified:

- The level 1 of tele-operation makes it possible for a human operator to actuate directly (in open loop mode) the Articulated Mechanical Structure. This level receives data from the level 1 of the ‘Perception’ part. This allows any operator to replace the autonomous loop of level 1. This is a remote control in open loop where the robot does not have any autonomy (Figure 17).

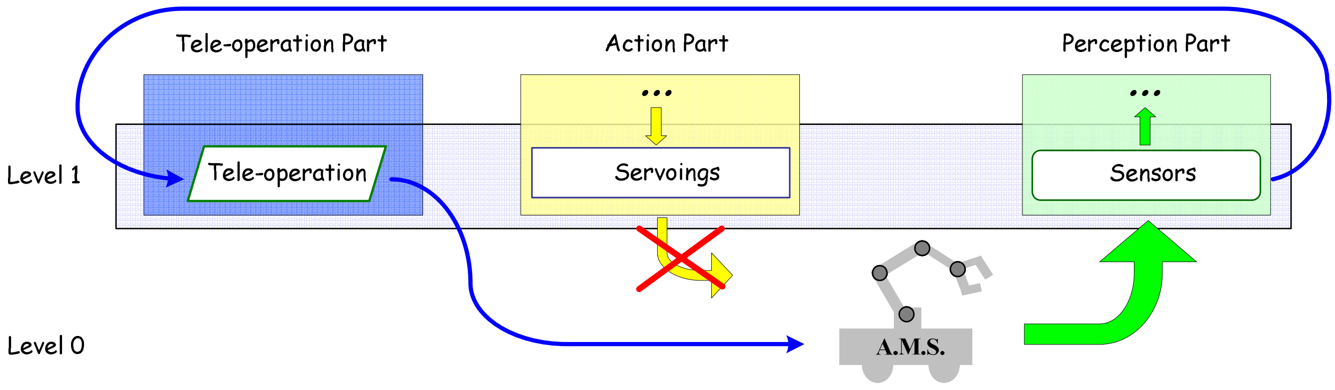

Figure 17 – The tele-operation supplying the level 1 of the action part

- The level 2 of the tele-operation uses information of the ‘Proximity Model’ to make it possible for a human operator to replace the level 2 of the ‘Action’ part, called the ‘Pilot’. It thus delivers setting points necessary to the level 1 to control the robot. This level of tele-operation is higher than previous, and can preserve the level 1 autonomous loop of the robot (Figure 18). Notice that there is no flow of information between level 1 and level 2 of the tele-operation; the reason being that they are excluded mutually: when there is a tele-operation of the robot, it is made either from level 1 or from the level 2.

