Computers in Industry 101 (2018) 81–90

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems

Nelson Rodrigues[a,b,c,*], Eugénio Oliveira[b,c], Paulo Leitão[a]

[a] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Braganca, Portugal
[b] Faculty of Engineering - University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[c] LIACC - Artificial Intelligence and Computer Science Laboratory, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

ARTICLE INFO

Keywords:
Service reconfiguration
Multi-agent systems
Service-oriented architectures
Intelligent manufacturing systems

ABSTRACT

Intelligent manufacturing systems rely on the capability to adapt and evolve to face the volatility of dynamic markets. The complexity of these systems increases with the demand of more customized and quality products, which requires more agile and flexible methods to support the dynamic and on-the-fly system reconfiguration aiming to respond quickly to product changes, by offering more efficient services. In this service-oriented manufacturing context, where process functionalities are modelled as services (e.g., quality control, welding and transportation), the dynamic reconfiguration of the services structure (e.g., in terms of quality, processing time and provided features) assumes a critical role to achieve the referred requirements. Despite the current research efforts, the service reconfiguration approaches usually use reactive event triggers, with decisions coming from a centralized decision-maker and performed manually. This means a lack of dynamic and run-time reconfiguration flexibility by discovering opportunities and needs to change, and, thus, exploring possible actions leading to new and appropriate system configurations. To overcome the mentioned issues, it is essential to provide solutions that answer to when and how to reconfigure a manufacturing system in an integrated, automatic and dynamic manner. For this purpose, this paper introduces an agent-based approach for service reconfiguration in manufacturing systems that allows the identification of opportunities in a pro-active and dynamic manner, and the on-the-fly implementation of new configuration solutions leading to a better production efficiency. The experimental results, using a flexible manufacturing system case study, allowed to verify the feasibility and benefits of the proposed agent-based service reconfiguration solution for competitive and collaborative industrial automation scenarios.

1. Introduction

Manufacturing industry is facing strong demands, in terms of products quality, customization and delivery time, imposed by the global market growth, which requires the capability to react rapidly and cost-effectively to condition changes. To address the mentioned issues and aligned with the vision of Industry 4.0 [1], Internet of Things and Internet of Services, established upon Cyber-Physical Systems (CPS) and complemented with other emergent ICT technologies, such as Big data, cloud computing and data analytics, are recognized as crucial to support the deployment of more flexible, robust, responsive, reconfigurable and interoperable systems. In this context, the development of Service-Oriented Architectures (SOA) [2] based solutions, requires the implementation of several features, namely service-discovery, -registration, -composition and -reconfiguration. In particular, the service reconfiguration is crucial to respond to condition changes that take place in unpredictable environments by dynamically adapt and improve the functionalities abstracted by the offered services. Usually, the service reconfiguration is performed to cope with unexpected condition changes, to improve the system competitiveness and to respond to new business strategies. Note that in manufacturing systems, services can be seen as functionalities offered by a device or system, e.g., pick-and-place or welding operations provided by a robot or an inspection operation provided by a quality control station.

Despite the significant research efforts found in literature, the existing service reconfiguration approaches are focusing limited scopes, e.g., components' replacement (to react to the harmful effects or breakdowns) and re-planning (to deal with the modified configurations' requirements) [3], and lack the automatic, dynamic and run-time reconfiguration procedures that enable discovering new reconfiguration

* Corresponding author at: Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Braganca, Portugal.
E-mail addresses: nrodrigues@ipb.pt (N. Rodrigues), eco@fe.up.pt (E. Oliveira), pleitao@ipb.pt (P. Leitão).

https://doi.org/10.1016/j.compind.2018.06.003
Received 31 October 2017; Received in revised form 15 May 2018; Accepted 23 June 2018
0166-3615/ © 2018 Elsevier B.V. All rights reserved.

opportunities and exploring new system configurations. Additionally, the deployed service reconfiguration approaches are usually performed manually and reactively in a centralized manner, usually requiring to stop a running process, diagnose the problem, perform the reconfiguring and restart the system/device. However, the determination of all possible service configurations' solutions in a repeated and consistent manner goes beyond the human capability in an acceptable time. In this sense, reconfiguring the service manually and then restarting the system is not acceptable if we want to comply with the dynamics and requirements of current industrial needs [4]. As a consequence, new service reconfiguration approaches are required to support the truly reconfiguration procedure by being performed automatically, dynamically and on-the-fly. Note that in industrial manufacturing systems, reconfiguration procedures are strongly dependent on physical resource constraints and response time.

Having this in mind, the paper describes a service reconfiguration approach for manufacturing systems that allows the pro-active and dynamic identification of opportunities for reconfiguration and the on-the-fly implementation of the best strategies for the service reconfiguration that will lead to an increase of the production efficiency. For this purpose, multi-agent system (MAS) is used to distribute the system intelligence aiming to run the service re-configuration process in an autonomous, modular and decentralized manner. The proposed MAS approach is enriched with intelligent mechanisms for the early detection of reconfiguration opportunities, e.g., a performance degradation or the introduction of a new product, and the selection of reconfiguration strategies for improving the service quality or updating the catalogue of offered services. The conflicts arising from the dynamic reconfiguration provided by the distributed agents, acting in a selfish mode, are avoided by considering a collaborative mechanism that considers the interests of the individual agents together with the interests of the overall system. This approach was tested in an experimental flexible manufacturing system, and the results show the benefits of this dynamic and pro-active service reconfiguration approach.

The remaining of this paper is organized as follows. Section 2 overviews the service reconfiguration concept and the existing related work and establishes the requirements for a truly dynamic, intelligent and pro-active service reconfiguration process. Section 3 introduces the MAS architecture for the dynamic service reconfiguration approach, particularly describing the "when" and "how" reconfiguration phases. Section 4 describes the process to select the best service reconfiguration alternative from the built space of solutions and introduces a mechanism to evaluate the pertinence of these solutions from a collaborative perspective. Section 5 describes the experimental case study and the implementation details, and analyses the experimental results. Finally, Section 6 rounds up the paper with the conclusions.

2. Related work

The SOA paradigm [5] is based on the concept of software systems offering and consuming services, each one encapsulating the functionalities of a service provider. The use of service-orientation brings significant benefits to design complex systems allowing to face interoperability and loose-coupled abstraction, being currently, amongst other areas, applied to model and design modern manufacturing systems [34], where services can encapsulate physical operations like welding and pick-an-place. In these systems, the concepts of services aggregation, composition, and orchestration are important, and strongly impact the service reconfiguration capability. Note that services can be composed of atomic services (i.e. simple and indivisible services) following a specific workflow.

