Paper:

Distributed Robotics Education

Henrik Hautop Lund and Luigi Pagliarini

Center for Playware, Technical University of Denmark
2800 Kgs. Lyngby, Denmark
E-mail: hhl@playware.dtu.dk, luigi@artificialia.com
[Received February 24, 2011; accepted July 12, 2011]

Distributed robotics takes many forms, for instance, multirobots, modular robots, and self-reconfigurable robots. The understanding and development of such advanced robotic systems demand extensive knowledge in engineering and computer science. In this paper, we describe the concept of a distributed educational system as a valuable tool for introducing students to interactive parallel and distributed processing programming as the foundation for distributed robotics and human-robot interaction development. This is done by providing an educational tool that enables problem representation to be changed, related to multirobot control and human-robot interaction control from virtual to physical representation. The proposed system is valuable for bringing a vast number of issues into education – such as parallel programming, distribution, communication protocols, master dependency, connectivity, topology, island modeling software behavioral models, adaptive interactivity, feedback, and user interaction. We show how the proposed system can be considered a tool for easy, fast, flexible hands-on exploration of these distributed robotic issues. Through examples, we show how to implement interactive parallel and distributed processing in robotics with different software models such as open-loop, randomness-based, rule-based, user-interaction-based, AI- and ALife-based, and morphology-based control.

Keywords: educational tool, distributed robotics, parallel processing, agent-based robotics, playware

1. Introduction

The teaching of robotics requires that many subdisciplines be considered and covered, including mechanical engineering, electrical engineering, computer science, and artificial intelligence. In this paper, we describe a concept for the distributed robotics education with an interactive hands-on approach that enables students to easily understand and develop control components for distributed robotics. The many research developments in distributed robotics make it evident that distributed robotics has become a major field of research and development in robotics (see [1] for a review) and therefore of high importance to robotics education. Such research developments include multirobotics, swarm robotics, modular robotics, and self-reconfigurable robotics, and also related developments within sensor networks, artificial life, and human-robot interaction with multirobot systems. Indeed, it can be argued that even behavior-based robotics, in its control of distributed parallel behavior, is an instance of distributed robotics, as was exemplified in explicit multirobot development based on behavior-based robotics [2].

In distributed robotics, coordination of the robotic system is distributed to a number of robotic units. The desired collective behavior of the distributed robotic system emerges from interactions between robotic units and interactions of robotic units with the environment – where a dynamic environment may include human beings. In contrast to centralized, single-robot systems, distributed robotic systems explore the features given by parallel and distributed systems thanks to the inherent flexibility and potential robustness that such systems provide for robotic application in a dynamic environment.

Over the last decade, we developed a new tool that enables building distributed platforms easy for physical interaction. The system has shown itself to be an excellent tool for educational purposes when introducing students to such complex problems as interactive parallel and distributed programming for distributed robotics. The foundation of parallel and distributed processing is indeed an important subject within distributed robotics – and even a major focal point in most computer science curricula and theoretical educational textbooks. This is because numerous other applications and systems are also based on the principle of parallel and distributed processing, including the Internet, cloud computing, parallel computers, multiagent systems, and swarm intelligence.

The proposed educational concept emphasizes enabling students to explore the complex, abstract themes of distributed robotics in a simple, playful hands-on manner with simple interactive building blocks that can be easily composed and manipulated by students. On purpose, these building blocks are stationary to facilitate students' cognitive entrance into and manipulation of the educational system, thereby focusing their attention on the basic challenges of distributed, parallel processing underlying distributed robotics and physical interactivity underlying human-robot interaction. Note that the mechanical challenges of multirobot systems are not addressed by this educational system.

Approaching education with the proposed educational concept for distributed robotics has several advantages. First, it may make the learning of the above computer science themes more interesting to students. Second, such systems may be a foundation of distributed robotics experience based on interaction, coordination, and/or manipulation of physical, parallel building blocks. Acquiring this distributed robotics foundation easily and inspirationally is crucial to enabling students to progress toward understanding and developing more advanced distributed robotics, e.g., using other resources.

2. Background

The distributed robotics field began in the late 1980s (Fukuda CEBOT [3], Arai [4], Asama [5]), and its rapid growth led to international conferences and symposium such as Distributed Autonomous Robotic Systems (DARS). In the 1990s, several universities started using the Khepera robot platform as a tool for distributed robotic systems education. The major educational focus on distributed robotic systems appeared, however, with the emergence of robot soccer tournaments such as RoboCup and FIRA in the late 1990s. These robot soccer tournaments focused on multirobot systems competing in ball games against multirobot opponents. The competitive nature of these tournaments quickly attracted researchers to use the tournaments as one way to educate students within the robotics field. A major challenge in generally exploiting this in education is, however, that it is extremely time-consuming and expensive to create multirobot platforms able to compete in robot soccer tournaments such as RoboCup.

In 1998, the authors of this paper invented RoboCup Junior with LEGO Mindstorms robots initially as a multirobot demonstration during RoboCup'98 in Paris in 1998 [6] and later as an interactive tournament for children and students during RoboCup'99 in Stockholm in 1999 and after [7]. With the development of user-guided evolutionary robotics and behavior-based systems for LEGO Mindstorms robots [7], we enabled learners from a very young age up to university develop control of such robot soccer players and enabled scaffolding where they could go step by step to more complex understanding and programming. Even though such educational systems enable easy access to robot programming, e.g., for RoboCup, because of the high-level abstraction needed to enable nonexpert learners to access educational systems, these systems often fail to go in depth with more basic issues – for example, distribution, communication protocols, master dependency, and adaptive interactivity. Other educational systems are needed to enable students to approach these complex basic issues underlying distributed robotics more easily.

For education in distributed physical processing systems, we also developed intelligent blocks – I-Blocks – focusing on the "programming by building" concept [8]. Students used I-Blocks for mathematics education, language education, and for IT education in Africa [9]. Similar systems include Smart-its, Cubelets, Roblocks, and Blinky Bots. As an educational system, I-Blocks showed the possibilities of using inspiration from distributed robotics to create an educational tool with distributed processing that, through its physical properties, gives students an easy hands-on way to manipulate and combine components of a distributed system.

