


BIOSOARM: a bio-inspired self-organising architecture for manufacturing cyber-physical shopfloors

João Dias-Ferreira¹ · Luis Ribeiro²  · Hakan Akillioglu¹ · Pedro Neves¹ · Mauro Onori¹

Received: 30 June 2016 / Accepted: 30 August 2016 / Published online: 10 September 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract Biological collective systems have been an important source of inspiration for the design of production systems, due to their intrinsic characteristics. In this sense, several high level engineering design principles have been distilled and proposed on a wide number of reference system architectures for production systems. However, the application of bio-inspired concepts is often lost due to design and implementation choices or are simply used as heuristic approaches that solve specific hard optimization problems. This paper proposes a bio-inspired reference architecture for production systems, focused on highly dynamic environments, denominated BIO-inspired Self-Organising Architecture for Manufacturing (BIOSOARM). BIOSOARM aims to strictly adhere to bio-inspired principles. For this purpose, both shopfloor components and product parts are individualized and extended into the virtual environment as fully decoupled autonomous entities, where they interact and cooperate towards the emergence of a self-organising behaviour that leads to the emergence of the necessary production

flows. BIOSOARM therefore introduces a fundamentally novel approach to production that decouples the system's operation from eventual changes, uncertainty or even critical failures, while simultaneously ensures the performance levels and simplifies the deployment and reconfiguration procedures. BIOSOARM was tested into both flow-line and “job shop”-like scenarios to prove its applicability, robustness and performance, both under normal and highly dynamic conditions.

Keywords Bio-inspired production systems · Self-organisation · Cyber-physical production systems

Introduction

The refinement in consumer's requirements and the fast paced development of socio-technical systems are promoting an increasingly globalized market with a high demand for fast time-to-market, sustainable, high quality and highly customized, or even personalized, low priced products (Pine 1999; Asif et al. 2008). This new reality is nowadays being addressed, almost unanimously, by all the current research agendas for the future factories on what it is now perceived as the 4th Industrial revolution (Kagermann et al. 2013; Monostori 2014; Jazdi 2014). Collectively they stress the importance of cyber-physical equipment to design shopfloor components. These have their own cyber-identity to autonomously and actively take part, dynamically, on the different shopfloor processes (Kagermann et al. 2013; MacDougall 2014).

Furthermore, in response to these new challenges, a number of modern production paradigms (MPP) has also emerged, such as: bionic manufacturing systems (BMS) (Ueda 1992), holonic manufacturing systems (HMS) (McFar-

✉ João Dias-Ferreira
joao.diasferreira@itm.kth.se

Luis Ribeiro
luis.ribeiro@liu.se

Hakan Akillioglu
haaki@kth.se

Pedro Neves
pmsn2@kth.se

Mauro Onori
onori@kth.se

¹ Production Engineering, The Royal Institute of Technology, Stockholm, Sweden

² Department of Management and Engineering, Division of Manufacturing Engineering, Linköping University, Linköping 581 83, Sweden

lane and Bussmann 2003), reconfigurable manufacturing systems (RMS) (Koren et al. 1999), evolvable production systems (EPS) (Onori 2002), and more recently cloud manufacturing systems (CMfg) (Ren et al. 2014; Zhang et al. 2014), etc. They complement the cyber-physical nature of modern shopfloors by providing the necessary conceptual and theoretical background to enact systems with the required capabilities to thrive in face of the currently uncertain, but highly competitive global markets. For this purpose, modern production systems must be built upon a set of new production requirements and characteristics that go beyond the traditional performance indicators (Monostori et al. 2006). These emerging requirements and characteristics (ERC) include, but are not necessarily limited to:

- increasing component and system level autonomy in decision making (ERC1);
- development of a cyber-physical oriented design promoting the individualization of equipment and products (ERC2);
- improved overall system robustness with a focus on tolerance to faults and system changes (ERC3);
- adaptive capabilities that ensure short and immediate system adaptation (ERC4);
- evolutive capabilities that ensure scalability as well as strategies for long term changes (ERC5).

These ERCs are characteristics that support the different stages of the production system life cycle, fundamental to cope with these new and highly dynamic environments. In this context, the ERCs must be harmonized and valued on par with the more traditional performance metrics.

By complying with these ERCs, shopfloors become naturally modular, autonomous, networked, adaptable and plug-gable. Nevertheless, their full potential can only be reached if the production control mechanisms and interaction patterns are able to exploit and reflect these characteristics. Hence, from a structural and dynamic perspective such solutions should become close to biological systems, particularly to collective biological systems (swarms). In fact, biological systems and their characteristics are a common analogy to express the high level design principles of MPPs (further discussed in Sect. 2). For instance, BMS *...draw parallels with biological systems and proposes concepts for realising essential properties of future manufacturing systems...* (Tharumarajah et al. 1998). HMS owe their main architectural characteristics to hierarchies and stable intermediate forms in living organisms and social organisations (Tharumarajah et al. 1998). Furthermore, A. Tharumarajah states that *the emerging concepts of manufacturing systems have been derived from some underlying similarity with naturally occurring systems, be they biological or social*. EPS embraces the concept of evolution by relying on many simple, reconfigurable,

task-specific elements, which closely resemble swarms (Onori et al. 2006). Within the RMS context, bio-inspired techniques may also contribute to obtain reconfigurable manufacturing systems with their desired characteristics, as postulated in Leitão et al. (2012). The same is valid for CMfg.

Many different system reference architectures, based on distributed control and supported by these MPPs have been proposed in the past decade in an attempt to embrace the ERCs (Van Brussel et al. 1998; Maturana et al. 1999; Barata and Camarinha-Matos 2003; Leitão and Restivo 2006; Monostori et al. 2006; Shen et al. 2006; Frank et al. 2011; Schutz et al. 2013; Vyatkin and of America 2007; Vyatkin 2011; Josuttis 2007; Brennan and Norrie 2001; Lepuschitz et al. 2011; Farid and Ribeiro 2015). Although each architecture may follow different structural and information patterns, they typically use functional decomposition to distribute the different functionalities to independent self-contained entities. Also, in most cases they follow the intelligent product pattern that relies on the establishment of process sequences (workflows) that support the production process of one or several intelligent products, which are responsible to oversee their own execution (McFarlane et al. 2013). In this sense, control is typically achieved by asynchronous negotiation and cooperation interactions.

Despite these efforts, the link between the system and bio-inspiration is often lost, either due to architectural design or to implementation choices that deviate from the conceptual guidelines. In the end, bio-inspired principles are only very seldom integrated into the architectural production control mechanisms, rendering in many cases the resulting systems shy of their main source of inspiration and consequently of the ERCs. Most of the successful application of bio-inspired concepts, in the production context, has been limited to the use of bio-inspired heuristics, dedicated to solve specific hard optimization problems (Leitão et al. 2012).

In this sense, the authors present BIOSOARM as a strongly bio-inspired system reference architecture for control of highly dynamic production environments. BIOSOARM strictly adheres to bio-inspiring structural organisation and self-organising control principles. It aims to provide an engineering framework that fully complies with the ERCs, by modelling its cyber-physical components so that they all cooperate towards the emergence of a collective production behaviour in an analogous way, to the way collective biological systems behave in their natural environment. This is very different from most current work that typically relies on negotiation-based control mechanisms to comply with product-driven approaches.

In the present work, BIOSOARM is supported by a known model of the self-organising behaviour of fireflies in the wild. Nevertheless, the BIOSOARM structure is decoupled from the control algorithm and consequently should support the implementation of any collective-biologically-inspired

model. As such, the approach presented is fundamentally novel and it requires a careful behavioural and performance analysis.

The subsequent details are organised as follows: Sect. 2 discusses the different characteristics of biological and manufacturing in order to highlight the challenges and limitations of implementing bio-inspired concepts in manufacturing; in Sect. 3 the related work is presented; Sect. 4 presents the main constructs and properties of the BIOSOARM architecture; while in Sect. 5 the bio-inspired self-organising control mechanism used to instantiate the architecture is described in detail; in Sect. 6 further details regarding the implementation as well as the representative testing scenarios are provided; the achieved results and their assessment are further detailed in Sect. 7; and the overall conclusions are provided in Sect. 8.

Biological concepts in light of manufacturing systems

In order to develop a proper framework for strong bio-inspiring design, that serves as base for the development of BIOSOARM, it is fundamental to highlight the different characteristics of biological and engineering systems at the light of production environments. Also, a quite substantial set of biological jargon has been frequently applied in the context of MPPs without a concrete definition of the concepts. In this section, MPP's main biological supporting concepts, frequently used in the literature, are therefore presented, discussed and redefined at the light of the current work.

One of the key concepts for both biological and engineering systems is autonomy. In particular, biological systems and their response are heavily related to the interaction between autonomous entities. Autonomy is permanently limited by design and may be temporarily constrained by the environment in which a biological or artificial systems or components are immersed.

Definition 1 (*Autonomy*) The ability of a component or a system to govern themselves, within predefined design limitation or otherwise in an open unbounded way, formalized as a decision making process that defines how such component or system exert their autonomy.

Autonomy is envisioned in this context as a precondition for systems and the components within those systems to evolve and adapt.

It is important to note that biological systems and their adaptive and evolutionary responses result from millions of years of trial and error. In addition, the species with the best absolute traits were not necessarily the ones that survived. The prevailing species presented instead the right traits relatively to an evolving environment at the right time (there is an important randomization factor to be accounted for in evolu-

tionary processes as well). These processes are also not about the evolution of specific individuals, but rather about the common traits affecting the entire species (Futuyma 1998).

Evolutionary processes are therefore an elusive concept in the context of bio-inspired production architectures and paradigms. In natural systems, evolutionary success is seen as the species overall reproductive success (Floreano and Mattiussi 2008). Although this principle makes sense from a bio-inspired optimization algorithms point of view, it is not applicable to engineering systems, where components do not “reproduce”. In this sense, evolution should be interpreted in a weak sense. In most cases it does not come from within the system but can be induced by an external actor, by adding or removing components. If, later on, such external mechanisms would be embedded in the system, as part of its architecture, then the grounds for generally discussing evolution in engineering control systems improve.

Definition 2 (*Evolution*) A system intrinsic, dynamic and long-termed improvement process, that results on the development of new or significantly improved functionalities, characteristics or behaviours not previously described as part of the system.

Adaptation is instead a more tangible concept, since it respects the individual. It relates design and function (Rafferty 2011) and it can be translated into being suited to a given situation or a set of circumstances. From a biological perspective, adaptation is the adjustment or changes in behaviour, physiology, or structure that got selected and integrated in the organism, so that it becomes more suited to the environment (UNA of Sciences 2015). Adaptive processes, at least some of them, can also be taught and/or learned, which makes it a more interesting concept in an engineering context in general.

Definition 3 (*Adaptation*) Is any behavioural, functional, or structural change enacted by the system that results in a better fit of the particular component or subsystem to handle the environmental conditions at a particular time and context.

The result of an adaptive response is reflected on a component/system adaptedness. Adaptedness can be biologically understood as the degree to which an organism is able to live and reproduce in a given set of habitats (Dobzhansky et al. 1970). It requires, as well, a reinterpretation of the production context as discussed in this paper.

Definition 4 (*Adaptedness*) An indicative measurement of the state of adequacy and quality of a component's or subsystem's functions for a given operational context.

Adaptive responses are normally enacted in biological systems through a self-organising response. Self-organisation denotes the ability of collective systems to dynamically

and adaptively acquire spatial, temporal or functional structure themselves, without external control (De Wolf and Holvoet 2004; Haken 2006).

Definition 5 (*Self-organisation*) A pervasive system-wide adaptive response triggered by one of several components, locally adapting to environmental or other component's induced changes, that causes a global structural or functional/behavioural re-arrangement.

Self-organisation is normally associated to the concept of emergence (De Wolf and Holvoet 2004), traditionally interpreted as “the whole is bigger than the sum of its parts”. However according to Bedau (2008), such an interpretation would imply “the creation of something out of nothing”. Emergence as a construct for an engineering system must therefore be carefully considered. Natural emergence can be explained from the observer's limitations in capturing and explaining the full flow of causality that leads to a specific system response. At the light of this interpretation, Bedau's notion of weak emergence suggests that such complex causality flows can be mainly explained by simulation processes (Bedau 2008). Emergence may manifest itself in engineering systems whose design promotes open adaptation and/or evolution. Therefore, although it is an elusive design construct, it cannot be disregarded as an effect, thus highlighting the need for simulation-based validation of certain system designs.

Structural and functional changes are often reflected on the size of the system. Another key characteristic of biological systems is therefore their ability to tune in the number of components in response to environmental changes. Scalability is also a key feature in engineered systems.

Definition 6 (*Scalability*) The ability of the system to grow or shrink autonomously, or otherwise accepting similar external induced changes, in order to adjust to new operational contexts keeping its performance levels within an acceptable threshold.

Component interfacing aspects are closely linked to the ability of dynamically re-organise. In fact, biological systems ensure that all their components interface instantly, implicitly or explicitly contributing to the speed at which adaptation may occur. From a production system's point of view, such characteristic has been perceived as key for the ability to introduce or remove new components without any major integration effort. Generally, this approach has been designated as Plug and Produce (Arai et al. 2000).