Basically, service reconfiguration is related to the adaptation of the existing services to deal with unexpected internal or external condition changes, such as failure of a service and loss of the quality of service (QoS) [3]. Although focusing primarily on fault mitigation, service reconfiguration may also consider reconfiguration opportunities that are not initially described but contribute to improve the system performance. In fact, over the time, services can become less competitive, e.g., not being requested by service consumers, requiring the execution of proper actions to improve their competitiveness. To exploit that situation, several types of services reconfiguration can be identified:

- **Improvement of the service's behaviour:** the functionality of the service is improved aiming to increase its performance, e.g., calibrating or switching components; this can be seen as a weak service reconfiguration and only needs an internal perception of the problem.
- **Changing the services' catalogue:** the catalogue of services offered by an entity can be modified to face the service demand through adding new services and/or removing others, e.g., a robot that changes a tool to be able to weld metal sheets; this can be seen as a strong service reconfiguration and needs to have an external perception of the problem.
- **Changing the structure of a composed service:** a composed service can be re-organized by changing the structure and workflow of atomic services (known as service choreography), e.g., adding or removing services to better accommodate the evolution in the available atomic services; this can be seen as a strong service reconfiguration and also needs to have an external perception of the problem.

The use of dynamic service orchestrators can provide these features, particularly to dynamically create workflows when required. Complementary, aiming to execute a dynamic, pro-active and on-the-fly service reconfiguration, considering the referred reconfiguration types, the following requirements need to be observed [6]:

- R1: The opportunity to execute a service reconfiguration must be identified internally (regarding the system), automatically and at run-time.
- R2: The system needs to have the capability to select an alternative reconfiguration solution and perform the reconfiguration on-the-fly, reducing the impact of condition changes (e.g., a performance degradation).
- R3: The service reconfiguration must be performed according to the "nervousness" state of the system, i.e. reacting smoothly or dramatically in such a way that mitigates the problem.
- R4: The service reconfiguration process should be performed in a distributed manner and consider competitive and collaborative scenarios.

In the literature, a significant part of the research in service reconfiguration is devoted to the change in the service composition aiming to compose the best service that meets the client's requirements. This composition can be performed in two distinct moments: design-time and run-time. Most of the works consider that reconfiguration is planned in the design phase, setting up the system to cope with expected changes [7–9]. In contrast, run-time reconfiguration reacts promptly to situations during the system execution to overcome unexpected events that were not considered at design-time, as illustrated by [10–12]. However, the reconfiguration is usually based only on the migration of services that are considered as a reconfiguration action in an automatic manner. The selection of different composition instants, in essence, remains a challenge creating a design-time/run-time trade-off situation [13]. Particular scenarios result in complex computational problems that cannot be efficiently solved in run-time and, in such cases, a design-time policy might be more beneficial in the long run, where time is not a constraint to apply more complex algorithms.

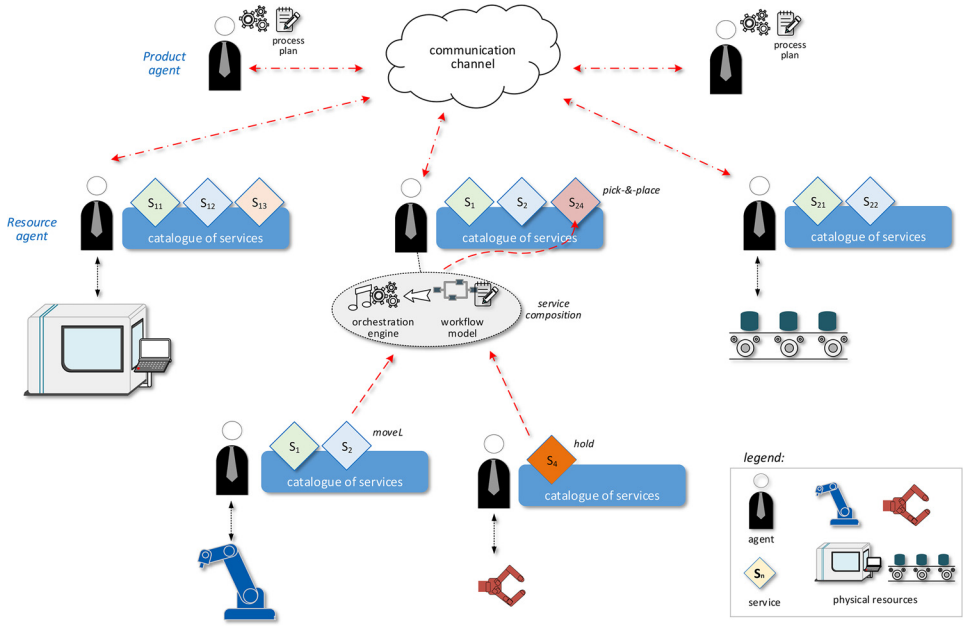


Fig. 1. Multi-agent system approach for the distributed, dynamic and on-the-fly service reconfiguration.

Despite this unresolved challenge, many run-time reconfiguration approaches that operate in an automatic mode are explored, e.g., by [6,8,10,12,14,15]. The remain operation types, such as semi-automatic [11,13,16–19] and manual intervention [7], are sensible to the type of composition, which implies that the semi-automatic asnd manual kind of interventions shall be performed in the design time.

Regardless the composition time, another important action comprehends the selection of alternative solutions that are available on the system. Traditionally, the discovery of appropriate services is performed in a centralized manner, e.g., using the UDDI (Universal Description, Discovery, and Integration) central service registry [5]. Innovative solutions, e.g., based on self-organization [20], refer to a cooperation process without any centralized decision, where a decentralization on the service discovery phase can be implemented by using the social plasticity of the providers, as presented in [8,11,12,15,21]. The structure of the new services composition is determined by considering a variety of techniques, from optimization techniques, which require heuristic-based algorithms to face the problem of combinatorial optimization (known to be NP-hard), to more sophisticated Artificial Intelligence (AI) based techniques, to achieve a near optimal solution and to accelerate the execution time. Regarding the detection of the moment of change, several works are focusing different reconfiguring conditions, e.g., new consumer policies and requirements [22] or request of new services [23]. An agent-based simulation was used to detect machine breakdowns, handling rush order arrivals, service performance degradation by using dynamic monitoring [24]. The specification of triggers during the design-time and using a common user interface was performed for manually added resource service's configurations [25].

In the manufacturing domain, service reconfiguration is also addressed in the literature. As examples, a services' model for the dynamic production system reconfiguration, particularly to reorganize the machinery to face a newly introduced product, was proposed in [26], and an approach that considers software and hardware reconfiguration is also proposed by [27], using a knowledge ontology and AI-planning for the service reconfiguration. The service reconfiguration can also benefit from integrating MAS and SOA to combine the best features of both worlds, namely decentralization, interoperability, loose coupling, intelligence and autonomy [28]. In this context, an automated service composition approach is proposed in [14], aiming to maximize the overall quality of the final composition, using agents that adapt the services' processes, in a continuous manner. A dynamic service reconfiguration is proposed by [12] that explores the use of agents to achieve consistent solutions focusing on fault-tolerant systems. An automatic plug & produce approach was used to easily change the setup services of a robot welding cell system to be more time and cost efficient in small lot sizes [16]. In this work, the continuous service discovery plays an essential key to support the development of automatic reconfiguration in real-time.

