# Collaborative rules operating manipulators

**Chapter** · April 2010

DOI: 10.5772/9667 · Source: InTech



**3 authors:**

Jose Martins

**15** PUBLICATIONS **72** CITATIONS

SEE PROFILE

Luiz Camolesi Junior

State University of Campinas (UNICAMP)

**36** PUBLICATIONS **52** CITATIONS

SEE PROFILE

Glauco Caurin

University of São Paulo

**126** PUBLICATIONS **828** CITATIONS

SEE PROFILE

7

# Collaborative rules operating manipulators

José Martins Junior[1], Luiz Camolesi Jr[2] and
Glauco Augusto de Paula Caurin[3]
[1] *Piracicaba School of Engineering*
[2]*University of Campinas,* [3]*University of São Paulo*
*Brazil*

## 1. Introduction

Collaboration among robots and human beings has inspired researchers and novelists since a long time ago. Apropos, the word "robot" first appeared in a theatre play ("R.U.R.", Karel Capek, 1921) referring to an automata character, a slave humanoid. Important advances for control strategies were presented by researchers, applied to service robots, toys and automata vehicles, concerning the interaction with human beings.

Over time, manipulator robots were massively used on industrial plants, performing predefined and repetitive tasks. Modern applications for manipulators, involving two or more robots on cooperative tasks, are now arising in industry. Most of the scientific publications on this area present solutions for some aspects involving humans, mainly related to the safety in robots' workspaces, and the flexibility to fast operate and reconfigure them. However, the way to operate manipulators remains rigidly based on imperative programming, through a HRI (Human-Robot Interface). On the other hand, a new approach proposed by (Brooks, 1986), based on behaviours, allows the definition of reactive models of control applied to mobile robots. The main limitation of this approach is its strictly reactive behaviour, i.e. all knowledge the robot will learn about the environment is unpredictable.

Current trends in several research areas are pointing to a possible occurrence of a new singularity, when the mankind will experiment the knowledge disembodiment, i.e. the human knowledge will be retrieved from brain, including its consciousness, and transferred to another place, or machine (Vinge, 2008). Psychologists (Pinker, 1999) defend that the mental states, as well as deliberations and emotions, can be represented by means of symbols of a mental language, known as "Mentalese". The free representation of signals and symbols for all mental states and their causalities is practically impossible, considering the current state of the art in technology. However, if conceived to specific domains, this can be fetched. Rules and policies for collaborative environments consist of well formed sentences, which describe states, causal relationships and their effects, applied to collaboration among human beings. These rules and policies have been used for several situations, involving computer supported cooperative work (CSCW).

This chapter presents and discusses the application of symbolic rules to coordinate collaborative environments with manipulators and humans. It also demonstrates how to express a set of collaborative rules, with common effects for machines and humans. We

know that the elephants don't play chess (Brooks, 1990), but at the end of this chapter was presented our "robotic elephant" which plays a Tic Tac Toe game.

## 2. Robot control strategies

The traditional approach for robots control is feedback based and hierarchically subordinated to a trajectory generator that is responsible for mapping tasks into sets of movement sequences (rotations and translations). These sequences are obtained by combining the individual motions defined by links or mobile parts of a robot. The generated information is presented to the control modules, which are responsible for motors actuation.
A different strategy was presented by (Brooks, 1986). It describes a multilayer architecture with several levels of abstraction, allowing reactive behaviours (Nwana, 1996) for mobile robots. Each layer consists of a distinct level of competence, which may be activated. Thus, a layer can modify the resultant output, by including its component on the signal, and also inhibiting the signal produced by the lower layers.

### 2.1 Manipulators and the traditional approach

The large majority of industrial robot manipulators, available in the market today, use the control architecture originally proposed by Engelberger/Devol and their UNIMATE robot more than 30 years ago. The movements are decomposed by the user during the programming task into a sequence of primitive movements, i.e. point to point, straight line, circle. Normally the robot controller programming interfaces are implemented in an interpretative form. All the movements are related to a "tool center point - TCP". Time intervals or velocity to be reached during the execution of each primitive movement is also user defined using the programming interface.
A strong point of this strategy is the capability to generate complex behaviour and movements independently on which kind of end-effector the manipulator is carrying on. A weakness of this approach remains in the ability of such equipment to interact with a non-static environment, as for example, in assembly tasks. Each primitive of an individual movement is further transformed into coordinates for the joints using inverse kinematic calculation. The coordinates and its derivatives are finally transmitted to the robot controllers, for each single joint, as a function of time.

### 2.2 Cooperative robots, humanoids and the behavioural-based approach

Cooperative applications research for multiple robots sprung in the last decade (Parker, 2003). (Cao et al, 1997) state that cooperative behaviour can be observed on more complex animals (vertebrates, for instance), including human beings. Such behaviour has social motivation, demanding each isolated participant to feel the need or desire to cooperate. A system with multiple robots can present cooperative behaviour if, when performing some task with any cooperative mechanism, the increase on the efficiency of the whole system emerge.
Several architectures were proposed to solve the distributed control problem for mobile multi-robots, but this subject is out of the main focus of this chapter. A good description about this research area can be found in (Parker, 2003). On the other hand, there are few

published works (Lau & Ng, 2006) that discuss solutions for control problems for robotic manipulators using distributed strategies.

#### Reactive behaviour models

Behaviour may be defined (Bishop & Potter 2004) as an observable and repeated pattern in the relationships among spatio-temporal events associated with an agent and its environment. Behaviour based robots use the information they gather from the environment through the sensors to react to specific situations. The internal representations of environment are extremely limited when not completely inexistent. Isolated models of simple behaviours are responsible to respond to specific sensor signal conditions. The overall robot behaviour results from the output combination of each model. It is exactly the synthesis of coherent reaction, i.e. emerging intelligence as the result of the fusion of each behaviour model constitutes an open challenging task.

Reactive models are very important for strategies involving robots learning, especially mobiles ones, because they allow to assume the world as its own best model (Brooks, 1990). This feature is primordial in situations involving adaptation for robots' behaviours acting in unknown environments, like other planets.

#### Deliberative behaviour models

In deliberative control, the robot takes all of the available sensory information, and compares this information with its internally stored knowledge. Therefore in this approach a complete representation of the environment is stimulated using all available internal robot computer resources. To accomplish its task, the robot must further plan its future actions. This requires the robot program to look ahead. As a consequence, the control structure must provide multi task real time capabilities allowing the robot to act strategically.

If we pretend to see robots replicating behaviours based on the knowledge previously obtained by a human (in opposite to a non-deterministic learning by observing the environment), this knowledge must be formally expressed and converted into deliberative actions, according to this interpretation. Currently, two different approaches may apply: the neuronal model of the brain and the symbolic model of the mind.

The first (connectionist) aims to reproduce in computers the basic functions of the brain inspired by its topology. The main limitations for this approach refer to the enormous quantity of neurons and synaptic connections, and also the plasticity of neural networks created by the brain.

The symbolic model is presented and discussed at session 3.1 of this chapter.