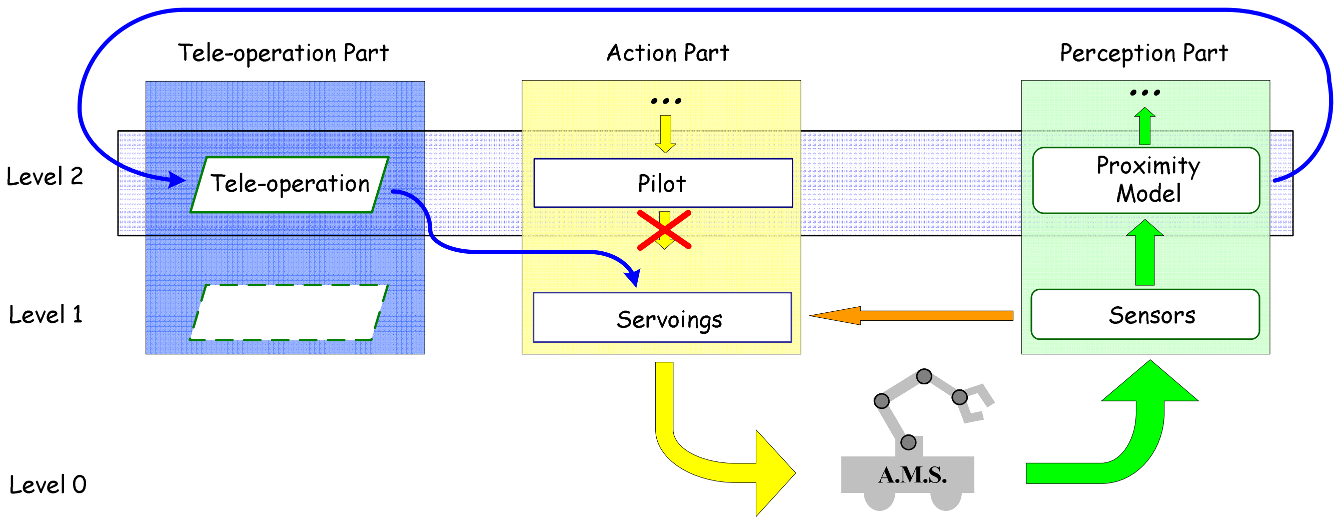

Figure 18 – The tele-operation supplying the level 2 of the action part

- The level 3 of the tele-operation allows an operator to generate a trajectory for the 'Pilot', hence taking the role of the 'Navigator'. The human operator who carries out this task uses data resulting from the 'Local Model' of the 'Perception' part. Thus, he acts in place of the loop of level 3 of autonomy. The operator tele-operates the robot still using the lower degrees of autonomy (2 and 1). For example, when a reflex reaction pilot is implemented in the level 2 of the 'Action' part, this one will act autonomously if the human operator sends a non-acceptable trajectory by the robot (Figure 19).

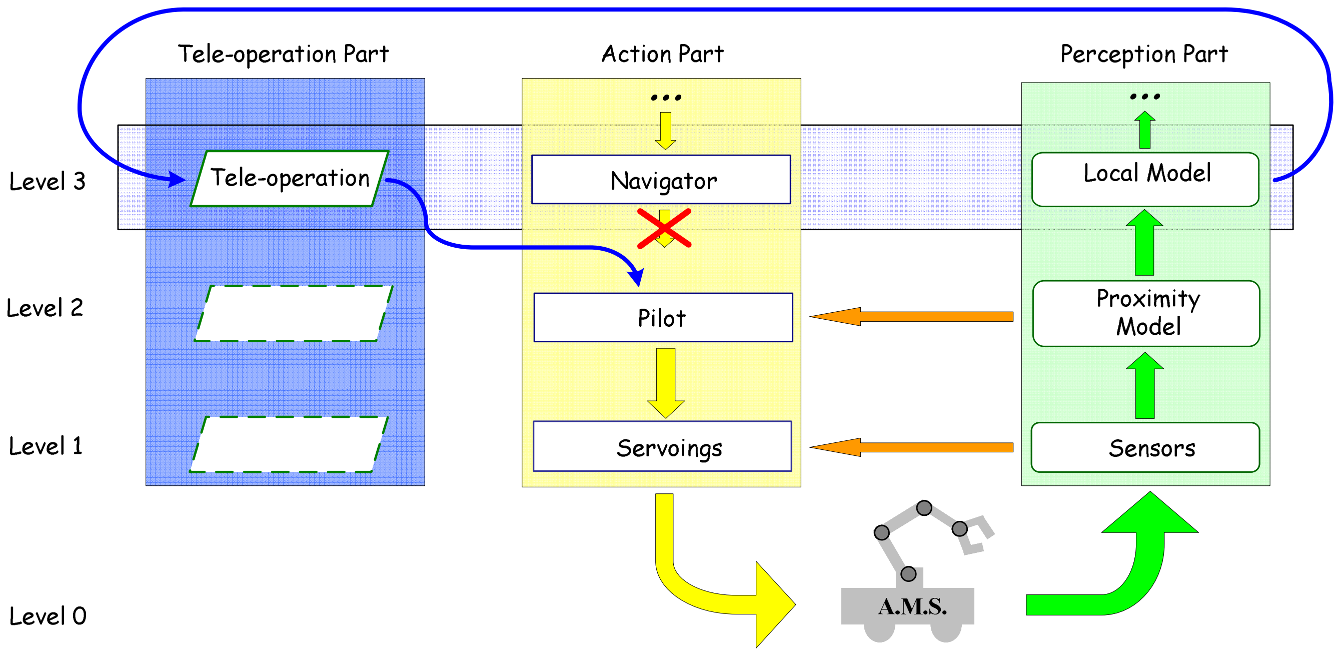

Figure 19 - The tele-operation supplying the level 3 of the action part

- The level 4 of the tele-operation makes it possible for the human operator to send a path to the 'Navigator' using the 'Global Model' of the 'Perception' part. He thus replaces the autonomous loop of the level 4 and is assisted in the control of the robot by the lower degrees of autonomy (Figure 20).