In the context of several European R&D projects, the service reconfiguration problem has also been addressed, namely the PRIME project [17] that uses a plug & produce approach combined with MAS, to allow the rapid reconfiguration and deployment, and the evolutionary system adaptation. The SOCRADES project focused on the reconfiguration of smart embedded devices [9], and the IDEAS project focused on the reconfigurable production systems by using agent technology to perform the on-line reconfiguration without the need of reprogramming efforts [18]. More recently, the PERFoRM project [19] focused on the seamless reconfiguration of machinery and robots as a response to operational and business events. However, these approaches do not focus primarily the reconfiguration under a service reconfiguration perspective and do not fully comply the identified requirements for service reconfiguration.

In conclusion, the literature survey shows that, until now, the existing service reconfiguration approaches are usually specified during the design phase and do not consider the need for a smooth reconfiguration process (as stated by R3), achieved by a centralized composition planner that does not provide the capability to execute the on-the-fly reconfiguration (as stated in R2), and focused in the occurrence of failure events or product changeovers. The analysis and execution of the reconfiguration process are usually carried out in an individual way without considering a collaborative analysis (as stated in R4).

Under this perspective, the relevant challenge regarding the service

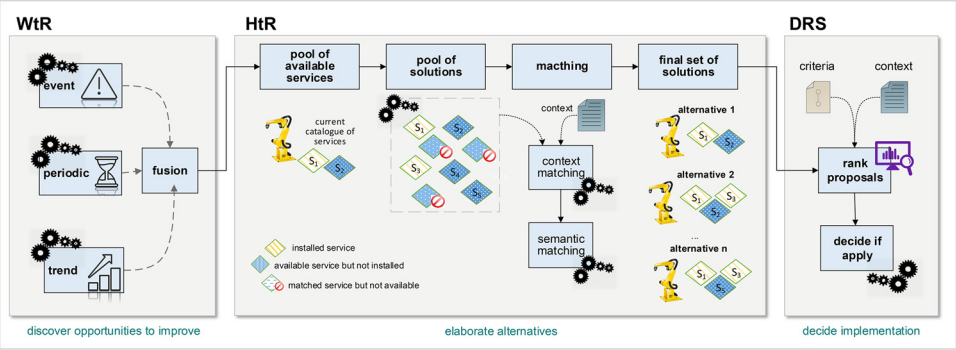


Fig. 2. Service reconfiguration module embedded in each agent.

reconfiguration is to develop an integrated approach that addresses the established requirements. This has to be done by detecting opportunities to evolve the system behaviour in a continuous and pro-active manner, elaborating and evaluating the alternatives to perform the reconfiguration smoothly and on-the-fly, i.e. without the need to stop, re-program or re-start the system, and always considering the competitive and collaborative environments.

3. Multi-agent system architecture for service reconfiguration

Having in mind the development of an adaptive system that can regulate, in a dynamic, distributed and on-the-fly manner, the catalogue of services offered by each entity to response to internal and external condition changes, complying with the requirements previously described, a MAS approach is proposed for the service reconfiguration process, as shown in Fig. 1.

The proposed approach considers a network of intelligent and autonomous agents that interact by following the service-based principles, i.e. each one exposing its functionalities as services. The resource agents (RA) act as service providers by encapsulating the physical operations provided by machines as services, e.g., drilling or welding operations. These services are published in a repository and can be reached via discovery mechanisms by product agents (PA), which act as service consumers that need to consume those services to execute their products aiming to meet the production demand. Agents can also compose several atomic services and provide the resulting aggregation as a composed service.

The product and resource agents have different reconfiguration needs, according to the service reconfiguration classes defined in the previous section: the product agents focus on changing the structure of a composed service, and the resource agents cover the improvement of the service's behaviour, the changing of the services' catalogue and the changing of the structure of a composed service.

For this purpose, agents are continuously monitoring and able to execute a service reconfiguration process by embedding a reconfiguration module, illustrated in Fig. 2, that comprises the When to Reconfigure (WtR), How to Reconfigure (HtR) and Decide Reconfiguration Solution (DRS) phases [15], which will be detailed in the next sections.

3.1. When to reconfigure phase

The first phase of the service reconfiguration approach is related to the continuous monitoring and analysis of the collected data, aiming to determine an opportunity to reconfigure, e.g., a degradation of the service's quality or the product's changeover. For this purpose, the WtR model relies on three different triggering strategies for the on-line detection of possible situations to reconfigure [15], supporting the discovery of opportunities to reconfigure:

- **Event**: uses an event-driven approach to detect the occurrence of events related to the service condition changes, e.g., a resource failure or the removal of an existing resource/service. This strategy allows a good reaction facing unexpected events.
- **Periodic**: uses a periodic check to verify the current service performance to detect possible opportunities to reconfigure. The triggering time interval should be dynamically adjusted to better fit the system dynamics, i.e. increasing or decreasing this value taking into consideration the application of proper machine learning algorithms [28]. As example, Q-learning [29] is a suitable method to implement this feature since it provides a positive/negative reinforcement feedback that allows converging to an optimal value at each moment.
- **Trend**: uses machine learning methods to detect, as earlier as possible, a trend or pattern in the service performance, allowing the prediction of an eventual problem, and consequently to identify an opportunity to earlier implement a service reconfiguration procedure to mitigate this possible loss of performance. Some examples of algorithms that can be used in this strategy are the anomaly detection, the regression models and the clustering analysis.

Triggers indicating opportunities to proceed with reconfiguration, generated from different strategies, can happen at the same time in relation to different features and contexts, without any explicit interference between them. When this happens, the fusion module is responsible for joining the concurrent triggers, selecting the most critical one and firing one event to the HtR module that will handle the elaboration of the space of service reconfiguration alternatives.

3.2. How to reconfigure phase

After identifying an opportunity to reconfigure, the HtR module is responsible to determine how the service reconfiguration can be implemented. The process comprises the elaboration of a pool of possible service reconfiguration solutions, which compatibility should be tested by using a context and semantic matching to reduce the dimension of the alternative solutions. The elaboration of the alternative solutions considers the improvement of the resource's utility and consequently the three classes of service reconfiguration previously referred, namely improving the service's behaviour, changing the service's catalogue, and changing the service composition, as previously presented.

The algorithm embedded in each agent to build the space of alternative solutions for the service reconfiguration is represented as follows.