Definition 7 (*Plug-ability*) The ability of a system to seamlessly support the runtime integration of new physical components and their logical counterparts, that may change the system behaviour, function or structure, without disrupting the system's normal operation and without requiring any other action from the operator including the stoppage of the system.

The scope of a system or components autonomy ends up influencing its ability to adapt and evolve functionally, behaviourally or structurally, which has a direct impact on the system's robustness. In biological systems, robustness is interpreted as the persistence of certain characteristics or traits under perturbations or uncertainty (Félix and Wagner 2008). In an engineered system the ability to tolerate perturbations is also of evident importance. However, the robustness of natural systems frequently arises from their high number of homogeneous components and from the fact that a certain percentage of those components is disposable. This is not acceptable in a production context which is also a more heterogeneous environment. Robustness is, in this context, not only a characteristic but also an effect.

Definition 8 (*Robustness*) The ability of a system or system's component to maintain a stable functional behaviour under perturbations or uncertainties while simultaneously keeping its performance above a desired threshold.

The importance of these bio-inspired principles will necessarily grow due to the also increasing trend of developing cyber-physical shopfloors. Indeed, the cyber-physical design promotes the individualization of the system's components and subsequently their autonomy as well as the need to self-contain all the relevant information concerning each individual within the cyber-physical component it-self. Such an approach creates a very interesting starting ground for the development of BIOSOARM to further embrace the above defined bio-inspired concepts towards the full adherence of the ERCs.

Related literature

Classification of control architectures

In Dilts et al. (1991), one of the first classification schemes for non-centralized designs is introduced. This preliminary proposal has more recently been adapted in Trentesaux (2009) to better categorize the differences between different levels of hierarchical/heterarchical arrangements:

- *Class I Fully hierarchical control system* based on a pyramidal structure organised across different levels of control whereby the subordinate levels strictly adhere to the commands from the higher order levels.
- *Class II Semi-heterarchical control system* based on the relaxation of hierarchical relationship, by temporarily allowing other forms of organisation, specially during disturbances.
- *Class III Fully heterarchical control system* based on local relationships between the components without any order or rank promoting overall flatter models.

Although the rigidity of traditional automation systems is frequently connected to their hierarchical structures, these are not necessarily an avoidable design if the interactions between the components are still managed in a dynamic and efficient way. In fact, many state-of-the-art architectures that follow the intelligent product pattern rely on the establishment of temporary hierarchies that support the production process, which is managed by the product itself (McFarlane et al. 2013). In case of disturbances, the system can dynamically re-establish new hierarchical control chains. While the runtime dynamic nature of these hierarchical structures solves the inherent problems of structural rigidity, their performance is highly dependent on the proper design of the control mechanism. More than the control structure, the control interactions design and concurrency handling are critical to attain the maximum levels of performance, as demonstrated in Ribeiro et al. (2012a).

A compromise between hierarchy and heterarchy has been proposed by several authors and the Holonic design principles closely entail the use of Class II architectures. As such, and in a way typically compatible with the intelligent product pattern, Class II architectures consider a nominal operating mode and a disrupted mode. Hence, in case of a critical event, the hierarchical recovery mechanism kicks in first. If that fails, the entire system or parts of it, will enter in disrupted mode where the hierarchy is temporarily dissolved (switching-down), improving the system's ability to adapt. Once recovered, a switch-back mechanism triggers the re-organisation of the system back to its nominal hierarchical control structure.

Heterarchical architectures have mainly been proposed due to ease of change. The lack of any hierarchy makes the system theoretically more pluggable and able to better handle internal or external changes by having more decision freedom to self-organise. Nevertheless, these designs are considerably affected by a myopic behaviour, due to the loosely linked nature of the individuals, which limits their access to global knowledge. For this reason it is difficult to achieve long-term optimization and maintain the levels of performance often attributed to the more hierarchical architectures. Approaches to minimize the system's myopia would have to possibly involve a chain reaction of information exchange (Zambrano et al. 2011; Rey et al. 2013) which may be time consuming and computationally intensive.

Reference control architectures

As mentioned in Sect. 1, since the emergence of modern manufacturing paradigms several reference architectures have been proposed in literature for the control of modern production systems. A considerable part of these developments has been introduced by the research community on multi-agent systems (MAS) and has used agent-based

technologies as the implementation support for the high level and frequently bio-inspired design principles (Leitão 2009; McFarlane and Bussmann 2000; Babiceanu and Chen 2006; Shen et al. 2006). In this sense, the main reference system architectures for shopfloor control are briefly surveyed.

The Product Resource Order Staff Architecture (PROSA) (Arch. 1) (Van Brussel et al. 1998) was among the first such architectures and was inspired by the holonic design principles. It defines a set of holons that must self-organise in order to optimize production. The original architecture has been later modified (Verstraete et al. 2008) [(Arch. 2)] to include more bio-inspired behaviours. The adaptive holonic control architecture (ADACOR) (Arch. 3) (Leitão and Restivo 2006) also follows holonic design principles and addresses the agile response of the shopfloor towards emergence, change and uncertain scenarios. In Barbosa et al. (2013) proposes a second iteration of ADACOR where a self-organising mechanism is introduced to smooth the overall system's response during disturbances and increase the system's robustness. The ORCA-FMS (Arch. 4) architecture proposed by Pach et al. (2014) follows an approach similar to ADACOR, in which the system may alternate between a nominal (hierarchical) and a disrupted mode (totally or partially heterarchical), to tackle perturbations or changes. In Maturana et al. (1999) proposed the MetaMorph I which follows a mediator-centric federation architecture (Classe I), aiming at providing a dynamic adaptation of form, structure and activity to emerging tasks and environmental changes. This architecture has been further improved in MetaMorph II (Arch. 5) (Shen and Norrie 1998), in order to integrate design, planning and scheduling, simulation, execution, material supply and marketing mediators. In Peeters et al. (2001) (Arch. 6) a pheromone based emergent shopfloor control system is presented. This architecture aims at managing disruptions and changes, both in process or product characteristics through a stigmergy mechanism. Another architecture inspired in natural phenomena (Arch. 7), potential fields, is presented in Zbib et al. (2012). It aims to dynamically and simultaneously resolve allocation and routing problems in FMS. Finally, in Ribeiro et al. (2012b) presents the IADE (Arch. 8) architecture which aims to support the rapid design and deployment of mechatronic systems.

Table 1 presents a mapping between the discussed architectures and the above mentioned ERCs. As it is possible to conclude, most architectures only partially adhere to ERC1. This is due to the class II nature of many of the analysed architectures or due to global layers that influence the system's behaviour. Regarding the ERC2, most architectures provide equipment individualization to different extents. Products are however not always specifically individualized, as in many cases the product is abstracted by more than one entity (e.g.

Table 1 Mapping between the reference architectures and the ERCs

Arch.	ERC1	ERC2	ERC3	ERC4	ERC5
Arch. 1 (Class II)	Components can be both autonomous or belong to an hierarchy	Resource holons abstract physical resources or resource aggregations. Product is not explicitly individualized. No product aggregation. Staff holon is not a physical agent	Resource holons abstract physical resources or resource aggregations. Product is not explicitly individualized. Staff holon is not a physical agent. Robustness is attained by changing autonomy levels of resource or order holons	Resource allocation algorithms are not pre-determined by PROSA	Evolution or plug-ability are not explicitly mentioned
Arch. 2 (Class III)	Components are autonomous but may delegate tasks to a swarm operating in parallel environment	Similar to PROSA with the exception of the staff holon which is not included	Order agent keeps exploring its options and communicates its intentions during execution	Delegate pattern allows to react to up-to-date information	Evolution or plug-ability are not explicitly mentioned
Arch. 3 (Class II)	Components can be both autonomous or belong to an hierarchy	Operational holons abstract physical resources. Resource aggregation is not considered. Product is not explicitly individualized. Supports product aggregation	Dynamically changes hierarchies and supports changes from hierarchical to heterarchical structure of system/subsystem	Adaptation managed by a autonomy regulating stigmergy-based mechanism	ADACOR holons of plug and product type. Evolution used in a different context
Arch. 4 (Class II)	Components can be both autonomous or belong to an hierarchy	Local optimizers (Resources) are associated with a physical component. Products are extended by intelligent products. No mention to resource or product aggregation	Robustness is achieved by making intelligent products following only optimal undated machine sequences	Switch between a hierarchical predictive and a heterarchical reactive architecture	Evolution or plug-ability are not explicitly mentioned.
Arch. 5 (Class II)	Components can be either autonomous with direct communication or respond to mediators	Resource agents abstract numerous types of entities. Mediator agents coordinate activities. No explicitly individualization of products	It is attained by changing structural arrangements and alternate between mediated and autonomous operations	Mediator centric approach allows dynamic coalition generation	Evolution or plug-ability are not explicitly discussed
Arch. 6 (Class III)	Autonomous components. Decision supported by layered algorithm dependent on global propagation	Resource agents abstract specific resources. Aggregation of resources is not mentioned. Order agents represent production orders and are linked to a specific workpiece	The exploration behaviour, the limited awareness of global system status and use of local information ensures robustness	Spreading of global information and update of local information allow adaptation	Interactions through the environment. Evolution or plug-ability not explicitly discussed
Arch. 7 (Class III)	Components are autonomous	Resources abstract physical resources. Products are properly individualized. No mention of resource or product aggregation	The decoupled nature of the interactions ensures the system robustness	Self-organisation adapts the resources allocation and path selection	Constructs are loosely coupled. Evolution or plug-ability not explicitly discussed
Arch. 8 (Class I)	Components can be autonomous or belong to a dynamic hierarchy	Resource agents abstract physical resources. Coalition leader agents abstract skill (processes) aggregations. Product is explicitly individualized through the product agent	Re-negotiation mechanism ensures the system robustness	Self-organising generation of coalitions enact the system adaptation	Designed for plug-ability. Supports evolutions in its weakest sense

order and product holon in Van Brussel et al. (1998) or product and task holon in Leitão and Restivo (2006)). Furthermore, all the architectures adhere to ERC3 and ERC4, although from different perspectives. On the other hand, ERC5 is only specifically addressed by Ribeiro et al. (2012b), and plug-ability is only briefly mentioned in Leitão and Restivo (2006).

Through the analysis presented above it is, therefore, possible to conclude that the major gap between the reference architectures and the ERCs is in the adherence to ERC5. There is also only a partial adherence of some architectures to ERC1 and ERC2.

BIOSOARM attempts to close these gaps by relying on a heterarchical architecture that, despite the limitation to achieve a fully optimized performance, provides a better structural organization, where the different autonomous entities promote the system's robustness and adaptation in the face of highly dynamic environments, by cooperating towards the emergence of a self-organising/optimizing behaviour. Furthermore, BIOSOARM also attempts to fully adhere to the concept of evolution (in its weakest sense, as defined in Sect. 2), by supporting the seamless addition/removal of components.

It is important to point out that Table 1 and the analysis presented does not aim to evaluate the different architectures but simply understand how they adhere to the ERCs in the context of this work.

BIOSOARM reference architecture

In this section the BIOSOARM architecture is carefully detailed and mapped against biological systems. It is very important to stress that BIOSOARM aims to provide a framework focused on highly dynamic manufacturing environments. In this sense, it introduces a new self-organised production pattern, where both shopfloor equipment as well as product parts equally contribute for the overall response of the system as a whole. BIOSOARM is decoupled from the self-organising control algorithm and therefore any swarm-like behaviour can be instantiated through it.

Architectural components

Collective biological systems perform many different self-organising behaviours that, in a more or less optimal way, explore the limited capabilities of the individuals to perform highly complex behaviours. In Fig. 1, a possible analogy between one of such behaviours and the main BIOSOARM architectural constructs is presented. In this case, the mating ritual of some species (e.g. birds) is considered. During these rituals, male entities perform specific behaviours to attract female entities. If the male succeeds to attract the female,

eventually they mate and produce offspring. Analogously, in the manufacturing domain, resources attract different parts which after the execution of a process may result in the generation of a new part, this process is repeated over and over, until a final product is produced.

However as opposed to the biological systems, parts usually do not have the ability to move. In this sense, from a BIOSOARM architectural perspective the male entity would be abstracted by the part together with the transport entity. Furthermore, different analogies can be made in order to produce different bio-inspired behaviours. For example, the male entity could be a food source or the shortest path to the nest, which would then attract the swarm individuals towards them, among others.

The different constructs, particularly the Part (PT) and the two superclasses Resource (RS) and Transport System (TS) are detailed below.

Resource The Resource is an abstraction of all the potential cyber-physical components with a processing role in the system. It attracts PTs and executes specific production processes, defined through templates (described below). A RS also encapsulates the common features and information of its specialized classes. It directly interfaces with the equipment hardware through a Reactive Layer Library (RLL) that harmonizes the equipments functionality with the networked cyber level, enabling its execution upon demand. A RS is also a buffer (size may vary), as it may need to store different PTs to process at a later stage. A specific resource can be defined as RS_r with $r \in [1, \text{no. of resources}]$.

The heterogeneous nature of the shopfloor components implies that some shopfloor components may have specific features. Hence, as shown in Fig. 1, the RS is further specialized into Source (RSo), Station (RSt) and Sink (RSi).