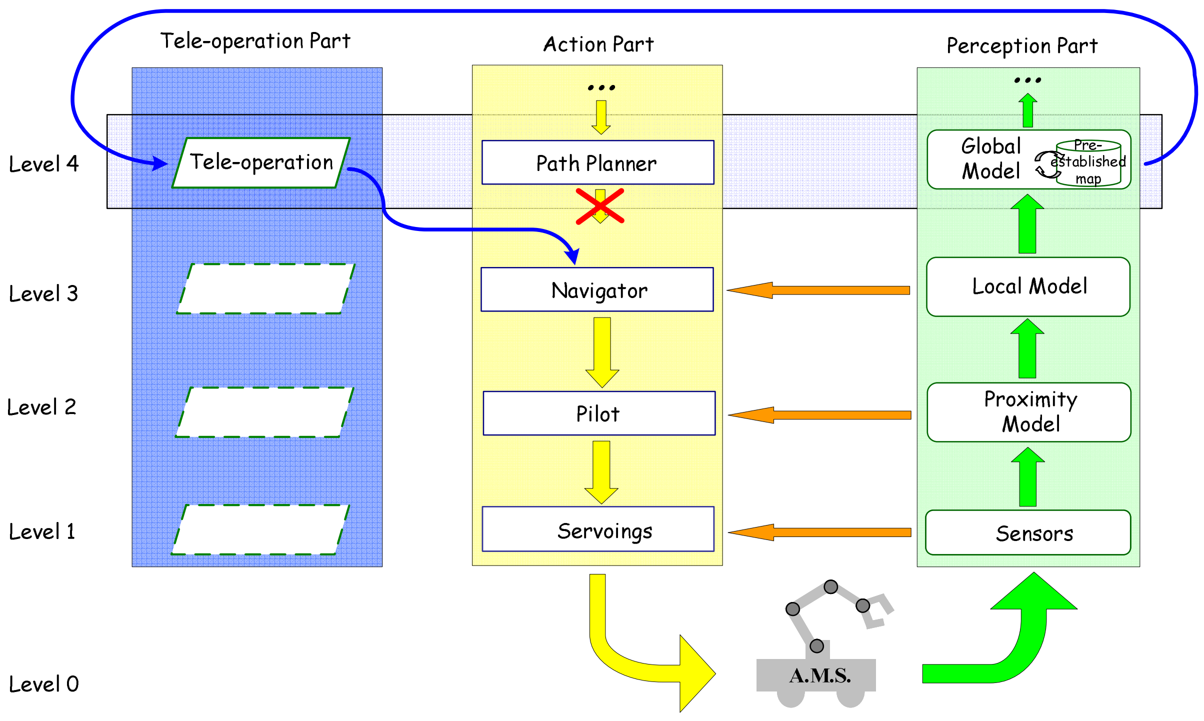

Figure 20 - The tele-operation supplying the level 4 and assisted by the lower levels of autonomy

- Finally, the level 5 of the tele-operation gives the possibility to an human operator to choose a task or a mission carried out by the robot autonomously, thanks to its levels of autonomy 4 to 1 (Figure 21).

