Collaborative routing of products using a self-organizing mechatronic agent framework: A simulation study

Luis Ribeiro, Andre Rocha, Angelo Veiga and Jose Barata

Linköping University Post Print

N.B.: When citing this work, cite the original article.

Original Publication:

Luis Ribeiro, Andre Rocha, Angelo Veiga and Jose Barata, Collaborative routing of products using a self-organizing mechatronic agent framework: A simulation study, 2015, Computers in industry (Print), (68), 27-39.

<http://dx.doi.org/10.1016/j.compind.2014.12.003>

Copyright: Elsevier

<http://www.elsevier.com/>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-117369>

Title: Collaborative Routing of Products using a Self-Organizing Mechatronic Agent Framework – A Simulation Study.

Abstract: Scheduling is a fundamental activity in modern shop floors. It is also known to be a highly complex problem which has motivated several sub-formulations and the subsequent models. Traditional approaches, typically enumerative or heuristic, struggle to contain the computation complexity and often present solutions for restricted cases that feature unrealistic assumptions in respect to the system size, flow of products and the system logistics/behavior. The multiagent-based architecture presented in this paper is aligned with a set of emerging architectures that seek to explore more heterarchical decision and control models to circumvent the limitations of the traditional approaches. The main distinguishing factor of the proposed architecture is that it directly addresses (re)routing/local scheduling of products in plug and produce systems. It does not make any assumptions on the alignment of the orders and, instead, it dynamically handles the potential rescheduling of the orders already on the system based on the available resources, and their state, in a time efficient way. The architecture was tested under a simulation environment, that is geometrically accurate and that supports plug and produce in runtime, to characterize its performance under dynamic conditions.

Keywords:

Multiagent Systems, Material Handling, Self-Organization, Scheduling, Product Oriented Systems, Mechatronics

1. Introduction

The scheduling of products, parts, assemblies or sub-assemblies to different shop floor resources is a well-known and complex problem. It has been typically formulated as an optimization problem. It is also known to be NP hard consisting in the selection of the best possible schedule out of $n!^m$ where n is the number of tasks (or jobs) and m the number of available machines. The complexity of determining effective schedules is subsequently aggravated when [1]: operators and tools are included in the process, optimization occurs both for planning and scheduling, unpredictable conditions impact the system (failures, breakdowns, system changes, production surges).

This has promoted the sub-formulation of the scheduling problem to meet distinct objectives and several performance indicators have been chosen as the optimization target. When the main objective is to improve the system's balance, typical problem formulations include: maximization of line utilization, minimization of number of stations given the cycle time, minimization of the cycle time given the number of stations, a compromise between the number of stations and cycle time [2, 3]. When the optimization objective is more focused on performance then makespan [4-13], minimization of tardiness [11, 14, 15], throughput [16, 17], energy efficiency [18], work in progress [12], activity based costing [19], etc., are commonly used to characterize the performance of the scheduling algorithms.

Scheduling has therefore been approached from different perspectives, with different objectives. The conventional approaches are based in enumerative or heuristic algorithms and are normally able to produce near optimal solutions. Known techniques and algorithms include: genetic algorithms [6, 14], ant colony optimization [8, 10], particle swarm optimization [9, 13, 15], fuzzy control [12, 13] and neural networks [7].

These techniques consider the system and the orders to be scheduled as a whole which is the main strategy for obtaining a near optimal solution. However these strategies are extremely

sensible to system perturbations. Any significant system change requires a complete rescheduling. The time efficiency of these algorithms is also directly related with the size of the system. In this context, they are not suitable for highly dynamic systems of a larger size. Computational tractability has been dealt with by introducing constraints in the problem formulation. These limitations normally include: reduced number of stations and jobs, reduced differentiation of stations, inexistence of machine breakdowns or system changes, unlimited buffering capacity, process flows without repeating stations, no accounting for transport costs, etc. The purpose of introducing these constraints is to significantly reduce the search space of the algorithms. However these also limit the application scope in real scenarios and these techniques are normally restricted to small job-shop-based systems.

Scheduling is therefore a pressing issue in bigger and more complex systems that have increasing reconfigurability requirements. The key aspects of a reconfigurable system include: module mobility, diagnosability, integrability, convertibility, scalability, “automatibility”, modularity and customization [20, 21]. All these characteristics promote and support frequent and dynamic system changes that are hardly tackled by the traditional approaches and that address the need for mass customization and sustainability. This problematic has originated, in the past fifteen years, a set of multiagent-based techniques that tend to replace exhaustive or heuristic search with negotiation [1] or some form of collaborative interaction. It is at this point however important to distinguish between:

- multiagent-based approaches that rely in the agent concept as a mere distributed computation construct and seek to explore the availability of distributed computational resources;
- and multiagent-based approaches that encourage an identity relation between purpose-specific agents and their physical counterpart and explore collective self-organizing phenomena.

The first case is closer to the conventional approaches, already described, and shares the limitations considered before.

The article focuses therefore in the latter case which has led to the emergence of several reference architectures. These architectures, collectively, provide the constructs to integrate: planning, scheduling, control, material handling, monitoring and diagnosis.

The multiagent systems community has been particularly active, in the past decade, in the development of agent-base architectures for manufacturing systems. This research has generated a wide range of system specific architectures [22] and set of more generic reference architectures among which one may mention: PROSA [23], HCBA [24], ADACOR [25], COBASA [26], Rockwell Automation Agents [27] and, more recently, ORCA-FMS [4] and the IDEAS Architecture [28]. The architectural design at this level is particularly relevant since it bridges the gap between more abstract concepts and paradigms, such as Holonic Manufacturing Systems [29], Bionic Manufacturing Systems [30], Evolvable Production Systems [31], and system engineering.

The present work positions itself in this context and places particular emphasis in handling manufacturing systems under dynamic conditions. In particular, it addresses systems where:

- stations and transport elements can be plugged and unplugged at any time (dynamic topology);
- and orders are of random nature.