This modular approach facilitates contextualized project-based education in a bottom-up approach focusing on individual competencies [10]. It emphasizes building up individual competencies through a problem-based approach in which individuals solve identified problems and challenges by gaining competency when needed to solve the problems at hand. This demands access to resources – not "passive resources" as in the case of more traditional curriculums but "active resources" used to create solutions for identified problems. Resources come into active use in forming students while simultaneously coming into direct use in society in the solutions created with these resources.

Educationally speaking, it is important to identify resources and tools enabling students to learn how to develop distributed robotics and create future applications for next-generation robotics, whose systems are based on interactive physical distributed processing, for example – a very interesting candidate because it provides a flexible physical system that can be set up and used by anybody anywhere within minutes [11, 12].

Other resources and tools include the Player/Stage, which started as a project at the USC Robotics Research Lab in 1999 to address interfacing and simulation for multirobot systems [13]. Player supports a variety of robots and provides a clean simple interface for robot sensors and actuators over a network. Stage provides a population of simulated efficient and configurable, rather than highly accurate, robots and sensors operating in a two-dimensional bit-mapped environment. Devices are accessed through Player as if they were real hardware. Stage simulates tens or hundreds of robots on a desktop PC and has been used in teaching undergraduate and graduate classes. With Player/Stage, the educational concept easily becomes one of *simulating* multirobot systems because of the obvious advantages of Player/Stage. Educationally, it remains important, however, to provide educational tools that enable manipulation with physical representation as is detailed in Section 3.

3. Educational Purpose: Concepts and Definitions for an Educational Course

For creating distributed multirobot systems, there are numerous basic issues related to parallel and distributed processing that a student must learn about, such as to what extend parallelism can improve efficiency and robustness and what algorithms can exploit parallelism. This leads, for instance, to the need to know about hierarchical and functional decomposition of problems. An educational

tool for this algorithmics learning should enable students to learn about when to use shared variables, e.g., in the master robot, and distributed variables in robots, when to use a scheduler in the master robot, how to use semaphore for critical sections, and, for instance, enable students to confront a mutually exclusive problem [14]. Low-level issues related to topology, communication, event-based control, prevention of deadlocks, data transfer, etc. (e.g., [15]) should be confronted together with high-level issues about distributed systems for understanding artificial neural network control, evolutionary robotics, multi-agent systems, swarm intelligence, artificial life, etc., as a basis for *distributed, multirobot systems*.

Many of these themes may appear quite abstract to engineering and computer science students interested in understanding and creating distributed robotic systems. A need clearly exists for an educational tool enabling students to confront these themes very concretely. We hold that the best way to learn about these abstract issues is through direct *hands-on problem solving*, following the pedagogical principles of Piaget [16], known as constructionism [17–19], and in computer science and robotics literature as guided constructionism [20]. We combine this with an approach contextualizing training for students by enabling them to work with technological building blocks [10]. Numerous experiments have shown that a hands-on problem-solving constructionist approach enables learners to confront abstract cognitive problem solved more simply through physical representation. For a mathematics education, for instance, Lakoff & Nunez [21] show how we project embodied or sensory motor reasoning onto abstract (mathematical) concepts to understand them. Researchers have shown the role of bodily activity in learning mathematics, e.g., [22]. One example is how the manipulative properties of interfaces may affect children's numerical strategies [23]. When designing objects to facilitate learning, we must consider the opportunities for action provided by the designed object or environment [24].

The fact that different representations, e.g., physical, may cause dramatically different cognitive behavior is termed "representational determinism" [25]. Zhang and Norman [26] propose a theoretical framework in which internal and external representation form "distributed representational space" representing abstract structures and properties of a task in "abstract task space" (p. 90). They developed this framework to support the rigorous formal analysis of distributed cognitive tasks and to assist their study of "representational effects [in which] different isomorphic representations of a common formal structure can cause dramatically different cognitive behavior" (p. 88). "External representation are defined as the knowledge of the structure in the environment, as physical symbols, objects, or dimensions (e.g., written symbols, beads of abacuses, dimensions of a graph, etc.), and as external rules, constraints, or relations embedded in physical configurations (e.g., spatial relations of written digits, visual and spatial layout of diagrams, physical constraints in abacuses, etc.)" (p. 180) [25].

To facilitate distributed representational space in distributed robotics education, we suggest using *interactive* parallel and distributed processing enabling students to easily and physically represent, interact with, and create their own parallel and distributed processing systems. Designing *interactive parallel and distributed robotics* software leaving behind conventional routes and finding another way of developing algorithms. This "other" programming paradigm demands that the programmer enter a new "state of mind," which is among the most difficult things to do. It is thus important to clearly understand the concepts and definitions underlying this interactive parallel and distributed processing paradigm, summarized in the sections that follow.

3.1. Interactivity

Interactivity here intends a physical and tangible interaction. The physical parallel and distributed system enables physically manipulating objects and material representations of information to be experienced. Technology embeds physical, conceptual, and cultural constraints. Mapping between physical affordances of objects with digital components – different types of output and feedback – is a design and technological challenge, since the physical properties of the objects serve both as representation and as control for their digital counterparts [27]. We make digital information directly manipulatable, perceptible, and accessible through the senses by physically embodying such information.

While playing with the system, users take advantage of distinct perceptual system qualities to make interaction tangible, lightweight, natural, and engaging. Interacting with a physical parallel and distributed system may mean jumping over, pushing, assembling, or touching physical objects and experimenting with a dialogue with the system in a very direct, nonmediated way. It is thus viewed as highly suitable, e.g., for student training. Undeniably, this enables direct hands-on experience and learning together with a funny and playful experience.

3.2. Parallel and Distributed

A computational process is called distributed [28] when a single computational atom is on one side autonomous and on the other insufficient to determine the desired outcome. A computational process is thus called distributed here when more than one computer (or robot) – communicating through any possible network – contributes to accomplishing the very same task by sharing different roles in a computational problem or process.

Whenever considering a *distributed (computational) process*, it is necessary to define the level of parallel vs. serial computational flow that the system should perform and to define "computational group" characteristics. Parallel computing is a form of computation in which many calculations are done simultaneously, operating on the principle that large problems can often be divided into smaller ones that are then solved concurrently ("in parallel"). Parallel computing comes in different forms – bit-

level, instruction level, data, and task parallelism. For distributed robotics education, we may be interested in the task parallelism problem, so an educational tool should be able to run distributed processes in at least three different ways – *fully-distributed, semi-distributed,* and *centralized*.