Algorithm: **how to determine the space of configurations**

Inputs:
Sets of services associated to an agent: $\{S_{current}, S_{available}, S_{problem}\}$
Context Rules: $\{C_{rules}\}$
Maximum number of installed services in the agent: *NS*
Output:
Set of feasible configurations provided by the agent: $C_{feasible}$

1: **for** $s_i \in S_{problem}$ **do**
2: $S_{available} \longleftarrow S_{available} + s_i^{improvement}$
3: **end for**
4: $C_{candidates} \longleftarrow S_{current}$
5: **for** $s_i \in S_{problem}$ **do**
6: **for** $s_j \in S_{available}$ **do**
7: $C_{candidates} \longleftarrow C_{candidates} \cup replaceImprove\ (C_{candidates}, s_i, s_j)$
8: **end for**
9: **if** $i == 1$ **then** $C_{candidates} - S_{current}$
10: **end for**
11: $C_{feasible} \longleftarrow applyContextMacthing\ (C_{rules}, C_{candidates})$

The algorithm finds all possible combinations for the catalogue of services, limited to a maximum value *NS*, by improving each identified service presenting poor performance indexes, i.e. $s_i \in S_{problem}$, through the execution of a set of actions regarding the optimization of the process encapsulated by the service, e.g., calibrating tools or replacing components, or through the replacement of problematic services by others that are available but are not currently installed, i.e. $s_j \in S_{available}$, which are promising to contribute for the improvement of the resource performance. Each configuration solution comprises the set of services to be provided by the resource, each one represented according to the tuple $\{s_i, a_i\}$, where s_i is the service and a_i is the action to be performed during the reconfiguration (namely nothing, improve, add or remove).

This process can generate a huge volume of service configuration alternatives, resulting in a time-consuming task to analyse and evaluate all of them. However, some of these alternatives are not technically adequate for the current state of the system, e.g., services that are not possible to be installed due to missing technical conditions. For this purpose, and aiming to reduce the space of alternative solutions, it is used a matching mechanism that analyses each configuration solution according to the context and current situation, complemented with machine learning and semantic reasoning techniques aiming to discard non-feasible, non-reasonable and non-beneficial solutions. Semantic reasoning tools, such as the JENA framework, permit to execute the semantical reasoning about the logical configuration and determine the feasibility of the service reconfiguration solution. In detail, as illustrated in the following rule, this mechanism determines if the set of attributes of a proposed service matches the technical constraints presented in the resource component (in this case the diameter of the new drill service should be less than the maximum diameter supported by the machine). Note that each service, resource and process is semantically described in a RDF/XML format.

[(?x ms:**canProcess**?y) ⟵ (?x rdf:type ms:Machine),
(?y rdf:type ms:drillService),
(?x ms:maxDiameter ?maxD),
(?y ms:diameter ?d),
ge(?d, ?maxD)]

At the end, the outcome of this process is a set of feasible service reconfiguration solutions that should be evaluated and ranked.

4. Decision of the service reconfiguration implementation

The last phase of the proposed service reconfiguration mechanism is responsible for selecting the optimal configuration, which requires an evaluation of the possible alternatives and a decision about the viability of its implementation (DRS module in Fig. 2).

4.1. Evaluation of the service reconfiguration alternatives

The agent conducts an evaluation procedure to rank the pool of possible feasible service reconfiguration solutions provided by the HtR module according to their effectiveness. For this purpose, a multi-criteria function is used to quantify the advantage of performing a certain reconfiguration solution, that basically is based on the benefits of the new configuration (which should be maximized) and the cost to execute the transition into this new configuration (which should be minimized),

Table 1
QoS indicators to estimate the benefit of a certain configuration solution.

Parameter	Equation	Description
Availability	$f(\phi) = \frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{r} \lambda_j + \sum_{j=1}^{r} \psi_j}$	Indicates the percentage of time that the service is available by considering the service uptime (λ) and the service downtime (ψ); *r* is the number of times that the service was executed
Execution time	$f(\theta) = \frac{\sum_{j=1}^{r} (\delta_j - \rho_j)}{r}$	Indicates the average execution time by considering the conclusion time (δ) and the requested time (ρ) for each one of the *r* number of times that the service was executed
Throughput	$f(\eta) = \frac{\gamma}{t}$	Indicates the performance of the service by considering the number of times that the service was executed (γ) in a certain period of time (*t*)

as illustrated in the following formula:

$$score = RB/RC \quad (1)$$

where *RB* is the reconfiguration benefit and *RC* is the reconfiguration cost. The RB parameter is further detailed by considering the analysis of several QoS indicators, expressed in the next formula:

$$RB = \sum_{i=1}^{s} f_i(\varnothing)^{w_1} \times f_i(\theta)^{w_2} \times f_i(\eta)^{w_3} \quad (2)$$

where *s* is the number of services considered in the configuration solution, and $f(\theta)$, $f(\varphi)$ and $f(\eta)$ are the QoS indicators, which meaning is represented in Table 1. These indicators, which evolve over the time according to the service performance and external context, provide an estimation of the benefit of the configuration solution.

On the other hand, it is also important to calculate the cost associated with the implementation of each configuration solution, which in this work, considers the effort for the implementation of the several modifications defined in the new configuration. For this purpose, the vector with the actual service configuration is compared with the vector with the new service configuration to determine the number of changes in the service catalogue [4], which should be grouped in three classes of actions: *nRS* that represents the number of removed services, *nAS* that represents the number of new added services and *nIS* that represents the number of service improvements. Having these values, the reconfiguration cost is calculated as follows:

$$\mathrm{RC} = nRS \times Cost_{remove} + nAS \times Cost_{adding} + nAS \times Cost_{improve} \quad (3)$$

where $Cost_{remove}$, $Cost_{adding}$ and $cost_{improve}$ represent the unitary average costs of removing, adding or improving a service.

At the end, after applying the multi-criteria function, the set of alternative service reconfiguration solutions is sorted according to the scores achieved by each configuration solution.

4.2. Decide the implementation of the service reconfiguration

After the evaluation process, the DRS module is responsible for deciding if the best service reconfiguration alternatives previously generated by the HtR module should be implemented or not, since the fact that they are possible does not necessarily represent an obligation to perform any of these options. In fact, the dynamic environment can produce undesirable situations, i.e. distributed reconfiguration procedures are performed unsynchronized and sometimes simultaneously. The DSR mechanisms can detect these unsatisfactory situations, preventing the system from going to states of unstable behaviour that limits the ability to predict, with accuracy, the future system landscape and compromises the service reconfiguration decision-making.

4.2.1. Nervousness stability mechanism

Resource agents are able to regulate the nervousness of their reconfiguration decisions through the use of stability mechanisms embedded in the DSR module. This mechanism ensures that a service reconfiguration is not executed every time a reconfiguration opportunity occurs, because it may bring negative effects and may cause a performance degradation, controlling the reconfiguration awareness frequency: a low-frequency value corresponds to a calm system that probably misses opportunities to improve, and a high-frequency value corresponds to a very nervous system that may lead to an unstable behaviour.

The stability mechanism imposes a restriction to limit the frequency of the service reconfiguration under a particular context. Agents are continuously adjusting this frequency based on the Q-learning [29] algorithm, e.g., defining upper and lower bounds. For this purpose, the algorithm uses the positive and negative feedback from previous service reconfigurations to increase or decrease the threshold value for reconfiguration.