The proposed architecture envisions the system as a directed graph where links and nodes are directly connected to specific agents. These continuously compute transport costs between locations and maintain local routing tables featuring the shortest path, and direction, to reach all relevant locations in the system. The architecture supports the integration of user defined metrics so as to cater for the requirements of a wider range of systems. The architecture was

deliberately designed to be independent of other external planning or scheduling tools and targets the orders that are already on the system for which time-effective and runtime decisions need to be considered. In particular, the paper seeks to demonstrate that the architecture is able to improve the overall makespan, in systems supporting plug and produce, while balancing the load on the different components. One fundamental aspect is that all the components are fully decoupled and hence there are no centralized points of failure other than the ones imposed by the mechanical limitations of the system under control. In this context, agents make their own decisions and scheduling is necessarily locally derived and seeks to explore collective phenomena to attain a global consistent state rather than a fully optimized solution.

The subsequent details are organized as follows: section 2 discusses the related literature in agent based manufacturing scheduling; section 3 presents the proposed architecture, supporting algorithms and relevant implementation details; section 4 summarizes the differences between this work and the state of the art; section 5 presents and assesses the main results; section 6 features the main conclusion and points future research directions.

2. Related Literature

2.1. Brief Survey

When dealing with complex and dynamic shop-floors there is the need for frequent rescheduling of the orders already in the system as well as the orders that are about to enter it. Hence, there is an interweaving between material handling and scheduling activities that is often ignored in the traditional models. There have been several contributions towards the definition and tackling of these problems. This section focuses on more recent work where the documented approaches are closely related with the one proposed. A complete survey on other approaches and techniques can be found in [1].

Material handling is a problem that is not exclusive of the manufacturing domain and is a complex problem on its own. In [32] the problematic of developing a generic automated material handling systems (AMHS) is discussed. The authors conclude that most of the AMHSs are sector specific and that the takes on a more generic approach often fail due to the specificities of particular sectors. Also, and not surprisingly, the authors reject centralized approaches due to their rigidity when handling dynamic conditions and voice their concerns regarding pure heterarchical approaches due to the high potential for deviations from optimized solutions. Finally, they propose a hybrid solution in [16] which features a three layer hierarchical system composed of a planner, a resource scheduler and a local traffic control. The local traffic control does specialized optimization in restricted regions of the system using routing rules while the two higher level blocks handle sequencing and coordination issues. The authors also focus their AMHS model on two particular sectors where they identified several structural commonalities. They were therefore also able to verify a high degree of code commonality between the solutions for the different sectors.

A similar and previous solution can be found in [33] where a Holonic-based architecture introduces a global scheduler. The main decision process is mediated by the order holons that are able to negotiate with the material handling holons as well as with the global scheduler to devise the execution plan. The order holons globally announce the transportation task and wait some time for the transportation bids. The local bids from the material handling holons tend to arrive faster and a local schedule is built upon them. The scheduling is later revised if the global scheduler is able to generate a timely solution. A follow up of this proposal is presented in [34] where local schedules are generated at material handling holon level (lowest hierarchical level in the proposed architecture) and recombined from a systemic perspective at the Global View Holon (highest hierarchical level).

Dynamic routing for AMHS is also reported in [3] where the authors follow a heterarchical approach whereby active route agents exchange link state information to enable the dynamic update of the node's routing tables. Exchange of link information is constrained by geographic zones to control the information dissemination. The routing decision is based on the computation between the shortest path between origin and destination. The approach has shown to be adaptive under failures and promoting the balancing of the load through the available resources.

In the manufacturing domain, material handling and routing is slightly more complex since it is often not only a matter of routing from origin to destination as the establishment of the routes must fulfil process precedence constraints.

In this context, several approaches have been proposed. Following the trend in AMHS the emerging architectures are adhering to more distributed and heterarchical models [35]. These models typically rely in structured negotiation/interaction [5, 17], normally using the contract net protocol [17], or in bio-inspired principles of self-organization such as the deposition of pheromones or any other indicator leading to an attractiveness field [4, 25, 36, 37] (see [38] for a complete survey on bio-inspired agent-based techniques also featuring the more conventional approaches).

Agent interaction has been explored in [5]. The work follows a monitor-analyse-plan-execution loop to maintain the reactivity of the holons in changing conditions. The order agents analyse the feasibility of the orders in respect to the timely completion of their process plan. Accepted orders are then step-wisely allocated to the different resources in the system so that a complete solution can be attained. At the same time order holons monitor their progress and assess the potential rescheduling of activities, the integrity of the system is maintained through a global and dynamic gantt chart holding all the allocated activities in the system.

Contract-net based negotiation was recently explored in [17] where product agents take the role of negotiating all the pending jobs on a failing station. The approach was tested under simulation using different dispatching rules and assuming station breakdowns in combination with the agent framework. The authors have found that the inclusion of the transportation costs in rescheduling has a beneficial impact in the improvement of the system throughput and that conveyor failures more significantly affect the overall performance than station failures.

More bio-inspired approaches include ADACOR [25] which proposes a pheromone spread mechanism that regulates the autonomy of the lower order holons. Under disturbances the increased autonomy of these holons compels them to reject the scheduling proposals of the higher order holons. As the system self-organizes towards recovering from the disturbance order is progressively restored by controlling the pheromone flow and global scheduling and control resumes.

In [39] faster than real time simulation is considered to provide continuous adjustments to the physical system. The main concept is to maintain a model of the system behaving at an accelerated pace, rendering it possible to predict the future behavior of the real system and test corrective actions.

Also, in [36] the delegate MAS pattern is presented in articulation with the PROSA [23] architecture. The main principle is to transfer the responsibility of certain tasks to a swarm of lightweight agents that collectively can solve the problem in an optimized way. Each agent in the PROSA architecture then relies in different types of delegation to advertise, search and commit to the utilization of the available resources. This infrastructure has been recently extended to integrate a higher order scheduler that attempts to steer the system towards globally optimal solutions [37].

A similar solution was recently discussed in [4, 18]. The supporting reference architecture combines a high level scheduler that provides an optimal solution for the scheduling problem with local reactive scheduling based in potential fields. The main concept is to balance global/optimal decision making. Under normal circumstances the intelligent products follow strictly the scheduling generated by the global scheduling and use the potential field's information for routing. When a disruption occurs the products manage their own resource allocation and routing using the potential fields alone.

2.2. Integrated discussion

There is an unquestionable trend towards the adoption of reference architectures that integrate several aspects of shop floor management and control. Recent approaches are converging to two agent/holon organizational models: heterarchical and hybrid.

The main advantage of heterarchical models is that they tend to remove centralized points of failure as the control is normally distributed between agents in a peer relation. On the downside, these models tend to be, by design, sensitive to decision myopia.