4. Educational Material

As the foundation for distributed robotics, interactive parallel and distributed system programming demands that student programmers shape specific abilities. After years of direct teaching experience, we believe that certain educational materials can simplify this learning process. The development of such educational material is guided by certain design policies.

4.1. Design Policies

We will present a number of interactive parallel and distributed subproblems that students must learn about. This leads to design policies outlined below for developing an open tool for dealing with all aspects of understanding and both low- and high-level programming and front- and back-end representation.

4.1.1. Classical Parallel and Distributed Process Subtasks

Coding parallel and distributed processes stress programming and the understanding of different levels such as physical – i.e., bit transmission; data link – i.e., packages, transmission error, and recovery; network – i.e., addresses and packages destinations; transport – i.e., message exchange between clients and master(s); session – i.e., defining and implementing sessions in priority and process-to-process communication; representation – i.e., working on data-format differences; application – i.e., end-user interaction and feedback; and understanding and implementing solutions for robustness – i.e., error diagnosis and recovery; reconfiguration – i.e., module assembly; unreliable communication – i.e., data loss, duplication, and corruption; parallelism and concurrency – i.e., language nondeterministic side-effects; and fixed and expanding parallelism – i.e., modifying the number of processors involved.

It is also essential when teaching information distribution to include problems such as system connection – i.e., total vs. partial connection; token-passing – i.e., how to share and act on critical information; deadlock prevention – i.e., wait-die, wound-wait, etc.; memory sharing – i.e., how to locate physical distributed system memory; topology – i.e., ordinary and complex topology algorithms, initial vs. run-time topology building, etc.; process transfer – i.e., distributing the work load, speeding up calculation, using hardware and software specialization among system modules; centralized vs. hierarchy vs. distributed approaches – i.e., centralized and decentralized information flow; and run-time adaptation – i.e., adapting system (re)actions on the fly.

In addition to all of the above "classical" subproblems of computer science, educational material must force the educational session to face other aspects that distributed robotic designers must deal with when learning parallel and distributed processing. Such subtasks include local and global connectivity, multifaceted hardware topologies, interactivity and adaptive interactivity, and multimodal feedback.

4.1.2. Connectivity

To realize a suitable interactive parallel and distributed platform, the educational tool must implement both a *local connection* system – through which hardware cells communicate with the neighbourhood and propagate such information both sides – and a *global connection* device through which to connect with neighbour platforms and any external tools.

4.1.3. Multifaceted Hardware Topologies

Since educational tool modularity implies the use of run-time de/attachable modules, hardware/software topology is strongly emphasized and demands great effort to understand programming and deal with such structures. In our model, we identified three specific topology subtypes:

1. *Regular*, i.e., a one-block platform – i.e., any given group of hardware cells attached contiguously and sharing a single master cell – with modules attached in a square or rectangle.

2. *Irregular*, i.e., a one-block platform arranged in any shape but nevertheless in which hardware cells must be continuous, meaning that assembly does not show any discontinuity and no cell or group of cells is isolated.

3. *Island configurations*, i.e., a platform consisting of more than one block – i.e., as defined in 1, and 2. It makes no difference whether master cells mutually communicate, communicate through an external device, or do not communicate at all.

4.1.4. Interactivity

Implementing software in a modular interactive educational tool implies designing – or at least dealing with – a relevant dynamic and interactive scenario, since in most cases, software use itself relies on physical and continuous user action. Students/designers must deal with completely different requirements based on whether single- or multi-user software is targeted. Students also must hypothesize a wide variety of behavioral situations, even including those – in our personal experience – in which a single-user platform will have many users or multiuser software will be run by a single user.

4.1.5. Adaptive Interactivity

The way we approach interaction in such a modular and distributed model leads beyond the classic idea of Human-Machine Interaction (HMI) and is of fundamental importance because it promotes and applies – under both physical and cognitive circumstances – user adaptation and user adaptivity. First, with our model being architecturally reconfigurable – eventually run-time-reconfigurable – represents of itself the essence of adaptation. Being focused on physical user action, such a system is easily tailored to user activity either in real time or over the long term. To achieve such a goal, the educational tool must be programmable using many different strategies that also depend on the quality and quantity of feedback that students/designers are willing to exchange with users. Feedback and multimodal feedback are introduced below. We have shown in more then one case [29, 30] that using modular interactive tiles enabled us to detect user characteristics and adapt software execution to those. In further tests, we found that by capturing a user's conditional attitude and adapting software execution to this that, in some cases, we could eventually modify user behavior itself [30].

4.1.6. Multimodal Feedback

When discussing HMI, we commit ourselves to the idea that "*how* you give is more important than *what* you give," and focus on software and tools that can both give and get feedback from users.

When developing software for a modular interactive tool, we constantly try to provide the user with *immediate feedback,* e.g., LED, score, and *delayed* or *long-term feedback,* e.g., adaptivity, documentation software, since we believe that both components are essential to modern robotics design. For immediate feedback from modular interactive tiles, we use a light (LED) configuration or colors. Any time a need exists for a stronger, more complex, or long-run "signal," we interface with external devices in a layered mode in which each feedback layer can be added/removed freely on top of each other – something we call layered multimodal feedback [31]. External devices may be "passive" as vision-oriented feedback – e.g., screen, projector, etc.; sound-oriented feedback – e.g., loudspeakers, buzzers, etc.; or "active" such as computational devices that run an analysis or link user action to specific databases through external communication – e.g., radio and the Internet.

In conclusion, to manage and teach the many features of parallel and distributed modular robotics programming, we require a system that is robust, reliable, and easily reconfigurable, so we present such an educational system based on these design policies to enable us to shift the level of representation from very abstract to very empirical.

4.2. Hardware Specifications

From an educational view of what is needed and would be of real value is a tool that enables studying and understanding parallel and distributed processing underlying distributed robotic systems while stressing the user and/or multiuser interactivity component. The Modular Interactive Tiles System (MITS) may provide educators and students with such a tool and approach, since the system is based on "robotic" modules with certain properties. Each robotic module has a physical expression and processes and communicates with its environment. Communication with the environment is through communication to neighboring robotic modules and/or sensing or actuation. A modular robot consists of many robotic modules.