### 2.3 Distributed and modular strategies for control architectures

Modular Robotics offers an answer for various complex tasks. Instead of designing a new and different mechanical robot for each task, simple module reconfiguration when connect in a suitable form may accomplish complicated things and meet the demands of different tasks or different working environments.

Each module is improved with individual capabilities for processing, sensing, communicating and actuating. The overall functionality of a modular system is only achieved when several modules are connected as a unique robot, i.e. a single module presents low utility.

Similarly, a manipulator robot can be decomposed and its parts individually analyzed. These parts present individual capabilities, like modules. Thus, the robot can be classified as an n-modular system, where n is the number of different types of modules.

### 2.4 Supporting collaborative behaviours

A multilayer control model, adapted from (Brooks, 1986), was proposed in (Martins Jr et al, 2008). This new approach includes cooperative and collaborative behaviours, and was designed to operate on distributed systems, defining different contexts – local and global – as shown in Figure 1.

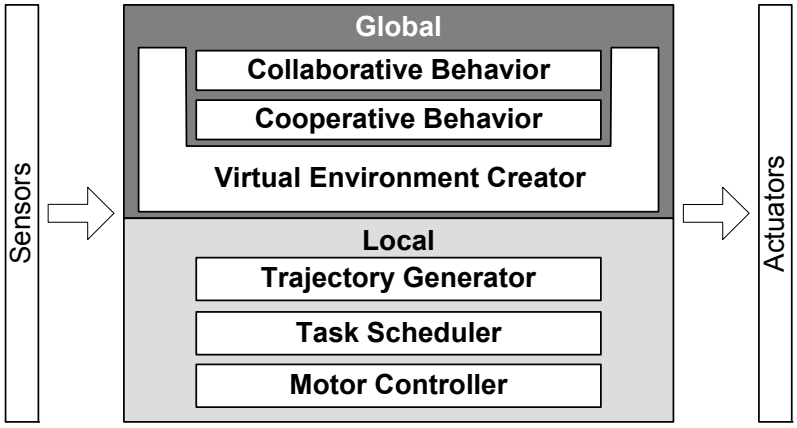


Fig. 1. Multilayer control model

Distinct parameters of criticality and strictness for agents operating on each of the two contexts (local and global) can be individually treated into its respective layer and provide means to define its coupling degree to the target (robot). Figure 2 shows the appropriated allocation of the layers on a distributed environment.

The local functions describe processes that are highly rigorous for execution time, but demand a small amount of resources (storage and processing power). They can be classified as local agents, tightly coupled to the target.

On the other hand, in the global context, the processes are less rigorous with respect to performance time, but need larger amount of resources. These features indicate that the designed agents must not be embedded into the target, but executed on loosely coupled remote computers. Local agents interact directly with the target using communication boards connected to actuators and sensors.

Each distinct part (link) of the robot, including its sensors, motors and mechanisms can be individually considered as different modules. Thus, the movements' composition for each module can be resolved on a higher level, as a cooperative task. This is one of the advantages provided by the architecture.

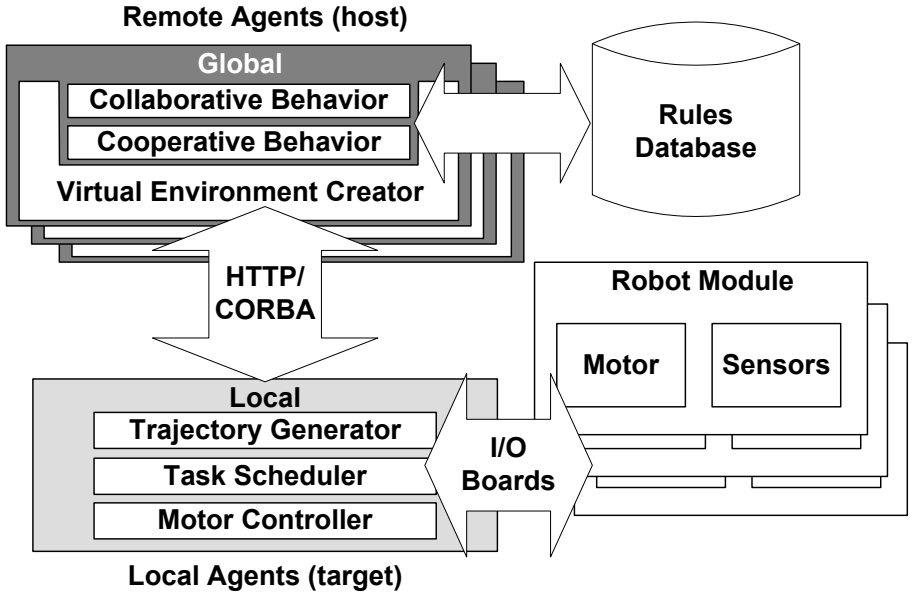


Fig. 2. Distributed architecture

Remote agents can be assigned for each module and interact with local agents to provide appropriate behaviour for the global model of the environment. This interaction can be done through a local network, and supported by data communication protocols (such as CORBA, or HTTP). The cooperative behaviour can be achieved by joint deliberation involving remote agents, based on cooperation rules and on environment model. Cooperation is frequently associated to a specific task execution.

The collaborative behaviour is placed on top of the architecture. Collaboration can be observed when a robot interacts with a set of tasks contributing to achieve a common goal to other agents in the environment, including humans. In the same manner, this behaviour is deliberated from decisions taken by agents, by analyzing collaborative rules and checking the current state of the environment.

The guiding rules for cooperation and collaboration are stored in a rules database, and can be accessed by global agents, at its respective actuation level.

A brief description of each architecture layer is presented on the following subsections.

**Motor controller**

Motor controller describes the bottommost layer, highly dependent and coupled to the hardware. Software agents developed to this level are locally executed (embedded) on the target. Real time requirements are highly rigorous, requiring performance time up to 1 millisecond.

The main function of this layer is to emit signals to the drivers that directly actuate on each of the motors. Data from sensors (encoders) attached to the motors are returned to the control mesh as feedback, to ensure correct performance of the actuators.

**Task scheduler**
Task scheduler plays an important role in the local context of the architecture, because it provides means to coordinate the processes that operate during each of the individual movement of the robot links. The execution of individual movements is part of the strategy defined by an upper layer (trajectory generator), allowing general repositioning of the robot in the environment.

**Trajectory generator**
Trajectory generator is the topmost layer in local context. It also imposes rigorous restrictions concerning real time, yet it allows larger deadlines, of around 10 milliseconds. Its main function is to define individual movements for each motor, by decomposing a desired trajectory between two points and sending it to the robot (inverse kinematics).

**Virtual environment creator**
The bottommost layer of global context is to implement the general model of the environment where the robot will be placed. At this level, geometric aspects are considered, allowing robot's workspace analysis to be performed. Visual and proximity sensors can provide data to be compared with the current state of the model, internally represented by a software agent. Global strategies to replace the robot in the environment can be defined at this level. These functions demand more computing power and, as compensation, allow flexibility to response times. The main advantage of an environment model which keeps close fidelity to the real world is the easiness to perform robot simulation on a virtual environment. In this manner, all the development and testing stages for high-level functionality, involving cooperative and collaborative behaviours, can be previously done on a simulated environment.