Hybrid models, on the other hand, attempt to combine a global view of the system that potentially eliminates myopia related problems and rely in a control mechanism that regulates the autonomy of the lower order components. More autonomy is granted when the higher order scheduler is not able to decide and the autonomy is progressively reduced when global decisions can be taken.

Although hybrid models are extremely appealing the literature corroborates the idea that often taking an optimal decision is still time consuming and only feasible in somehow stable systems. In fact, hybrid approaches may suffer from the same constraints as the traditional scheduling approaches since they still attempt a globally optimal scheduling. One can argue in the direction that heuristic and meta-heuristic models can be used to reduce the global search

space. However, the more the solutions produced by the global scheduler, in the hybrid architectures, deviate from optimality, the closer the hybrid approaches are to the heterarchical model and the decision myopia issues. In fact, a hybrid model that spends most of its time using the reactive scheduling components behaves effectively as a heterarchical system.

The second difficulty with the hybrid approaches is regulating the switch between global and local control and establishing the generic conditions under which it can occur in a wide range of systems.

This is not to say that the heterarchical architectures are exempt of problems. When reactive scheduling/routing is considered timely information exchange is the key to keep the consistency of the system. It is hence difficult to find an ideal solution for the information exchange timing.

In either case the regulation of the nervousness of the system is a fundamental point to obtain consistent results.

In comparison to the traditional scheduling algorithms both the hybrid and heterarchical approaches have shown the ability to properly handle systems under dynamic conditions. The literature has widely explored the behavior of such models under machine failures and breakdowns and several benchmarks and test-beds have been documented[40].

One of the features that has been so far elusively explored is the behavior of these emerging systems under topological changes (i.e., when new stations and transport elements are introduced and/or removed in runtime).

The obvious mechanism inducing topological changes are reconfiguration actions. So far only a very small set of systems as made use of this type of changes since the technology required to properly support them is not sufficiently matured yet. Some of the challenges that need to be

overcome are directly related with the present work and the associated literature. The ability to swiftly change a system's topology has however an important influence in the way future production systems can be designed and managed. It supports the use of shared resources in different areas of a specific system and the rapid integration of new equipment to support temporary production surges. A considerable number of systems in ran in shifts. This entails that in some cases part of the equipment is stopped even if a considerable number of resources could be reused in the running processes.

This ability is of high interest for modular equipment providers and has motivated recent research efforts (see as an example the IDEAS project [41]). Rapid integration also impacts commissioning directly as the very same mechanisms that support plug and produce in a generic way also support the creation of hybrid systems where simulation is integrated with the real system. Therefore it is possible to assess in a very tangible way the impact of change in the current system and swiftly change from the simulated environment to the new system implementation.

Overall, the ability to perform quick reconfiguration is an enabler of more efficient ways of running and managing a system.

Aside the straightforward reconfigurability aspects a system's topology is also affected by faults. Failing components can prevent certain part of the system from work. Traditionally this would lead to an overall blockage. The ability to react to this kind of disturbances by dynamically managing the orders already on the system is relevant in today's systems regardless if they would be frequently reconfigured or not.

These sort of runtime system changes normally have a dramatic impact on the system as they potentially create the possibility of establishing new routes on the system that can influence the performance indicators.

In the next section the authors propose an architecture that specifically targets system under such conditions.

3. Reference Agent Architecture

3.1. Reference Agent Architecture – Model and Agent Interactions

The agent architecture presented in this section is derived from the IDEAS reference architecture [41, 42]. It shares with this architecture some concepts and their implementation.

Among them concepts are:

- the notion of skill [43] as the basic representation of the stations' functionalities;
- the deployment agent [44] as a software construct that has the ability to handle agent serialization, deserialization and the subsequent deployment across compatible controllers;
- and the product agent as the top level entity that takes decisions on the locations and associated costs where its process plan is to be executed.

The deployment agent is a pure technological construct. It is used in the agent deployment procedure and in the automation of the tests.

Although some names of the transport related agents are similar its internal behavior and interaction dynamics have been substantially modified in respect to the work reported in [45].

In this context, the proposed architecture focuses entirely on the interactions between the product agents and the transport elements. In respect to the work detailed in [45] the present architecture is more generically defined, featuring no specific technological couplings, and hence is able to tackle a wider family of systems.

The link with the IDEAS architecture is worth a mention since the present implementation has retained the generic technological elements that have been demonstrated under the IDEAS

industrial test cases and that render it applicable in real world scenarios and not only in simulation.

The proposed agent architecture (Figure 1) is supported by a heterarchical agent-based model composed of four main agents: Transport Entity Agent (TEA), Routing Entity Agent (REA), Skill Management Entity Agent (SMEA) and the Product Agent (PA).

All these agents are formally specializations of the Transport Element Agent abstract class that encapsulates all the generic variables and functions related in the management of products in runtime.

The architecture is additionally constituted by three interfaces and one abstract class that enable:

1. the development of customized transport cost metric algorithms (Cost Metric Algorithm Interface (CMAI));
2. the development of customized path computation algorithms (Path Computation Algorithm Interface (PCAI));
3. the generic interconnections of the agents with the physical world and subsequent control in the form of the Low level Control Integration Interface (LLCII) and the Low level Control Integration Library Interface (LLCILI)(partially inherited from the the IDEAS architecture following the principles described in [43, 44]).

Figure 1 – Reference Architecture

The main interactions are restricted to three pairs of agents. The first set of interactions occurs between the PAs and the REAs and their specializations, between the REAs and the TEAs and between the REAs

The PAs are the top level decision makers in the architecture and they interact with the REAs in order to obtain the transport cost associated with their displacement to a location where the next step of their process plan will execute. The stepwise negotiation concerning each skill on the PA's process plan is an architectural design decision that seeks to limit the allocation of a high number of resources. This allows the transport elements more flexibility in re-routing when changes are introduced in the system. It also introduces some decision myopia since the stepwise negotiation does not necessarily drive the PA through the optimal path for its entire process plan. However, the present architecture compromises between PA level decision myopia and the self-organizing effect of transport elements when handling PAs with weaker

allocation commitments to the available resources. The PAs keep track of their current location by memorizing the last transport element they have interacted with. The transport elements keep track of all the PAs in the system.