In the MITS, the term "robotic module" is used in the broad sense. The module is not mobile but physical output possibilities consist of colored light and sound. MITS modules are purposefully simple, with no mobile actuation to focus students' attention on the basic issues of interactive parallel and distributed processing as a basis for developing interactive distributed robotics. Mechanical issues – e.g., related to mobile actuation of modules – are therefore excluded from this approach and tool, to enable students to first learn about the other basic issues before moving on to mechanical issues, which inevitably involve one more level of complexity.

The MITS approach inherits the behavior-based robotics method [32] and uses it assuming that behavior-based systems can include both the coordination of primitive behavior in terms of control units and include the coordination of primitive behavior in terms of physical control units. We assume a physical module being a primitive behavior, so the physical organization of primitive behavior is, together with interaction with the environment, what decides overall system behavior. Similar to controlling robot behavior by coordinating primitive behavior, we imagine the overall behavior of a robotic artifact to emerge from the coordination of a number of physical robotic modules each representing a primitive behavior, eventually opened to single/multiuser interaction.

Modular interactive tiles attach themselves to each other to form the overall system. Tiles are designed to be flexible and motivational in providing immediate feedback based on physical user interaction following design principles for modular playware [33].

Each modular interactive polyurethane tile is quadratic, measuring $300 \times 300 \times 33$ mm – see **Fig. 1** and **Table 1** for specifications. In the center is a quadratic dent of 200 mm wide with a raised circular platform of 63 mm in diameter in the center. The dent can contain a Printed Circuit Board (PCB) and electronic components on the PCB, including an ATmega 1280 as the main processor in each tile. At the center of each of the four quadratic sides is a small tube 16 mm in diameter through which infrared (IR) signals are sent and received from neighboring tiles. On the back of a tile are four small magnets. Those on the back enable a tile to be mounted on a magnetic surface – e.g., a wall. Each side of a tile has a jigsaw puzzle pattern to provide opportunities for tiles to attach themselves to each other. This jigsaw pattern ensures that when two tiles are put together, they will become aligned, which is important for ensuring that tubes on two tiles for IR communication are

Fig. 1. Modular tiles used for foot or hand interaction.

Table 1. Specification of a modular interactive tile.

	Amount	**Type**
Processor	1	ATmega1280
Sensor	1	FSR
Sensor	1	2-axis accelerometer
Effector	8	RGB color LEDs
Communication	4	IR transceivers
Communication	1	XBee radio chip
Energy	1	Li-Io Polymer battery
Switch	1	On/Off switch
Connector	4	Jigsaw puzzle
Size	300 mm × 300 mm ×33 mm	
Weight	1 kg	

aligned. On one side of the tile is a small hole for a charging plug used for connecting a battery charger, including an on/off switch.

A small groove on the top of the quadratic dent wall enables a cover to be used on the dent. The cover consists of two transparent satin-ice plates on top of each other, with a sticker in between as a visual cover for the PCB.

A Force Sensitive Resistor (FSR) on the center of the raised platform underneath the cover serves as a sensor enabling analog measurement on the force exerted on the top of the cover. A 2-axis accelerometer (5G) on the PCB detects horizontal or vertical tile placement. Eight RGB light emitting diodes (LED SMD 1206) are spaced equally in a circle on the PCB to light underneath the transparent satin-ice circle.

Modular interactive tiles are individually battery-powered and rechargeable by a rechargeable Li-Io polymer battery on top of the PCB. A fully charged tile runs continuously for 30 hours and takes 3 hours to recharge. The battery status of individual tiles is seen when switching on tiles and is indicated by white lights. When all eight lights appear, the battery is fully charged, and only one white light lit means the tile must be recharged. This is done by turning tiles and plugging the intelligent charger into the DC plug next to the on/off switch.

The PCB has connectors for an XBee radio communication PCB add-on, including a MaxStream XBee radio communication chip. The two types of tiles are thus master tiles with a radio communication chip and slave tiles without. The master tile may communicate with the game selector box and initiates programs on the built platform. Every platform must have at least one master tile if communication is needed, e.g., to a game selector box or a PC.

With these specifications, a system consisting of modular interactive tiles is a fully distributed system, with each tile containing processing (ATmega 1280), its own energy source (Li-Io polymer battery), sensors (FSR sensor and 2-axis accelerometer), effectors (8 color LEDs), and communication (IR transceivers, and possibly XBee radio chip), as indicated in the specifications in **Table 1**. Each tile is thus self-contained and runs autonomously. The overall behavior of the system consisting of such individual tiles is, however, the result of assembly and coordination by all of the tiles.

Modular interactive tiles are easily set up on a floor or wall within one minute by attaching them similar to a jigsaw puzzle. No wires are involved. Tiles register whether they are horizontal or vertical and thus make software games behave accordingly.

Tiles may be grouped into tile "islands" and groups communicate mutually wirelessly (with radio communication). An application may, for example, be running distributed in a group of tiles on the floor and a group on the wall, requiring that users interact physically with both the floor and wall.

5. Proposed Educational Course

Based on the definition of our educational intentions and educational material – hardware and software tools – we have built for bringing these educational intentions to life, we describe a pedagogical work-flow for classes based on our experience. We have been using the MITS as an educational tool since 2002/2003 (see [34] for further information), as part of university bachelor's, master's, and Ph.D. degree classes, and as a platform for thesis studies and research. These classes, e.g., adaptive robotics and robotic agent courses, and studies were for obtaining engineering degrees in robotics. Through such long experience, we have been able to verify the MITS pedagogical value and to refine applicable models for theoretical and hands-on education.

The examples and phases that follow were built thanks to on-site experience and didactic practice. We will thus try to reconstruct what is and has been the best practice of almost a decade of empirical teaching and supervision, since we could observe how quick and powerful it was for student learning. We strongly believe that such an educational tool and paradigm provides easy and painless access to the above topics while letting alumni gently slip into hard-core experience in a – vice versa – actually quite scary and challenging discipline as distributed robotics with underlying parallel and distributed interactive computing.

Fig. 2. PCB and components of a modular interactive tile.

Fig. 3. Assembly of the modular interactive tiles as a simple jigsaw puzzle.

Fig. 4. Physical interaction with the modular interactive tiles placed on the ground.

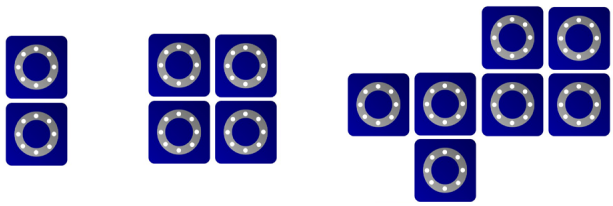


Fig. 5. Examples of different topologies.