**Cooperative behaviour**
Cooperative behaviour describes relations among robots (or parts of them) and must be implemented by global agents, based on environment model and predefined rules (rules database). It is possible to notice behavioural capacities, both reactive and deliberative, of these agents. The reactive capacity is provided by environment analysis, which represents its internal model, differently from the deliberative, which results from decisions about cooperation rules.

**Collaborative behaviour**
In the same manner, collaborative behaviour for human interaction can be also provided. Previously defined collaborative rules are stored in a database and allow analysis by agents, by comparing to the current state of the environment model.

**Rules database**
The rules database is an important artifact of the whole architecture, and it was designed to store all cooperative and collaborative rules. These rules were defined using the M-Forum model (Camolesi Jr & Martins, 2006), and describe interaction policies by means of collaborative situations, involving different agents. M-Forum is presented in the next sessions.

## 3. Languages and rules

Languages define written or spoken symbols that are used for communication purposes. Written symbols can be jointly combined into words that, and depending on the context, provide the meaning of transmitted ideas. The sentences composition in a language is previously defined by a grammar. Sentences are constituted by a finite sequence of symbols from some finite alphabet (Slonneger & Kurtz, 1995).
The syntax of a programming language describes how the symbols may be combined to create well-formed sentences (or programs). The meaning, obtained by interpreting the words of a sentence, defines its semantics (originally conceived in "Mentalese", the language of thought). Thus, intentions about a desired behaviour can be described by means of rules, using an interpretable language.
Restricted and non-ambiguous formal grammars were proposed to express programming languages for computers. BNF (Backus-Naur form) is a widely adopted formal model to specify grammars that describe terminal symbols of a valid alphabet, non-terminal symbols and production rules. The main benefit of using a BNF style language is the easy to implement programs that work as its lexical interpreters.

### 3.1 Symbolism and the mind-brain dilemma

Symbolism can be described as a movement that defends symbolic representation of mental activities, inspired by computer like way of operation. In this sense, a constructive approach is defined using a top-down strategy (Minsky, 1990): begin at the level of commonsense psychology and try to imagine processes that could simulate it. The central idea consists, assuming a greater challenge, to search for a solution by decomposing it into simpler parts. This refers to a reductionist method, a typical approach commonly applied in AI (Artificial Intelligence), and known as heuristic programming.
Two relevant aspects can be highlighted from the concept of the mind presented in (Pinker, 1999), which constitutes the foundation of computational theory of the mind. The first states that the mental computations are applied to information, and this can be expressed using an internal symbolic representation. The other refers to the functional composition of the mental modules, which perform the computations. Thus, no matter what kind of subject (physical) where mental computation is performed, the functionality of the modules that compose the mind and the symbols are submitted to it. As a consequence, beliefs and desires can be seen as information, physically embodied as configurations and symbols.
In the last decades, with the advances in AI research, a new approach for philosophy of the mind – not dualist either materialist – has emerged, the functionalism. Functionalism introduces the concept of causal role, in which a mental state can be described by their causal relations with other mental states. Functionalism is based on the distinction established by computer science about hardware (physical components) and software (programs). From this point of view of psychology, a system can describe a human being or a machine, and its basic constitution (neurons or electronics) is not what really matters, but how parts are organized (Fodor, 1981). Thus, functionalism does not rule out the possibility of a mechanical or electronic system having mental states and processes.
The central subject of this chapter is related to collaboration among robots (machines) and human beings. In this sense, we have adopted a top-down model, where the behavioural rules, with common sense for both types of actors, have been stated using a formal language

(constituted by symbols). Then, the rules were implemented and their functionality observed during the coordination of a collaborative activity (more specifically, a game).

### 3.2 Behavioural rules, policies and collaborative environments

Interaction Policies are norms for the interactions in an environment; those can be established by logic grouping of rules with well defined goals or objectives. In the definition of a Collaboration and Control Policy (CCP) model for human-robot interaction, a policy must observe the relationship among robotic and human agents in a same environment, regarding collaborative task performance.

The research (Camolesi Jr & Martins, 2005) has achieved excellent progress for structure and ontology definition. However it still has a lot to advance on applications, such as robots control. Towards the approach of these questions, the M-Forum model supports collaborative interactions modelling through the definition of rules by providing support to five dimensions: actor; activity; object; time and space. A comparison between M-Forum and the other models for rules (Tonti et al, 2003) is presented in Table 1.

| | **Kaos** | **Rei** | **Ponder** | **M-Forum** |
|---|---|---|---|---|
| **Ontology based** | Yes | Yes | No | Yes |
| **Specification language** | DAML/OWL | Prolog based | Ponder Language | L-Forum |
| **Tool for specification policy** | KPAT – Graphical Editor | No (GUI under development) | Graphical Editor | No (GUI under development) |
| **Reasoning support** | Java Theorem Prover | Prolog engine; Event-condition-action model. | Event calculus | Activity theorem; Deontic theorem; Event-condition-action model. |
| **Enforcement mechanisms** | Policy automation being explored for the next version | Action execution is outside the Rei engine | Java interfaces for enforcement agents | Rule execution is outside the engine |
| **Flexibility** | Ontology can be extended with domain dependent descriptions of local entities | Ontology can be extended with domain dependent descriptions of local entities | Management domain as a structuring technique for partitioning complex object | Ontology can be extended with domain dependent descriptions of local entities |
| **Elements represented** | Actors, actions, groups, places | Subject, activity, object | Subject, activity, object, domain | Actor, activity, object, time, space, association, domain, composition and generalization abstractions |

Table 1. Comparison between M-Forum and other models for rules

### 3.3 The M-Forum model

In M-Forum (Camolesi Jr & Martins, 2006), the Actor dimension allows the representation of an agent in a collaborative environment through activity rights, prohibitions and

obligations. The actors of a collaborative environment can be classified in human or not-human. Every human actor has an identifier (*Ach_id*), a current state (*AchState*) and a set of attributes (*Ach_AttS*). Given *qh* as the number of human actors at the environment and *qs*, the number of not-humans, the formal statements are:

$$\begin{aligned}
&AchS = \{Ach_1, Ach_2, ..., Ach_{qh}\}\ ,\ AcsS = \{Acs_1, Acs_2, ..., Acs_{qs}\}\\
&AchS \neq \varnothing \vee AcsS \neq \varnothing\\
&AchS \cap AcsS = \varnothing\\
&AcSS = AchS \cup AcsS\\
&Ach_i = (Ach_id_i\ ,\ AchState_i\ ,\ Ach_AttS_i\ )\\
&Acs_i = (Acs_id_i\ ,\ AcsState_i\ ,\ Acs_AttS_i\ )
\end{aligned}$$

Actors are responsible for the execution of individual or collective activities, thus being able to reach objects, an actor or actors group.