The entire transport procedure is transparent to the PA. In this context, the PA interacts with the routing elements and their specialized classes only when: entering the system, choosing the next execution location or reacting to the failure of the SMEA where they attempt to execute.

Figure 2 provides an overview of the main interactions between the agents when disturbances do not occur.

Figure 2 – Main interactions without disturbances

In the portrayed scenario the PA is entering the system. Hence, it interacts with the REA that will trigger the transport to its first execution location. The communication starts through the request of the transport cost to all the locations in the system where *Skill x* can be executed. The REA replies with the corresponding list. If the skill is not available the PA is not allowed to

enter the system or, if it is already in the system, it is routed to the destination where it can be removed.

The PA will subsequently evaluate the costs and choose one of the locations. It will normally choose the lowest cost solution unless its process plan prescribed the execution of *Skill x* in a specific station. Once a destination is selected the PA requires the corresponding transport to the REA.

The REA will then forward the PA to the TEA listed in its routing tables as the one conducting to the destination along the shortest path. This procedure is used to progressively process the PA until it reaches its destination.

When the PA arrives at its destination, the receiving SMEA, which is a specialization of the REA, recognizes that the PA wishes to use the resources therein and issues a confirmation that the PA has arrived at the required destination.

The PA will subsequently handle the execution, mediated by SMEA, and upon conclusion will retrigger the procedure of selecting the next execution location.

If the final destination of a PA disappears from the system the transport procedure must be interrupted (Figure 3).

Figure 3 – Main interactions with disturbances.

If during the transport process a REA deems the target destination of a PA unreachable it temporarily stops the routing procedure and messages the PA with a set of possible alternatives for the execution of *Skill x*. In the figure above, a station, which by inheritance is also a REA, receives new link state information that reveals that the desired resource is no longer present. The PA will then reassess its process plan and decide if it can take one of the alternatives or if it should be removed from the system. If the cost of the potentially alternatives is the same then the PA chooses a random location. This is particularly important during the setup stage of the system where several stations may be available at the same cost and the load must be balanced.

The interactions between the REAs are to a certain extent governed by the TEAs. In effect, each TEA continuously computes its transport cost (**+computeCost(capacity : int, contents : Item [0..*], timeStamp : long) : long**) using the user defined class that implements the CMAI. The CMAI also features a function (**+updateCosts() : boolean**) that instructs the TEA if it should send its cost update to the associated REA. Only the REA that precedes a TEA receives its

traversing cost information. In this context, it is possible to control the nervousness of the overall system by regulating the output of the **updateCosts** function. The interaction between the TEA and the REA is therefore a simple request-reply communication triggered by the TEA.

The same is valid for the interactions between REAs. Whenever a REA receives information from a TEA it recomputes the routing tables using the function **(+updateRoutes(systemInfo : hashMap) : treeMap)** in the user defined class that implements the PCAI and forwards the neighbourhood changes to all the other REAs in the system. These generic interactions and basic behaviours constitute the generic core of the architecture that render it applicable to a wide range of systems. However, to extract concrete performance figures and evaluate the architecture the user defined algorithms need to be instantiated. Information about recently plugged resources and the associated skills are broadcasted everytime a station is plugged.

3.2. On the instantiation of the of the user defined algorithms

Since the user defined algorithms make use of the internal variables of the different transport elements it is worth detailing their main purpose.

These variables include the maximum capacity of the transport element **(-capacity : int)**, a flag representing the filling level **(-full : boolean)**, a identifier detailing the next item to be dispatched by the transport element **(-nextToDispatch : Item)**, a flag stating the status of the item being dispatched **(-dispatched : boolean)** and the destination of that item **(-nextDestination : Transport Element Agent)**, the current contents of the transport element **(-contents : Item [0..*])**, a flag stating the preparedness of the next item to be despatched **(-readyToDispatch : boolean)**, the set of neighbouring transport elements that are able to dispatch items to the current transport element **(-inputNeighs : Transport Element Agent [0..*])** and the set of neighbouring transport elements to where the current transport element can dispatch items to **(-outputNeighs : Transport Element Agent [0..*])**, a internal behaviour that aggregates all the monitoring and control behaviours of the agent **(-controlLoop :**

Behaviour) which is overridden by the specialized classes and finally a flag that tracks the status of the hardware execution commanded by the transport element (**-executionStatus : boolean**).

The transport element class also features a set of abstract functions that handle: dispatching (**+dispatchItem(item : Item)**) and reception of items (**+acceptItem(item : Item)**) and the runtime management of the neighbours (**+addInputNeigh(neigh : Transport Element Agent)**, **+addOutputNeigh(neigh : Transport Element Agent)**, **+removeInputNeigh(neigh : Transport Element Agent)**, **+removeOutputNeigh(neigh : Transport Element Agent)**).

As mentioned before the CMAI is a central algorithm in controlling the nervousness of the system. In the present context, the cost computation function (**+computeCost(capacity : int, contents : Item [0..*], timeStamp : long) : long**) is calculated as follows:

$$Cost = \begin{cases} \frac{\sum TT_i}{N} & \text{if at least one carrier exceeds the MTT} \\ MTT & \text{if 0 carriers exceed the MTT} \end{cases}$$

where TT_i is the transport time of a carrier i whose transport time exceeds the minimum transport time MTT and N is the total number of carriers exceeding the MTT value.

The MTT is computed as follows:

The minimum transport time (MTT) is computed taking into consideration the TEA characteristics. In the test case later described where all the TEAs are conveyor belts the minimum transport time is computed as:

$$MTT = \frac{Capacity}{Conveyor\ speed}$$

The time based metric produces an integer value whose sum across the different links provides each PA with an accurate estimate of the transport costs.

The cost updating function (**+updateCosts()** : **boolean**) instructs the TEA if it should propagate its information to the associated REA. This function directly controls the reactivity of the system and its proper design contributes to improve the overall stability.

In the present context the update is regulated in an event driven fashion.

The cost is updated if the last computed cost is the double of the previous cost or if it has been reduced to half:

$$update \begin{cases} true & if \frac{NewCost \times 100}{PreviousCost} > UpperThreshold \cup \frac{NewCost \times 100}{PreviousCost} < LowerThreshold \\ false & otherwise \end{cases}$$

In the present case the value of the upper threshold is 200 and the value of the lower threshold is 50 to promote the stability of the system unless significant events impacting the processing time at the SMEAs or the routing time at the REAs occur.