- The Source, abstracts the entry points of the parts. Their activities are limited to the introduction of parts in the shopfloor.
- The Station, instead, abstracts any shopfloor module that is able to process any type of parts. This implies both value adding and non-value adding processes.
- Finally, the Sink abstracts any resource that represents an exit point of the shopfloor. It removes products from the shopfloor.

Part The Part abstracts any component in the shopfloor to be processed. For example, a product is typically the result of the process or aggregation of different parts, which can in turn be a resulting product of other branches or production lines. Each part $[PT_p \text{ with } p \in (1, \text{no. of parts})]$ in the shopfloor, from the raw materials to the final product, is represented as a PT of different types $[type(PT_p)]$, depending on its characteristics.

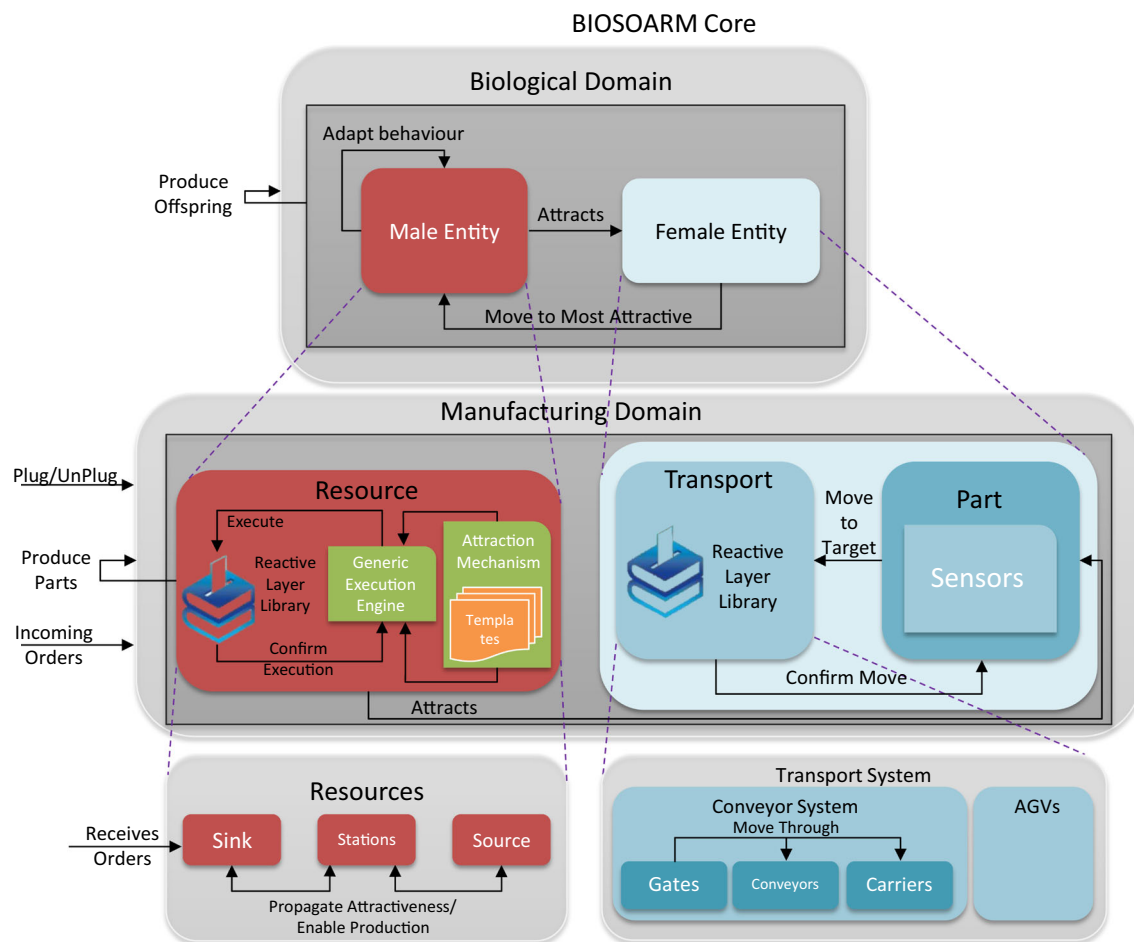


Fig. 1 BIOSOARM architecture

Transport - The Transport is responsible for moving the PTs through the shopfloor. TP is also a superclass, it encapsulates the common features and information of the specialized transport classes. Hence, considering the completely different nature of the transport systems in industrial shopfloors, TP may be further divided into the specialized class AGV (TAGV) or into another superclass conveyor system entity (TCS), as shown in Fig. 1. A specific transport entity can be defined as TP_i with $i \in [1, \text{no. of transport entities}]$.

- **TAGV** - The AGV abstracts automated guided vehicles. Each AGV needs to be able to locate itself, calculate its own route, and to have the ability to avoid obstacles such as other AGVs, etc.
- **TCS** - Conveyor systems, instead, are restricted to fixed paths. In this context, the TCS abstracts all the generic variables and functions common to the management of a conveyor-based system. The TCS may be further divided into the following specialized classes (Fig. 1): Conveyor (TCO), Gate (TGa) or Carrier (TCa).

- The Conveyor abstracts all the point to point sections of the conveyor system. It is unidirectional and has its own speed and length.
- The Gate abstracts all the n to 1 , 1 to n and n to n routing sections of the conveyor system. Similarly to the conveyor, each gate may have different speeds and lengths.
- The Carrier abstracts the carriers that carry and hold the different PTs while moving through the TCos and TGas.

Templates The concept of template was introduced in analogy to swarm-based communication patterns. Collective biological systems perform different swarm behaviours for different and particular situations. Similarly in a shopfloor, different PT flows are necessary to produce different products. In this sense, a template $[T_i]$ with $i \in (1, \text{no. of templates})$ represents the main construct that supports and regulates the interactions between the RSs and the PTs, in order to promote the emergence of a coherent and robust self-organising behaviour and consequent PT flows.

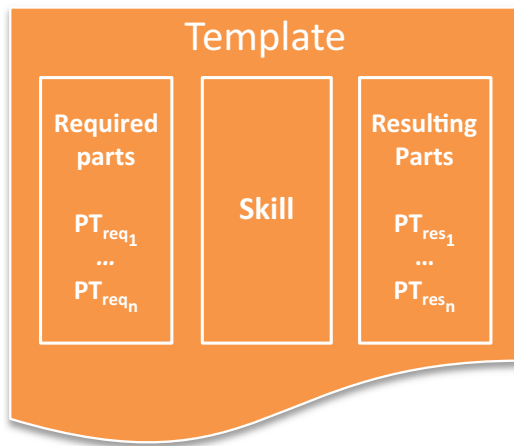


Fig. 2 Main constituents of a template

Each particular resource therefore holds a set of templates [$templates(RS_r)$]. As Fig. 2 shows, a template is composed by the required PTs, a skill and the resulting PTs. A set of required PT type(s) for T_i is therefore defined as $req(T_i)$. The template's skill is instead defined as $skill(T_i)$, and it is instantiated and executed once all the $req(T_i)$ are available. Its execution then results in the production of a set of new resulting PT(s) of certain type [$res(T_i)$].

A skill, is the main executing construct in BIOSOARM. It translates the physical capabilities of a component, by accessing a RLL, into the functionalities that the cyber counterpart can execute and consequently offers for execution. The skill is based on the concept introduced by the EPS paradigm. For more details please refer to Ribeiro et al. (2012b).

Interaction patterns

Swarming is the resulting behaviour of interaction patterns between numerous entities that strive to reach a common goal. As previously mentioned such goals may vary in nature, but ultimately they abstract an attractor that influences the behaviour of the system.

As in swarm-like biological systems, the value of the BIOSOARM architecture is not in the isolated individuals but rather in their interaction patterns and resulting system's wide response.

Resource-part interactions

As depicted in Fig. 1, the RSs are therefore the “attractors” that influence the behaviour of the system by providing the execution of production processes. This is reflected in variations of the production flows by making themselves more or less attractive towards their specific required parts. For this purpose each RS has a certain attraction radius (defined by the system integrator) and an attraction value that trans-

lates the attractiveness (β) of the RS for a specific PT type (each PT type as a different value $\beta_{type(PT_p)}$). Furthermore, the PTs detect the attraction fields, when inside the attraction radius, which then use to decide towards which RSs are they attracted to (typically towards the most attractive). In this way, RSs and PTs are loosely coupled and only interact indirectly with each other.

Different approaches and algorithms can be coupled and used to calculate β .

Part-transport interactions

As soon as a PT is attracted towards the attraction field of a certain RS, it request the TS, which is carrying it, to move towards that specific attracting RS. Once the RS is reached, the TS confirms its arrival by sending a move confirmation to the PT.

The PTs are therefore completely decoupled from the transportation system. The transportation system is managed by the TS entities that simply provide transportation services to the PTs. More details regarding the transport systems are presented below.

Transport system

As stated before, the TAGVs have the freedom to move in any different direction. Hence, they provide a more natural solution to attain swarm-like transport behaviours. In such case, upon reception of the specific target, the TAGV follows the specific attraction field of the target towards which the PT is being attracted, simply by avoiding possible obstacles until it reaches its target.

Since in preliminary versions of the BIOSOARM architecture (Dias Ferreira et al. 2013; Dias-Ferreira et al. 2014) a AGV-based transport solution was successfully tested, the authors decided to focus, in the this work, on conveyor-based transport systems. Not only they are the most challenging, within this context, as well as they are probably the most common automated transportation mechanism in current shopfloors.

In this sense, a solution based on the same attraction principles was developed to self-organise and manage a conveyor based transport system. One important thing to point out is the fact that for transportation purposes any RSt is considered to be a normal conveyor with relatively small length, capable to hold only a single carrier.

For the implementation of the proposed transport management mechanism it is assumed that each of the TCSs and RSs have a specific pivotal point ($PvP()$) defined by the system integrator, which does not necessarily need to be central. It serves as reference for the propagation of the transport attractiveness.

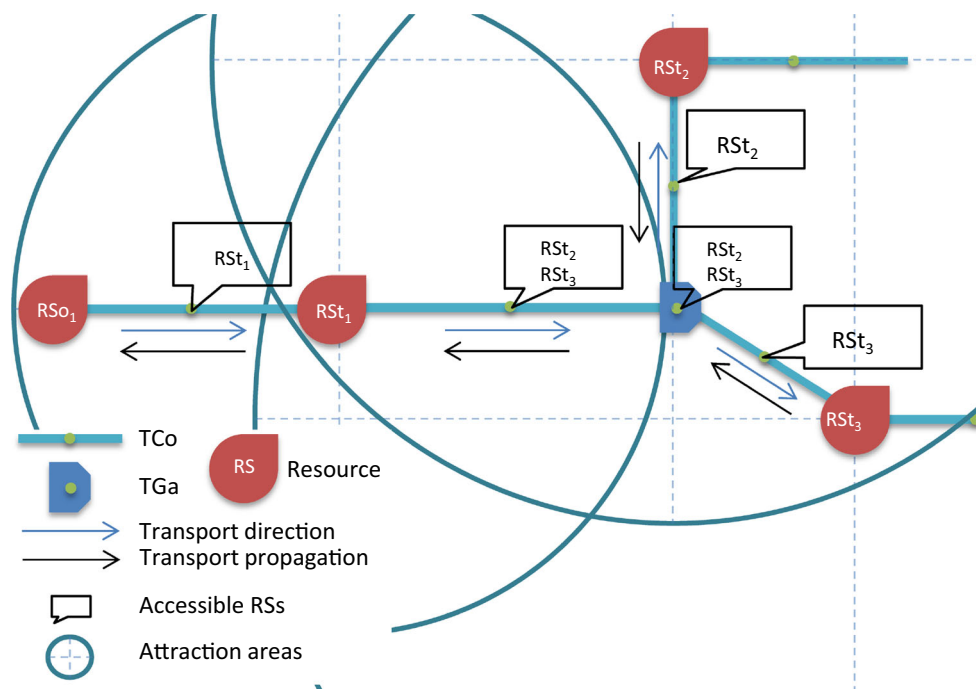


Fig. 3 Transport propagation

With this in mind, when a TCS (except the TSos) or a RS (except the RSos) is deployed, it starts by checking with the upstream neighbour (it can be either a RS or TCS) with which RSs it is connected. If the deployed TCS or a RS is also within the attraction radius of the RSs to which its upstream neighbour is connected, having as reference its $PvP()$, then the deployed TCS or a RS assumes the connection to those RSs. That information is then further propagated to its downstream neighbours. This back-propagation mechanism stops when the entering points of the conveyor system are reached.

An example is depicted in Fig. 3. In this case, the plug of the TGa is considered. Hence, the TGa starts by asking to its exit TCo to which RSts they are connected. In this particular case, one is connected to RSt_2 and the other to RSt_3 . Since TGa is also inside the attraction radius of RSt_2 and RSt_3 , then TGa assumes the connection to both RSt_2 and RSt_3 . These connections are then back-propagated to RSt_1 , which will go through a similar process. The back propagation in this particular case stops at the RSo_1 .