Activity is an execution unit that can be carried through by an actor or group. Normally, activities involve the manipulation or transformation of an object. Activities must be defined using Activity Operators and logic Operators representing rights, prohibitions and obligations. Activity Operators are required to specify the interaction between actors and objects. Activities have identifiers (*At_id*), a state (*AtState*), an activities subset (*At_S*), a set of operations (*OpS*) and a set of attributes (*At_AttributeS*). Given *qa* as the number of activities in an environment:

$$\begin{aligned}
&AtSS = \{At_1\ ,\ At_2\ ,\ ...,\ At_{qa}\}\\
&At_i = (At_id_i\ ,\ AtState_i,\ AtS_i\ ,\ OpS_i\ ,\ At_AttributeS_i\ )\\
&AtS_i \subseteq AtSS\\
&OpS_i \subseteq OpS
\end{aligned}$$

Object is any element that can be used in actions on objects or actors. An object represents the structural characteristics and the behaviour of reality. Activities can be carried through in objects to modify its characteristics. An object modelling in such a way establishes uniformity of vision and treatment, useful for collaborative environment projects. An object may be composed by others objects (*CompObS*) and has an identifier (*Ob_id*), a state (*ObState*) and a set of attributes (*Ob_AttS*). Activities and Operations may be performed on Objects that allows its state or attributes changing.

$$\begin{aligned}
&ObSS = \{Ob_1\ ,\ Ob_2\ ,\ ...,\ Ob_{qo}\} \vee ObSS \neq \varnothing\\
&Ob_i = (Ob_id_i\ ,\ CompObS_i\ ,\ ObState_i\ ,\ Ob_AttS_i\ )\\
&CompObS_i \subseteq ObSS
\end{aligned}$$

Spaces are localization areas of actors or objects and the specific areas used for activities. Like other elements presented in this section, the spaces are essential for modelling a collaborative environment.

On the collaborative interaction, elements of the dimension space must be defined using the Space Operator (*SpOp*) to specify the position or the size of actors and objects in collaborative environments. The space element has an identifier (*Sp_id*), a state (*SpState*) and a set of attributes (*SpAttS*). If *qe* is the number of spaces into an environment:

$Sp_i = (Sp\_id_i , SpState_i, SpAttS_i)$
$SpSS = \{Sp_1 , Sp_2 , ..., Sp_{qe}\}$
$SpOp \in So = \{ <,<=,>,>=,=,<>,==$(attribution), not equal,
inside, outside, intersect, meet, overlap, north, south, east, west $\}$

In time modelling, Duration, Date and Occurrence have basic semantic for temporal references establishment. These semantics are used to define a logical action with duration, occurrence date or occurrence interval of activities defined on interactions between actors and objects.
The formal basis for temporal elements describes the natural set of numbers (*N*), and representations for years (*Ty*), months (*Tm*), days (*Td*), hours (*Th*), minutes (*Tmi*) and seconds (*Ts*), for a Moment or Interval. Enumerated sets of relative values (*Tmr, Tdr, Thr, Tmir, Tsr*) are used to represent dates for a specific calendar. Given *qt* as the number of time moments or intervals occurring on an environment:

$Ty, Tm, Td, Th, Tmi, Ts \in N$
$Tmr \in \{1, 2, 3, 4, ..., 12\}$
$Tdr \in \{1, 2, 3, 4, 5, ..., 31\}$
$Thr \in \{0, 1, 2, 3, ..., 23\}$
$Tmir \in \{0, 1, 2, 3, ..., 59\}$
$Tsr \in \{0, 1, 2, 3, ..., 59\}$
$Time = \{Ti\_id ,(Ty , Tm , Td , Th , Tmi , Ts)\}$
$Datetime = \{Ti\_id ,(Ty , Tmr , Tdr , Thr , Tmir , Tsr)\}$
$Tb, Te \in Datetime , Interval = \{Ti\_id , (Tb, Te)\}$
$TiSS = Time \cup Datetime \cup Interval$
$TiSS = \{Ti_1 , Ti_2 , ..., Ti_{qt}\}$
$TiOp \in To = \{ <,<=,>,>=,=,<>,==$ (attribution),
precedes, succeeds, directly precedes, directly succeeds,
overlaps, is overlapped by, occurs during, contains, starts,
is stated with, finishes, is finished with, coincides with $\}$

The dimensional elements of a rule are defined in three contexts:

- **Applicability**: condition for the execution or activation of a rule and definition of the scenes (values of attributes or space aspects) in which it can be applied;
- **Execution**: a set of expressions that establishes the actions or conditions for the interactions between elements, being able to optionally involve transitory aspects of time and space;
- **Survivability**: it is an optional context specifying the other rules with the same applicability. Also the scenes can be defined (values of attributes or space aspects) to establish the instant at which the rule must be activated or deactivated.

### 3.4 The L-Forum syntax

L-Forum is a language developed to formalize the concepts specified by the M-Forum model. The language allows the definition of rules for an environment, increasing their precision and improving disambiguation for collaborative environment designers. Its

overall structure may be described by clauses, which are defined for three particular purposes:

- **Context:** this clause is composed by performing or activating parameters of a rule and comply with applicability conditions of the scenario (value of attributes, spatial or temporal aspects) for a rule adoption;
- **Definition (body):** it is composed by a set of expressions where actions or conditions are established for interacting elements and may involve transitory aspects of time and space. Rules and actions may be directly invocated at the body of a rule, which allows to compose the expressions;
- **Regime:** this is the scope of a rule, and refers to an optional set composed by interrelated rules having the same orientation to be performed or applied. Scenarios, involving a rule activation or deactivation, can be also described.

The main elements of L-Forum syntax are presented in Table 2.

### 3.5 Collaborative rules for human-robot interactions

At this point, we address to the problem of defining the collaborative rules among robots and humans. To illustrate it, a simple task was considered: a tool passing between the human and the robot.

For the robot to identify the different collaborative situations, involving the task, a visual code was established. If the human presents his(er) open hand over the common workspace (a table surface), the situation "tool passes from robot to the human" must be assumed. If the human presents his(er) hand holding the tool, the opposite situation must be considered.

Summarizing, the dimensional elements to elaborate collaborative rules, are:

- **Actors:** robot and human. The robot is an actor composed by different links. The human being is also an actor established by the composition of single parts, detaching the hand;
- **Objects:** the tool, which will be collaboratively shared by the actors;
- **Space:** there are three involved spaces in the problem: the common workspace (table surface) where will be shared to pass the tool; the individual spaces, where the human and the robot stay. Each individual space is exclusive. Only the common area must be shared collaboratively by both actors;
- **Activity:** ordering and delivering are activities that may be realized by both actors (human and robot). The activities will be recognized by both actors analyzing the state their parts, i.e., the human's hand and the robot's gripper (end-effector). A hand or a gripper on "open" state means the tool ordering; a hand or a gripper holding the tool is associated to a tool delivering. Grasping and releasing are also related activities on the working process.

