29

EMBODIED INTELLIGENCE TO TURN EVOLVABLE ASSEMBLY SYSTEMS REALITY

Regina Frei and José Barata
Department of Electrotechnical Engineering, New University of Lisbon
{regina.frei, jab}@uninova.pt

Evolvable Assembly Systems may successfully resolve industry's problems with low volume / high change productions; thanks to Embodied Intelligence, systems can play an active role in the engineering procedure. Modules do not only consist of their physical body including small local controllers, but also represent themselves in Virtual Reality. Interacting with each other, they Self-Organize to fulfill the ever-changing production requirements. Systems become user-friendly, distributed and more autonomous.
This article concretizes the required control solution by detailing the different control-related issues and tasks at hand. Examples of information needed for computer-readable specifications of parts, processes and system modules are given. A navigator system is proposed to transform resource-independent assembly instructions into layout-specific executables.

Keywords: Evolvable Assembly Systems, Reconfigurable Production Systems, Automation, Autonomous Systems, Embodied Intelligence

1. INTRODUCTION

Industry-branches which have to cope with small lot sizes and frequent changes need innovative solutions such as Evolvable Assembly Systems, EAS (Frei et al. 2007; Onori 2002). They are based on a thorough analysis of product properties and production processes; the systems' modularity must be supported by a correspondingly modular control solution (Marik et al. 2007; Rizzi et al. 1997). Solutions consist of agents or other technologies like Service Oriented Architectures (Jammes and Smit 2005) or Function Blocks (Lewis 2001), which however need to be completed with proactivity.

One of the most important remaining questions is how to realize these control systems, which must be easy to use, dependable, flexible and scalable. Specialized and extensive programming must be avoided. Being user-friendly implies the systems to be as autonomous as possible, hiding complexity and providing the user with well-represented, specific information resp. requests. Chances are best for Multi-Agent Systems (MAS) with extended system capabilities, able to organize themselves to a high degree.

Ideally, system modules are plugged together the way a new mouse is plugged into a laptop, which, if necessary, automatically searches for the right driver and installs it, or eventually asks for user interaction in a precise way. Similarly, the module controllers will communicate with each other, verify their compatibility, create complex skills based on their individual simple skills, and offer them to the user. Functionality is readily offered.

A possible approach towards achieving such a distributed, autonomous control solution with Embodied Intelligence is discussed in this article. Embodied Intelligence is a concept widely used in the area of intelligent and cognitive systems in the Artificial Life (AL) research domain. The main idea is that intelligence requires a body to interact with (Pfeifer and Scheier 2001). Intelligent behavior emerges from the interaction of brain, body, and environment. In the specific case of EAS the idea is not to consider the concept strictly as understood in the AL domain but using an interpretation in which each assembly component has computer power (brain), its hardware (body), and is placed within its external environment. As in the AL domain, the actions of the systems (realizing assembly according to the requirements), which can be regarded as intelligent actions, are the result of the interactions between the components, its onboard computer, and the environment.

Chapter 2 presents control-related tasks in a conceptual overview, chapter 3 details module specifications and chapter 4 takes up the issue of the autonomous generation layout-specific assembly operations and layout improvements and the way it could be achieved. Finally, conclusions and outlook are provided.

2. CONTROL-RELATED TASKS

Automated assembly consists of various phases and / or levels which lead one to another, following the flash upwards, in time as well as in logic (Figure 1). They are often iterative and closely related, like running the assembly operations and monitoring them. The next few paragraphs describe the phases in some detail, starting with the lowest.

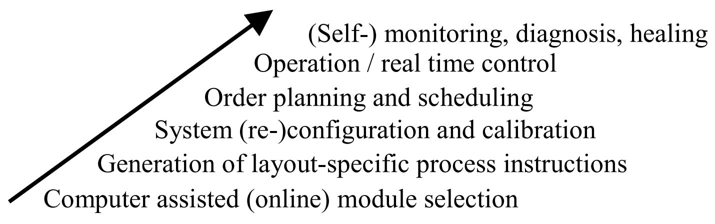

(Self-) monitoring, diagnosis, healing
Operation / real time control
Order planning and scheduling
System (re-)configuration and calibration
Generation of layout-specific process instructions
Computer assisted (online) module selection

Figure 1. Control-related tasks in EAS

2.1. Module Selection with Embodied Intelligence

Embodied Intelligence opens up much more possibilities for interactive system design. All available standard modules, i.e. those present on the shop floor, those stored in the repository and eventually those offered for rent or sale by a network of system suppliers, also exist in a “second world”, a Virtual Reality (VR). This is not a simple user interface with simulation. Modules represent themselves and their current or future interactions in the Module Pool. The user specifies the types of products to be treated. According to the product class, suitable module class will propose itself to the user, based on a well-elaborated ontology of processes, products and systems, as explained in (Barata et al. 2007a; Onori et al. 2004). The user can then select the wished modules.