Finally, when a PT_p is deployed and it is attracted to a RS_r , the PT_p requests the TCa_r , which is carrying it, to move. The TCa_r then further interacts with the TCo_r or TGa_r in which it is located, in order to move towards the acquired target. Particularly if it is a TCo , the carrier is simply transported to the end of the conveyor, while if it is a TGa the carrier is instead routed to the specific exit which leads to the target.

Hybrid solutions that consider both AGVs or TCSs are supported.

Pull-based production control

BIOSOARM follows a pull production philosophy. This means that downstream work centres pull parts from previous stations, as needed [Spearman and Zazanis \(1992\)](#). For this purpose, the concept of orders (O)s was introduced. Orders are contained on a file that translates the requests of the customers. An order is composed by an order ID, the products requested which includes the product name, id and the desired quantity, and finally a due date for the completion of the order, as it is shown in the example below:

```
ORDER
ORDER_ID: 1
PRODUCTS
  PRODUCT_NAME: Car
  PRODUCT_ID: 4
  PRODUCT_QUANTITY: 100
END_PRODUCTS
DUE_DATE: 1460988496400
END_ORDER
```

To exemplify, in Fig. 4 an order is submitted to produce 100 PTs of type 4 (Car), as described above. Upon submission of the order (case A), the RS_{i1} activates its template that has as required PT the PT type that matches the requested product (in this case 4). Once the template is activated, automatically the attraction area of that required PT type is also

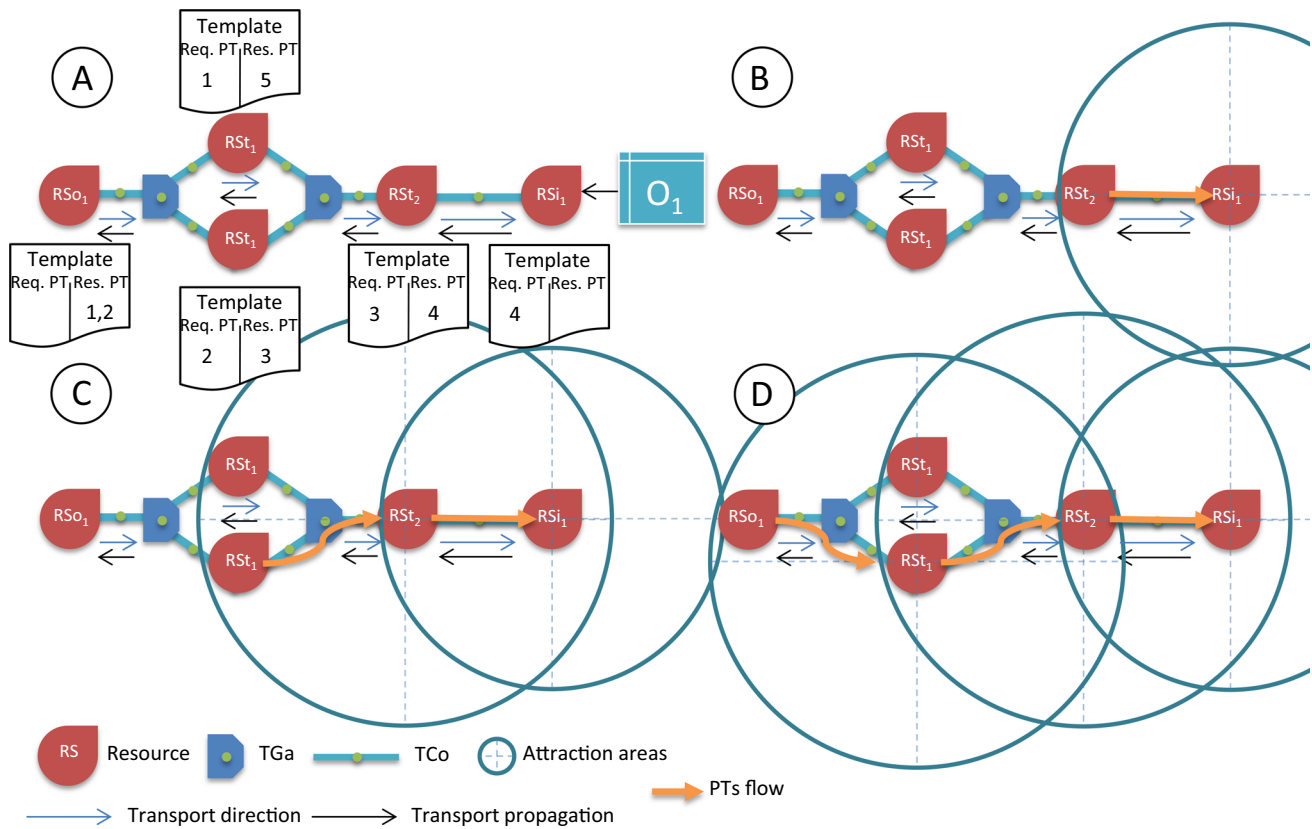


Fig. 4 Pull production mechanism

activated (case B). The activation of RSi₁'s attraction area for product type 4 then triggers the activation of the other upstream attraction areas of RSs that have as resulting PT the PT type 4. In this case RSt₂ has the PT type 4 as resulting PT of a template that requires PT type 3. Then RSt₂ starts attracting PTs of type 3 (case C). Finally, the activation of the attraction area for PT type 2 in RSt₁ trigger the release of PTs by the RSo₁. Once the required attracted PTs arrive to the respective RSs the template's skills are instantiated parameter-wise, according to those PTs, and then executed. As a result of the skill execution, new PTs are introduced into the shopfloor. This cycle continues until eventually the PT type is equal to the product, in this case until PT type 4 is produced by RSt₂. As soon as all the desired products are produced and absorbed by the RSis, a similar propagation mechanism takes place, but in this case to deactivate the templates and consequently the respective attraction areas.

Since the production of one RS, depends on the attraction of the next (without attraction the parts do not arrive and therefore the RSs are not able to produce), the system inherently assumes a self-organising pulling behaviour. A PT is only produced if there is a downstream RS that needs it.

Plug and produce

In swarm-like biological systems each individual is relatively disposable, as long as a critical number, enough to hold the whole, is maintained. Because of the loosely coupled nature of the interactions, the addition and removal of elements should have limited or no affect on the system, as long as there are the necessary numbers that ensure its correct behaviour or performance.

In plug-oriented state-of-the-art architectures the process of plugging a module typically entails the need to implement the particular process(es) at the module level (RLL), design and implementation of new workflows that include the newly plugged process(es) and, in some particular cases, the implementation of higher level processes including the already plugged process(es). Although it is a fairly simple procedure when compared to changes in traditional automation solutions, it is still rather laborious depending on the complexity of the system and more importantly of the product.

In BIOSOARM however, the different entities are autonomous, self-contained and loosely linked to each other. Moreover BIOSOARM introduces a different self-organising production pattern where there is no need for strict control over a pre-defined sequence of processes. In other words,

there is no notion of workflow. In this sense, the plugging procedure is basically reduced to the definition of the attraction radius and creation of template files that relate the particular PT(s) that a RS should process, with the skill itself and the expected resulting PT(s). Of course, there is also the need to develop the RLLs. However, due to its cyber-physical nature, it is assumed that the implementation of the RLL is done by the module provider. Once these are ready, the RS can then be plugged. The plug-in of a new RS simply affects other RSs in the computation of β as it will be demonstrated in Sect. 5.

Furthermore, the plug-ability is also extended to the transport system being it AGV or conveyor based. In the case of the AGVs, a new AGV is possibly simply an extra element on the shopfloor to take into consideration while navigating through it. For a conveyor system instead, once a CS is plugged the back-propagation mechanism is triggered and all the conveyor system self-organises itself in order to support new routes to the attracting locations.

As opposed to the unplug of a module, which requires the execution of a certain procedure, critical failures might result in the instant inoperability of a component. Because of the loosely coupled nature of BIOSOARM's interactions, the addition or removal of components also does not affect the system, as long as there is the necessary redundancy at the shopfloor level. Hence, both the unplug or a critical failure of a module are simply translated in one less point of attraction, less routes or less transport vehicles. The system maintains its normal behaviour possibly with some changes in the parts flow. The equipment can then be repaired and re-plugged at a later stage.

Such plug-ability facilitates the deployment and re-configuration procedures which ease the constant change and re-configuration of the system in the short term, as well as it promotes the evolution of the system (in its weakest sense) in the long-term, extending the shopfloor's life cycle.

Deployment methodology

As stressed before, the attraction of PTs by the RSs, according to the active templates, creates flows of PTs throughout the shopfloor. In this sense, and despite the fact that all RSs are completely decoupled, these dynamic flows promote the emergence of a network perspective of the shopfloor.

Hence, considering the pull production principles, it is important that the deployment of the shopfloor respects the flows that can be extrapolated from the templates by linking the required parts of a downstream RS with the resulting parts of an upstream RSs. In this way, if the attraction radius is precisely defined and the RSs are methodically deployed, so that the attractiveness areas encompasses the adjacent RSs

respecting the PT flows [as depicted in Fig. 4)], then the performance of the system should be "optimal" (considering BIOSOARM and the instantiated system), as demonstrated in Dias-Ferreira et al. (2014).

Major differences to other reference control architectures

BIOSOARM is a class III control architecture. In this sense, there is no notion of master and slave among RSs, PTs or TSs. Templates are used to regulate the interactions between shopfloor components and PTs, established through an attraction mechanism that promotes the attraction of the right PTs to the right RSs. Once there, the PTs are processed and new resulting PTs are deployed on the shopfloor to be further processed. Therefore, no architectural component is fully aware of the production sequences. Instead, the production flows emerge as the result of the self-organisation mechanism. This is a major difference to other state-of-the-art architectures, since it is a fundamentally novel production mechanism, opposed to the more common product-driven approach.

Furthermore, another important difference that results from the newly presented production mechanism is the fact that in BIOSOARM there is no need to define workflows, since there are no entities that need to be fully aware of the processing sequences. This simplifies the deployment and configuration procedures, as workflows are typically defined manually and grow in complexity in par with the product and the production system.

Moreover, by interacting through the attraction mechanism, the system operation is decoupled from eventual changes, uncertainties or even critical failures. Any possible critical failures or unplug of RSs is simply reflected in one less point of attraction. Also, the removal of a PT, simply means one less PT to attract. This means that no alternative measures need to take place to compensate for the removed PT. Since the PT is not directly bonded to a specific product entity, another similar PT will be attracted and take the place of the removed one. In short, BIOSOARM naturally handles changes and uncertainty without need to assume alternative operational states.

Additionally, by treating all the different parts on the shopfloor as independent units, BIOSOARM can also contemplate waste management. Ultimately, waste is also the result of the execution of some processes. Hence, specific RSs can be instantiated and attract the PTs (waste) from the shopfloor in order to further process them.

Finally, BIOSOARM's was designed so that the bio-inspired architectural components and decoupled interactions seamlessly support the implementation of different swarm-based self-organising behaviours.

Self-organising production control

For the purpose of this work, the authors used the Firefly Algorithm (FA) as the base for the bio-inspired self-organising control mechanism. FA is an algorithm based on the flashing characteristics of fireflies. It proposes an optimization attraction mechanism that can be both local (foggy weather) or global (clear sky), depending on the visibility of the fireflies considering the environment. In this sense, FA can present completely different behaviours that range from a standard Particle Swarm Optimization (PSO), when the sky is clear (all fireflies have a global view of the system), or a random walk, when the sky is extremely foggy (fireflies visibility range is minimum). This is a useful characteristic for tackling complex systems where the partitioning of the shopfloor into zones could be an important factor. It is also one of the major reason to select FA over other swarm-like approaches.

Fireflies in the natural world emit 'cold light'. It is emitted in short and rhythmic flashes creating particular patterns associated to specific species. It is mostly used to attract mates, during which the rate of flashes and their duration defines the signal pattern to which females respond to.

The FA (Yang 2009, 2010) is a population-based iterative procedure with numerous fireflies (agents) concurrently solving a considered problem. The communication is performed via 'bio-luminescent signals' that efficiently guide the individual fireflies through the search space (Łukasik and Zak 2009). The flashing characteristics of the FA can be summarized by the following three basic rules:

- All fireflies are unisex so that one firefly is attracted to other fireflies regardless of the gender.
- Attractiveness is proportional to their brightness, therefore of two flashing fireflies the less brighter one is always attracted towards the brighter one. The attractiveness is proportional to the brightness and both decrease with the distance. If there is no brighter one than a particular firefly will move randomly.
- The brightness is affected by the landscape of the objective function to be optimized.

There are, however, some conceptual differences from the FA to the proposed application in BIOSOARM. The first and probably most noticeable is the fact that fireflies are not unisex. Instead, as in nature, male fireflies are attracted towards female fireflies. Analogously, the PTs (male fireflies) are attracted towards the RSs (female fireflies). The light patterns that in nature regulate the attraction of different species, are in BIOSOARM abstracted by the templates that regulate the attraction of the right PTs to the right RSs. The female fireflies then together with the male fireflies produce new offspring, which in this case are new PTs introduced into

the shopfloor. Furthermore, since fireflies are able to move, which is not necessarily true for a PTS as mentioned before, the moving capability of a male firefly is undertaken by the TPs. A male firefly is therefore abstracted by a PT together with a TP.