When modelling the collaborative actions, the human is the actor with primary actuation, and so, he (or she) establishes the frequency and sequence for actions. In this sense, considering the dimensional elements of the collaborative work scenery, previously presented, some rules to compose the Collaboration and Control Policy (CCP) are shown in Table 3.

| | |
|---|---|
| <rule> ::= | ´Rule´ <rule name> ´[´<status>´]´ ´{´ <context> ´Body ::´<definition> [<regime>] ´}´ |
| <context> ::= | ´Parameters: (´<parameters>´)´ [ <applicability> ] |
| <definition> ::= | <condition> \| <action> \| <rule call> [<definition>] |
| <regime> ::= | <survivability> [´Priorities:´ <priority>] |
| <parameters> ::= | (´any´ \| ´all´ \| <identifier>)´:´<element> [´,´ <parameters>] |
| <element> ::= | <actor> \| <group> \| <object> \| <space> \| <time> \| <association> \| <activity> \| <operation> |
| <type> ::= | ´actor´ \| ´group´ \| ´object´ \| ´space´ \| ´time´ \| ´association´ \| ´activity´ \| ´operation´ |
| <applicability> ::= | ´Applicability::´ <condition expression> |
| <survivability> ::= | ´Survivability::´ <condition expression> |
| <condition> ::= | ´If ´ <condition expression> ´then {´ <definition> ´}´ [´else {´ <definition>´}´] |
| <action> ::= | ´Action: (´ <actor> <normative operator> {<activity> (<actor> \| <object>)} [<space attribution operation> <space>] [<time attribution operation> <time>] ) [´);´ <action>] ´);´ |
| <supreme action> ::= | <actor> <normative operator> <primitive operator> (<element>\|<domain>\|(´is part of´\|´is a´) <element>) \| <actor> <normative operator> <primitive group operator> <group> <element>) |
| <definition action> ::= | <actor> ´set´ <status> |
| <attribution action> ::= | <actor> ´attribute´ (<value>\|<formula>\| ( (next \| prior) (<value domain>\|<domain name>) ) <attribute> |
| <condition expression> ::= | ´(´ (<attribute><attribute condition operator>(<value>\| ([´all´\|´any´] (<value domain>\|<domain name>)) \| (<condition expression> (and \| or)) ´)´ |
| <rule call> ::= | ´Rule (´ <rule name> ´ (´<parameters>´)´ <normative operator> [´);´ <rule call>] ´);´ |
| <priority> ::= | <name> [´,´ <priority>] |
| <group> ::= | <name>´:Group´ |
| <actor> ::= | <name>´:Actor´ |
| <activity> ::= | <name> ´:Activity´ |
| <operation> ::= | <activity>´.´<name> ´:Operation´ |
| <object> ::= | <name>´:Object´ |
| <space> ::= | <name>´:Space´ |
| <time> ::= | <name>´:Time´ |
| <association> ::= | <element>´.´<name> [´.´<association>]´:Association´ |
| <attribute> ::= | <element>´.´<name> [':Attribute´] |
| <domain> ::= | <name> ( <value domain> \| <grouping> ) |
| <value domain> ::= | ´(´ (<numeric value> {´,´ <numeric value>}) \| (<string> {´,´ <string>}) ´)´ |
| <grouping> ::= | (<type> <name> <attribute condition operator> ( <value>\|([´all´\|´any´](<value domain>\|<domain name>))) {(and\|or)<grouping>} ) \| (<element> {´,´ <element>}) |
| <element status> ::= | <element> <status attribution operator> <value> |
| <status> ::= | [´active´] \| [´inactive´] |
| <primitive group operator> ::= | ´insert´ \| ´delete´ \| ´update´ |
| <primitive operator> ::= | ´create´ \| ´destroy´ |
| <group element operator> ::= | ´∈´ \| ´∉´ \| |
| <group group operator> ::= | ´⊆´ \| ´⊄´ \| ´⊂´ |
| <activity condition operator> ::= | [´not´] ´has´ |
| <normative operator> ::= | ´right´ \| ´prohibition´ \| ´obligation´ \| ´dispensation´ |
| <activity attribution operator> ::= | ´receive´ |
| <status attribution operator> ::= | ´put on´ \| ´move to´ |
| <space attribution operator> ::= | ´==´ \| ´inside´ \| ´outside´ \| ´north´ \| ´south´ \| ´east´ \| ´west´ |
| <time attribution operator> ::= | ´in´ \| ´on´ \| ´at´ |

Table 2. L-Forum syntax

| |
|---|
| Rule **Human Delivers Tool** [active] {<br><br>Parameters:: (hu: human, ro: robot, too: tool, ts: table Surface)<br>Applicability:: (hu.hand not is open) and (ro.gripper is open) and<br>(hu.hand not is on ts) and (ro.gripper not is on ts)<br><br>Body:: Action (hu obligate hand put on ts);<br>Action (hu obligate release too on ts);<br>Action (hu.hand obligate put of ts);<br>Rule **Robot Moves to the work** (ro , ts)<br>Action (ro obligate hold too);<br>Action (ro.gripper obligate put of ts); } |
| Rule **Human Orders Tool** [active] {<br><br>Parameters:: (hu: human, ro: robot, too: tool, ts: table Surface)<br>Applicability:: (hu.hand is open) and (ro.gripper not is open) and<br>(hu.hand not is on ts) and (ro.gripper not is on ts)<br><br>Body:: Action (hu.hand put on ts)<br>Action (hu.hand obligate put of ts)<br>Rule **Robot Moves to the work** (ro , ts)<br>Action (ro obligate release too on ts);<br>Action (ro.gripper obligate put of ts);<br>Action (hu.hand obligate put on ts)<br>Action (hu obligate hold too);<br>Action (hu.hand obligate put of ts) } |
| Rule **Robot Moves to the work** [active] {<br><br>Parameters:: (ro: robot, ts: table Surface)<br>Applicability:: (hu.hand not is on ts)<br><br>Body:: Action (ro.vector_a prohibition move on ts)<br>Action (ro.vector_b prohibition move on ts)<br>Rule **Moving Vector_c** (ro, ts)<br><br>Survivability:: Priorities: **Human Delivers Tool , Human Orders Tool** } |
| Rule **Moving Vector_C** [active] {<br><br>Parameters:: (hu:human; ro: robot, ts: table Surface)<br>Applicability:: (hu.hand not is on ts)<br><br>Body:: Action (ro.vector_c obligate move to ts) } |

Table 3. Rules for collaborative tool passing

## 4. Case study: Tic Tac Toe game

In this session we present an experimental case study, involving human and robot interaction faced as opponents in a board game, known as Tic Tac Toe. The game was chosen because its rules are very easy to learn, allowing it to be played by people with different levels of skill, from children to adult.

The decision was also influenced by another feature, that the game is played on a predefined board (field), facilitating the coordination of movements done by opponents (robot and human) into the common workspace.
This case study was proposed as a proof of concept for using collaborative rules to govern the interactions among different types of actors, a robot and a human. It was also important to demonstrate the need for a strategy definition when selecting rules in collaborative environments, in order to surpass the unpredictability of some human behaviour.

### 4.1 The game

The Tic Tac Toe is a two player game where the participants take turns drawing tokens (X or O) on a 3 x 3 grid. Winning the game involves a player placing three tokens in a row, column or diagonal. When the grid is completely full and no sequence of three equal tokens occur (row, column or diagonal), they got a draw.
Figure 3 shows a particular and possible situation during a Tic Tac Toe match. In this case, the player of X tokens won.