If there is no suitable module available for a certain task, the user is advised to reconsider the product and its processes in order to solve the problem with standard modules, or could otherwise ask a module supplier to create the required special module. The modules guide the user in this selection procedure, giving indications about their compatibility, performance, emplacement constraints, etc.

2.2 From resource-independent assembly instructions to specific executables

A product to be assembled brings generic assembly instructions, which determine how the parts shall be joined to each other. This description is not specifically made for a certain system layout; most products could even be assembled by hand. It does not mention, neither, how the parts get from their storage place to the assembly location.

In the scenario considered here, the "navigator" (Figure) will then combine the generic assembly plan with the existing layout or the chosen modules and generate concrete steps to realize the assembly. This means that the instructions will now be interpreted for the actual layout and thus transformed into executable programs, using the modules' skills. Such programs must, however, stay easily modifiable in case the layout is changed again later or in case one or several modules should become unavailable.

The way this navigator is realized could be similar to the navigating systems used in cars: a dynamically created system map (Embodied Intelligence!) is compared with the initial situation, i.e. the parts as fed into the system, and the goal, which is the completely assembled product. The paths to bring the parts from their initial location to their final target position are then calculated (chapter 4).

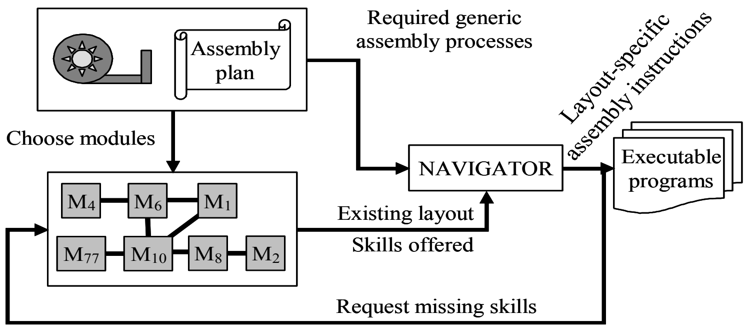


Figure 2. From generic assembly instructions to layout-specific operations

2.3. System (re-)configuration, layout improvements and calibration

When the modules are being connected to each other, they will announce their presence and contact their immediate "geographical" neighbors. Each module provides one or several functions (simple skills). The user can decide about the complex skills to be formed, but even better, the modules know themselves with whom they can collaborate (or not) and know in which way their skills can be combined (chapter 3). Thanks to the matching between the requirements of the tasks to be executed on one hand, and the skills offered by the layout on the other hand, discrepancies will be detected. Also unsuitable module combinations or locations will cause a user alert, asking for a layout change (chapter 4).

The modules will be able to autonomously calibrate themselves and their interactions with others. This requires that a module, e.g. a basis axis, can identify if there are other modules fixed on top of it, e.g. second or third axis including a gripper. Calibration data will be stored and used for monitoring effects of fatigue and other deviations from the original values.

CoBASA (Barata 2005) can already today fulfill most of these tasks. An increased version with more functionality is currently under construction.

2.4. Order planning and scheduling

Many agent-based planning and scheduling algorithms are available in literature, e.g. (Valckenaers and Van Brussel 2005), or even already implemented in industry, such as the "Truck Scheduler" (Magenta Technology) described in (Rzevski and Skobelev 2007); therefore, this issue is not further detailed in the scope of this work.

2.5. Operation / run-time control and Orchestration

Real-time control must be highly efficient and robust. Tiny controllers to include in modules allow truly distributed operation. The challenge is to develop MAS for those small real-time controllers, to be used at fine levels of granularity – i.e. as an integrated part of modules. Hardware as well as software is required. MAS environments such as JADE is computationally too heavy for tiny controllers, which will not be full PCs but something smaller, lighter.

Agent-oriented Architectures (AoA) as well as Service-oriented Architectures (SoA) are suitable choices for autonomous, adaptive, decoupled and distributed systems as required for EAS. In AoA and SOA, the logical and physical aspects of an entity are uncoupled, enabling a Holonic approach.