The architecture also supports the association of different CMAIs to different instances of the TEAs. This is particularly important in systems where different forms of transport are considered for instance: AGVs, conveyor belts, pick and place units, etc.

While the previous computations address the behavior of the system when its topology is stable and trigger changes in the routing tables through the PCAI. The PCAI is also responsible for tracking and reacting to topology changes. In particular, it reacts to the invocations of the functions adding and removing neighbours. The topological updates follow the link state algorithm. This means that REAs and its specialized classes are modeled as nodes and TEAs are considered links in a directed graph $DG(REAs, TEAs)$. Each REA has a set of output neighbours (**-outputNeighs : Transport Element Agent [0..*]**) to where it can dispatch items/PAs following their CMAIs. These neighbours constantly compute their cost. When cost changes are sent to the REA it broadcasts these changes to all the other REAs in the environment as detailed

before. The link information is structured in the form of a table featuring the link and the transport cost leading to SMEA hosting a set of skills (Table 1)

Table 1 – Link state information example

Link/Destination X	Transport Cost
TEA1/SMEA1	50
TEA2/SMEA2	10

Each REA uses this information to infer the entire network. Once the REA has a graph of the network it uses the Dijkstra algorithm to compute its routing table. The Dijkstra algorithm is used since it ensures the discovery of the shortest path if one exists. The algorithm assumes that the cost associated with each link is a positive integer, which is the case with the time based metric, and outperforms other more generic algorithms that would relax this constraint. In particular, since at the moment there is no preferred heuristic to guide the selection of the next node to be explored there would be any advantage of using the A* algorithm whose behavior would revert to the Dijkstra algorithm when the heuristic component is 0.

The REA delegates the computation of the shortest path algorithm on the user defined code and checks the availability of the results through the `computationReady` function (**+computationReady()** : **Boolean**) retrieving the results using the `getRoutes` function (**+getRoutes()** : **treeMap**)

The REA maintains a table that associates the available skills with the different execution locations. In this context, the determination of all the locations where a skill can be executed results from crossing information from the routing table and the SMEA-to-skill association. Finally, the Low level Control Integration Library Interface ensures that all the transport elements are able to interact with physical or simulated devices maintaining the same skill description in accordance with the strategy detailed in [44].

The proposed algorithms, although generic, account for the particularization of the self-organizing response of the proposed architecture. They are by no means unique and, as detailed, the architecture can support a multitude of possible combinations of the different algorithms. It is the authors' opinion that the one selected positively contributes to attaining the objectives of the present paper.

3.3. Comparison with similar approaches

The proposed architecture is similar in objectives and purpose to the works detailed in [4, 5, 17, 25, 32, 33, 36, 37]. However it is rather different in respect to the approach taken. In this context, [4, 25, 32, 33, 37] promote hybridized approaches that feature the interaction between the agent system and a global scheduler while the present architecture promotes a more heterarchical model. As discussed, the purpose of doing so is to avoid the time consuming effort in generating global schedules which may prove to be unfeasible in systems prone to frequent or more dramatic topological changes.

The proposed work is therefore architecturally closer to [3, 5, 17]. In respect to [17] the proposed architecture does not feature a contract net based negotiation on the reallocation of orders when failures occur in runtime. It rather uses a simpler and faster request reply protocol since all the decision alternatives are inherently computed by the shortest path algorithm. Hence, the negotiation procedure is replaced by a choice procedure which tends to reduce the complexity of the decision process. In the same line, [5] focus on the interactions between orders and resources and in the distributed and negotiated construction of a global schedule. In the present approach the system does not seek the construction of a global schedule but rather attempts to handle the orders already in the system as efficiently as possible and in a more self-organizing way. A stronger similarity is found with the work reported in [3] that also proposes a link state algorithm for automated material handling systems. However, [3] constraints the link state information dissemination to specific zones

while in the proposed approach it was chosen to broadcast this information compromising between the number of messages exchanged and the speed at which each node/REA is able to update its routing tables. Further, and as detailed before, it is possible to directly control the nervousness of the system and subsequently the frequency of information exchange. The proposed approach also has some contact points with [36] in the direction that it uses delegation. However the behavioral mechanisms that are used in the delegation and in the articulation with the agent platform are fundamentally different.

Another distinguishing point of the present architecture is the explicit support for plug and produce which is also reflected in the testing and validation methodology. Most of the literature so far has focused on evaluating the system behavior under faults and the capacity of the system in reacting to different fault scenarios. The present architecture has an additional focus in allowing the system to handle runtime topological changes namely the introduction and removal of stations and elements of the transport system.

4. Main implementation details and test cases

The presented architecture is fundamentally platform agnostic. Its main technological requirements entail an object oriented language supporting the implementation of the architecture, already supported in standard automation languages (IEC 61131), and a networked environment connecting the several components.

For practicality reasons the architecture and the test case were implemented in JAVA and the agents are supported by the JADE platform. The communication follows a direct interaction message exchange pattern without negotiation using the FIPA Request Protocol.

The main control loop in the agents is implemented using JADE's cyclic behavior to support the intensive nature of the computations and improve the reactivity of the system.

The validation and testing scenario is supported by a customized simulation environment that simulates the displacement of items in a conveyor based system whereby several stations can be connected. The main elements of the simulation environment include: conveyor belts, entry points, exits points, stations and routing gates. These elements handle carriers. All the active elements in the simulated elements have a size and a speed defined in generic time and distance units. The stations have an additional processing time and the carriers only have one size.

The simulation is geometrically correct in the sense that it accounts for the precise positioning of all the carriers in each simulated time step. In this sense, it accurately simulates collisions, accumulating effects and control faults.

External systems interact with the simulated components through a front end interface that grants individual access to each component and to the control of their behavior in the simulation environment.

The mappings between the simulation environment and the agent environment are detailed in **Figure 4**.

Figure 4 – Mapping between the multiagent and the simulation environments.

As detailed in Figure 4 all the generic agent behaviors are retained by the architectural constructs and the interaction details are provided by the LLCILI. The LLCILI controls the simulation interfaces of the different simulated components. The simulation interfaces emulate the control of the components in particular:

- **Station** – supports the actuation and sensing of a fixture that locks the carriers in place for execution, actuation and verification of the execution of the station process (emulated as a timespan).
- **Conveyor** – supports the retention and release of carriers using a locking mechanism at the head of the conveyor.
- **Gate** – supports the retention of carriers and the re-routing to a selected simulated component directly connected to the gate.
- **Entry and Exit points**- support the retention and release of carriers using a locking mechanism at the head of the entry point. The exit point controls the removal and buffering of the carriers.