4.2.2. Collaborative mechanism

At this stage, distributed agents have produced service reconfiguration solutions individually in an autonomous and self-interested way, without sharing its objectives with the other agents. This approach fits well with competitive environments where the several agents act individually, competing to maximize their individual profit. However, in collaborative environments, the lack of regulation or a normative environment using self-interested agents can lead to situations that are degrading the entire system performance, namely:

- **Deadlock situations**, where the simultaneous self-fish service reconfiguration based on the interest of the most valuable services can lead to situations where no one offers the lowest profitable but necessary services.
- **Non-beneficial solutions**, where in spite of the benefits of the service reconfiguration for the proponent, the proposed reconfiguration is not beneficial for the entire system.

The design of a collaborative interaction protocol allows to reach a mutual agreement that benefits the collaborative system behaviour, improving the competitiveness of the system and avoiding a service reconfiguration carried out in an uncoordinated and chaotic way, and particularly preventing the existence of deadlocks [30]. For this purpose, the proposed protocol, illustrated in Fig. 3, considers the initiator agent as the one that wants to perform a service reconfiguration, and the participant agents as the other system's agents.

The Initiator agent, after deciding to implement a service reconfiguration, notifies its intention to all the other Participant agents by sending the "REQUEST" message, inquiring if someone can provide the service(s) that will be removed from the catalogue of services, i.e. asking the non-objection of the participants. Each Participant agent will reason about its local catalogue of services and will reply with "INFORM" if it already provides the service or "REFUSE" if it does not provide the service. After compiling the replies from all participants, the initiator should cancel the opportunity to reconfigure if none "INFORM" message is received since despite being beneficial for the initiator, the proposed service reconfiguration solution leads to a non-feasible configuration (in the collaborative perspective). In opposite, if at least one "INFORM" message is received, which means that a feasible collaborative reconfiguration solution was achieved, the initiator should proceed to the second phase of this protocol that is related to the evaluation of the reconfiguration proposal. This phase starts with the election of the rapporteur, elected from the peers, that will be responsible to analyse if the service reconfiguration proposal is beneficial for the entire system. The Rapporteur uses the collected historical data, and without critical time restrictions, simulates several scenarios to get conclusions about the service reconfiguration proposal benefits. In case that the reconfiguration proposal is found beneficial for the whole collaborative system, the initiator agent gets the green light to proceed with the implementation of the service reconfiguration proposal; otherwise, the initiator should cancel the intention to reconfigure.

An important assumption in this collaborative mechanism is that all agents are acting in good faith, which is expected since they are running in a collaborative environment.

5. Experimental validation

5.1. Description of the case study

The proposed service reconfiguration approach was tested by considering the AIP-PRIMECA flexible manufacturing system [31], which comprises 5 workstations (WS), interconnected by a conveyor system. Each workstation offers a catalogue of services (with a maximum of 2 services for M2 and M3, and 3 services for M4) related to the execution of several operations, as illustrated in Table 2. As example, the "I_comp" service can be executed by the workstation M4 and the "L_comp"

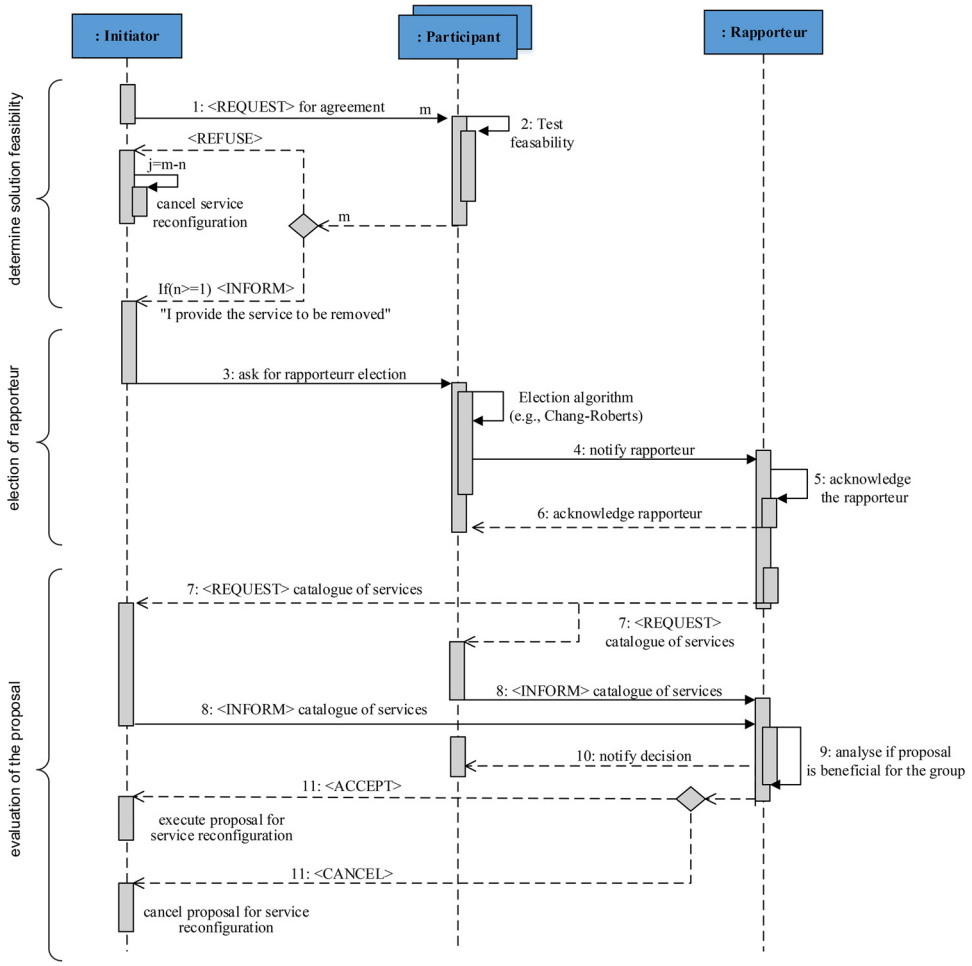


Fig. 3. Interaction protocol to determine the reconfiguration viability in collaborative scenarios.

Table 2
Catalogue of services provided by each workstation (processing time in seconds) (adapted from [31]).

	Workstations				
Service	**M1**	**M2**	**M3**	**M4**	**M5**
Loading	I (10)				
Unloading	I (10)				
Axis		I (20)	NI (20)		
r_comp		NI (20)	I (20)		
I_comp				I (20)	
L_comp		I (20)		NI (20)	
Screw_c			I (20)	I (20)	
Inspection					I (5)

Legend: I – offered in service catalogue; NI – available but currently not offered in service catalogue.

service can be executed by M2. Some services are available but are not currently offered in the machines' catalogue, e.g., the "L_comp" service is not currently offered by M4 but can be offered in the future if a service reconfiguration is performed (i.e. changing its catalogue of services).