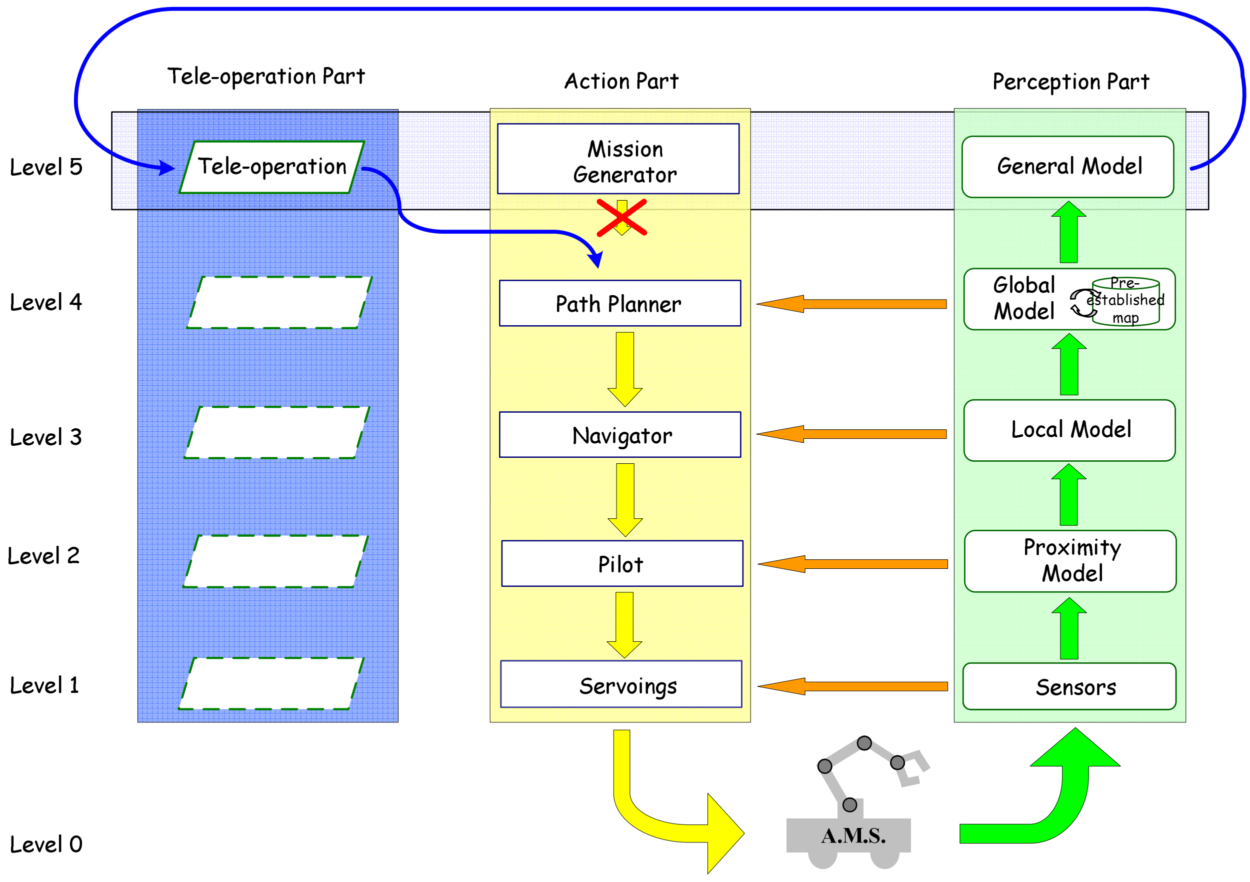

Figure 21 - The tele-operation supplying the level 5 of action part

Depending on the robot applications, only one level (or no level) of tele-operation is used by a human operator. However, the human operator can also choose his level of tele-operation according to the course of the robot mission. This dynamic evolution of the tele-operation gives the possibility to keep the human control on all the levels of an autonomous robot. To allow a dynamic evolution between the degrees of autonomy and tele-operation, a multiplexer module in each cluster of the ‘Action’ part is needed. This multiplexer module drives the input data either from the tele-operation or from the data coming from the upper autonomous level.

This dynamic evolution of autonomy/tele-operation degree also gives an operational security for the autonomous robot: currently, a human operator can take over the robot when it cannot solve problems on its own or because of material or software failures.

Of course, according to the tele-operation level where the human operator acts, the man-machine interface is different. It can be simple a keyboard/joystick for the lower levels, a graphic environment for the median levels or a textual analyser for the higher level. Finally, nothing prohibits the tele-operation to be ensured by computers instead of a human operator.

The complete PAT Architecture

The PAT architecture that we are proposing is based on 3 parts, the “Perception”, “Action” and “Tele-operation”, layered in 5 levels. As it has been shown, this PAT architecture forms a frame (Figure 22). A robot designer places on the PAT frame the modules that he needs to carry out the mission. He also determines the flow of data between the modules.

Robotics is also a technology of integration and the designer has to use various hardware and software modules. For example, he must integer a GPS sensor or a DC-power variator on which he can access only to the inputs and the outputs according to the builder specifications. Heterogeneous devices can be placed in the PAT architecture. The designer organizes them in priority and after, he places the rest of the modules (generally software). Each module is then developed afterwards according to the software (OS, languages, RT kernel…) and the hardware (CPUs, networks…) choices. The choice of the theory for each module is totally free: different theories/methods can be used and combined in the architecture (e.g. SLAModels, reactive behaviours based on neural networks, and a navigator based on heuristics can be used).