Attractiveness

Attractiveness β is usually dependent on the distance towards the attractor. In this sense, the strength with which something is attracted, it is not only dependent upon the attractor, but also on the distance to it. From a production perspective, however, this makes the system considerably dependent on its layout, which in many cases it is constrained by the physical properties of the shopfloor or of the transport system. Hence, within the scope of this work, β is considered constant inside an attraction area βRS_r of a certain specific maximum radius [$\max Radius(\beta RS_r)$] around RS_r .

Additionally, in nature different firefly species are attracted towards different light patterns. Similarly, in the present work, specific PTs shall be attracted towards specific RSs. Hence, each RS_r has as many independent attraction areas ($\beta_{type(PT_p)} RS_r$) as different required PT types in the set of templates. In that case, if a PT_p is inside an attraction area with the same type as that specific PT_p ($\beta_{type(PT_p)} RS_r$), the PT_p is attracted towards RS_r .

Despite the fact that there is no notion of distance-based gradient inside the attraction areas, the attractiveness of a certain $type(PT_p)$ for a certain RS_r , takes into consideration a number of different variables.

The RS's attractiveness for a specific PT_p of type ($type(PT_p)$), for the purpose of this work, is therefore the result of three different factors:

- *ABRatio* (1): represents the ratio between the maximum attractiveness ($\max Att$), maximum buffer size ($\max BSize$), and the number of incoming and stored PTs of type $type(PT_p)$ ($stoIncP$). It is the main factor in the computation of the attractiveness. In essence, as the number of stored and incoming PTs of type $type(PT_p)$ increases, the attractiveness decreases. The *ABRatio* value will then be affected by the *APenalty* and *APPenalty* factors.

$$ABRatio = \max Att(\beta_{type(PT_p)}, RS_r) - \frac{\max Att(\beta_{type(PT_p)}, RS_r) \times stoIncP}{\max BSize(type(PT_p), RS_r)} \quad (1)$$

- *APenalty* (2): expresses the penalty factor that the attractiveness exerted by other RSs ($attF()$) has on the attractiveness of PT_p of type $type(PT_p)$, considering

the number of incoming and stored PTs of that same type. The $attF(type(PT_p), RS_r)$, therefore, represents the maximum value of attractiveness that a resulting PT of any type (for the active templates of RS_r), that has as required PT, a PT of type $type(PT_p)$, is being subjected to by downstream RSs. In other words, given two RSs (RS_1 and RS_2) and considering that RS_2 attracts a PT of type $P1$ that is produced by RS_1 , then the less $P1$ is attracted, the less should RS_1 attract the required PTs to produce $P1$.

$$APenalty = \left(\frac{1}{1 + 1 - \frac{attF(type(PT_p), RS_r)}{\max Att(\beta_{type(PT_p)}, RS_r)}} \right)^{stoIncP} \quad (2)$$

- $APPenalty$ (3): represents the penalty factor that translates the difference in number between the stored PTs of type $type(PT_p)$ and the total number of times it is possible to execute the RS's template(s), that have as required PT, the PT_p of type $type(PT_p)$ ($nTET(type(PT_p), RS_r)$), considering the number of incoming and stored PTs of that same type. This means that the smaller the number of PTs that can be produced is, of any type that requires a PT of type $type(PT_p)$, the less that same $type(PT_p)$ should be attracted.

$$APPenalty = \left(\frac{1}{1 + \frac{\frac{stoP(type(PT_p), RS_r) \times 100}{\max BSize(type(PT_p), RS_r)} - \frac{nTET(type(PT_p), RS_r) \times 100}{\max BSize(type(PT_p), RS_r)}}{100}} \right)^{stoIncP} \quad (3)$$

Hence, the RS 's attractiveness of $type(PT_p)$ can be described by the following function:

$$\beta_{type(PT_p)} RS_i = ABRatio \times APenalty \times APPenalty \quad (4)$$

with

$$stoIncP = stoIncP(type(PT_p), RS_r) = stoP(type(PT_p), RS_r) + incP(type(PT_p), RS_r). \quad (5)$$

In this way, once a PT is deployed in the shopfloor, it starts by checking its surroundings to see if it is attracted towards any RS. On the one hand, if there are several attracting RSs, the PT is drawn towards the most attractive one. If more than one RS is attracting PT with the same force, the PT is drawn towards the closest one. If more than one RS is attracting the PT with the same force and is located at the same distance, the PT chooses randomly to which RS it is attracted to. On the other hand, if a PT is deployed in the shopfloor

and it is not attracted by any RS (due to critical failure or unplug of the RS), then the PT shall start moving randomly, as fireflies in nature do. This 'random walk' is performed during a certain period within which if an attractive RS is not found the PT will remove itself from the shopfloor. The 'random walk', as demonstrated in previous iterations of this work (Dias Ferreira et al. 2013), ensures the fulfilment of the orders independently of the shopfloor layout, as long as the necessary RSs are eventually plugged. It might not meet the deadlines, and for that reason the deployed methodology mentioned in Sect. 4.6 should be followed, but it will nevertheless ensure the convergence of the system.

This brings us to the fact that self-organisation only makes sense in the presence of redundancy. With no redundancy, traditional highly optimize systems are rather more efficient and, therefore, more profitable to adopt.

Implementation

Although BIOSOARM has been described in a platform agnostic way, its implementation requires a minimal level of technical requirements, such as: an object-oriented programming language and a networked environment which supports the virtual representation, extension and interaction of the cyber-physical components.

For the purpose of this work, the present BIOSOARM architecture and test cases were developed in JAVA using the JADE platform. JADE is an agent framework which supports and implements all the main multi-agent functionalities. The attractiveness mechanism as well as all the other interactions are abstracted and supported through direct interaction message patterns (based on FIPA ACLMessage Performative REQUEST and INFORM messages), which explore JADE's asynchronous message paradigm.

Furthermore, due to hardware limitations, the validation and testing scenarios used a customized modular automation simulation environment that simulates a conveyor-based shopfloor. The simulation tool used was also implemented in JAVA and it is partially described in Ribeiro et al. (2015).

Testing scenarios

In order to demonstrate the versatility of the BIOSOARM architecture, two different testing scenarios were devised and considered.

Flow line scenario

The first testing scenario to be considered is an "assembly line" that simulates the production of a car (Fig. 5). It is a fictional scenario and was devised in an attempt to test BIOSOARM under conditions that simulate a more tra-

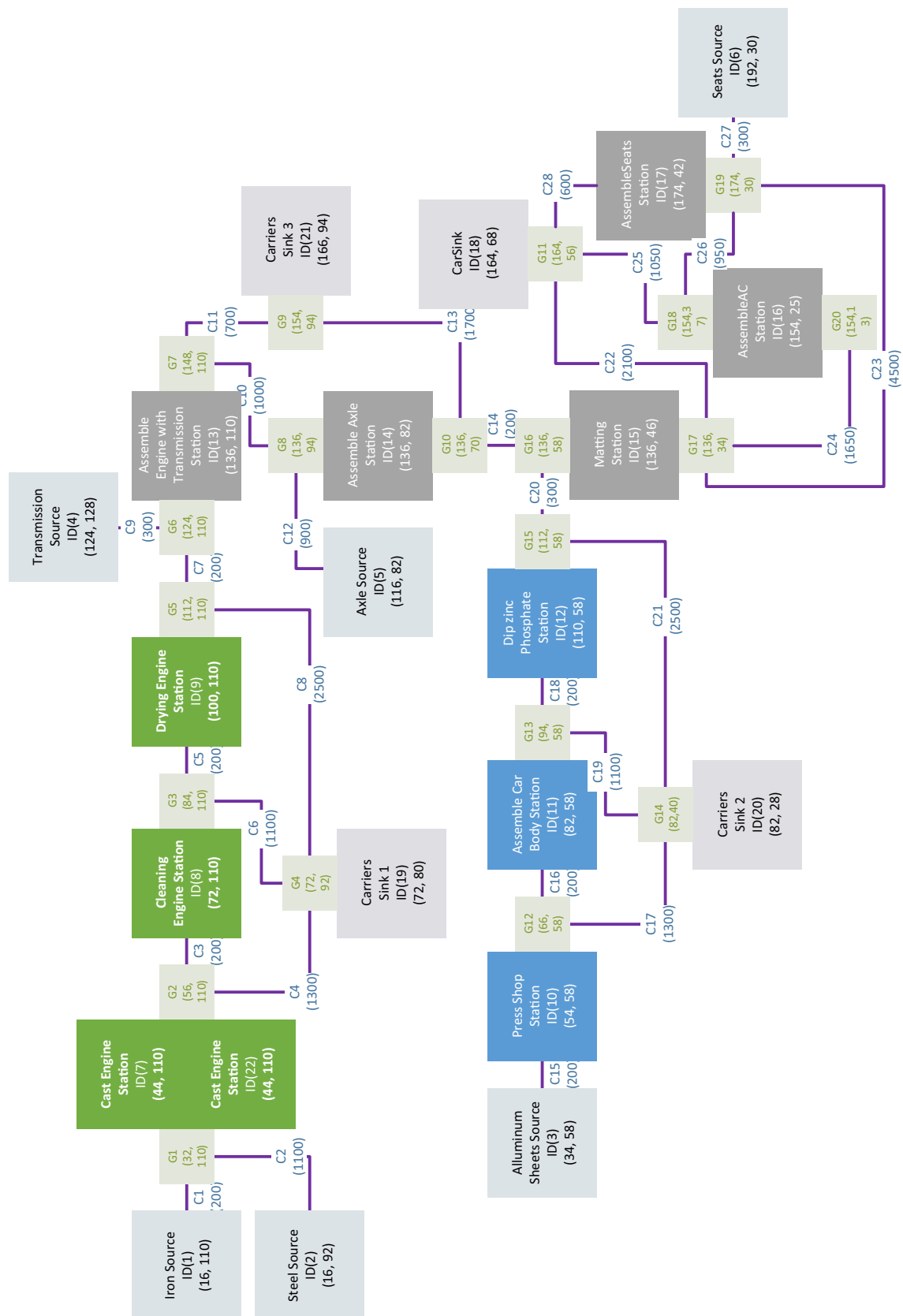


Fig. 5 Testing scenario 1

ditional production line. In this way, the authors aim to understand the performance of BIOSOARM, but also how the system reacts as a whole to perturbations such as dynamic plug and unplug of components, with reference to scenarios commonly found in nowadays production facilities.

The proposed assembly line is composed by two branches (engine and body) that converge into a third main branch, where the final assembly is performed. As depicted in Fig. 5, the engine branch is composed by two RSos, four RSt, five TGas, eight TCos and one RSi. At an initial stage, RSt₂₂ is not present and it is only plugged in some test cases, as described in Sect. 7. The body branch has a similar composition, but only one RSo and four TGas instead. Finally, the main branch is composed by three RSos, five RSts, eleven TGas, thirteen TCos and two RSis. For all purposes, all the RSts and TGas have the same length as a TCa (20 distance units). The conveyors have different lengths [under the TCo name in (Fig. 5)]. The speed of the transport system is constant and it is rated at 20 distance units/time.

In short, the engine branch feeds the engines to the main branch where they are coupled with the transmission. Then, the axle is added and the resulting PT is then mated with the body of the car. Next, there is an optional RSt where the air conditioning (AC) can be added, which then leads to the assembly of the seats RSt. After the assembly of the seats the car is ready and can be removed from the assembly line.

The templates in Table 2 were, therefore, defined, in order to generate the required flows to ultimately produce the car in its two available variants (with/out AC).

As described in the Table 2, the first column represents the ID of the template; in the second column, the ID of the RSt that owns that template is presented; then it comes the required PT(s) ID; followed by the skill and the respective processing time, in time units; and finally, the resulting PT(s) from the execution of that template. The respective PTs ID can be matched with the respective PT in Table 3.

“Job Shop”-like scenario

The second testing scenario is, instead, a more “Job Shop”-like shopfloor (Fig. 6), that simulates the assembly of four different and fictional products (FP).

In this case, the different RSts are able to produce more than one type of PTs. Hence, different product flows will go through the same RSts. Particularly, three of the products flows go through two RSts, while one will go through three RSts. FP1 goes through RSt₄ and RSt₆, FP2 goes through RSt₅ and RSt₇, FP3 goes through RSt₄, RSt₅ and RSt₇, and finally FP4 goes through RSt₅ and RSt₁₀. Although RSt₁₀ is present in Fig. 6, it is never present at the initial state of the systems. The RSt is instead added in real time, if FP4 is to be produced.