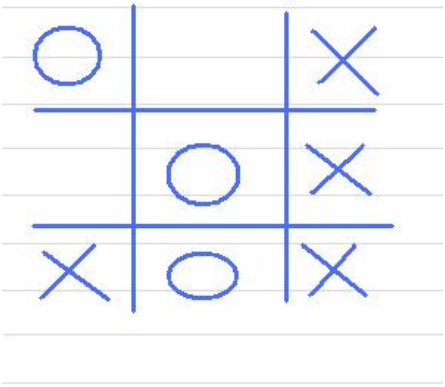

Fig. 3. A Tic Tac Toe situation

An expert performance for Tic Tac Toe game can be described as a set of few rules (Crowley & Siegler, 1993), as shown in Table 4. These rules are enough to describe every faced situation, during a Tic Tac Toe game, but often occurs that more than one rule can be applied, pointing to the need to define a criteria to select the most appropriated.

#### Adaptability for different levels of skill

Despite the rules simplicity, selecting them in order to make a move depends on several factors, like attention, knowledge and others. These factors are clearly influenced by the player's age, and thus, the system must be able to use appropriate strategies for different levels of skill presented by its opponent. Otherwise, a child will never be able to win, and could become bored.

| Move type | Conditions | Action |
|---|---|---|
| Win | If there is a row, column or diagonal with two of my pieces and a blank space | Play the blank space |
| Block | If there is a row, column or diagonal with two of my opponent's pieces and a blank space | Play the blank space |
| Fork | If there are two intersecting rows, columns or diagonals with one of my pieces and two blanks<br>AND<br>If the intersecting space is empty | Move to the intersecting space |
| Block fork (1) | If there are two intersecting rows, columns or diagonals with one of my opponent's pieces and two blanks<br>AND<br>If the intersecting space is empty<br>AND<br>If there is an empty location that creates a two-in-a-row for me | Move to the location |
| Block fork (2) | If there are two intersecting rows, columns or diagonals with one of my opponent's pieces and two blanks<br>AND<br>If the intersecting space is empty<br>AND<br>If there is NOT an empty location that creates a two-in-a-row for me | Move to the intersecting space |
| Play center | If the center is blank | Play the center |
| Play opposite corner | If my opponent is in a corner<br>AND<br>If the opposite corner is empty | Play the opposite corner |
| Play empty corner | If there is an empty corner | Move to an empty corner |
| Play empty side | If there is an empty side | Move to an empty side |

Table 4. Set of rules for expert performance on a Tic Tac Toe game

### 4.2 Defining the rules of the game

The next step consists on translating rules to L-Forum format. According to L-Forum syntax, described above, a rule may be stated using some elements of the language. A rule must have a unique name and declare its status. The parameters of a rule specify the involved elements, like actors, space and objects. The applicability refers to the conditions that cause a rule selectable. The body of a rule describes actions to be performed.

Two examples for rules mapping are presented in Table 5. The first refers to the "Play Center" rule for a Tic Tac Toe expert match, and may be applied when the center is empty and the game is not finished. The second implements a rule that may be selected in four situations, relating to each corner of the board.

```
Rule PlayCenter[active] {

Parameters::    (pl :player:actor; tboard :TicTacToeboard : space; tok :token:object, g
                :game:object)
Applicability:: (tboard.center is empty) and (g not is finished)

Body::          Action (pl right play tok inside tboard.center); }
```

```
Rule PlayCorner[active] {

Parameters::    (pl :player:actor; tboard :TicTacToeboard : space; tok :token:object)
Applicability:: (g not is finished)

Body::          if (tboard.corner_high_right is empty)
                  then { Action (pl right play tok inside tboard.corner_high_right );
                }
                else {
                  if (tboard.corner_high_left is empty)
                    then { Action (pl right play tok inside tboard.corner_high_left);
                  }
                  else {
                    if (tboard.corner_low_right is empty)
                      then { Action (pl right play tok inside tboard.corner_low_right);
                    }
                    else {
                      Action (pl right play tok inside tboard.corner_low_left);
                    }
                  }
                }
}
```

Table 5. "Play center" and "play corner" rules translated to L-Forum

### 4.3 The hardware infrastructure

An IBM 7545 SCARA robot retrofitted with open platform (Aroca et al, 2007), was used in this project. The target's hardware, assembled on a CompactPCI rack, contains the following components:

- **Boards**: Inova AMD K6, Acromag Carriers, National Instruments I/O;
- **Industry Packs (IP)**: Tews 48 Digital I/O, Tews IP Quadrature, Tews DAC.

Figure 4 presents the whole view of the architecture.

Since the main purpose of this project was not related to research accurate positioning and fine control, and aiming to simplify the robot operation, a strategy based on fixed points was adopted. All the nineteen positions, representing valid locations of the game, were predefined and marked into an 800x400 mm board, as shown in Figure 5. Ten of these positions (five at each side of the game field) were designed as pieces repositories.