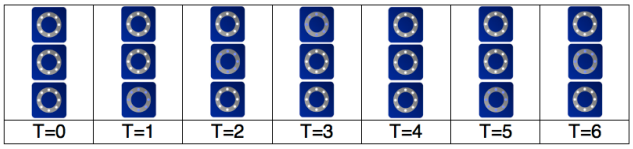


Fig. 6. Open loop behavior, 7-state sequence.

5.1. Introduction Phase

The teacher/tutor should introduce students to the hardware platform (**Figs. 2**, **3**, and **4**) and ask the class to implement all needed protocols for obtaining a robust, efficient, reliable parallel and distributed system. This requires and encourages students to face the basic algorithms and protocols that the subtasks of parallel and distributed systems need – e.g., physical, data link, network, transport, session, and representation levels.

5.1.1. High-Level Experience

Once such a start-up system is obtained from student work or a premade system, the second step is, for example, testing the system by working on problems such as application, robustness, communication, system connection, token-passing, deadlock prevention, parallelism, reconfiguration, memory sharing, topology, and process transferring.

The MITS model is ideal for implementing all of the above challenges because hardware components are minimalistic and distributed system complexity can be developed and tested quickly and easily (**Fig. 5**).

Once students have reached this level of competency, the tutor leads their attention to a higher level of representation and asks them to implement end-user-interaction-based applications that highlight distributed robotics issues. An attractive option involves linking robotics behavior understanding and development to gaming development to enhance student motivation. This demands an educational focus on extracting knowledge from gaming examples to robotics issues. Examples follow.

5.2. Hands-On Education Phase

Once a specific topology is chosen, software engineering students can implement and run a large variety of tasks. Here, we start by considering examples to be applied to a *semi-distributed*, *single-user application* on a *regular topology* platform.

5.2.1. Open Loop and Randomness-Based Behavior

The simplest case, a naïve one, could be the following open loop behavior (**Fig. 6**).

In open loop behavior, light is "passed" from one module to either an adjacent or a distant module using a predefined open loop or randomness-based algorithm. In both cases software is cycling and we must introduce an *interactivity level* – e.g., the player scores a point when the lighted tile is hit – to stop it and, by doing so, transforming the two into games for very young children. When the user press a tile, dynamics stop and tiles freeze in a pattern until the user presses the lighted tile again, after which the light shift sequence starts again.

5.2.2. Rule-Based Behavior

One step further is rule-based software characterized by a pattern sequence – either predefined or random – governed by a specific rule or rule set. The simplest case we consider is one in which, given any machine state and configuration – e.g., two tiles – states that are on turn off and

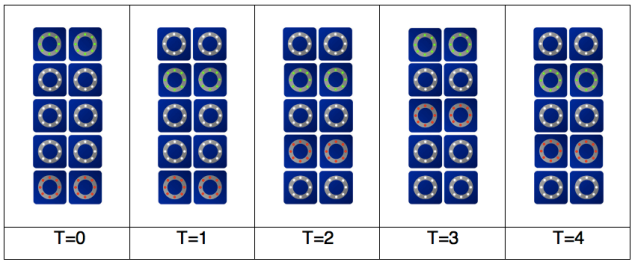


Fig. 7. *American Football*, a 5-state sequence.

Fig. 8. left: RoboCupJunior, and right: Roomba, two examples of behavior-based robotics application.

those that are off turn on. A much more complex setting can, of course, be designed but essentially this is the logic used in rule-based software.

When an interaction element is introduced in such rule-based software, a more dynamic scenario appears, denoted by rules and users acting coactively and contributing step by step to the system state. Such a situation is clearly observed in *American Football* (**Fig. 7**).

American Football is one-against-one game where, given, say, a 5 (width) per 2 (height) cluster of modular interactive tiles, interactive software acts so that when the game begins, the platform extremes appear activated – i.e., light goes on – and two different colors – i.e., blue and red – are used. Squeezing tiles, a user "pushes" color/activation forward in the row – i.e., switches off the squeezed tile and switches on the adjacent one in the direction toward the opponent. The user who first pushes a color to the opposite end of the game platform wins. Such practice introduces students to a robotic behavioral approach or logic – e.g., RoboCupJunior, Roomba (a commercially available floor vacuum cleaner), etc., (see **Fig. 8**) – illustrating a common approach to constructing robot behavior and one of the approaches most widely used in industrial applications.

5.2.3. User-Interaction-Based Behavior

A user-interaction-based program is per se an interactive software concept in which the user directly contributes to the next machine state – i.e., tile color or activation. Such a software model is quite similar to the interactive version of rule-based software because users themselves cannot determine machine states without the aid of an underlying algorithm. It basically differs from the rule based software in terms of strain on increasing the user role and contribution to the next machine state, and

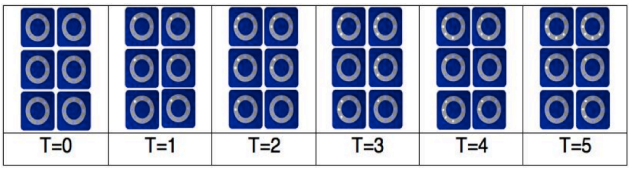


Fig. 9. *Final Countdown*, a 6-state sequence.

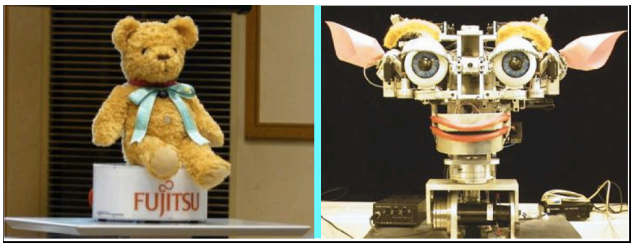


Fig. 10. left: Fujitsu Teddy-Bear Robot, and right Kismet, two examples of interactive robotics applications.

an attempt to reduce the rule component. A good example is *Final Countdown* (**Fig. 9**), in which the tile platform varies both in aspect and size, since application components all behave in the same way. It consists of a number of tiles that, when the application is initiated, are fully lit – i.e., any color will do. After initialization and with a given interval, e.g., one second, they all start "fade-out," switching off their 8 lights clockwise one at a time. If all of the lights of one go completely off, the game ends. To restore a single tile to the initial state, the user must squeeze it. The wider the platform, the more important the strategy becomes in keeping the game/application alive.