Table 2 Flow line templates

ID	RS ID	Req. PTs ID	Skill (processing time)	Res. PTs ID
1	1		FeedIron(720)	1
2	2		FeedSteel(720)	2
3	3		FeedAluminumSheets(720)	3
4	4		FeedTransmission(720)	4
5	5		FeedAxle(720)	5
6	6		FeedSeats(720)	6
7	7/22	1,2	Cast(1000)	7
8	8	7	Clean(150)	8
9	9	8	Dry(200)	9
10	10	3	Press(100)	10
11	11	10	AssembleCarBody(250)	11
12	12	11	DipZincPhosphate(200)	12
13	13	4,9	AssEngWTran(200)	13
14	14	5,13	AssembleAxle(150)	14
15	15	12,14	Mating(300)	15
16	16	15	AssembleAc(300)	16
17	17	6,15	AssembleSeats(500)	17
18	17	6,16	AssembleSeats(500)	18
19	18	17	RemoveCar(10)	
20	18	18	RemoveCar(10)	

Table 3 Parts mapping

ID	Part
1	Iron
2	Steel
3	Aluminium sheets
4	Transmission
5	Axle
6	Seats
7	Engine block (EB)
8	Cleaned EB
9	Final EB
10	Body part
11	Car body
12	Final car body
13	Engine + transmission
14	Powertrain
15	Powertrain and chassis (P&C)
16	P&C with AC
17	Car without AC
18	Car with AC

The present testing scenario is, therefore, composed from an initial phase by three RSos, four RSts, two RSis, eleven TGas, and fourteenth TCos. Again, if required, another RSt (RSt₁₀) may be plugged to contemplate the production of FP4.

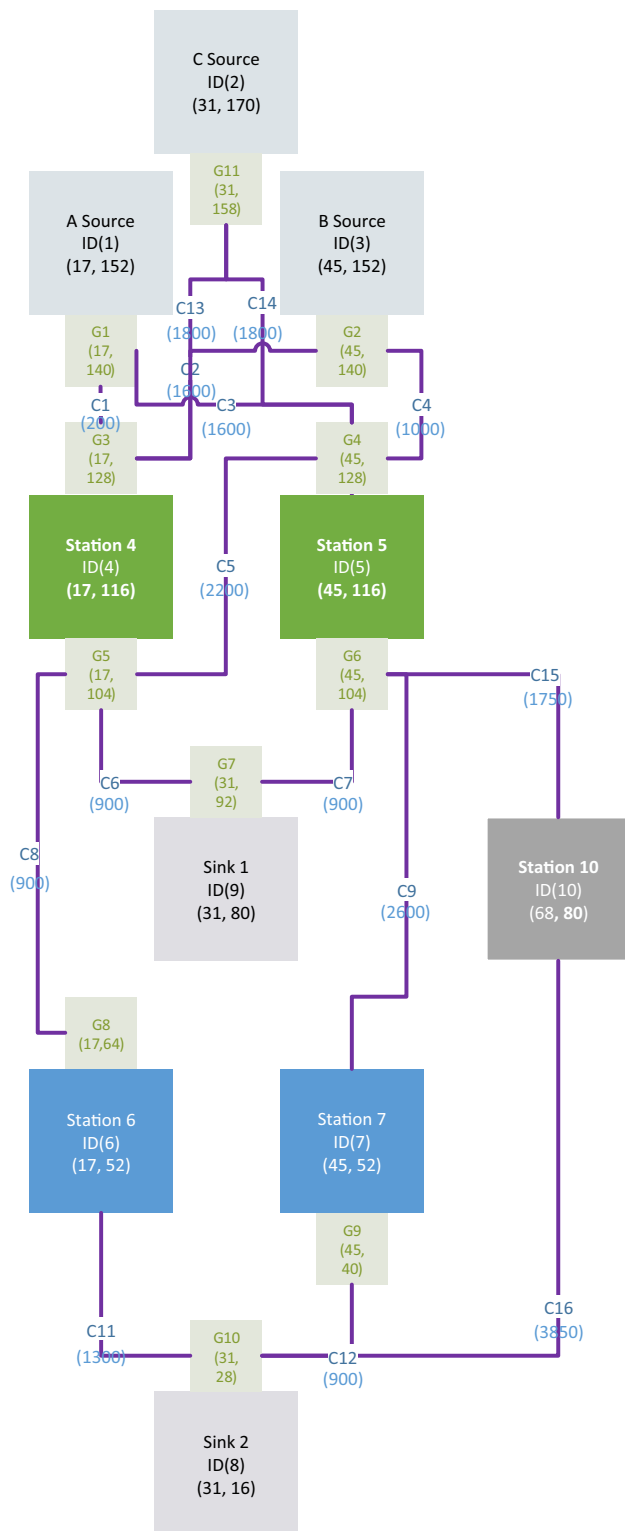


Fig. 6 Testing “Job Shop”-like shopfloor

Similarly to the previous scenario, the RSts, TGas and TCAs have the same properties. Moreover, the TCos have different lengths as depicted in Fig. 6, with a transfer speed also rated at 20 distance units/time units.

Table 4 “Job Shop”-like shopfloor templates

ID	RS ID	Req. PTs ID	Skill (processing time)	Res. PTs ID
1	1		FeedA(10)	a
2	3		FeedB(10)	b
3	2		FeedC(10)	c
4	4	a,c	RSt4Skill1(1500)	p1
5	4	a,b	RSt4Skill2(1000)	p3
6	5	b,c	RSt5Skill1(1000)	p2
7	5	p3	RSt5Skill2(1250)	p4
8	5	a,b	RSt5Skill3(1750)	p5
9	6	p1	RSt6Skill1(1500)	FP1
10	7	p2	RSt7Skill1(1000)	FP2
11	7	p4	RSt7Skill2(1500)	FP3
12	10	p5	RSt10Skill1(2000)	FP4
13	8	FP1	RemoveProduct(10)	
14	8	FP2	RemoveProduct(10)	
15	8	FP3	RemoveProduct(10)	
16	8	FP4	RemoveProduct(10)	

In this scenario, the PT flows are the result of the set of templates in Table 4.

Test cases

For the two different testing scenarios, two different test cases were considered.

For the flow line, the first test case, aims to test BIOSOARM’s performance under normal operational conditions. This will establish a baseline which will allow to compare the system’s performance, under more dynamic conditions, tested in the second test case. Hence, in the second test case, the dynamic conditions are simulated through the introduction of a new RSt (RSt₂₂), able to execute the same templates as the bottleneck RSt (RSt₇), during the ramp-up of the system (period during which the system is already under stress).

Similarly, for the “Job Shop”-like scenario, the first test case also aims to test BIOSOARM’s performance under normal operational conditions and the second under more dynamic conditions. However, in this scenario, the dynamic conditions are simulated by making the system to accommodate the production of a new final PT, enabled through the introduction of a new RSt (RSt₁₀), at random stages of the test.

For all test cases, the rate at which the Pts are introduced in the shopfloor is dependent on the physical properties and characteristics of the RSos, as well as on the attraction of the PTs. Also, the elements named as carrier sinks are meant to absorb and return carriers to the sources. This is necessary for RSts that require more than one PT to execute a process. As

each RSt can only hold one carrier, it is not possible to deliver another PT until the previous carrier is released. Hence, the first carrier leaves the RSt empty and it is absorbed by the carrier sinks which will then return them to the sources.

Finally, it is important to stress that the performance assessment of the proposed architecture for the different testing scenarios presented in Sect. 7 is done through the analysis of the system makespan in relation to the minimum makespan boundary. The minimum makespan boundary corresponds to the total processing time of the bottleneck RSt plus the total minimum processing time, after the bottleneck, necessary to complete a product among the product types available. Although in some cases the minimum makespan boundary may not represent the minimum feasible makespan, it can never be lower and for that reason it will be used as a reference. This approach was followed as there are no well defined benchmarks that provide the grounds for a meaningful and insightful comparison. This claim is supported by the different approaches used in the different research works in the field (Peeters et al. 2001; Mařík et al. 2005; Leitão and Restivo 2005; Colombo et al. 2006; Ribeiro and Barata 2012; Wang and Koren 2012; Akillioglu et al. 2013; Pach et al. 2014; Ribeiro et al. 2015).

Results and assessment

For the flow line testing scenario, the first test case considered was a simple continuous production of 100 parts of both type 17 and 18 (without or with AC respectively), for a total of 200 parts, intended to test BIOSOARM under normal operational conditions. In this test case, RSt₂₂ was not considered. For each test case, 10 tests were performed.

Figure 7 depicts the resulting processing behaviour of the different branches in one of the test runs. For readability purposes only the 30,000 initial time units are presented. For each time unit a RS can only assume two possible values, 0 if it is idle or 1 if it is busy. Hence, as it is possible to observe the repetitive patterns of the different branches demonstrates the coherent and stable behaviour of the system.

Particularly for the first branch, it is possible to see that the processing map stabilizes immediately after the production of the first PT. This is explained due to the fact that RSt₇ is the bottleneck RSt which constraints the operation of the downstream RSts. In both the body and the main branch, the ramp-up time is around 21,000 and 24,000 time units respectively. For the purpose of this work, ramp-up times translates the time since the start of an operation of the system or system component, until it reaches a stable operational state. These higher ramp-up times of these branches represents the adjustment of their production rate to the engine branch production rate. After this period, both the body and the main branch also present a clear repetitive pattern which highlights

the coherent and stable behaviour of the system. This behaviour is verified in all the test simulations.

These results are also supported by the cycle times (the cycle time is the sum of the time a RS is idle with the time it takes to process the next PT) detailed in Table 5. As it is shown, the different RSts have similar cycle times (the cycle time is constrained by the bottleneck RSt, which in this case is RSt₇ with the processing time of 1000) close to 1000 time units, which once again reinforces the system's stability under flow line scenarios. Only the RSt₁₆ has a cycle time twice as big as the other RSts, which is expected since RSt₁₆ only processes half of the parts.

Figure 8 shows the resulting makespan for the 10 different test runs (green). The results are on average only 4.6 % higher than the minimum makespan boundary (201,500 time units), which together with the very low standard deviation reinforces the quality of the results achieved.

In order to test BIOSOARM's adaptability, self-organisation, plug-ability and scalability, a second test case was designed considering more dynamic conditions. In this case, RSt₂₂ is plugged into the system during ramp-up. This will enable the analysis of the impact of decreasing the system's bottleneck and of adding a RSt during an already sensitive period. It is important to mention that the addition of a new RSt may require the adjustment of the TS and of other RSs. In this particular case, in order to plug RSt₂₂, RSt₇ needs to be unplugged and re-plugged together with the new RSt₂₂. Also, TGa₁ (G1) and TGa₂ (G2) need to be unplugged, reconfigured and re-plugged to support both RSts. The same testing parameters such as processing times, number of products and runs have been considered.

The results, depicted in Fig. 8 (red), show that for most of the test runs the makespan presents considerable improvements, as expected. This gain highlights the robustness and the system's ability to scale production. Through the processing maps of the different branches, it was also possible to conclude that the system managed to converge very quickly to a stable operating state. This demonstrates the architecture's ability to adapt its behaviour to new circumstances. Nevertheless, the only way to quantify this adaptation period is through the analysis of the output of the system. Since the addition of RSt₂₂ requires RSt₇, TGa₁ and TGa₂ to be removed, there is a short period during which the production is halted, as there is no production of cast engines. That period took on average 2403.74 time units, which represents 1.38 % of the total production time. It is, nevertheless, necessary to mention that the logical procedure of adding an architectural components and the simulation runs in different time scales. While the simulation runs at very high speeds (404 time units/s on average), the logical plug of a component works in normal time. This means that the adaptation in a real system should be, in principle, considerably faster and, therefore, its impact almost negligible.

