An Axiomatic Design of a Multi-Agent Reconfigurable Mechatronic System Architecture

Amro M. Farid, *Member, IEEE,* and Luis Ribeiro, *Member, IEEE*

***Abstract*—In recent years, the fields of reconfigurable manufacturing systems, holonic manufacturing systems, and multi-agent systems have made technological advances to support the ready reconfiguration of automated manufacturing systems. While these technological advances have demonstrated robust operation and been qualitatively successful in achieving reconfigurability, their ultimate industrial adoption remains limited. Amongst the barriers to adoption has been the relative absence of formal and quantitative multi-agent system design methodologies based upon reconfigurability measurement. Hence, it is not clear 1.) the degree to which these designs have achieved their intended level of reconfigurability 2.) which systems are indeed quantitatively more reconfigurable and 3.) how these designs may overcome their design limitations to achieve greater reconfigurability in subsequent design iterations. To our knowledge, this paper is the first multi-agent system reference architecture for reconfigurable manufacturing systems driven by a quantitative and formal design approach. It is rooted in an established engineering design methodology called axiomatic design for large flexible engineering systems and draws upon design principles distilled from prior works on reconfigurability measurement. The resulting architecture is written in terms of the mathematical description used in reconfigurability measurement which straightforwardly allows instantiation for system-specific application.**

***Index Terms*—multi-agent system, reconfigurability, reconfigurable manufacturing systems, axiomatic design**

I. Introduction

Manufacturing has become increasingly characterized by continually evolving and ever more competitive marketplaces. In order to stay competitive, manufacturing firms have had to respond with a high variety of products of increasingly short product lifecycle [1], [2]. One particularly pertinent problem is the need to quickly and incrementally adjust production capacity and capability. To fulfill the needs of enterprises with extensive automation, reconfigurable manufacturing systems have been proposed as a set of possible solutions [3].

Definition 1. Reconfigurable Manufacturing System [4]: [A System] designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or regulatory requirements.

Over the last decade, many technologies and design approaches have been developed to enable reconfigurability in manufacturing systems [5]–[7]. Modular and distributed system design has been consistently acknowledged as a pillar of modern automation practices. Standard automation technology has been following this rationale with the introduction of object orientation in the IEC 61131, subsequent agent oriented developments [8], [9] and the development of IEC 61499 [10]. This technological oriented development has also included modular machine tools [11]–[15] and distributed automation [16]–[18]. Additionally, a wide set of IT-based paradigms such as Multi-Agent Systems [19]–[25], Holonic Manufacturing Systems [26]–[29], Evolvable Assembly Systems [30], and Fractal Factories [31] have emerged. They include a number of notable reference architectures including PROSA [32]–[34], HCBA [35] and ADACOR [22], [36]. While these technological advances have demonstrated robust operation and been qualitatively successful in achieving reconfigurability, their ultimate industrial adoption remains limited [37].

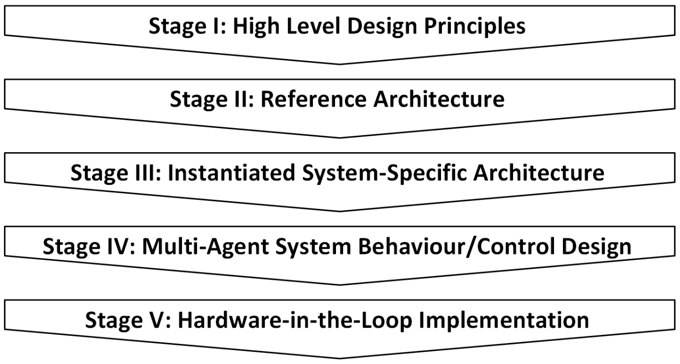

Stage I: High Level Design Principles
Stage II: Reference Architecture
Stage III: Instantiated System-Specific Architecture
Stage IV: Multi-Agent System Behaviour/Control Design
Stage V: Hardware-in-the-Loop Implementation

Fig. 1. A Five Stage MAS Design Methodology

Amongst the barriers to adoption, and the primary motivation for this work, has been the relative absence of quantitative multi-agent system design methodologies based upon reconfigurability measurement. In fact, the research in quantitatively supported design methodologies has been lagging behind in comparison with more technology grounded research specially when compared with the developments around the IEC 61499 for which several design patterns have been documented [38]. Although these design patterns are a fundamental contribution to later stages of the design process, as shown in Figure 1, a design methodology based on reconfigurability measurement would facilitate a logical and seamless transition between five stages of design: 1.) high level design principles (e.g. bionic, holonic, fractal) 2.) a reference architecture that is sufficiently general to apply to the scope of the manufacturing systems it seeks to address 3.) the associated instantiation as a system-specific architecture 4.) the development of multi-

Amro M. Farid is with the Thayer School of Engineering Dartmouth College Hannover, New Hampshire 03755 and also the Technology and Development Program, Massachusetts Institute of Technology. Cambridge MA 02139 amfarid@mit.edu