This education in intelligent artifacts and human-robot relationships is the basis for interactive robotics. User-interaction-based robotics is a widespread and important topic in robotics, e.g., Fujitsu's Teddy-Bear Robot, Kismet, etc. (see **Fig. 10**). This makes it important that students practice on simple use but behaviorally articulated platforms, such as MITS, that give students an opportunity to explore underlying interactive robotics issues straight-forwardly.

5.2.4. Morphology-Based Behavior

MITS strength lies in its physical modularity – and with this, a wide variety of applications can be hypothesised. Instead of looking at interactivity as a mostly digital experience, this approach views human-robot behavioral interaction as the rule itself where the user assembles modules – i.e., tiles, in our case – to fulfil task requests and accomplish the "goal" of reconfigurable robotics practice, which could be experienced when handling more complex robotics modules – e.g., Atron, Conro (see **Fig. 11**).

The simplest example is the *AndOr Game* in which the player must identify which particular tile configurations turn tile lights green (see **Fig. 12**). Another example is the *ColorMix* application, in which colors flooding and mixing between tiles are put together by the user [11].

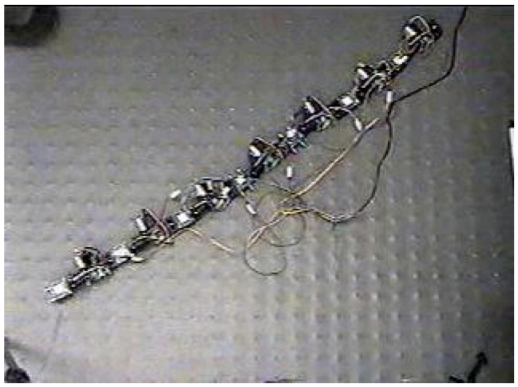


Fig. 11. Conro modular robots used for user configuration and self-reconfiguration.

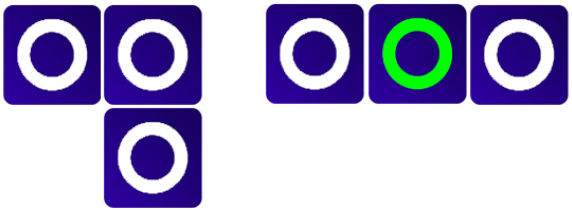


Fig. 12. *AndOr*, a sequence of 2 possible states.

5.2.5. AI- and ALife-Based Behavior

The AI- and ALife-based software are an extension of what we define as rule-based systems, which essentially rely on the same principles for autonomous and interactive versions, although the quality of the computational experience is much higher in software behavioral equality/variety, (un)predictability, etc. Since modular interactive tiles tend to resemble pixel structures, they appear to easily incorporate a consistent number of classical and modern AI paradigms. A good example is Cellular Automata (CA), a discrete model used in computability theory and different fields, which consists of a regular grid of cells, each with a finite number of possible states – e.g., on, off – that can change their state based on their neighbourhood activation states [35]. To initiate students into such complex relationships – i.e., ALife-based robotics – we use MITS and have them implement one of the most well-known CA algorithms, *Conway's Game of Life* (see **Fig. 13**). We ask them to add an interactive aspect – e.g., pressing a tile toggles the tile state from off to on or vice versa – transforming it into an intriguing game in which the player must keep the platform alive – i.e., modules change their configuration – as long as possible. We believe that using such a paradigm is pedagogical in instructing students about interactive robotics and in learning on how to create different "living" artifacts.

5.2.6. Distributed Physical Teleplay

Combining some of the above development possibilities enables students and developers to create more advanced physical systems in a contextualized action-based

Fig. 13. *Game of Life*, a particular moment in the interactive game, implemented on an older tile version.

situated scaffolding approach to education. An example is how distributed systems were developed for stroke and cardiac rehabilitation in hospitals and for soccer game practice.

In the soccer game example, tiles are placed on a wall and a specific number lights up in different colors. Each counts down with eight LEDs. Players must hit a tile with a ball before LEDs all turn off and get points for how many LEDs are turned on when a tile is hit. Points are multiplied by a factor for how high the tile is positioned – row 1, 2, or 3. One of the tiles will have its LEDs spin quickly at random intervals, indicating that if the tile is hit, a bonus round will start during which players can get extra points when hitting lit tiles.

To increase motivation for interacting with the system, multimodal immediate feedback was used so that the player would receive immediate feedback directly from tiles in changing colors, but we also added sound and graphic feedback in time and score via a host computer to enhance the system as social playware. When a player hit a lit tile, the light would go off on that tile and jump to another tile, a loudspeaker would sound, and the score would be shown on a monitor. When the game ended, high scores would be shown.

In teleplay, high scores were shared over the Internet on different continents so that players were playing directly and simultaneously against each other in soccer games, e.g., in Asia and Africa. To explore such physical teleplay, the application was tested over broad ranges of cultural differences in users and environments, e.g., tested simultaneously by over 1.000 users in Denmark, South Africa, and Japan during FIFA World Cup 2010 (**Fig. 14**). In Asia, the system was tested in highly metropolitan areas, such as in Shibuya, Tokyo, whereas in South Africa, it was tested in such diverse places as an orphanage for HIV/AIDS children, townships, a public market, a village, an official FIFA Fan Park, a science center, a university, a bar for soccer fans, and a public park in Soweto (see [31] for details).

The simple physical teleplay over distance introduces students to robotics telepresence in a practical, hands-on manner. Telepresence has been studied intensively in

Fig. 14. Playware soccer in Atteridgeville township, South Africa, during FIFA World Cup 2010.

robotics, e.g., by Ishiguro with the studies of a human-like presence using teleoperated androids [36, 37]. Such robotic studies have promoted a human-like technology to study presence, and some large videoconferencing setups have enabled a soccer teleplay [38]. The educational system presented here is a first step with a much simpler yet playful technology for mediating social interaction in line with simple yet effective FeelLight robot technology for mediating social interaction as introduced by Suzuki and Hashimoto [39]. Hence, as a first step in student learning, we enable the study of telepresence sans any large bulky infrastructure and sans anthropomorphic expression, but with simply expressed light patterns, sound, and scoring. In the specific example, such simple telepresence was studied and experienced by connecting players in Asia, Africa, and Europe in a soccer game with the MITS.