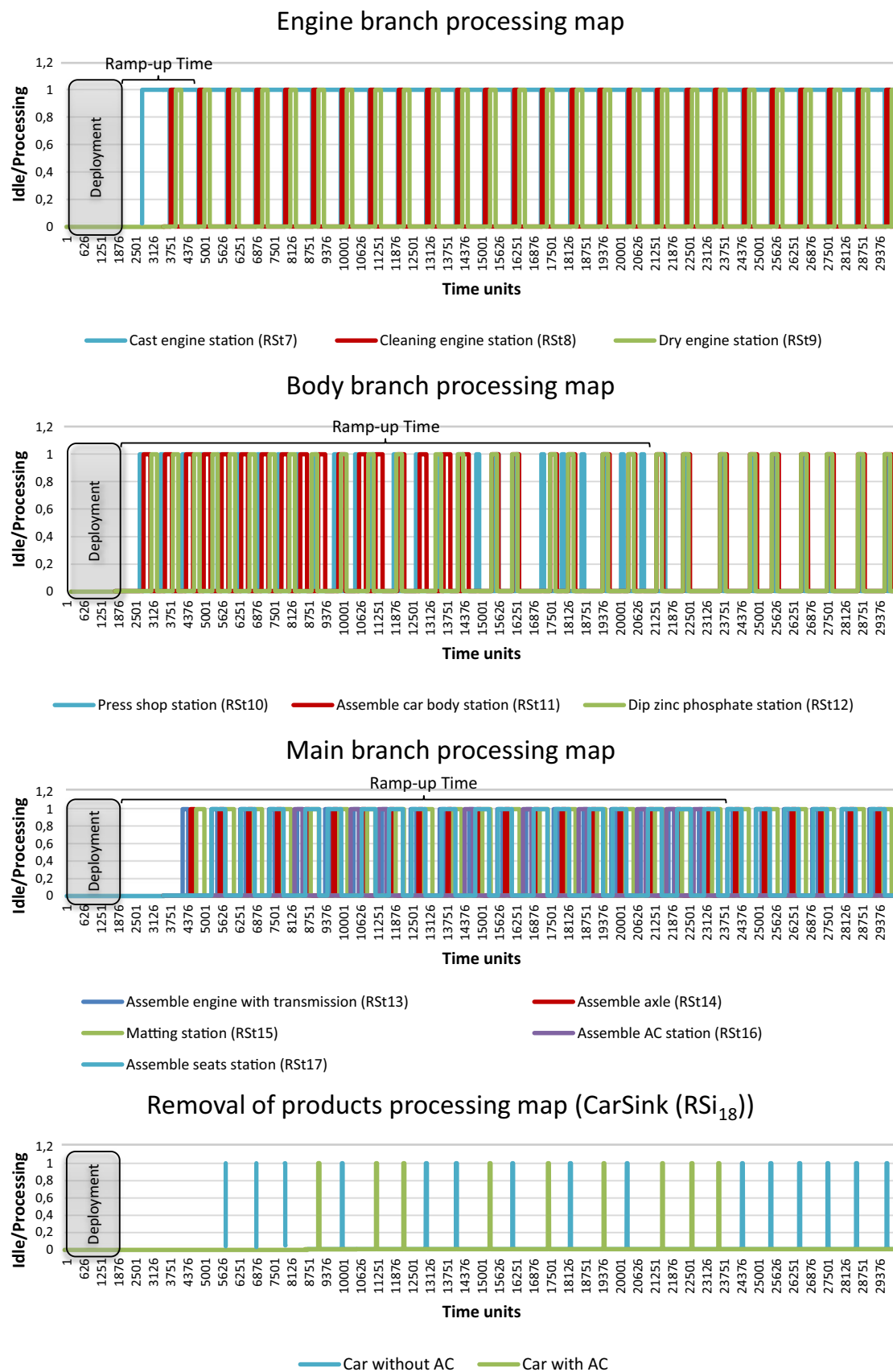
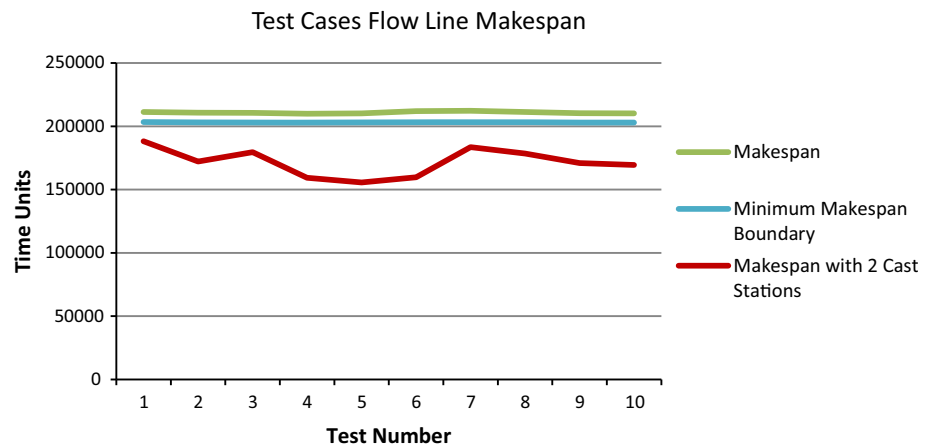
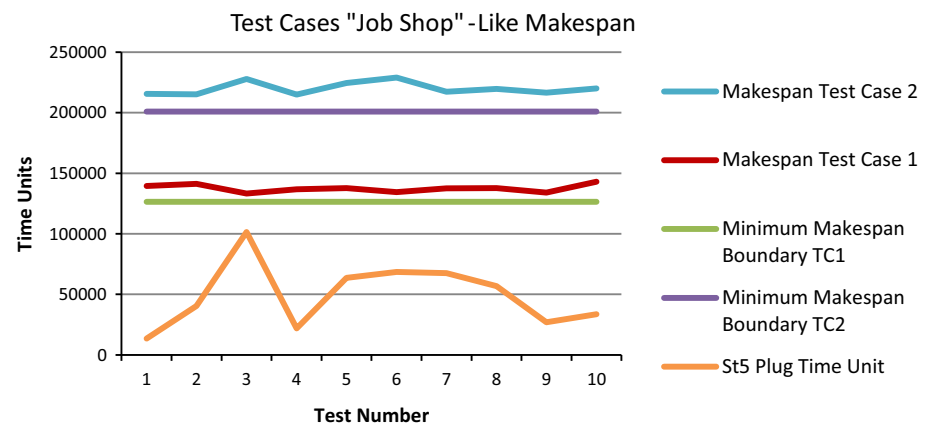


Fig. 7 Flow line processing map

Table 5 Flow line cycle times

	RSt ₇	RSt ₈	RSt ₉	RSt ₁₀	RSt ₁₁	RSt ₁₂	RSt ₁₃	RSt ₁₄	RSt ₁₅	RSt ₁₆	RSt ₁₇	RSt ₁₈
Sample mean	1037.4	1037.3	1037.3	998.3	1012.1	1025.5	1037.7	1037.7	1037.7	2075.8	1038.4	1038.4
S. SD	3.34	3.43	3.43	3.53	3.60	3.75	3.62	3.62	3.62	15.47	3.60	3.60
No. tests	10	–	–	–	–	–	–	–	–	–	–	–
Confidence	95 %	–	–	–	–	–	–	–	–	–	–	–
Significance	5 %	–	–	–	–	–	–	–	–	–	–	–
Critical t	2,262	–	–	–	–	–	–	–	–	–	–	–
Upper limit	1039.8	1039.8	1039.8	1000.8	1014.68	1028.18	1040.3	1040.3	1040.3	2086.9	1040.97	1040.97
Lower limit	1035.0	1034.8	1034.8	995.8	1009.5	1022.8	1035.1	1035.1	1035.1	2064.7	1035.8	1035.8

Fig. 8 Makespan of the 10 test runs for 100 products of type 17 and 100 products of type 18**Fig. 9** Test cases “Job Shop”-like shopfloor makespans

For the testing of the “Job Shop”-like shopfloor scenario, another two different test cases were considered. As in the flow line scenario, the first test case aims to test BIOSARM’s ability to tackle “Job Shop”-like scenarios during normal operation. For this reason, in this case RSt₁₀ was not considered. A set of 10 different runs were performed in which 50 units of three different product types (FP1, FP2, FP3) were produced. As depicted in the Table 4, each RSt can have more than one skill. Also, each final product type requires different

PTs which consequently leads to the emergence of different flows in the shopfloor.

Figure 9 depicts the obtained makespan (red) compared again to the minimum makespan boundary (green; 126,500 time units). In the results presented, it is possible to observe that the obtained makespan maintains a relatively stable value throughout the 10 different runs, averaging at 137,556.6 time units with 133,345 as the lower makespan for test number 3. According to test run 3, the system is therefore able to present

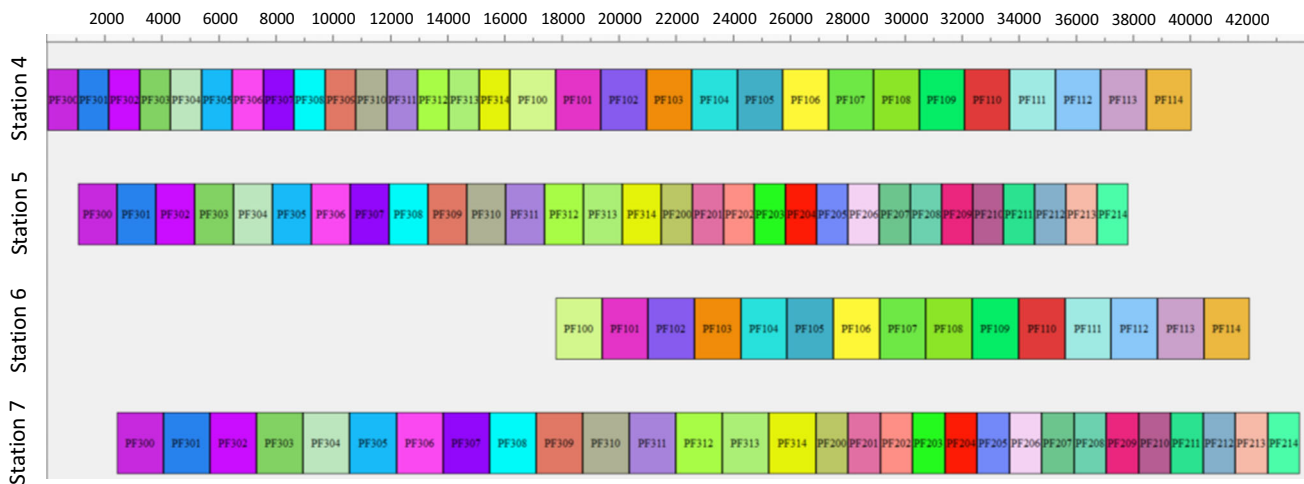


Fig. 10 LEKIN schedule using the lowest processing time rule for test case 1 of the “Job Shop”-like scenario

results only 5.4% over the optimum, which highlights the good performance of BIOSOARM in managing “Job Shop”-like scenarios.

Furthermore, a smaller version of the same test case was devised in order to better compare BIOSOARM’s performance against more traditional approaches. For this purpose a scheduling tool named LEKIN was used. However, only 15 units of each product type were produced due to limitations related with LEKIN. Hence, for the ten different runs, the obtained average makespan was 46,952.1 time units with the minimum obtained value of 43,851 time units. In LEKIN, the lowest makespan achieved was 43,890 time units for the LPT (Lowest Processing Time), as depicted in Fig. 10, which is only slightly higher than the best test of BIOSOARM. For the available heuristics, however, the minimum value was slightly lower at 42,540 time units for the Shifting Bottleneck/Tmax. Nevertheless, these results demonstrate, once again, that despite being highly focused on change, uncertainty, adaptation and plug-ability BIOSOARM is able to obtain results close to the ones achieved by more traditional and dedicated approaches.

Finally, for the last test case of the “Job Shop”-like scenario, a new product (FP4) is randomly introduced during runtime. In this way, the authors aim to simulate highly dynamic conditions where the system will have to accommodate the production of a new product at different stages of its operational phase. The new product FP4 requires the execution of two skills. The first available in RSt₅ and the second not readily available in the shopfloor. Hence, a new station, in this case RSt₁₀ (Fig. 6) with skill RSt₁₀Skill1 needs to be plugged.

Through the makespan (blue) in Fig. 9 it is possible to conclude that the system presents again a stable behaviour. Performance wise the results are 7.5% off the optimal case. It has nevertheless, to be considered that the new order and

components are introduced randomly in different stages of the system’s operation. In Fig. 9, the RSt₅ Plug Time Unit graph (orange) represents the time unit at which the new order was introduced. By introducing the new order and consequent station at a later stage, the system has less ground to self-optimize as the bottleneck shifts from RSt₁ and RSt₄ to RSt₂ which is reflected in the system makespan. Nevertheless, independently of the plugging time the system was still able to present relatively similar performances, which demonstrates its ability to distribute the workload through the different RSts, in a self-organising way. The system also did not assume any undesired behaviour. Instead, it maintained its coherent execution and level of performance, demonstrating once again its plug-ability, robustness and ability to deal with uncertainty.

The above results clearly demonstrate not only BIOSOARM’s ability to deal with disturbances and uncertainty but also its performance, that despite not being globally optimal, it is almost in par with the levels of more traditional approaches.

One of the most important and differentiating aspects of BIOSOARM, as stressed before, is the emergence of a production pattern supported by the bio-inspired architectural structure and self-organising behaviour, that isolates the system’s operation from unexpected events. In other words, the fully decoupled nature of BIOSOARM’s structural components minimizes the impact that failures (removal of RSs, removal of parts, etc.) have on the system’s operation. Hence as the results showed, BIOSOARM managed to adapt itself and seamlessly recover from any unexpected event while maintaining the level of performance.

With the results in mind and in light of the ERCs, it is, therefore, possible to conclude that the main constructs of BIOSOARM have a high degree of autonomy and decision making capabilities. In fact, the virtualization and individual-

ization of both RSs and PTs alike, enables the prompt reaction to the different environmental conditions in a robust manner, without a higher level coordination. Despite the introduction of random events (plug/unplug of components), the architecture showed a coherent and efficient adaptation, allowing it to maintain acceptable levels of performance. Finally, the seamless addition/removal of components enables the evolution of BIOSOARM systems, in its weakest sense, as demonstrated by the introduction of new products and components not supported by the initial system.

Conclusions

In the present work, BIOSOARM, a heavily bio-inspired reference architecture for highly dynamic production environments, is presented. BIOSOARM aims to adhere to the modern architectural-control requirements and characteristics for the future factories, by adopting and implementing control structures and self-organising mechanisms that one may find in collective-biological-like systems, while taking into account the limitations of the manufacturing environment. For this purpose, BIOSOARM proposes a system-wide self-optimising/organising response that promotes the emergence of production flow patterns. Such production approach is considerably different from the more “traditional” product-driven control mechanisms. It relies on all the different instantiated decoupled architectural constructs to interact through an attraction mechanism that simultaneously ensures the levels of performance and decouples the system’s operation from unexpected events. The use of swarm-based interaction patterns enacts components to become fully decoupled and consequently not directly constrained by other components. This is a critical aspect in order to meet the ERCs. Any changes or disturbances have, therefore, little or no repercussion on the system’s operation, improving the system robustness. It also renders BIOSOARM a highly pluggable architecture. Furthermore, the emergence of the production flows greatly improves the deployment and reconfiguration of the shopfloor, since it does not require the generation of possibly highly complex workflows.