The main question is how to make two devices, with no previous knowledge of each other's type, recognize each other and start interacting. To solve this problem, devices need knowledge processing capabilities. Using semantic web services, knowledge is explicit through machine-interpretable semantics and can be inferred by machine-reasoning. Applying this approach to the manufacturing domain, components previously unknown can be recognized straightforward and be ready to interact with existing ones. It could be also possible to select the best available service following search parameters, such as QoS (Quality of Service), supplier, past activities, etc. After identifying the services, it is possible to compose them to complex processes through orchestration and choreography. Combining AoA with SoA, a new paradigm (or evolution of an existing two) can arise by joining "the best of two worlds", allied to crescent technology improvements that allow to put even more intelligence in ever tinier devices.

2.6. System (self-)surveillance, (self-)diagnosis, (self-)healing

Autonomous systems are able to maintain themselves in good condition. Self-monitoring is required at the level of each individual module as well as on cluster level and (global) system level. The modules locally manage their maintenance and service schedules and inform the user about forthcoming events.

Creating a way of autonomously supervising and diagnosing higher (emergent) system levels is not easy in a situation characterized by frequent changes. Of course the monitoring / diagnosis system could be manually adapted whenever the layout has changed – but this would be a breach with the goal of creating autonomous systems with minimal user interaction. Methods for system self-diagnosis are currently being developed (Barata et al. 2007b).

In certain cases, the system will be able to "heal" itself, e.g. by restarting a blocked controller or exchanging a problematic gripper. If the failure is more serious, the shop floor staff will be alerted. In the meantime, the navigator reconsiders the requirements and the current layout (now with unavailable modules) and tries to find alternative ways of fulfilling the current tasks, eventually delegating the operations to modules which have the same skills but work slower or which are already executing other tasks. What counts is keeping production running.

3. MODULE, PART AND OPERATION SPECIFICATIONS

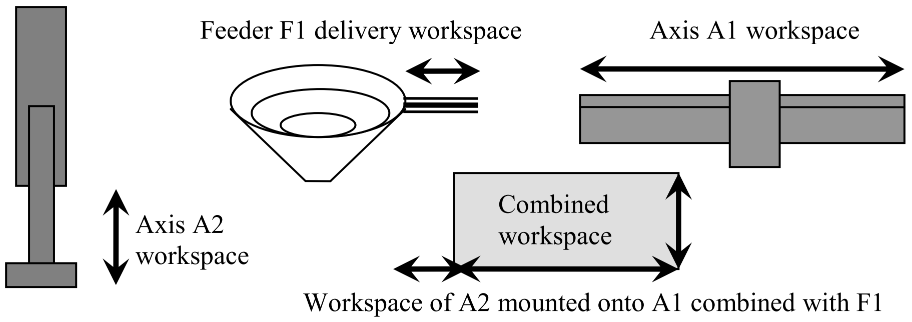
Embodied Intelligence means that agents must know themselves, their physical bodies, their workspace and their interactions with others. This implies specifying parts, modules and operations / skills in a computer-readable way.

3.1. Module specifications

Modules need an internal functional model of themselves and their working space, in the geometrical sense, as well as specifications of their pneumatic, electric and electronic interfaces. They could then match with each other much in the way jigsaw puzzle pieces match with each other.

When modules come together, their workspaces merge. This may result both in an expansion or in a limitation; modules can give each other more freedom or constrain each other. Generally, the consequences may be calculated by the agents, using vector addition (Figure 3). Doing this properly is crucial for collision avoidance by the means of the control software.

Besides their interface descriptions, the modules may also carry a changeable list of preferable partner modules, and accordingly establish a "black list" of modules which are known for causing trouble or being unsuitable. Similarly, modules will carry rules for forming complex skills in collaboration with their partners. The most frequent combinations might be explicitly stated – while more compositions may emerge when encountered. Such a mechanism prepares the system for the eventual emergence of functions or combinations which the system designer did not originally plan.