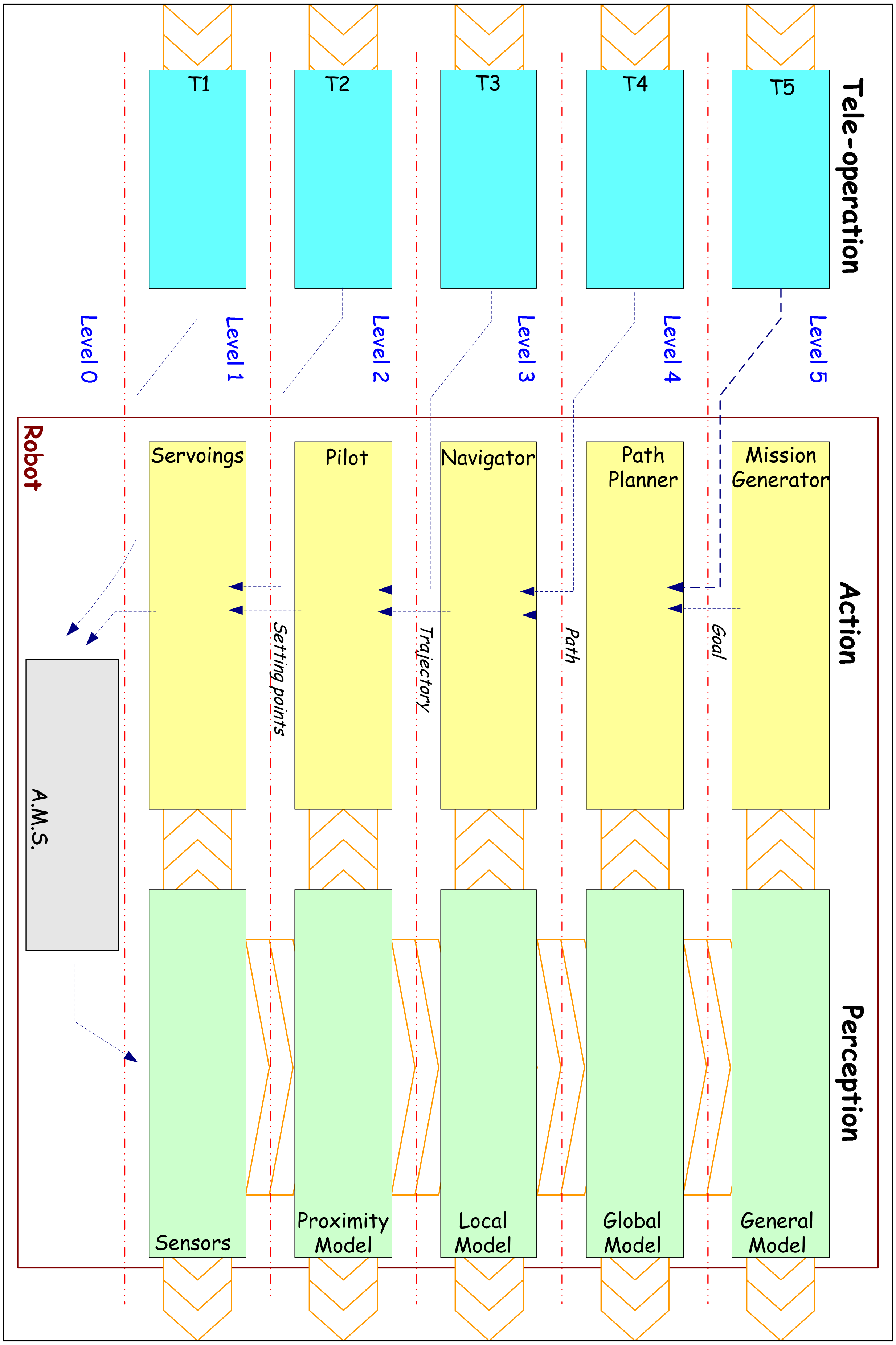

Figure 22 - PAT Architecture (frame)

Example of a mobile robot: “Raoul”

Several robots have been developed in our lab (LVR-Bourges); each of them had different hardware and software platforms. The PAT architecture was used to design, develop and control these totally different robots.

Raoul is an autonomous mobile robot, able to run in an unknown environment, finding alone a trajectory to perform and avoiding collision with static or mobile obstacles. Raoul is built on a Robuter platform (Robosoft) with an additional computer (PC/RT-Linux) and exteroceptive sensors: two Sick telemeters laser range and one goniometer laser.