Luis Ribeiro is with the Division of Manufacturing Engineering, Department of Management & Engineering, Linköping University, Linköping, SE-58183 luis.ribeiro@liu.se

agent system behavior as a control approach and 5.) a detailed implementation with connected hardware. Previous work on the reconfigurability of automated manufacturing systems has shown that reconfigurability depends primarily on architectural decisions made in Stages I, II, III, and V and much less so Stage IV [39]–[43]. That said, the manufacturing system designer must still take care to consider the manufacturing system's operational performance (e.g. makespan, throughput etc) ***after*** a reconfiguration has already occurred. The relative absence of such a quantitative design methodology based upon reconfigurability measurement has left it relatively unclear 1.) the degree to which these existing designs have achieved their intended level of reconfigurability, 2.) which systems are indeed quantitatively more reconfigurable 3.) how these designs may overcome their inherent design limitations to achieve greater reconfigurability in subsequent design iterations.

This paper contributes a multi-agent system reference architecture for reconfigurable manufacturing systems driven by a quantitative and formal design approach directly in line with Figure 1. In doing so, it demonstrates the design approach as well as the design result. First, it roots itself in an established engineering design methodology: axiomatic design for large flexible engineering systems which was has been previously used to develop quantitative reconfigurability measures [39]–[48]. Second, these quantitative measures are distilled into qualitative high level design principles. The qualitative design principles *and* quantitative reconfigurability measures are then demonstrated in order to produce the reference architecture in the mathematics of reconfigurability measurement – thus facilitating implementation. Finally, the reference architecture and its associated mathematical description straightforwardly address instantiation for system-specific implementation. The subsequent discussion is described as a demonstration of the high level design principles. These specific advantages have yet to be demonstrated in existing reference architectures such as PROSA [32]–[34], HCBA [35] and ADACOR [22], [36].

A. Paper Outline

The remainder of the paper proceeds as follows. Section II qualitatively articulates the the design strategy and rationale for the architecture. Section III details the underlying mathematical description of the architecture. Section IV then details the architecture's data model implementation. Section VI then discusses the merits of the architecture relative to the existing literature. Finally, Section VII brings the work to a conclusion.

B. Scope of Work

This paper restricts its reconfigurability discussion to the shop-floor activities of discrete-part automated manufacturing system as defined in Levels 0-3 of ISA-S95 [49] where

Definition 2. Reconfigurability [47]: The ability to add, remove and/or rearrange in a timely and cost-effective manner the components and functions of a system which can result in a desired set of alternate configurations; chosen here to be the addition/removal of new products and resources.

II. Architecture Design Strategy & Rationale

The aim of this work is to develop an Axiomatic Design of a Multi-Agent Reconfigurable Mechatronic System (ADMARMS) Architecture. To this end, the central goal of the design is to conceive a multi-agent (control) system architecture that enables a highly reconfigurable manufacturing system when it is integrated with its physical devices. Here, reconfigurability is understood as the principle life cycle property which enables the desired behaviors described in the previous section. It is dependent upon the characteristics of the productions system structure [39], [42]. That said, Axiomatic Design states generally [50] and previous works on this subject [39], [42], [43] have discussed that a well conceptualized architecture is also a *necessary prerequisite* for excellent production system performance. Furthermore, as a design methodology, Axiomatic Design is able to highlight potential design flaws at an early conceptual stage; well before final implementation where production system performance data becomes available.

The architecture design strategy and rationale directly follow Figure 1. It begins with the identification of high level design principles which are directly distilled from recent work in the reconfigurability measurement of discrete-part automated manufacturing systems [39]–[48]. These works showed that a high degree of reconfigurability could be achieved by fostering greater ***reconfiguration potential*** (i.e. the number of possible configurations of the system) as well as greater ***reconfiguration ease*** (i.e the effort required to change from one configuration to another). The associated design principles for reconfiguration potential and reconfiguration ease are treated in turn and are actively used in the discussion of Section III. While a deep treatment of reconfigurability measurement is not feasible here, the interested reader is referred to these background references for the details of the mathematical developments in this work. The necessary aspects of these mathematical developments are revisited in Section III.

A. Design Principles for Reconfiguration Potential

The aspects of reconfigurability measurement related to reconfiguration potential were founded upon axiomatic design theory in which reconfigurable manufacturing systems may be classified as large flexible engineering systems.

Definition 3. Large Flexible Engineering System (LFES) [50]: an engineering system with many functional requirements (i.e. production processes) that not only evolve over time, but also can be fulfilled by one or more design parameters (i.e. production resources).

Furthermore, according to Axiomatic Design, this mapping of system processes to system resources requires adherence to the Independence Axiom.

Axiom 1. The Independence Axiom: Maintain the independence of the functional requirements [50].

In practice, this axiom means that the identified production processes must be mutually exclusive and collectively exhaustive. Furthermore, each production process must be matched to its associated production resource. Once this match occurs at a high level of design, Axiomatic Design proceeds to lower, more detailed, levels of design where greater specialization and decomposition can occur. The high level independence decisions taken earlier become design constraints at these lower levels that the designer must now respect [50].