Feeder F1 delivery workspace
Axis A1 workspace
Combined workspace
Axis A2 workspace
Workspace of A2 mounted onto A1 combined with F1
Figure 3. Example of individual and combined workspaces (symbolic representation)

3.2. Part specifications

In order to be treated by autonomous agents (or alternatively, to be autonomous agents themselves), parts must be precisely specified, analogous to the module specifications. There is plenty of potentially useful information, including:

- Geometry: plate, sphere, hemisphere, cylinder, cube, bar, stick, triangle, etc.
- Material: steel, cupper, aluminum, rigid plastic, rubber, composite fiber, etc.
- Properties: insulating, conducting, magnetic, transparent, etc.
- Mass
- Rigidity (or stiffness), elasticity
- Conditioning: in bulk, on pallet, in band, etc.
- Way of gripping

Even a CAD file for every part might be included in the part specifications. This would allow graphically specifying the locations of assembly operations, of gripping points and other relevant spots (example: Figure 4).

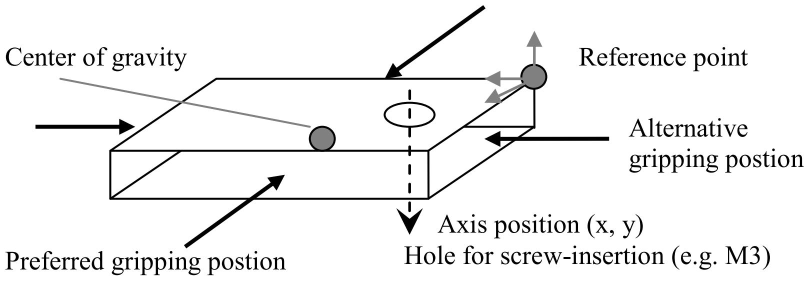

Center of gravity
Reference point
Alternative gripping postion
Axis position (x, y)
Hole for screw-insertion (e.g. M3)
Preferred gripping postion

Figure 4. Specifications of a plate with hole

3.3. Operation specifications

On the resource side, the operations a module can do are called skills; on the requirement side, they correspond to the processes needed to assembly a product. The goal of specifying them generically is to describe operations independent from the concrete part to be treated and independent from the resource (modules) which will execute the tasks. This gives the system a certain level of abstraction and thus the liberty to attribute any module with suitable skills to the operation in question. The execution of a certain task will not be blocked if a certain module is unavailable or out of service; having the abstract description, the agents may find one or several other modules corresponding to the requirements and able to do the task at hand. Neither will a part stay untreated in case a certain (maybe composite) process becomes unavailable; knowing the part and its needs allows finding alternative processes.

4. FROM RESOURCE-INDEPENDENT ASSEMBLY INSTRUCTIONS TO SPECIFIC EXECUTABLES

A simplified scenario will serve to explain the concept mentioned in section 2.2. We imagine the automated and autonomous assembly of a tape roller, consisting of four parts and a carrier, as symbolically shown on Figure ***8***5. The main input to the system are part specifications including precise but layout-independent assembly instructions. The part descriptions indicate where the parts come from, i.e. if they are contained in a feeder or stored on pallets, and how they are preferably grabbed by which kind of gripper. The already existing system layout is as schematically represented in Figure ***10***7 a).

Two different approaches are possible: Either the assembly plan is an agent, which has the task to procure the (passive) parts and assemble them, or the parts themselves are agents which must find their destined “neighbors” according to the plan, which is a passive piece of information. An intermediate solution could be that both assembly plan and parts are agents and collaborate in reaching their common goal.

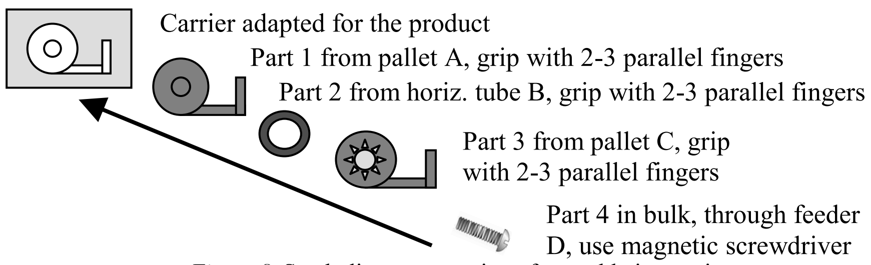

Carrier adapted for the product
Part 1 from pallet A, grip with 2-3 parallel fingers
Part 2 from horiz. tube B, grip with 2-3 parallel fingers
Part 3 from pallet C, grip with 2-3 parallel fingers
Part 4 in bulk, through feeder D, use magnetic screwdriver