The PAT architecture was implemented on Raoul with the following modules (Figure 23):

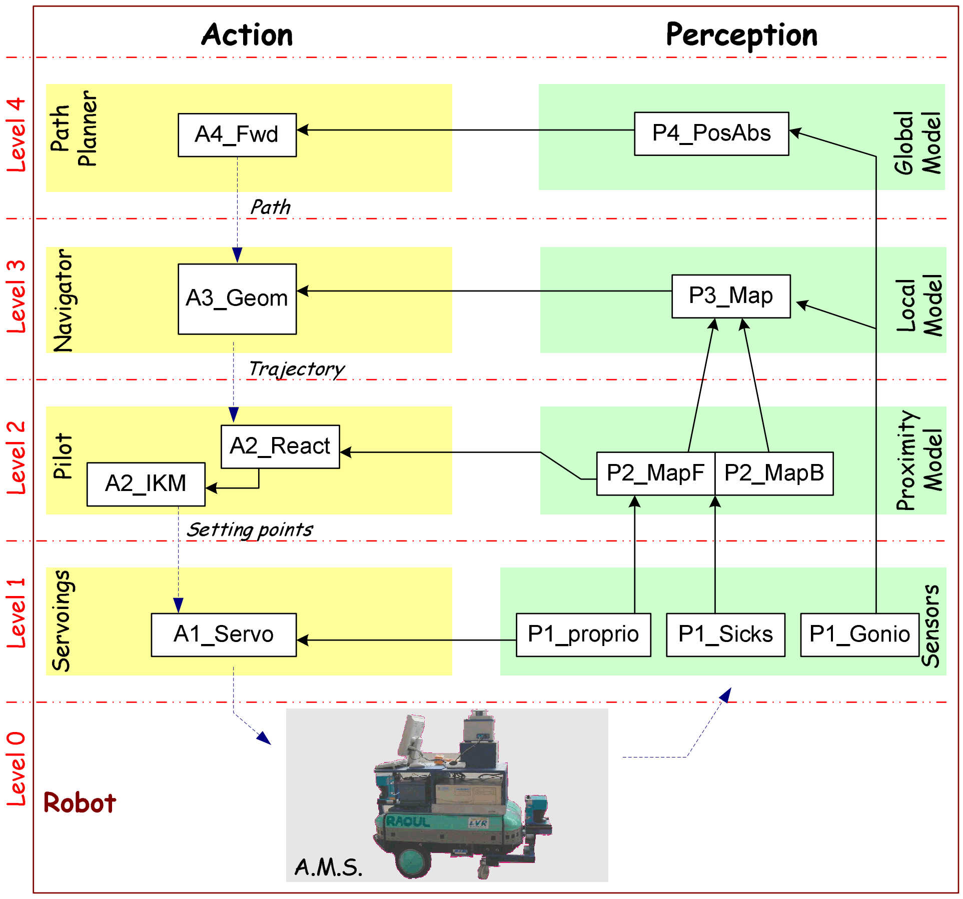

Figure 23 – Architecture of the mobile robot Raoul

‘A1_Servo’ is the module for the servoing of the 2 motorized wheels. Robuter platform is a standard differential wheels robot. This module uses the measures incoming from the proprioceptive sensors (motor ticks, odometry) ‘P1_proprio’. We can notice that these two modules, which forms the articular servoings loop, are implemented in the Albatros Robuter CPU. We cannot modify them; we have only access to inputs (setting points) and outputs (odometry, linear speed…).

‘P1_Sicks’ is the module driving the front and the rear laser range telemeters. Data that it delivers for the upper level is a list of polar distances depending to the angle of measurements.

‘P1_Gonio’ is the module driving the absolute laser goniometer. It delivers an absolute position/orientation in the plane.

The ‘Pilot’ cluster contains two modules. ‘A2_IKM’ module converts the desired motion of the robot in setting points for the servoings. ‘A2_React’ module performs the desired trajectory avoiding unforeseen obstacles using the two local maps made by ‘P2_MapF’ and ‘P2_MapB’ modules in the

'Proximity Model' cluster. 'A2_React' is based on DVZ formalism [9] and provides the robot with a reflex behavior to avoid obstacles.