In case the agents decide to perform a service reconfiguration, the physical resource is stopped during the execution of the maintenance intervention to improve the service performance (which takes 50 s) or to replace the provided service (which takes 30 s for each replacement). Aiming the simulation of realistic scenarios, it was considered that workstation M2 have a probability of failure of 4% for all services in their catalogues, staying out-of-service during 300 s for the execution of recovery actions. After the execution of an intervention to recover or improve the service performance, its probability of failure is reduced by 1,5%.

Several sub-products are being produced in this system, namely the parts in the form of the letters A, B, E, I, L, P and T, which combined can produce the final products BELT and AIP [31]. The complete execution of each product requires the execution of a set of operations defined in its process plan, as illustrated in Table 3 (see [31] for more details), which is performed by the different workstations according to their catalogue of services.

The proposed agent-based approach for the service reconfiguration was implemented using the JADE framework [32], namely the resource and product agents. A service layer creates an abstraction to real physical devices, where the virtual environment replicates each workstation emulating all the functionalities of the AIP-PRIMECA case study for running the experimental tests. The communication among the agents performed by using FIPA-ACL (Foundation of Intelligent Physical Agents - Agent Communication Language) compliant messages. Several agents were launched to represent the workstations and the products requested to be manufactured in the system.

Each one of the agents representing workstations has embedded the WtR module to identify opportunities to reconfigure, which in this work, considers the periodic and trend strategies. The trend strategy uses statistical and machine learning (ML) techniques that keep a track of the performance of services and allows predicting anomalies and triggering reconfiguration opportunities. The statistical technique permits to deal with the temporal continuity of data, being used the R "*AnomalyDetection*" library [33]. This package implements the seasonal

Table 3
Products process plans (adapted from [31]).

Operation	B	E	L	T	A	I	P
#1	Loading	Loading	Loading	Loading	Loading	Loading	Loading
#2	Axis	Axis	Axis	Axis	Axis	Axis	Axis
#3	Axis	Axis	Axis	Axis	Axis	Axis	Axis
#4	Axis	Axis	Axis	Rcomp	Axis	Icomp	Rcomp
#5	Rcomp	Rcomp	Icomp	Lcomp	Rcomp	Screw	Lcomp
#6	Rcomp	Rcomp	Icomp	Inspection	Lcomp	Inspection	Inspection
#7	Icomp	Lcomp	Screw	Unloading	Icomp	Unloading	Unloading
#8	Screw	Inspection	Screw		Screw		
#9	Inspection	Unloading	Inspection		Inspection		
#10	Unloading		Unloading		Unloading		

hybrid ESD (S-H-ESD) algorithm that takes into account the trend and data seasonality, which permits to detect both global as well as local anomalies, like structural breaks, in the time series data. The ML technique used was the unsupervised K-Means Clustering algorithm to detect the outliers. In particular, after performing the cluster analysis, the instances that do not belong to any cluster are potentially indicated as anomalies, i.e., those with largest distances to the cluster center. In both cases, the agents access to the R libraries [33] via RServe API connected by TCP/IP that acts as a back-end for services.

In the periodic strategy, the Q-learning algorithm [29] was used to dynamically adjust the sampling frequency to check the current service performance based on the positive/negative reinforcement feedbacks from previous service reconfigurations. The Q-learning provides feedback to each *Action* of the agent given a particular *State*:

- The *State* = $< \mathrm{St}, \beta >$ is composed by the available set of services St at the instant t to execute the production batch $\beta = < \xi, \rho >$, where ξ represents the number of orders and ρ the product type;
- The Action = $<\uparrow\Delta, =\Delta, \downarrow\Delta >$ can be related to increase the time interval ($\uparrow\Delta$), to maintain the time interval ($=\Delta$) or to decrease the time interval ($\downarrow\Delta$);
- The reward *R(State, Action)* is calculated by considering the *Action* taken in a specific *State* and the reconfiguration triggering feedback. If the reconfiguration feedback had led to a better configuration structure, the Q-value of *State-Action* pair chosen will be increased. Otherwise, it will suffer a penalty for the chosen Δ value, given by the following equation, $\mathrm{R(State, Action)} = \begin{cases} +n, & \textit{if reconfigure} \\ -\left(\frac{n}{2} * \mathrm{penalty}\right), & \textit{if not reconfigure} \end{cases}$, where n is the reward constant.

After some interactions, the agent is able to select an optimal triggering frequency, in a given State. This strategy leads to a reduction of the event triggering strategy since the agent is executing the service reconfiguration more efficiently in a preventive manner.

In the same manner, agents incorporate the HtR module that allows determining the best strategy to reconfigure, namely generating potential reconfiguration solutions based on the weak reconfiguration type (i.e. improving the service performance) and strong reconfiguration type (i.e. changing the service catalogue). JENA was used to implement the semantic verification of the matching between the pool of alternative configurations and the existing resources' context, allowing to reduce the number of these alternatives.

5.2. Experimental results

The experimental tests were conducted by considering different batch orders, namely 10, 20 and 30 BELT products, and the following testing scenarios:

i.) The service reconfiguration mechanism is disabled.
ii.) The service reconfiguration mechanism is enabled but without collaborative capabilities.
iii.) The service reconfiguration mechanism is enabled with collaborative capabilities.

Fig. 4 illustrates the experimental results for the different described scenarios, considering the makespan (i.e. the necessary amount of time to execute the entire batch of products) that illustrates the production efficiency, and the workload distribution (i.e. the average of the standard deviation of the service utilization for each machine) that illustrates how well balanced is the production (values close to zero represent a well-balanced production).

Initially, the system was running with the service reconfiguration and collaborative mechanisms disabled. The second scenario considers that each resource agent is running in a self-fish mode, which means that agents will execute their service reconfiguration individually and without any collaborative procedure. The achieved results show the benefits of using reconfiguration, with the best improvement achieved for the 10 BELT batch scenario (improvement of 19%), which means that for larger batches the occurrence of disturbances are more diluted and the reconfiguration has a shorter percentage impact. A more balanced workload among the machines is also noticed when applying the

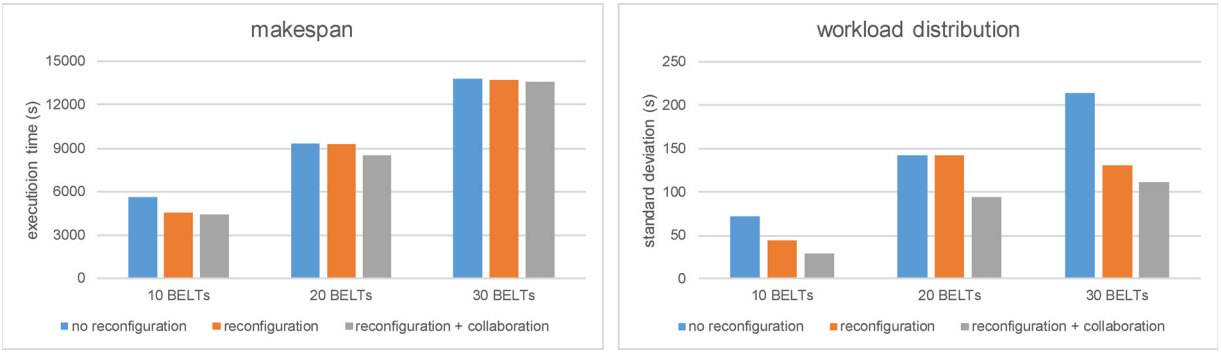


Fig. 4. Experimental results for the lead time and workload distribution, considering different batches and types of service reconfiguration.

service reconfiguration and the collaborative service reconfiguration.