All the simulated components support the addition and removal of neighbors in runtime. If these runtime changes generate errors such as jams or the disappearance of pallets these are detected and reported. It is the responsibility of the agent environment to ensure the consistency of the simulated system which does not impose any control restrictions.

Although the simulation can operate in a time scale that is faster than real time it has been tuned to operate on the limit of the decision making capabilities of the agent implementation in JADE. This tradeoff enables a fine granularity in the time and distance units and the subsequent increment in the precision of the carrier movements in the simulation environment.

The simulation tool also allows the definition of the precise time instants when the agents and simulation components will be deployed and acts as a front-end that manages and automates both the simulation and the agent environment.

The test case considered is detailed in Figure 5. It features 7 stations with two pairs of stations, station 1/station 2 and station3/station4, with repeating skills A and B and stations 5, 6, and 7 with skills C, D and E respectively. The execution time in each station is 2000 time units.

The system also includes 6 gates that can route carriers in the indicated directions, one entry point and two exit points. Each gate is able to store one carrier at the time. In this context, the maximum number of carriers that can be associated with a gate is two, when one carrier is leaving the gate and a new one is entering it.

These elements are connected by a conveyor network. The capacity of each conveyor, in number of pallets, is detailed in Figure 5, in parenthesis, next to corresponding conveyor name. The effective size of each conveyor is computed by multiplying its capacity by the size of each carrier that in the present case is 20000 distance units. The speed of all the transport elements is 50 distance units/time units.

Figure 5 – Graph of the test case

The Simulation tool associates each PA with a carrier and the agent controls the release of carriers/PAs in the entry point.

Three products types are considered P1, P2 and P3. All the P1's process plan includes visiting stations A, B and C, P2's process plan is A, D, E and finally P3's features A, B, A.

The PAs (P1 and P2) are randomly introduced in the system with equal probably until the time instant 40000. Precisely at this time instant PAs of type P3 are introduced with twice as much probably as P1 or P2.

These conditions attempt to simulate a surge of unexpected orders in the system.

Three different test cases are therefore considered.

In the first test case (TC1) the system runs without ST4, the Exit Point 2 and the dashed conveyors. This establishes the worst case scenario for the system in Figure 5 since it will increase the load after gate 5 which is the beginning of the exit route for all problems.

In the second test case (TC2) the system starts with all the components available in the system. This is theoretically the most favorable case since the load after gate 5 is reduced and PAs requiring B have an increase range of solutions. Also P2s benefit from a shortest production path.

The final test case (TC3) features the plugging of ST4, Exit point 2 and the associated conveyors precisely at the same time instance at which the production surge of P3's occur. This test case should feature a performance that is in between the TC1 and TC2 and demonstrate the capacity of the architecture to react to topological changes.

5. Results and Discussion

Given the random nature of the PAs' introduction 10 trials of each test case are considered. Table 2 details the results obtained for TC1. The best results are fairly consistent across the different rounds. They are produced by the first set of PAs entering the system that does not encounter any bottlenecks.

As the system starts to accumulate PAs, the REAs begin exploring alternative solutions for the scheduling/routing of the parts in the system. At this point in time the random nature of the PAs introduced contributes significantly to the variation of the results across the different simulation rounds.

In particular, in the less favorable cases, the REA's compensate by using more routes some of which are only temporary and are used by a reduced number of PAs. These decisions tend to contribute to the worst cases since a few PAs will have to travel through less favorable paths. This also subsequently creates more variation in the results.

Table 2 – TC1 Results for 10 simulation runs (sorted by average makespan)

Sample	Average Makespan	Standard Deviation	Unique Routes	Best Case	Worst Case
1	101279,5	36335,6	17	40830	196012
2	111856,7	49126,3	17	49572	201142

3	115894,6	62640,8	18	41913	238347
4	142609,9	55917,4	25	55877	277767
5	154310,4	78751,1	24	54937	282553
6	240033,3	166104,6	24	46575	541103
7	263909,4	215691,5	27	42882	721634
8	308011,6	157150,7	29	61115	654323
9	338357,6	202456,8	26	46620	707187
10	427524,2	360745,5	26	46919	967005

If the previous table shows the results for, predictably, the worst case, Table 3 details the results for the best case. These are consistent with the previous results in respect to the best and worst cases and in respect to the number of routes considered. As anticipated there is a substantial reduction in the average makespan across all simulation runs. The increased performance is directly linked to the more favorable layout that features two exit points close to the main stations involved in producing PAs of type P1 and P2.

Table 3 – TC2 Results for 10 simulation runs (sorted by average makespan)

Sample	Average Makespan	Standard Deviation	Unique Routes	Best Case	Worst Case
1	72474,5	126872,1	20	37821	126872
2	82783,9	28487,6	18	38335	137727
3	90251,5	28519,4	19	39432	143569
4	95648,5	37654,5	17	40710	171298
5	100034,7	47331,3	24	46780	184988
6	147528,2	55554,7	23	42604	259261
7	158196,7	115913,5	21	39176	385989
8	173043,1	88790,4	20	43715	364898
9	191054,4	112141,3	26	46664	421886
10	193780,8	107220,6	25	43347	397292

A similar behavior can be observed for the TC3 results (Table 4). The interesting aspect to retain is that the system that suffers the plug and produce of components features a performance that lays in between the best (TC2) and the worst case (TC1). This is a clear indication that the agents were able to quickly adapt and explore the new topology of the system. This is also reflected in the fact that, on average, the REAs are using more routes. That

is directly related with the adaptation of the system and the transient behavior during the plug and produce process.

Table 4 – TC3 Results for 10 simulation runs (sorted by average makespan)

Sample	Average Makespan	Standard Deviation	Unique Routes	Best Case	Worst Case
1	105098,2	47690,3	19	47621	239138
2	117185,2	53180,6	25	43911	228529
3	118330,6	45845,5	26	46376	211500
4	128007,8	64344,2	19	43281	279132
5	156468,6	90373,1	26	37428	330073
6	163596,4	73828,4	27	52592	319048
7	183627,7	124868,9	24	40720	435942
8	187288,7	98236,3	28	41099	371578
9	238972,6	144084,4	28	57287	520243
10	239248,5	130568,7	30	51167	463007

Across the three test cases two main factors were identified that cause the system to accumulate PAs:

1. Less favorable introduction of PAs (whose type is randomly generated).
2. Decision times at REA level under heavy load.