6. Discussion and Conclusion

Robotics education must largely extend itself based on seminal literature [17–20, 40], characterized by converging approaches of technology, computer science, and cognitive psychology [32, 41–43]. From a psychological and educational point of view, robotics implements three main ideas:

- Knowledge as *action* stressed by Piaget's and Papert's theoretical contributions [16–18].
- Knowledge as a process basically dependent on cultural and technological *scaffolding* [44, 45].
- Knowledge as a *situated* reciprocal interaction between an organism and its environment [20, 46].

Developing a robotic system, the learner acts concretely on problem-solving strategies [19] based on *high-tech scaffolding*.

Creating action-based, situated distributed robotics scaffolding is a major challenge, with *distributed robotics* itself highly complex. A need thus exists for a simple educational concept and tool that enables action-based, situated scaffolding starting from basic issues in distributed robotics.

We have developed an educational concept that uses MITS – a modern parallel and distributed modular robotics platform – that can be used to teach both traditional and modern educational topics on distributed interactive robotics. We have also defined pedagogical paths that should, step by step, deal with all of the problems and subproblems a discipline such as robotics may present. Our educational purpose focuses on physical interaction with parallel and distributed robotic systems and to highlight the many challenges that computer science and engineering students may face in understanding and designing interactive parallel and distributed robotic systems.

It is our belief that a system such as modular interactive tiles is a tool for easy, fast, flexible learning and exploration of these challenges, e.g., as shown in examples of how to implement interactive parallel and distributed processing with different behavioral software models such as open loop, randomness-based, rule-based, user-interaction-based, AI-based, and ALife-based control. MITS provides an educational hands-on tool, a basic starting point that enables a change in the representation of abstract problems related to designing interactive parallel and distributed robotic systems. Through the use of such a tool, students can learn about both classical and modern aspects of parallel and distributed systems and collective robotics, and easily create their own physical systems. Such practice will enable them to migrate easily to more complex systems that, as for an example taking the case of Atron modular robots [47], include motor action. As is the nature of scaffolding, not all issues should be covered initially, and the proposed educational tool emphasizes initially, through action-based, situated learning, addressing issues at the foundation of distributed robotics preparatory to mechanical issues, as part of scaffolding.

It is our conviction that the proposed educational concept and tool will enable many teachers to enjoy a high degree of motivation among students by enabling them to address the cognitive complex and abstract basic issues of distributed robotics in a simple, playful, and interactive manner. By changing the representation of complex and abstract basic issues to a physical representation with modular tiles, students are able to experiment with these issues playfully while forming a sound basis of knowledge before proceeding to other issues in the robotics curriculum.

References:

[1] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in Multi-Robot Systems," IEEE Trans. on Robotics and Automation, Vol.18, No.5, pp. 655-661, 2002.

[2] M. Mataric, "Behaviour-based control: Examples from navigation, learning, and group behaviour," J. of experimental and theoretical artificial intelligence, Vol.9, No.2, pp. 323-336, 1997.

[3] T. Fukuda and S. Nakagawa, "A dynamically reconfigurable robotic system (concept of a system and optimal configurations)," Proc. of IECON, pp. 588-595, 1987.

[4] T. Arai, H. Ogata, and T. Suzuki, "Collision avoidance among multiple robots using virtual impedance," Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 479-485, 1989.

[5] H. Asama, A. Matsumoto, and Y. Ishida, "Design of an autonomous and distributed robot system: ACTRESS," Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 283-290, 1989.

[6] H. H. Lund, J. A. Arendt, J. Fredslund, and L. Pagliarini, "Ola: What Goes Up, Must Fall Down," J. of Artificial Life and Robotics, Vol.4, No.1, 1999.

[7] H. H. Lund and L. Pagliarini, "RoboCup Jr. with LEGO Mindstorms," Proc. of Int. Conf. On Robotics and Automation (ICRA2000), IEEE Press, NJ, 2000.

[8] H. H. Lund, "Intelligent Artefacts," In Sugisaka and Tanaka (Eds.), Proc. of 8th Int. Symposium on Artificial Life and Robotics, I11-I14, ISAROB, Oita, 2003.

[9] H. H. Lund and M. Vesisenaho, "I-Blocks in an African Context," In Proc. of 9th Int. Symposium on Artificial Life and Robotics, pp. I-7-I-12. ISAROB, Oita, 2004.

[10] H. H. Lund and E. Sutinen, "Contextualised ICT4D: a Bottom-Up Approach," Proc. of 10th Int. Conf. on Applied Computer Science, WSEAS, Japan, 2010.

[11] H. H. Lund, M. D. Pedersen, and R. Beck, "Modular Robotic Tiles – Experiments for Children with Autism," Proc. of 13th Int. Symposium on Artificial Life and Robotics (AROB'13), ISAROB, Oita, 2008.

[12] H. H. Lund, "Modular Robotics for Playful Physiotherapy," Proc. of IEEE Int. Conf. on Rehabilitation Robotics, IEEE Press, pp. 571-575, 2009.

[13] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," Proc. of the Int. Conf. on Advanced Robotics (ICAR), pp. 317-323, Coimbra, Portugal, 2003.

[14] D. Harel, "Algorithmics," Addison-Wesley, 1987.

[15] A. Silberschatz, J. Peterson, and P. Galvin, "Operating System Concepts," Addison-Wesley, 1991.

[16] J. Piaget and B. Inhelder, "La psychologie de L'enfant," Paris, P.U.F., 1966.

[17] S. Papert, "Mindstorms: Children, Computers, and Powerful Ideas," Basic Books, New York, 1980.

[18] S. Papert, "Constructionism: A New Opportunity for Elementary Science Education," A proposal to the National Science Foundation, 1986.

[19] F. Martin, "Ideal and Real Systems: A Study of Notions of Control in Undergraduates Who Design Robots," Y. Kafai and M. Resnick (Eds.), Constructionism in Practice: Rethinking the Roles of Technology in Learning, MIT Press, MA, 1984.

[20] H. H. Lund, "Robot Soccer in Education," Advanced Robotics Journal, Vol.13, No.8, pp. 737-752, 1999.

[21] G. Lakoff and R. Nunez, "Where Mathematics Comes From," New York: Basic Books, 2000.