A third scenario considers that the collaborative mechanism is enabled, running over the distributed self-fished service reconfiguration process. In this case, the achieved results show an increase of the system performance (illustrated by the reduction of the makespan in 21,2%), with the reduction for larger batches following the same pattern as identified for the previous scenario. In the same manner, a reduction of the workload distribution is also noticed, which means a better balancing of the resource utilization. This scenario shows the benefits of applying the collaborative interaction protocol to balance the service reconfiguration performed individually by the distributed agents.

This set of experiments allowed to verify the benefits of using service reconfiguration, illustrated by the increase in the system production efficiency, and particularly the advantage of combining the collaborative interaction protocol to better balance the service reconfiguration performed individually by the distributed agents. These results also show that the proposed service reconfiguration approach is dependent on the dimension of the batch size, the frequency of service failures and also from the recovery time, which requires the consideration of learning mechanisms to decide if the application of the service reconfiguration is beneficial or could have a counter-productive effect.

6. Conclusions and further work

The dynamic service reconfiguration process is, nowadays, a hot topic in smart manufacturing systems, aligned with the CPS context, and particularly with the Industry 4.0 initiative. However, a literature review in this field showed that service reconfiguration is usually performed in a manual, offline and centralized manner, without fulfilling the requirements for a truly automatic service reconfiguration.

This paper proposes an agent-based approach to execute a dynamic, online and decentralized service reconfiguration, where intelligent software agents apply different strategies to identify opportunities to reconfigure, e.g. facing service failures, service performance degradation or production changeover, in a pro-active manner. These agents, after identifying opportunities to evolve, are able to determine the alternative possibilities to reconfigure their catalogue of services, and decide for the most promising one. The individual service reconfiguration approach is directly mapped in competitive environments, where the selfish behaviour of the agents leads to a truly dynamic and decentralized service reconfiguration. In case of collaborative environments, a decentralized collaborative interaction protocol was designed to ensure that the intended reconfiguration triggered by individual agents is only executed if it is beneficial for the whole system, avoiding reaching non-feasible configurations and also promoting a more efficient system configuration.

The proposed service reconfiguration solution was implemented by using the agent-oriented JADE framework and tested in a flexible manufacturing system use case. The experimental results show the feasibility of the proposed approach, with the proposed approach displaying better system performance than the normal operation, and the use of the collaborative mechanism leading to even better results for collaborative situations. The dimension of the product batch size also influences the benefits of the proposed services reconfiguration approach.

Future work will be devoted to a further analysis of the effects of the batch size in the system performance, and to a deeply regulation of the system “nervousness” phase during the reconfiguration process, avoiding falling into chaotic situations and allowing to run the system in a smooth manner when facing very dynamic environments.

References

[1] H. Kagermann, W.W. Wahlster, J. Helbig, Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0, ACATECH - German National Academy of Science and Engineering, 2013.

[2] OASIS, Reference model for service oriented architecture, Public Rev. Draft 2, (2006), pp. 1–28. August.

[3] C. Zeginis, D. Plexousakis, Web service adaptation: State of the art and research challenges, Technical Report 410, ICS-FORTH, (2010) Available at https://www.ics.forth.gr/tech-reports/2010/2010.TR410_Web_Service_Adaptation.pdf (Accessed on 30/08/2017).

[4] N. Rodrigues, P. Leitão, E. Oliveira, Dynamic service reconfiguration with multi-agent systems, in: T. Borangiu, D. Trentesaux, A. Thomas, P. Leitão, J. Barata (Eds.), Service Orientation in Holonic and Multi-Agent Manufacturing, Springer, 2017, pp. 307–318.

[5] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice-Hall, 2005.

[6] N. Rodrigues, P. Leitão, E. Oliveira, et al., An agent-based system approach for service reconfiguration in collaborative scenarios, in: V. Mařík (Ed.), Proceedings of the 8th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS'17), Lecture Notes in Artificial Intelligence, vol. 10444, Lyon, France, 28-30 August, 2017, pp. 140–154 Springer.

[7] G. Candido, C. Sousa, G. Di Orio, J. Barata, A.W. Colombo, Enhancing device exchange agility in service-oriented industrial automation, Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE'13), (2013), pp. 1–6.

[8] S. Karnouskos, et al., The IMC-AESOP architecture for Cloud-based industrial cyber-physical systems, Industrial Cloud-Based Cyber-Physical Systems, (2014), pp. 49–88.

[9] A.-W. Colombo, S. Karnouskos, J.-M. Mendes, Factory of the future: a service-oriented system of modular, dynamic reconfigurable and collaborative systems, in: L. Benyoucef, B. Grabot (Eds.), Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management, Springer, London, 2010, pp. 459–481.

[10] I.M. Delamer, J.L. Martinez Lastra, Self-orchestration and choreography: towards architecture-agnostic manufacturing systems, Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'16) vol. 2, (2006) 573–577.

[11] W. Dai, Wanqi Huang, V. Vyatkin, Enabling plug-and-play software components in industrial cyber-physical systems by adopting service-oriented architecture paradigm, Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON'16), (2016), pp. 5253–5258.

[12] W.T. Tsai, W.W. Song, Y.N. Chen, R. Paul, Dynamic system reconfiguration via service composition for dependable computing, Reliable Syst. Unreliable Networked Platforms 4322 (2007) 203–224.

[13] P. Zoltan, E. George (Eds.), Runtime Reconfiguration in Networked Embedded Systems, Springer, Singapore, 2016.

[14] F. Lécué, Y. Gorronogoitia, R. Gonzalez, M. Radzimski, M. Villa, SOA4All: an innovative integrated approach to services composition, Proceedings of the IEEE International Conference on Web Services (ICWS'10), (2010), pp. 58–67.

[15] N. Rodrigues, P. Leitão, E. Oliveira, Triggering strategies for automatic and online service reconfiguration, Proceedings of the 11th Iberian Conference on Information Systems and Technologies (CISTI'16), (2016), pp. 76–82.