The first factor can be clearly observed in Figure 6 and Figure 7.

Figure 6 – Number of Products on the conveyors around gate 1 (most favourable case in TC1)

Figure 7 - Number of Products on the conveyors around gate 1 (less favourable case in TC1)

The random nature of the orders has the potential for creating jamming problems. In particular, it can be seen that in the most favourable case (Figure 6) conveyors 1, 2 and 7 have only reached their maximum capacity a few times. This means that the alignment of the orders was inherently adequate for the layout considered. The system had to work less to keep the parts flowing and this is also supported by the fact that fewer routes were used.

The previous behavior opposes the one verified in the worst case (Figure 7) that shows the conveyors reaching their maximum capacity several times. This is an indication that products were getting periodically blocked in conveyors 1, 2 and 7. These blockages are likely to happen in TC1 when there is a surge of Products requiring stations ST1 and ST2 and the system starts recirculating parts to keep them moving as can be seen by the increase in the number of routes used.

The current implementation strongly promotes the recirculation of products to minimize the blockages. However, this also means that a non-negligible number of products will spend more time on the system. This, in turn, contributes to the relatively high standard deviation. The tests show that the deviation tends to increase with the number of routes being explored by the system. Some of these routes are only used by one product agent.

The first factor is an indication that the present architecture should be complemented by a rough pre-scheduling mechanism that takes into consideration the capacity of the system. This complementary system does not necessarily need to seek global optimization but rather consider a pre-selection of the orders that should enter the shop-floor production. The time based metric provides an important indication of the system status that can support these pre-selections.

It is also important to stress that the results may vary in systems with different topology. The proposed architecture does not replace the good practices of system layout design and it obviously denotes better performance in well-designed systems.

The second factor is directly related with the current technological limitations. When the system is under heavy load more information needs to be processed in order to ensure the best routes. The use of JAVA related technologies is particularly penalizing due to the interference of the garbage collecting mechanism. These architecture independent aspects can eventually be mitigated by improving the system implementation and possibly considering a distinct supporting technology.

6. Conclusions and Future Research Directions

The present paper introduced a heterarchical architecture featuring local scheduling/routing of the order already under production. The rational for adopting a more heterarchical model is avoiding the long decision times of hybrid and traditional approaches in systems that are prone to frequent changes.

The results suggest that the proposed architecture can cope well with systems under extreme dynamic conditions such as runtime topological changes.

The results also show that the architecture and the particular instantiation considered are sensitive to mechatronic constraints. Some are directly related to the structure of the system while others have a more technological background. It is therefore rather difficult to

benchmark the emerging architectures and the proposed work is not an exception. In this context, the proposed test cases were designed to expose the adaptation capabilities of the proposed architecture.

In particular, the random nature of the orders has, in some testing scenarios, led to less performing responses. As detailed this is an indication that a pre-selection mechanism should be considered.

The development of such a mechanism is currently being considered as a mean to further improve the dynamics of the system. This strategy will set the proposed architecture in a more hybrid format. However in order to avoid the potential pitfalls of these architectures, it is the authors' opinion that global optimization should not be attempted but rather a more agile and adaptive approach should be considered.

References

- [1] W. Shen, L. Wang, Q. Hao, Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey, *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36 (2006) 563-577.
- [2] N. Boysen, M. Flidner, A. Scholl, Assembly line balancing: Which model to use when?, *International Journal of Production Economics*, 111 (2008) 509-528.
- [3] H.Y. Lau, S. Woo, An agent-based dynamic routing strategy for automated material handling systems, *International Journal of Computer Integrated Manufacturing*, 21 (2008) 269-288.
- [4] C. Pach, T. Berger, T. Bonte, D. Trentesaux, ORCA-FMS: a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling, *Computers in Industry*, 65 (2014) 706-720.
- [5] M. Rolón, E. Martínez, Agent-based modeling and simulation of an autonomic manufacturing execution system, *Computers in Industry*, 63 (2012) 53-78.
- [6] D. Lin, C.K.M. Lee, W. Ho, Multi-level genetic algorithm for the resource-constrained re-entrant scheduling problem in the flow shop, *Engineering Applications of Artificial Intelligence*, 26 (2013) 1282-1290.
- [7] G.A. Rovithakis, S.E. Perrakis, M.A. Christodoulou, Application of a neural-network scheduler on a real manufacturing system, *Control Systems Technology, IEEE Transactions on*, 9 (2001) 261-270.
- [8] C. Blum, M. Sampels, An ant colony optimization algorithm for shop scheduling problems, *Journal of Mathematical Modelling and Algorithms*, 3 (2004) 285-308.
- [9] D. Sha, C.-Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, *Computers & Industrial Engineering*, 51 (2006) 791-808.