[22] R. Nemirovsky, C. Tierney, and T. Wright, "Body motion and graphing," Cognition and Instruction, Vol.16, pp. 119-172, 1998.

[23] A. Manches, C. O'Malley, and S. Benford, "The role of physical representations in solving number problems: A comparison of young children's use of physical and virtual materials," Computers & Education, Vol.54, pp. 622-640, 2010.

[24] J. J. Gibson, "The Ecological Approach to Visual Perception," Boston: Houghton Mifflin, 1979.

[25] J. Zhang, "The Nature of external Representations in Problem Solving," Cognitive Science, Vol.21, No.2, pp. 179-217, 1997.

[26] J. Zhang and D. A. Norman, "Representations in Distributed Cognitive Tasks," Cognitive Science, Vol.18, pp. 87-122, 1994.

[27] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms," Proc. of CHI: Human Factors in Computing Systems, pp. 234-41, 1997.

[28] D. Rumelhart, and J. McClelland, "Parallel Distributed Processing, Vol.I," Cambridge, Mass.: MIT Press, 1986.

[29] A. Derakhshan, F. Hammer, and H. H. Lund, "Adapting Playgrounds for Children's Play Using Ambient Playware," Proc. of IEEE Intelligent Robots and Systems (IROS'06), IEEE Press, Hong Kong, 2006.

[30] H. H. Lund and A. T. Thorsteinsson, "Adaptive Playware in Physical Games," Proc. of Foundations of Digital Games 2011, ACM, 2011.

[31] H. H. Lund and T. Thorsteinsson, "Social Playware for mediating tele-play interaction over distance," Proc. of 16th Int. Symposium on Artificial Life and Robotics, ISAROB, Japan, 2011.

[32] R. Brooks, "A robust layered control system for a mobile robot," IEEE J. of Robotics and Automation, Vol.2, No.1, pp. 14-23, 1986.

[33] H. H. Lund and P. Marti, "Designing modular robotic playware," IEEE Int. Workshop Robots Human Interactive Commun, Toyama, Japan, IEEE Press, pp. 115-121, 2009.

[34] A. Derakhshan, F. Hammer, H. H. Lund, Y. Demazeau, and L. Pagliarini, "Adapting Playgrounds using Multi-Agent Systems," Proc. of the Ninth Scandinavian Conf. on Artificial Intelligence (SCAI2006), pp. 151-158, Espoo, Finland, October 25-27, 2006.

[35] J. v. Neumann, "The general and logical theory of automata," in L. A. Jeffress (Ed.), Cerebral Mechanisms in Behavior – The Hixon Symposium, John Wiley & Sons, New York, pp. 1-31, 1951.

[36] H. Ishiguro, "Android science: Conscious and subconscious recognition," Connect. Sci., Vol.18, No.4, pp. 319-332, 2006.

[37] D. Sakamoto, T. Kanda, T. Ono, H. Ishiguro, and N. Hagita, "Android as a telecommunication medium with human like presence," Proc. 2nd ACM/IEEE Int. Conf. Human-Robot Interact, 2007.

[38] F. Mueller, A. Agamanolis, and R. Picard, "Exertion Interfaces: Sports over a Distance for Social Bonding and Fun," Proc. of CHI 2003, ACM, Vol.5, No.1, pp. 561-568, 2003.

[39] K. Suzuki and S. Hashimoto, "FeelLight: A Communication Device for Distant Nonverbal Exchange," Proc. of ETP'04, New York, USA, ACM, 2004.

[40] O. Miglino, H. H. Lund, and M. Cardaci, "Robotics as an educational tool," J. of Interactive Learning Research, Vol.10, No.1, pp. 25-48, 1999.

[41] V. Braitenberg, "Vehicles: experiments in synthetic psychology," MIT Press, Cambridge, MA, 1984.

[42] S. Woodcock, "Game AI the state of industry, Game Developer," Miller Freeman, 1999.

[43] F. L. Lewis, "Neural network control of robot manipulators," IEEE Expert/Intelligent System & their Applications, Vol.1, No.3, 1996.

[44] J. S. Bruner et al., "Studies in Cognitive Growth," John Wiley & Sons, Inc., New York, 1966.

[45] L. S. Vygotsky, "Thought and language," Cambridge; MIT Press, 1986.

[46] A. Clark, "Being There: Putting Brain, Body and World Together Again," Cambridge, Mass., Bradford/MIT Press, 1997.

[47] E. H. Ostergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the ATRON lattice-based self-reconfigurable robot," Autonomous Robots, Vol 21, No.2, pp. 165-183, 2006.

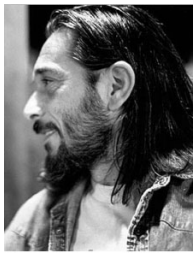
Name:
Henrik Hautop Lund

Affiliation:
Center for Playware, Technical University of Denmark

Address:
Building 325, 2800 Kgs. Lyngby, Denmark

Brief Biographical History:
2000-2009 Professor, University of Southern Denmark
2009- Professor, Center for Playware, Technical University of Denmark

Main Works:
- Founded and headed the LEGO Lab in 1997-2000.
- Founded the RoboCluster industrial promotion organization.
- Inventor of the RoboCup Junior robot football game.
- Principal investigator of the HYDRA project, developed ATRON reconfigurable robot.
- RoboCup Humanoids Free Style World Champion 2002.
- Inventor and developer of the Laudrup, Hogh & Lund RoboSoccer, which was used at the FIFA World Cup 2010 in South Africa.
- Inventor and developer of the RoboMusic in collaboration with World Music Award winner, remix musician Funkstar De Luxe.



Name:
Luigi Pagliarini

Affiliation:
Center for Playware, Technical University of Denmark

Address:
Building 325, 2800 Kgs. Lyngby, Denmark

Brief Biographical History:
2004- Professor, Theory of Perception and Psychology of Shape, and of Introduction to Computers at the Academy of Fine Arts of Bari, Italy
2009- Adjoint Associate Professor, Center for Playware, Technical University Denmark

Main Works:
- Musician, Electronic Artist, Curator, Founder and Director of the Pescara Electronic Artists Meeting.
- President of the Cultural Association Artificialia.
- Inventor of the RoboCup Junior robot football game.
- Founder of RoboCup Junior and Member of its International Committee.