BIOSOARM is self-organising mechanism agnostic and consequently any swarm-based self-organising behaviour can be instantiated by it. Nevertheless, for the purpose of this work the optimization Firefly Algorithm was used.

It is, however, important to stress that highly distributed heterarchical control architectures are still relatively scarce and, in this sense, the tests and the results obtained are of extreme importance to understand the potential and limitations of such approaches. Two different testing scenarios were, therefore, devised in order to test BIOSOARM and its adherence to the proposed ERCs. The first was dimensioned and inspired in a car assembly line. The results showed that

the system presents a coherent and robust behaviour, both under normal as well as in dynamic and uncertain conditions. Moreover, the system also presented a very acceptable performance level compared to the optimal solutions. For the second testing scenario, a more “Job Shop”-like shopfloor was used. Again, the system presented not only a very good performance as well as a coherent and robust behaviour both under normal and uncertain conditions. For this testing scenario, a smaller test case was also devised to compare the performance towards more traditional scheduling approaches. The system managed in some test runs to marginally outperform the more traditional scheduling rules. It was, nevertheless, surpassed through the adoption of an heuristic scheduling approach but also only for a minimal margin. In this sense, these results allowed to objectively evaluate the proposed architecture and testify its good performance but, above all, its great adherence to the ERCs.

Despite the obtained promising results, it is the authors’ belief that more testing scenarios need to be studied under more extreme uncertainty conditions.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Akiloglu, H., Ferreira, J., & Onori, M. (2013). Demand responsive planning: Workload control implementation. *Assembly Automation*, 33(3), 247–259.
- Arai, T., Aiyama, Y., Maeda, Y., Sugi, M., & Ota, J. (2000). Agile assembly system by “plug and produce”. *CIRP Annals-Manufacturing Technology*, 49(1), 1–4.
- Asif, M., de Bruijn, E. J., Fisscher, O. A., & Steenhuis, H. J. (2008). Achieving sustainability three dimensionally. In *Proceedings of the 4th IEEE international conference on management of innovation and technology, ICMIT* (pp. 423–428). IEEE.
- Babiceanu, R. F., & Chen, F. F. (2006). Development and applications of holonic manufacturing systems: A survey. *Journal of Intelligent Manufacturing*, 17(1), 111–131.
- Barata, J., & Camarinha-Matos, L. M. (2003). Coalitions of manufacturing components for shop floor agility-the cobasa architecture. *International Journal of Networking and Virtual Organisations*, 2(1), 50–77.
- Barbosa, J., Leitão, P., Adam, E., & Trentesaux, D. (2013). Structural self-organized holonic multi-agent manufacturing systems. In *Industrial applications of holonic and multi-agent systems* (pp 59–70). Springer.
- Bedau, M. A. (2008). Is weak emergence just in the mind? *Minds and Machines*, 18(4), 443–459.
- Brennan, R. W., & Norrie, D. H. (2001). Agents, holons and function blocks: Distributed intelligent control in manufacturing. *Journal of Applied Systems Studies*, 2(1), 1–19.
- Colombo, A. W., Schoop, R., & Neubert, R. (2006). An agent-based intelligent control platform for industrial holonic manufacturing

- systems. *IEEE Transactions on Industrial Electronics*, 53(1), 322–337.
- De Wolf, T., & Holvoet, T. (2004). Emergence and self-organisation: A statement of similarities and differences. *Engineering Self-Organising Systems*, 3464, 1–15.
- Dias Ferreira, J., Ribeiro, L., Onori, M., & Barata, J. (2013). Bio-inspired self-organising methodologies for production emergence. In *IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 3835–3841). IEEE.
- Dias-Ferreira, J., Ribeiro, L., Akillioglu, H., Neves, P., Maffei, A., & Onori, M. (2014). Characterization of an agile bio-inspired shop-floor. In *12th IEEE international conference on industrial informatics (INDIN)* (pp. 404–410). IEEE.
- Dilts, D. M., Boyd, N. P., & Whorms, H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*, 10(1), 79–93.
- Dobzhansky, T., et al. (1970). *Genetics of the evolutionary process* (Vol. 139). New York: Columbia University Press.
- Farid, A. M., & Ribeiro, L. (2015). An axiomatic design of a multiagent reconfigurable mechatronic system architecture. *IEEE Transactions on Industrial Informatics*, 11(5), 1142–1155.
- Félix, M. A., & Wagner, A. (2008). Robustness and evolution: Concepts, insights and challenges from a developmental model system. *Heredity*, 100(2), 132–140.
- Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge: MIT press.
- Frank, U., Papenfort, J., & Schütz, D. (2011). Real-time capable software agents on iec 61131 systems—developing a tool supported method. In *Proceedings of the 18th IFAC World Congress*. Mailand.
- Futuyma, D. J. (1998). *Evolutionary biology* (3rd ed.). Sunderland, MA: Sinauer Associates.
- Haken, H. (2006). *Information and self-organization: A macroscopic approach to complex systems*. Berlin: Springer Science & Business Media.
- Jazdi, N. (2014). Cyber physical systems in the context of industry 4.0. In: *IEEE international conference on automation, quality and testing, robotics* (pp. 1–4). IEEE.
- Josuttis, N. M. (2007). *SOA in practice: The art of distributed system design*. Sebastopol: O'Reilly Media Inc.
- Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). In Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group. Forschungsunion
- Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., et al. (1999). Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology*, 48(2), 527–540.
- Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7), 979–991.
- Leitão, P., & Restivo, F. (2005). Experimental validation of adacor holonic control system. In: *International conference on industrial applications of holonic and multi-agent systems* (pp. 121–132). Springer.
- Leitão, P., & Restivo, F. (2006). Adacor: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, 57(2), 121–130.
- Leitão, P., Barbosa, J., & Trentesaux, D. (2012). Bio-inspired multi-agent systems for reconfigurable manufacturing systems. *Engineering Applications of Artificial Intelligence*, 25(5), 934–944.
- Lepuschitz, W., Zoitl, A., Valleé, M., & Merdan, M. (2011). Toward self-reconfiguration of manufacturing systems using automation agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 41(1), 52–69.
- Łukasik, S., & Zak, S. (2009). Firefly algorithm for continuous constrained optimization tasks. In *Computational collective intelligence. Semantic Web, social networks and multiagent systems* (pp. 97–106). Springer.
- MacDougall, W. (2014). Industrie 4.0 smart manufacturing for the future. http://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf.
- Mařík, V., Vrba, P., Hall, K. H., & Maturana, F. P. (2005). Rockwell automation agents for manufacturing. In *Proceedings of the fourth international joint conference on autonomous agents and multiagent systems* (pp 107–113). ACM.
- Maturana, F., Shen, W., & Norrie, D. H. (1999). Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, 37(10), 2159–2173.
- McFarlane, D., Giannikas, V., Wong, A. C., & Harrison, M. (2013). Product intelligence in industrial control: Theory and practice. *Annual Reviews in Control*, 37(1), 69–88.
- McFarlane, D. C., & Bussmann, S. (2000). Developments in holonic production planning and control. *Production Planning & Control*, 11(6), 522–536.
- McFarlane, D. C., & Bussmann, S. (2003). Holonic manufacturing control: Rationales, developments and open issues. In S. M. Deen (Ed.), *Agent-based manufacturing* (pp. 303–326). Berlin, Heidelberg: Springer.
- Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, 17, 9–13.
- Monostori, L., Váncza, J., & Kumara, S. R. (2006). Agent-based systems for manufacturing. *CIRP Annals-Manufacturing Technology*, 55(2), 697–720.
- Onori, M. (2002). Evolvable assembly systems: A new paradigm? In *33rd international symposium on robotics*.
- Onori, M., Barata, J., & Frei, R. (2006). Evolvable assembly systems basic principles. In W. Shen (Ed.), *Information technology for balanced manufacturing systems*. IFIP international federation for information processing, (Vol. 220, pp. 317–328). Boston: Springer.
- Pach, C., Berger, T., Bonte, T., & Trentesaux, D. (2014). Orca-fms: A dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. *Computers in Industry*, 65(4), 706–720.
- Peeters, P., Van Brussel, H., Valckenaers, P., Wyns, J., Bongaerts, L., Kollingbaum, M., et al. (2001). Pheromone based emergent shop floor control system for flexible flow shops. *Artificial Intelligence in Engineering*, 15(4), 343–352.
- Pine, B. J. (1999). *Mass customization: The new frontier in business competition*. Boston, MA: Harvard Business Press.
- Rafferty, J. P. (2011) *New thinking about evolution*. New York : Britannica Educational Pub. : in association with Rosen Educational Services.
- Ren, L., Zhang, L., Wang, L., Tao, F., & Chai, X. (2014). Cloud manufacturing: Key characteristics and applications. *International Journal of Computer Integrated Manufacturing*, 1–15. (ahead-of-print)
- Rey, G. Z., Pach, C., Aissani, N., Bekrar, A., Berger, T., & Trentesaux, D. (2013). The control of myopic behavior in semi-heterarchical production systems: A holonic framework. *Engineering Applications of Artificial Intelligence*, 26(2), 800–817.
- Ribeiro, L., & Barata, J. (2012). Ims 10validation of a co-evolving diagnostic algorithm for evolvable production systems. *Engineering Applications of Artificial Intelligence*, 25(6), 1142–1160.
- Ribeiro, L., Rosa, R., Barata, J. (2012a). A structural analysis of emerging production systems. In *10th IEEE international conference on industrial informatics (INDIN)* (pp. 223–228). IEEE.
- Ribeiro, L., Rosa, R., Cavalcante, A., Barata, J. (2012b). Iade-ideas agent development environment: Lessons learned and research directions. In *4th CIRP conference on assembly technologies and systems*, (pp 91–94).

- Ribeiro, L., Rocha, A., Veiga, A., & Barata, J. (2015). Collaborative routing of products using a self-organizing mechatronic agent framework simulation study. *Computers in Industry*, 68, 27–39.
- Schutz, D., Wannagat, A., Legat, C., & Vogel-Heuser, B. (2013). Development of plc-based software for increasing the dependability of production automation systems. *Industrial Informatics, IEEE Transactions on*, 9(4), 2397–2406.
- Shen, W., & Norrie, D. H. (1998). A hybrid agent-oriented infrastructure for modeling manufacturing enterprises. *Proceedings of KAW, CiteSeer*, 98, 1–19.
- Shen, W., Hao, Q., Yoon, H. J., & Norrie, D. H. (2006). Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics*, 20(4), 415–431.
- Spearman, M. L., & Zazanis, M. A. (1992). Push and pull production systems: Issues and comparisons. *Operations Research*, 40(3), 521–532.
- Tharumarajah, A., Wells, A., & Nemes, L. (1998). Comparison of emerging manufacturing concepts. In *IEEE international conference on systems, man, and cybernetics* (Vol. 1, pp. 325–331). IEEE.
- Trentesaux, D. (2009). Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, 22(7), 971–978.
- Ueda, K. (1992). A concept for bionic manufacturing systems based on dna-type information. In *Proceedings of the IFIP TC5/WG5. 3 Eight international PROLAMAT conference on human aspects in computer integrated manufacturing* (pp 853–863). North-Holland Publishing Co.
- UNA of Sciences. (2015). Definitions of evolutionary terms. <http://www.nas.edu/evolution/Definitions.html>.
- Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference architecture for holonic manufacturing systems: Prosa. *Computers in Industry*, 37(3), 255–274.
- Verstraete, P., Saint Germain, B., Valckenaers, P., Van Brussel, H., Belle, J., & Hadeli, H. (2008). Engineering manufacturing control systems using prosa and delegate mas. *International Journal of Agent-Oriented Software Engineering*, 2(1), 62–89.
- Vyatkina, V. (2011). Iec 61499 as enabler of distributed and intelligent automation: State-of-the-art review. *IEEE Transactions on Industrial Informatics*, 7(4), 768–781.
- Vyatkina, V., & IS of America. (2007). IEC 61499 function blocks for embedded and distributed control systems design. In *ISA-international society of automation*. ISBN: 978-1-936007-93-6.
- Wang, W., & Koren, Y. (2012). Scalability planning for reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 31(2), 83–91.
- Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In W. Osamu, & Z. Thomas (Eds.), *Stochastic algorithms: Foundations and applications* (pp. 169–178). Berlin, Heidelberg: Springer.
- Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. University of Cambridge.
- Zambrano, G., Pach, C., Aissani, N., Berger, T., Trentesaux, D. (2011). An approach for temporal myopia reduction in heterarchical control architectures. In *IEEE international symposium on industrial electronics (ISIE)* (pp 1767–1772). IEEE.
- Zbib, N., Pach, C., Salles, Y., & Trentesaux, D. (2012). Heterarchical production control in manufacturing systems using the potential fields concept. *Journal of Intelligent Manufacturing*, 23(5), 1649–1670.
- Zhang, L., Luo, Y., Tao, F., Li, B. H., Ren, L., Zhang, X., et al. (2014). Cloud manufacturing: A new manufacturing paradigm. *Enterprise Information Systems*, 8(2), 167–187.