[16] G. Reinhart, S. Krug, S. Hüttner, Z. Mari, F. Riedelbauch, M. Schlögel, Automatic configuration (plug & produce) of industrial ethernet networks, Proceedings of the 9th IEEE/IAS International Conference on Industry Applications (INDUSCON'10), (2010).

[17] A. Rocha, G. di Orio, J. Barata, N. Antzoulatos, E. Castro, D. Scrimieri, S. Ratchev, L. Ribeiro, An agent based framework to support plug and produce, Proceedings of the 12th IEEE International Conference on in Industrial Informatics (IECON'14), (2014), pp. 504–510.

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

[19] P. Leitão, J. Barbosa, A. Pereira, J. Barata, A.W. Colombo, Specification of the PERFoRM architecture for seamless production system reconfiguration, Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON'16), Florence, Italy, 24–27 October, 2016, pp. 5729–5734.

[20] W.R. Ashby, Principles of the self-organizing dynamic system, J. Gen. Psychol. 37 (1947) 125–128.

[21] E. del Val, M. Rebollo, V. Botti, Combination of self-organization mechanisms to enhance service Discovery in Open systems, Inform. Sci. (2014) 138–162.

[22] Y. Bar-Yam, Dynamics of complex systems, Comput. Phys. 12 (2003) 864.

[23] E.M. Maximilien, M.P. Singh, A framework and ontology for dynamic web services selection, IEEE Internet Comput. 8 (5) (2004) 84–93.

[24] M. Bruccoleri, P. Renna, G. Perrone, Reconfiguration: a key to handle exceptions and performance deteriorations in manufacturing operations, Int. J. Prod. Res. 43 (19) (2005) 4125–4145.

[25] T. Borangiu, S. Răileanu, D. Trentesaux, T. Berger, Open manufacturing control with agile reconfiguring of resource services, Control Eng. Appl. Inform. 12 (4) (2010) 10–17.

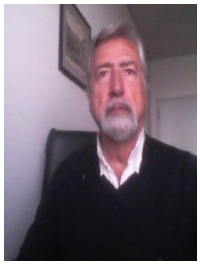
[26] D. Marcos-Jorquera, F. Macia-Perez, V. Gilart-Iglesias, J.A. Gil-Martinez-Abarca, Service model for the management of industrial environments. Dynamic reconfiguration of production elements, Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN'07), (2007), pp. 249–254.

[27] T.-H. Yang, W.-P. Lee, Intelligent service reconfiguration for home robots, in: X. Ding, X. Kong, S.J. Dai (Eds.), Advances in Reconfigurable Mechanisms and Robots II, Springer, 2016, pp. 735–745.

[28] N. Rodrigues, P. Leitão, E. Oliveira, Dynamic composition of service oriented multi-agent system in self-organized environments, Proceedings of the Workshop on Intelligent Agents and Technologies for Socially Interconnected Systems (IAT4SIS'14), Prague, Czech Republic, 2014.

[29] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[30] G.B. Chafle, S. Chandra, V. Mann, M.G. Nanda, Decentralized orchestration of composite web services, Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, (2004), pp. 134–143.
[31] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, J. Barbosa, Benchmarking flexible job-shop scheduling and control systems, Control Eng. Pract. 21 (no. 9) (2013) 1204–1225.
[32] F.L. Bellifemine, G. Caire, D. Greenwood, Developing Multi-Agent Systems With JADE, John Wiley & Sons, 2007.
[33] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2015.
[34] F. Gamboa Quintanilla, O. Cardin, A. L'Anton, P. Castagna, A modeling framework for manufacturing services in service-oriented holonic manufacturing systems, Eng. Appl. Artif. Intell. 55 (2016) 26–36.

Nelson Rodrigues has a MSc in Information Systems at Polytechnic Institute of Bragança, and is currently a Ph.D. candidate at the University of Porto. His main research topics focus on Intelligent and Reconfigurable Manufacturing Control Systems, with interesting ideas on how to explore the benefits of Artificial Intelligence for self-adaptative and evolutive manufacturing systems. Additional research interests are Industry 4.0, Cyber-physical systems and Multiagent systems. He is a research at IPB since 2010, participating in H2020 PERFoRM & GO0D MAN projects and in the past in FP7 projects ARUM & GRACE, in which his research topics play a relevant role. He has about 29 papers published. He is also a member of the IEEE Technical Committee on Industrial Agents and CeDRI (Research Centre in Digitalization and Intelligent Robotics) and collaborator on LIACC (Artificial Intelligence and Computer Science Laboratory) at the University of Porto.



Eugénio Oliveira is a Full Professor in Artificial Intelligence at the University of Porto. Co-Founder of LIACC (Artificial Intelligence and Computer Science Laboratory) at the University of Porto and its Director 2011-2016. He got his Ph.D. in Artificial Intelligence at New University of Lisbon in 1984. Awarded with Gulbenkian Prize for Science and Technology in 1984. "Guest Academic" at IBM/IEC in Belgium (84-85). He supervised more than twenty Ph.D. students in the area of AI and Software Agents. Coordinator and partner in both national and international (EU funded) projects. Most recent project team coordination: H2020 SIMUSAFE (2017–2020). He published about 300 papers. h-índex (Google Scholar) = 30, n. citations = 4849. Current topics of interest include Software Agents architecture and strategies for cooperation, Trust and Reputation Models, Intelligent Transportation Systems, "Emotional-like" Agents, Text Mining and Multi-agent systems applications and Deep Learning techniques. He was General co-Chair of the 18th EPIA Conference on Artificial Intelligence, September 2017.

Paulo Leitao received the M.Sc. and Ph.D. degrees in Electrical and Computer Engineering, both from the University of Porto, Portugal, in 1997 and 2004, respectively. He joined the Polytechnic Institute of Bragança in 1995, where he is Professor at the Department of Electrical Engineering and Coordinator of CeDRI (Research Centre in Digitalization and Intelligent Robotics). His research interests are in the field of industrial informatics, intelligent and reconfigurable systems, cyber-physical systems, Internet of Things, distributed data analysis, factory automation, multi-agent systems, holonic systems and self-organized systems. He participate/has participated in several national and international research projects (EU FP7 and H2020) and Networks of Excellence, has published 4 books and more than 200 papers in high-ranked international scientific journals and conference proceedings (per-review). He is co-author of three patents, received four paper awards at INCOM'06, BASYS'06, IEEE INDIN'10 and INFOCOMP'13 conferences, and served as general co-chair of several international conferences, namely IEEE INDIN'18, SOHOMA'16, IEEE ICARSC'16, HoloMAS'11 and IFAC IMS'10. Dr. Leitão is Senior member of the IEEE Industrial Electronics Society (IES) and Systems, Man and Cybernetics Society (SMCS), past Chair of the IEEE IES Technical Committee on Industrial Agents and member at-large of the IEEE IES Administrative Committee (AdCom). Currently is also chair of the IEEE Standards Association P2660.1 Working Group.