Figure 8. Symbolic representation of assembly instructions

The fact that each module knows its neighbors allows implicitly knowing the system composition; this is knowledge can easily be made explicit by forming a kind of map in Virtual Reality. This makes it easy to form routing tables based on those used in telecommunications. This VR could then be used in a similar way as a navigator in modern cars. Parts are located on their storage spot and can calculate their possible trajectories to the carrier resp. the final assembly (Figure 9). This can imply the transfer over a series of conveyors and the handling by several robots on the way – partners which all must be interacted with. Obstacles need to be avoided, and timing respected – the before-mentioned orchestration of the assembly modules. Based on the assembly instructions and these calculations, a resource usage plan (Table 1) is generated. The specific control programs for the modules to execute all these actions are then derived automatically; asking the user for confirmation is obviously necessary.

Table 1. Resource usage plan (P&P= "Pick & Place", R = robot, g = gripper)

	Operation plan	**Resource**
1	Bring carrier	Conveyor
2	P&P part 1	R1 G1
3	Pick part 2	R1 G1
4	Insert part 2 into part 1	R1 G1
5	Pick part 3	R1 G1, R2 G3
6	Place part 3 on part 1, align +/- 2°	R1 G1, R2 G3
7	P&P part 4, screw part 4: 5x 360°	R2 G1
8	P&P finished product (unload)	R2 G3, R1 G1

In case of difficult, long or impossible handling paths, the system may propose layout changes, based on a set of relatively simple rules such as:

- Check if a part is sent from a point to another and back again; if yes, try moving the storage location or the assembly place.
- Check if a part has a long trajectory compared to the size of the system; if yes, try moving the storage location or the assembly place.
- Check if all the axes / robots are used at least at (e.g.) 60% of their time; if not, try replacing the ones with lower usage. Try to equally distribute the work charge to all the robots, eventually requesting the addition of suitable grippers or other supplementary modules from the storage.

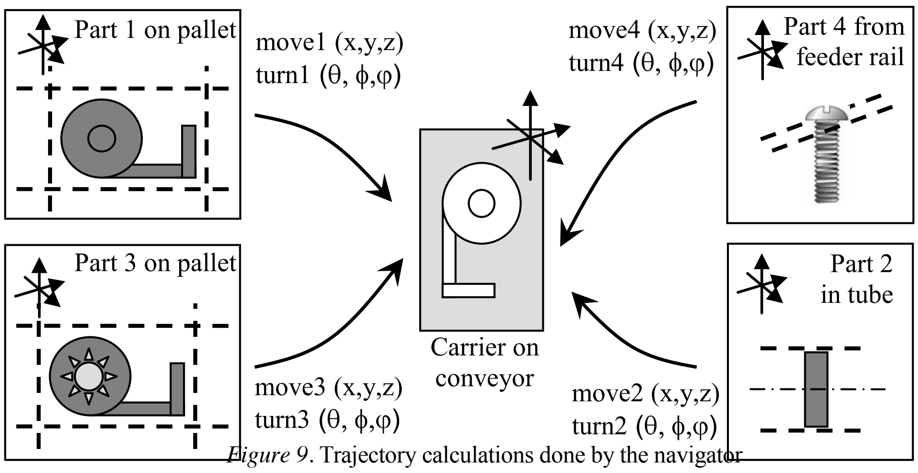


Figure 9. Trajectory calculations done by the navigator

As an example, the layout in Figure 10 a) could be improved as shown in b). It could mean that robot R1 has a parallel 2-finger gripper (G1) and can do all the needed movements, but it is a rather slow robot. Robot R2 has a magnetic screw-driver (G2) and is fast. A compatible parallel 2-finger gripper (G3) is available in the storage, as the system detects in the list of available modules. It would thus propose to add G3 and to move pallet C as well as the unloading station from R1 to R2.

Obviously, the system is neither capable of moving modules and pallets nor should it be allowed to take such decisions autonomously – at least not before the system has been proven absolutely dependable. The right approach is to inform the user about the detected possibility for improvement and letting her decide.