The 'Local model' cluster contains the module 'P3_Map' which uses the method developed by J. Canou to built a global map on-line using the motion of the mobile robot to feed and refresh it [20]. The 'A3_Geom' module of the 'Navigator' cluster is a very simple module: it only places pre-processed trajectories of the robot in the local map. It deletes trajectories which intersect obstacle and choose the nearest trajectory from the desired path.

The 'A4_Fwd' uses the absolute position of the mobile robot to generate a straight path of 1 meter in front of the robot. It does not take into account any obstacle of the environment. The 'P4_PosAbs' module is only a calculation of the absolute position of the robot. A global map has not been integrated in this cluster. With this level 4, our goal is to validate the following concept: with only level 2 and level 3 control loops, a mobile robot is quite autonomous to be able to evolve in a totally unknown environment.

With this implementation, Raoul mobile robot is able to run in an unknown environment, avoiding static and mobile obstacles. It is able to make maneuvers (backward-forward) to escape from dead end zones.

Example of the tele-operated robot: "OTELO-1"

Otelo is a European project (IST-2001-32516) leaded by the LVR laboratory on the tele-echography using a light robot. During this project, 3 dedicated robots have been developed. Otelo1 robot is a fully tele-operated robot holding an echography probe. It can follow the motions than a medical expert realizes at distance with the input device. The expert receives ultrasound image, modifies the probe holder orientation and makes a diagnosis at a distance [20].

Otelo1 is programmed with the following architecture (Figure 24):

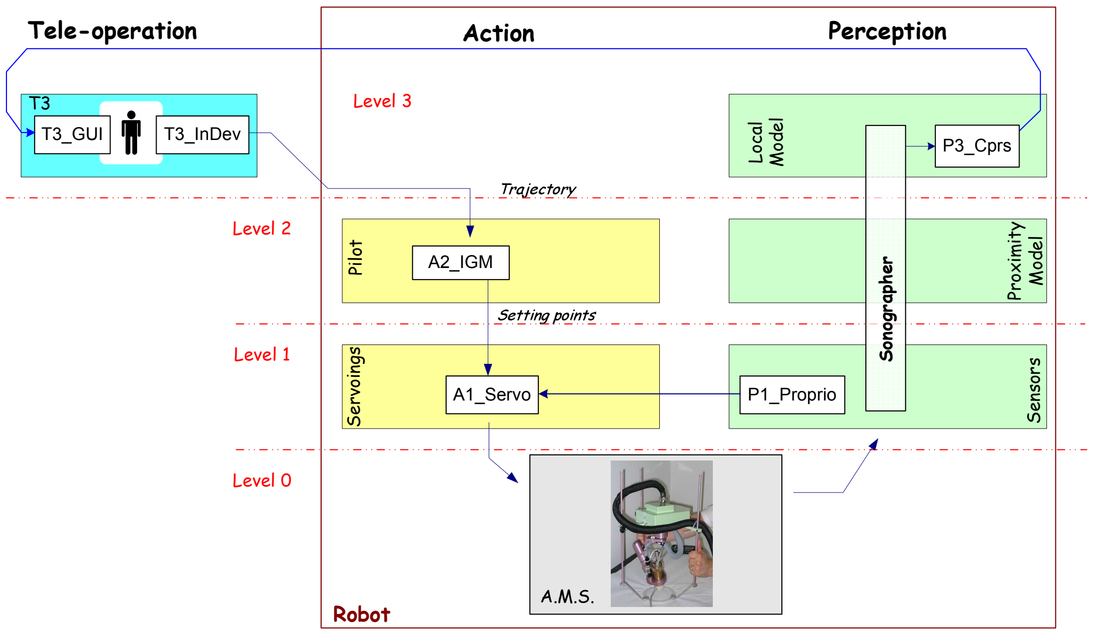

Figure 24 – Architecture of the tele-operated robot Otelo1

'A1_Servo' module performs the position servoings of the 6 axes of Otelo1 robot. This dedicated robot is technologically complex: it has 3 DC-motors, 3 step-motors, incremental encoders, analog absolute encoders, lvdt, force sensor and various digital I/Os like optical switches. 'P1_Proprio' module dives all the inputs. In fact, these two modules have its software distributed in many visual-C++ functions/objects, using functionalities of the advanced PMAC boards. The level 1 ensures the articular servoings of the 6 axes of the robot.

An ultrasound probe of a sonographer is held by the robot. However, companies manufacturing ultrasound devices with advanced functionalities restrict the access to the transducer signal; they only deliver ultrasound dynamic images. The ultrasound device is thus considered as an isolated device covering 3 levels of the 'Perception' part. The 2D ultrasound image is considered as a local model of the environment and it compressed by the 'P3_Cprs' module.