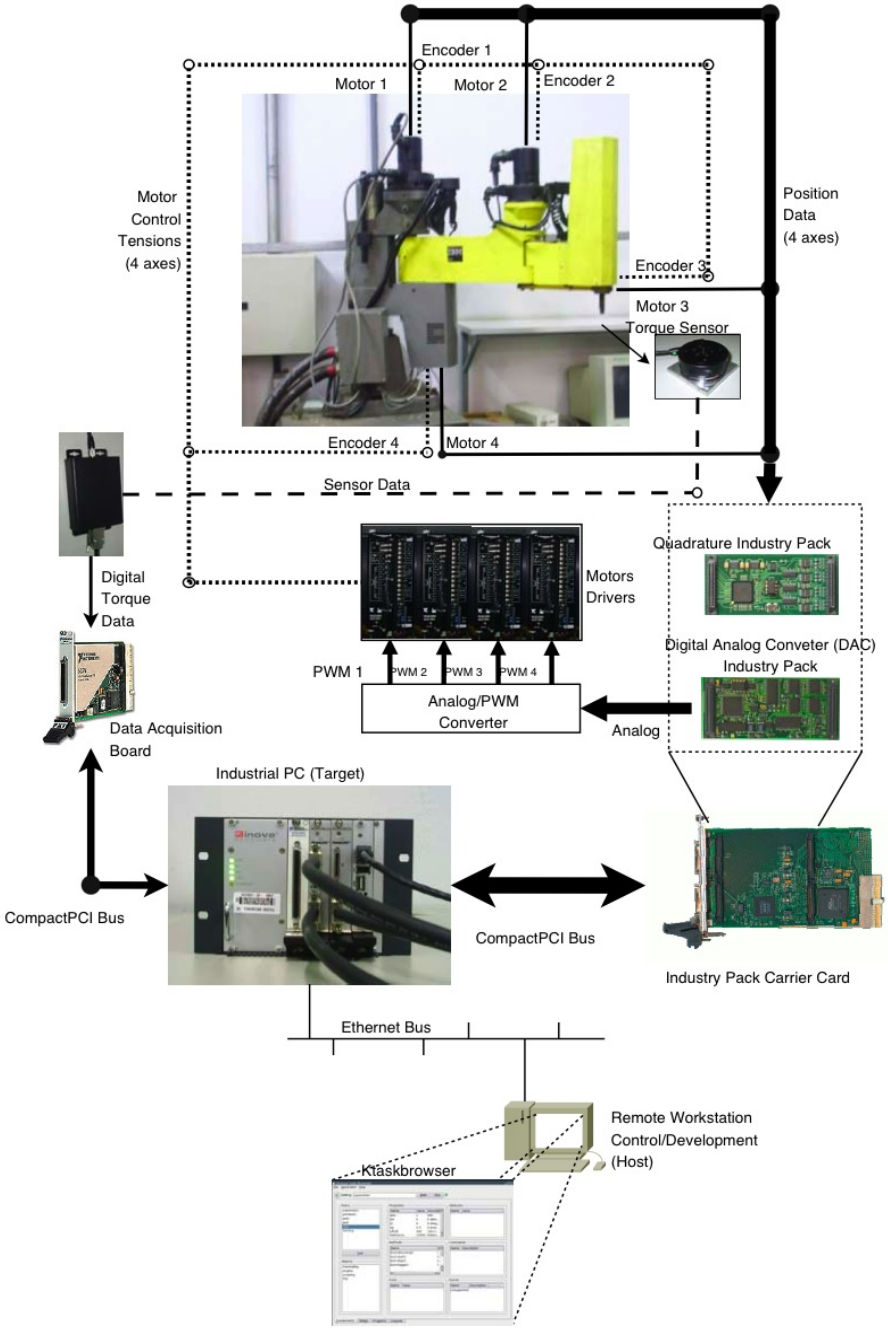


Fig. 4. The retrofitted IBM 7545 SCARA robot

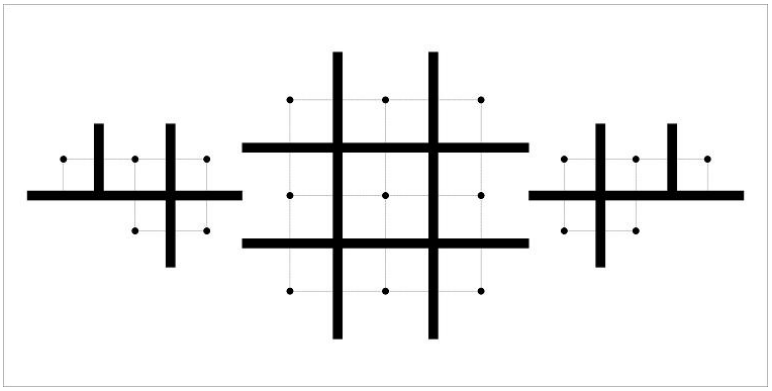

Fig. 5. Design of the Tic Tac Toe board

Nineteen reed switches (magnetic presence sensors) were fixed on the board, at each position for pieces locations (marked with small circles on the board). The reed switches are used to detect magnetic field generated by magnets, which in this case, were inserted in each piece of the Tic Tac Toe game, as shown in Figure 6.

Fig. 6. Magnetic pieces

All sensors were connected to a board, with 32 digital inputs. This board multiplexes all digital entries through a parallel interface, connected to a computer port.
Since the main focus of this project was to present coherent means to allow robot and human collaboration, another subject comes up. Several researches in Robotics have pointed to the relevance of robot's appearance when interacting with humans. Human beings usually feel more comfortable to interact when the other subject looks familiar.
Considering that this robot must interact with human beings, including children, we decide to give it a playful look. The SCARA robot was dressed in an elephant costume, making it very fun and with a less formal aspect.
Figure 7 shows the robot during a Tic Tac Toe match. The picture also presents the board and the pieces over it.

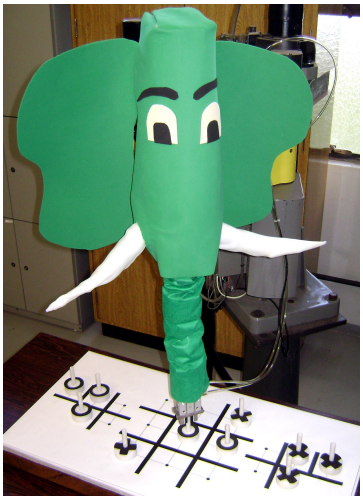

Fig. 7. Robot dressed as an elephant

### 4.4 Software design and implementation

The local software agents were designed to execute specific roles, implementing the bottommost levels of the distributed model, i.e., the trajectory planner, the task manager and the motor controller (Aroca et al, 2007).

The base system was developed using a real time interface to the Linux kernel – RTAI (RealTime Application Interface). Some of its features are the actuator for motors, a PID (Proportional Integral Derivative) controller, and sensor acquisition, through an industrial PC. The infrastructure also allows interacting with the robot, across the local network.

The overall solution was based on host-target model. A computer (host) was being used to develop and compile the software, before it was embedded in the industrial PC (target). Both computers run Linux operating system, but only the target's kernel was increased by the real time modules and the RTAI interface. Other facilities were also implemented, allowing more flexibility in robot reconfiguration and high level protocols for data communication (Tavares et al, 2007).

A three-dimensional HRI (Human-Robot Interface), called Scara3D, was presented in (Martins Jr et al, 2008). The interface was developed to perform tests for high-level layers integration into the architecture. The obtained results were satisfactory and proved the feasible implementation for the "Virtual Environment Creator" layer of the proposed model.

When designing the game we considered the interaction among a SCARA Robot and human beings, which can be classified as actors according M-Forum specifications. The SCARA Robot is composed by translational and rotational links and has a pneumatic gripper, while a Tic Tac Toe game contains a board and pieces (X and O); these individual parts are represented as objects in Forum model. These relationships are presented in a class diagram, as shown in Figure 8.

Current state of the real environment can be monitored by sensors integrated into the architecture, and so the virtual model can be updated, representing the interaction among several objects and actors within the game.