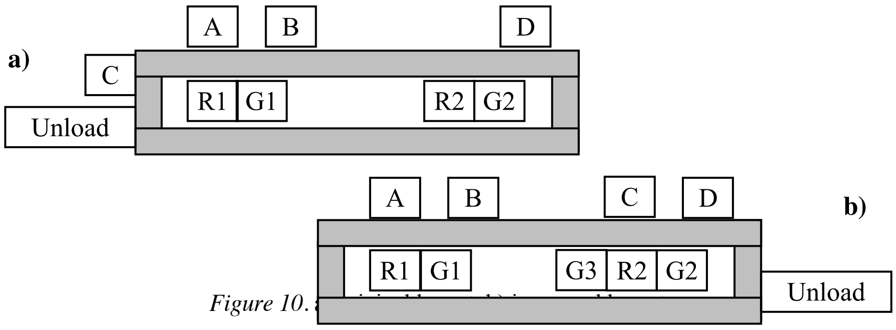


Figure 10. [illegible]

5. CONCLUSIONS AND OUTLOOK

The control-related tasks in EAS require innovative solutions. The realization of user-friendly, evolvable and more autonomous control systems for EAS relies on the use of Embodied Intelligence. It implies the computer-readable specification of parts, modules and processes. System modules consist of their physical body, their local controller and their software agent, representing them in Virtual Reality. They thus need thorough knowledge of themselves and their interaction characteristics. A well-founded ontology for EAS is currently being made and will soon be available. This article structured the different steps and suggests solutions such as the automatic creation of a dynamic system map in VR and the use of a "Navigator", similar to the navigation systems in cars. Generic, layout-independent assembly instructions can thus be transformed into layout-specific operations (i.e. executable programs) which will lead to the assembly of the product.

Most ideas presented in this article are still in theoretical form; their implementation has just begun. The authors are aware of the fact that many difficulties and complications will only surface when the ideas are put into practice. Nevertheless, the concepts are expected to turn out to be highly useful for future systems and make their handling user-friendly and fast.

It is well-known that production industry is a very traditional business and requires systems to be predictable, reliable and traceable. System autonomy is for most old-school engineers a red flag. However, the world is moving fast, and especially in Swarm Robotics great advances are being made. Even if not directly applicable to production systems with heterogeneous agents / modules and complex, specific tasks, such advances foster the development of new ideas and approaches. In order to convince the research departments of innovative industrial companies to collaborate in such futuristic research, a careful and stepwise approach is necessary.

6. REFERENCES

1. Barata, J. Coalition Based Approach For ShopFloor Agility, Amadora - Lisboa, Edições Orion, 2005
2. Barata, J., Frei, R., Onori, M. "Evolvable Production Systems: Context and Implications". In ISIE'07 - IEEE International Symposium on Industrial Electronics, Vigo - Spain, IEEE, 2007a
3. Barata, J., Ribeiro, L., Onori, M. "Diagnosis on Evolvable Production Systems". In ISIE'07 - IEEE International Symposium on Industrial Electronics, Vigo - Spain, IEEE, 2007b
4. Frei, R., Ribeiro, L., Barata, J., Semere, D. "Evolvable Assembly Systems: Towards User Friendly Manufacturing". In ISAM'07 - IEEE International Symposium on Assembly and Manufacturing, Ann Arbor - Michigan - USA, IEEE, 2007, pp 288-293.
5. Jammes, F., Smit, H. Service-oriented paradigms in industrial automation. IEEE Transactions on Industrial Informatics 2005; v1, n1: 62-70.
6. Lewis, R.W. Modelling control systems using IEC 61499 - Applying function blocks to distributed systems, IEE, 2001
7. Marik, V., Vyatkin, V., Colombo, A.W., eds. Holonic and Multi-Agent Systems for Manufacturing, Heidelberg, Springer, 2007
8. Onori, M. "Evolvable Assembly Systems - A New Paradigm?". In ISR2002 - 33rd International Symposium on Robotics (Stockholm), 2002, pp 617-621.
9. Onori, M., Lastra, J.L.M., Alsterman, H. "Evolvable Assembly Platforms - Definitions, Approaches and Requirements". In IPAS (Bad Hofgastein, Austria), 2004
10. Pfeifer, R., Scheier, C. Understanding intelligence. Cambridge, MA, MIT Press, 2001
11. Rizzi, A.A., Gowdy, J., Hollis, R.L. "Agile Assembly Architecture: an Agent Based Approach to Modular Precision Assembly Systems". In International Conference on Robotics and Automation, 1997
12. Rzevski, G. and Skobelev, P. Emergent Intelligence in Multi-Agent Systems. Windsor, Berkshire, UK: Magenta Technology, 2007.
13. Valckenaers, P., Van Brussel, H. Holonic Manufacturing Execution Systems. CIRP Annals - Manufacturing Technology 2005; v54, n1: 427-432.