The compressed ultrasound image is transmitted to the 'T3_GUI' module of the level 3 of the 'Tele-operation' part. This module prints the dynamic image (on a screen) for the medical expert. He uses a virtual probe as an input device, the 'T3_InDev' module which transmits trajectories to the 'Pilot' cluster. The 'Pilot' cluster contains the 'A2_IGM' module which translates trajectories in articular setting points for the level 1.

Otelo1 was tested during trans-continental experiments (Cyprus-Spain or Morocco-France) using various communication links (ISDN, Satellite, Internet), and has proved the validity of the tele-echography concept in the medical environment.

Another robot Otelo3 based on a different mechanical structure is currently being developed in the lab and we are implementing some autonomous abilities to perform motions when problems occur on the transmission links (lacks or delays). To achieve that goal, we close the loop of the level 3 is closed and modules are added in the clusters of level 3 and 4.

Conclusion

The main objective of the PAT architecture was to clearly specify the domain of application of each theory or method developed in robotics. This architecture allows to use together several different methods.

We used the common concept of level and push it to its extremity to exit of the scheme reactive/deliberative. We have designed the PAT architecture in order to accept a maximum of robotic methods either in control or in perception. The main consequence is that the PAT architecture is sufficiently generic to model any robot, mobile or not, autonomous or tele-operated. The second strong asset of the PAT architecture is to give the possibility to manage the tele-operation AND the autonomy of a robot. This mixed mode can be graduated and dynamically modified.

All the robot developments of the laboratory use this PAT architecture, even if the mechanical structure, the hardware and the software are different. We are thus building a library of modules for the various levels of the architecture which can be used in any other robotic applications. When we fix a common hardware and software frame (CPU, languages, OS, RT-kernel, network links…), this library becomes available to an open community where each user can develop his/her own modules or use existing modules.

Bibliography

[1] Brooks R.A., "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, vol. 2, n°. 1, pp. 14–23, 1986.

[2] Albus J. S., "4D/RCS A Reference Model Architecture for Intelligent Unmanned Ground Vehicules", Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, April 1 – 5, 2002

[3] Volpe R., et al., "CLARAty: Coupled Layer Architecture for Robotic Autonomy." *JPL Technical Report D-19975*, Dec 2000.

[4] Alami R., Chatila R., Fleury S., Ghallab M., and Ingrand F., "An architecture for autonomy", *The International Journal of Robotics Research*, Special Issue on Integrated Architectures for Robot Control and Programming, vol. 17, no 4, pp. 315-337, 1998.

[5] Rosenblatt J., "DAMN: A Distributed Architecture for Mobile Navigation", In proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, AAAI Press, Menlo Park, CA, 1995.

[6] Arkin R.C. "Behavior-based Robotics", MIT Press, 1998

[7] Dalgalarrondo A., "Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome", Thèse de doctorat, Université Paris-Sud, Orsay, 2001.

[8] Hong-Ryeol K., Dae-Won K., Hong-Seong P., Hong-Seok K.and Hogil L. "Robot control software framework in open distributer process architecture", Korean and World patent, WO2005KR01391 20050512, IPC1-7: G06F19/00, May 2005.

[9] R. Zapata, "Quelques aspects topologiques de la planification de mouvements et des actions reflexes en robotique mobile", Thèse d'état, Université Montpellier II, juillet 1991.

[10] Joseph Canou, Gilles Mourioux, Cyril Novales and Gérard Poisson, "A local map building process for a reactive navigation of a mobile robot", ICRA'2004, IEEE International Conference on Robotics and Automation, April-Mai 2004, New Orleans.

[11] Gilles Mourioux, Cyril Novales and Gérard Poisson, "A hierarchical architecture to control autonomous robots in an unknown environment", ICIT'2004, IEEE International Conference on Industrial Technology, December 2004, Hammamet.

[12] European OTELO project : "mObile Tele-Echography using an ultra Light rObot", IST n°2001-32516, leaded by the LVR (Sept. 2001 – Sept 2004), consortium : LVR (F), ITI-CERTH (G), Kingston Universite (UK), CHU of Tours (F), CSC of Barcelona (E), Ebit (I), Sinters (F), Elsacom (I) and Kell (I).