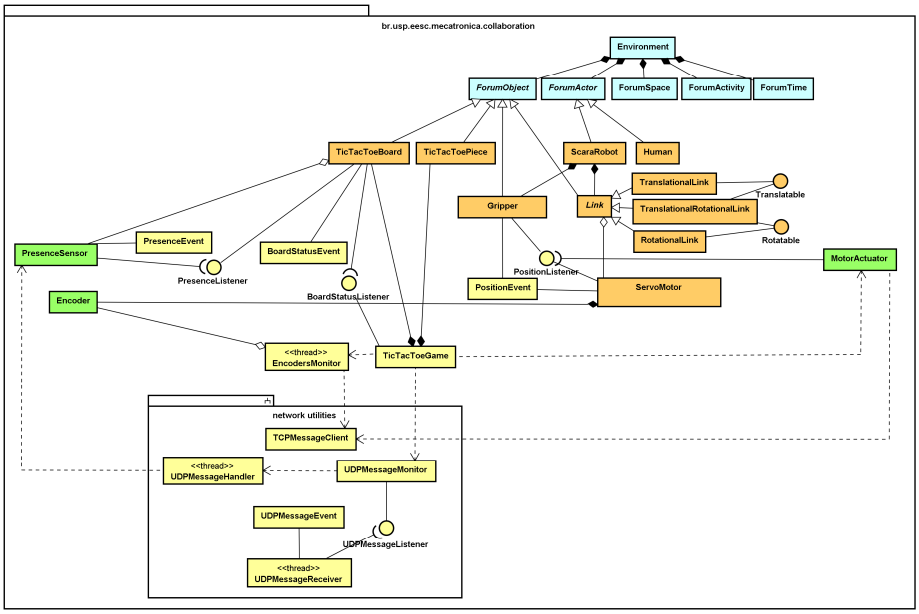


Fig. 8. Class diagram for Tic Tac Toe game

As described above, sensors were integrated to the system to detect the presence of pieces on nineteen different positions of the board. Another presence sensor (currently, a single switch) was also included to detect the presence of a human being into the shared workspace. The states of these twenty presence sensors are monitored by a client application that fires UDP (User Datagram Protocol) messages into the local network. Thus, using an UDP server (*UDPMessageReceiver*), the system allows asynchronous messages reading and performing event passing through appropriated listener implementations.
The current positions for servomotors were obtained by the system using an encoder monitor, which submits TCP (Transmission Control Protocol) requisitions to the target that controls the robot. TCP messages are also sent to the target by *MotorActuator* to reposition the servomotors, according the current states of their virtual representations by means of *ServoMotor* class instances.
As mentioned, listeners were used to provide events communication about states changing across objects into the virtual environment representation. Every change among virtual and real environments is communicated using TCP or UDP messages, allowing the distribution of the system components and the integration between high and low-level layers of the architecture.

## 5. Conclusion

In this chapter was presented a new architecture for robot control, which provides layers including deliberative behaviour on robot operation. The other features of the proposed

model refer to the explicit definition of local and global contexts and its operating support for distributed environments.
The collaboration among robots and human beings was described using a symbolic representation, through a formal model of rules. This approach was successfully experimented in restricted situations, describing human-robot interactions. An experimental case study was also presented for this purpose, involving a collaborative game among a manipulator and humans.
Future research about this subject can be applied evolving the model to support representations of other mental states and allowing the extraction of rules from knowledge databases. It is also encouraged the use of the model for other situations, including collaboration among other subjects (mobile robots or other machines), as uncovered by this chapter.

## 6. Acknowledgment

The publication fee of this chapter was supported by Municipal Education Foundation of Piracicaba (FUMEP – Fundação Municipal de Ensino de Piracicaba).
The authors would like to thank researchers of Mechatronics Laboratory at São Carlos School of Engineering (University of São Paulo) by all help with the experiments described in this chapter.

## 7. References

Aroca, R.; Tavares, D.M. & Caurin, G.A.P. (2007). Scara Robot Controller Using Real Time Linux. *Proceedings of International Conference on Advanced Intelligent Mechatronics*, pp. 1-6, ISBN 978-1-4244-1264-8, Zürich – Switzerland, Sep 2007, IEEE, New York.

Bishop, J. N.; Potter, W.D.; (2004). Towards Developing Behavior Based Control Architectures for Mobile Robots Using Simulated Behaviors. *Proceeding of the International Conference on Artificial Intelligence (ICAI'04)*, Las Vegas, Nevada.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. 2, Issue 1, Mar 1986, pp. 14-23, ISSN 0882-4967.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems,* Vol. 6, Issue 1, Jun 1990, pp. 3-15, ISSN 0921-8890.

Camolesi Jr, L. & Martins, L. E. G. (2006). A Model for Interaction Rules to Define Governance Policies in Collaborative Environments. In: *Lecture Notes in Computer Science*, Vol. 3865, Shen, W.; Chao, K.-M.; Lin, Z.; Barthès, J.-P.A.; James, A. (Eds.), pp. 11-20, Springer Berlin, ISBN 978-3-540-32969-5, Heidelberg.

Camolesi Jr, L. & Martins, L.E.G. (2005). Specifying Powerful Rules to Govern Collaborative Environments. *Proceedings of 9th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2005)*, pp. 810-815, ISBN 1-84600-002-5, Coventry – UK, May 2005, IEEE, New York.

Cao, Y. U.; Fukunaga, A. S. & Kahng, A. B. (1997). Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, Vol. 4, No. 1, Mar 1997, pp. 7-27, ISSN 1573-7527.

Crowley, K. & Siegler, R. S. (1993). Flexible strategy use in young children's tic-tac-toe. *Cognitive Science: A Multidisciplinary Journal*, Vol. 17, Issue 4, October-December 1993, pp. 531-561, ISSN 1551-6709.

Fodor, J. A. (1981). The Mind-Body Problem. *Scientific American*, Vol. 244, No. 1, Jan 1981, pp. 114-123.

Lau, H. Y. K. & Ng, A. K. S. (2006). Immunology-based Motion Control for Modular Hyper-redundant Manipulators. *Proceedings of the 16th IFAC World Congress*, ISBN 978-0-08-045108-4, Prague, Jul 2005, Elsevier, New York.

Martins Jr, J.; Camolesi Jr, L. & Caurin, G. A. P. (2008). Scara3D: 3-Dimensional HRI integrated to a distributed control architecture for remote and cooperative actuation, *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008)*, pp. 1597-1601, ISBN 978-1-59593-753-7, Fortaleza - Ceara - Brazil, Mar 2008, ACM, New York.

Minsky, M. (1990). Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. In: *Artificial Intelligence at MIT, Expanding Frontiers*, Vol. 1, P. H. Winston & S. A. Shellard, pp. 218-243, MIT Press, ISBN 978-0262231541, Cambridge.

Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review*, Vol. 11, Issue 3, Sep 1996, pp. 205-244.

Parker, L. E. (2003). Current research in multirobot systems. *Artificial Life and Robotics*, Vol. 7, No. 1-2, Mar 2003, pp. 1-5, ISSN 1614-7456.

Pinker, S. (1999). *How The Mind Works*, Penguin UK, ISBN 9780140244915, London.

Slonneger, K. & Kurtz, B. L. (1995). *Formal Syntax and Semantics of Programming Languages: a laboratory based approach*, Addison-Wesley Publishing Company, ISBN 0-201-65697-3, New York.

Tavares, D.M.; Aroca, R.V. & Caurin, G.A.P. (2007). Upgrade of a SCARA Robot using OROCOS. *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, ISBN 978-0-88986-685-0, Würzburg – Germany, Aug 2007, ACTA Press, Calgary.

Tonti, G.; Bradshaw, J. M.; Jeffers, R.; Montanari, R.; Suri, N. & Uszok, A. (2003). Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In: *Lecture Notes in Computer Science*, Vol. 2870, Fensel, Dieter; Sycara, Katia; Mylopoulos, John (Eds.), pp. 419-437, Springer Berlin, ISBN 978-3-540-20362-9, Heidelberg.

Vinge, V. (2008). Signs of the Singularity. *IEEE Spectrum*, Vol. 45, No. 6, Jun 2008, pp. 68-74, ISSN 0018-9235.