- [10] R.-M. Chen, S.-T. Lo, C.-L. Wu, T.-H. Lin, An effective ant colony optimization-based algorithm for flow shop scheduling, *Soft Computing in Industrial Applications*, 2008. SMCia'08. IEEE Conference on, IEEE, 2008, pp. 101-106.
- [11] A. Tharumarajah, A self-organizing model for scheduling distributed autonomous manufacturing systems, *Cybernetics & Systems*, 29 (1998) 461-480.
- [12] K. Tamani, R. Boukezzoula, G. Habchi, Application of a continuous supervisory fuzzy control on a discrete scheduling of manufacturing systems, *Engineering Applications of Artificial Intelligence*, 24 (2011) 1162-1173.
- [13] F. Zhao, A. Zhu, Z. Ren, Y. Yang, Integration of process planning and production scheduling based on a hybrid PSO and SA algorithm, *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, IEEE, 2006, pp. 2290-2295.
- [14] K. Kianfar, S.M.T. Fatemi Ghomi, A. Oroojlooy Jadid, Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA, *Engineering Applications of Artificial Intelligence*, 25 (2012) 494-506.
- [15] R. Zhang, C. Wu, An effective immune particle swarm optimization algorithm for scheduling job shops, *Industrial Electronics and Applications*, 2008. ICIEA 2008. 3rd IEEE Conference on, IEEE, 2008, pp. 758-763.
- [16] S.W.A. Haneyah, P.C. Schuur, J.M.J. Schutten, W.H.M. Zijm, A generic material flow control model applied in two industrial sectors, *Computers in Industry*, 64 (2013) 663-677.
- [17] M. Merdan, T. Moser, W. Sunindyo, S. Biffi, P. Vrba, Workflow scheduling using multi-agent systems in a dynamically changing environment, *Journal of Simulation*, 7 (2013) 144-158.
- [18] C. Pach, T. Berger, Y. Sallez, T. Bonte, E. Adam, D. Trentesaux, Reactive and energy-aware scheduling of flexible manufacturing systems using potential fields, *Computers in Industry*, 65 (2014) 434-448.
- [19] F.-S. Hsieh, C.Y. Chiang, Collaborative composition of processes in holonic manufacturing systems, *Computers in Industry*, 62 (2011) 51-64.
- [20] C. Rösiö, K. Säfsen, Reconfigurable production system design—theoretical and practical challenges, *Journal of Manufacturing Technology Management*, 24 (2013) 998-1018.
- [21] M.G. Mehrabi, A.G. Ulsoy, Y. Koren, Reconfigurable manufacturing systems: key to future manufacturing, *Journal of Intelligent Manufacturing*, 11 (2000) 403-419.
- [22] L. Monostori, J. Váncza, S.R. Kumara, Agent-based systems for manufacturing, *CIRP Annals-Manufacturing Technology*, 55 (2006) 697-720.
- [23] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, *Computers in industry*, 37 (1998) 255-274.
- [24] J.-L. Chirn, D.C. McFarlane, A holonic component-based approach to reconfigurable manufacturing control architecture, *Database and Expert Systems Applications*, 2000. Proceedings. 11th International Workshop on, IEEE, 2000, pp. 219-223.
- [25] P. Leitão, F. Restivo, ADACOR: A holonic architecture for agile and adaptive manufacturing control, *Computers in industry*, 57 (2006) 121-130.
- [26] J. Barata, L.M. Camarinha-Matos, Coalitions of manufacturing components for shop floor agility-the CoBASA architecture, *International journal of networking and virtual organisations*, 2 (2003) 50-77.
- [27] P. Vrba, P. Tichy, V. Mařík, K.H. Hall, R.J. Staron, F.P. Maturana, P. Kadera, Rockwell Automation's Holonic and Multiagent Control Systems Compendium, *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, IEEE Transactions on, 41 (2011) 14-30.
- [28] L. Ribeiro, J. Barata, M. Onori, C. Hanisch, J. Hoos, R. Rosa, Self-organization in Automation - The IDEAS Pre-Demonstrator, 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), IEEE, Melbourne, Australia, 2011.
- [29] D.C. McFarlane, S. Bussmann, Holonic manufacturing control: Rationales, developments and open issues, *Agent-based manufacturing*, Springer2003, pp. 303-326.
- [30] K. Ueda, A concept for bionic manufacturing systems based on DNA-type information, *Proceedings of the IFIP TC5/WG5. 3 Eight International PROLAMAT Conference on Human*

Aspects in Computer Integrated Manufacturing, North-Holland Publishing Co., 1992, pp. 853-863.

[31] L. Ribeiro, J. Barata, G. Cândido, M. Onori, Evolvable production systems: an integrated view on recent developments, Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology, Springer, 2010, pp. 841-854.

[32] S.W.A. Haneyah, J.M.J. Schutten, P.C. Schuur, W.H.M. Zijm, Generic planning and control of automated material handling systems: Practical requirements versus existing theory, Computers in Industry, 64 (2013) 177-190.

[33] R. Babiceanu, F. Chen*, R. Sturges, Framework for the control of automated material-handling systems using the holonic manufacturing approach, International Journal of Production Research, 42 (2004) 3551-3564.

[34] R.F. Babiceanu, F.F. Chen, Distributed and centralised material handling scheduling: Comparison and results of a simulation study, Robotics and Computer-Integrated Manufacturing, 25 (2009) 441-448.

[35] D. Trentesaux, Distributed control of production systems, Engineering Applications of Artificial Intelligence, 22 (2009) 971-978.

[36] P. Verstraete, B. Saint Germain, P. Valckenaers, H. Van Brussel, J. Belle, H. Hadeli, Engineering manufacturing control systems using PROSA and delegate MAS, International Journal of Agent-Oriented Software Engineering, 2 (2008) 62-89.

[37] J.M. Novas, J. Van Belle, B. Saint Germain, P. Valckenaers, A Collaborative Framework between a Scheduling System and a Holonic Manufacturing Execution System, Service Orientation in Holonic and Multi Agent Manufacturing and Robotics, Springer 2013, pp. 3-17.

[38] P. Leitão, J. Barbosa, D. Trentesaux, Bio-inspired multi-agent systems for reconfigurable manufacturing systems, Engineering Applications of Artificial Intelligence, 25 (2012) 934-944.

[39] Y. Sallez, T. Berger, D. Trentesaux, A stigmergic approach for dynamic routing of active products in FMS, Computers in Industry, 60 (2009) 204-216.

[40] S. Schreiber, A. Fay, Requirements for the benchmarking of decentralized manufacturing control systems, Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, IEEE, 2011, pp. 1-8.

[41] M. Onori, N. Lohse, J. Barata, C. Hanisch, The IDEAS project: plug & produce at shop-floor level, Assembly Automation, 32 (2012) 124-134.

[42] IDEAS-consortium, Instantly Deployable Evolvable Assembly Systems, <http://www.ideas-project.eu/>, 2010.

[43] L. Ribeiro, R. Rosa, A. Cavalcante, J. Barata, IADE – IDEAS Agent Development Environment: Lessons Learned and Research Directions, CIRP Conference On Assembly Technologies And Systems (CATS 12) Ann Arbor, 2012.

[44] L. Ribeiro, J. Barata, Deployment of Multiagent Mechatronic Systems, Industrial Applications of Holonic and Multi-Agent Systems, Springer Berlin Heidelberg 2013, pp. 71-82.

[45] L. Ribeiro, A. Rocha, J. Barata, A product handling technical architecture for multiagent-based mechatronic systems, IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society, IEEE, 2012, pp. 4342-4347.