In practice, this high-level axiom means that each functional requirement (i.e. production process) must be related mathematically to a design parameter (i.e. production resource) [39], [40], [42], [50]. Ultimately, each match of production process to production resource is assigned an event (in the discrete-event sense) called a production degree of freedom [40] which individually and collectively have a number of interesting properties:

1) Individually, they represent ***all*** of the available capabilities of the physical production system.
2) Collectively, they represent the configuration of the production system.
3) Their (countable) number represent the production system's reconfiguration potential.
4) The production of any product can be described as a sequence of the production degrees of freedom.

In this regard, production degrees of freedom adhere to a relatively strict analogy to the mechanical degrees of freedom in a purely mechanical system [40]. Furthermore, it is important to recognize the difference between the ***existence*** and the ***availability*** of a production degree of freedom. The former is the presence of a capability regardless of whether it is currently functional or not. The latter addresses the condition of its functionality as a binary state. Therefore, the description of shop floor phenomena such as machine breakdowns may apply sequence-independent constraints that limit the number of production DOFs [40]. Additionally, there may exist sequence-dependent constraints that do the same between pairs of production degrees of freedom. Sequence-dependent constraints may arise from a rigidly implemented supervisory controller; although the relative (physical) geometry of resources ***always*** applies some sequence-dependent constraints [40].

With this production degree of freedom primer in mind, a number of design principles are distilled that maximize reconfiguration potential

Principle 1. Application of Independence Axiom: Explicitly describe the agent architecture in terms of the production system's production degrees of freedom.

Principle 1 entails the main contribution and development strategy underlying the ADMARMS architecture. Because the production of an entire production line can be described as sequences of individual production degrees of freedom, aligning the agents with them creates an explicit relation between form and function. An agent architecture, on its own, describes the interactions between the agents' classes but cannot prescribe their instantiation. Different systems will be subject to different instantiations. Reconfigurability is, in a cyber-physical sense, a measurement of the articulation between form and function and its availability in one system. Hence, any arbitrary degree of freedom not included in the architecture immediately hinders reconfigurability measurement and prevents the agent-system from interpreting it. Structural degrees of freedom can therefore be seen as the quantitative equivalent of agent semantic ontologies [51].

Principle 2. Existence of Physical Agents: As a decision-making/control system, the multi-agent system must maintain a 1-to-1 relationship with the set of physical capabilities that ***exist*** on the shop floor.

The principle of physical agency is multiply reported in the literature. The cohesive integration between form and function facilitates the implementation of Principle 1. The 1-to-1 relationship ensures that physical changes have an equivalent logical change and, in articulation with Principle 1, those changes can be measured. A violation of Principle 2 will destroy the cyber-physical relation promoted by the agent architecture and will subsequently result in the introduction of control ambiguities whenever a new design iteration needs to be considered.

Principle 3. Heterogeneity: The production degrees of freedom within the agent architecture must respect the heterogeneity of capabilities found within the shop-floor be they various types of transformation, transportation or storage processes.

This is a key concept in agent-based design. Similarities between structure and function must be captured and modelled in a balanced way to support the abstraction of the set of possible physical interactions that may occur between the physical resources in the system. It is this generalization of behaviour that enables agent adaptability (function) that results in system reconfigurability (form). Consider two extremes. A design based on a single omnipotent agent would result, aside from the inherent complexity of modelling its internal behaviour, in the definition of a interface that would describe the set of active roles of that agent at any given instance so that other agents could interact with it. On the other hand the discrimination, and subsequent agent specialization, of all possible cyber-physical relations that could ever be considered at shop floor would result in an overwhelming explosion of agent classes that is not manageable either. Heterogeneity is therefore best captured at a granularity level that suits the purpose for wich the architecture has been designed.

Principle 4. Physical Aggregation: The agent architecture must reflect the physical aggregation of the objects that they represent.

Physical resources in a production line are frequently aggregated in specific arrangements to support specific sequences of production processes. Failing to capture the aggregation of the system components would lead to the need of developing new models that describe the contributions of the aggregation's constituents.

Principle 5. Availability: The agent architecture must explicitly model the potential for sequence independent constraints that impede the *availability* of any given production degree of freedom.

The existence of one resource does not require its availability. Availability is affected by failures, maintenance, selective shut-downs, addition of new resources and the dynamic of the system itself. The agents must explicitly be aware of the available capabilities in order to enact a decision that contributes to the resilience of the entire system.

Principle 6. Interaction: The agent architecture must contain agent interactions along the minimal set of physical sequence-dependent constraints.

For example, a material handling process must end where another material handling process begins. Agent's interactions must therefore respect the physical continuity of the system components and actuate accordingly.

Principle 7. Maximum Reconfiguration Potential: Aside from the minimal set of physical sequence-dependent constraints, the agent architecture should avoid introducing any further agent interactions (which may impose further constraints).

The introduction of additional interactions could potentially destroy the cyber-physical relation and cause agents to maintain interactions at a logical abstraction level not reflected at the physical level (for example due to the failure of one resource). Each agent interaction should have a clear reason for existence motivated by some physical phenomena.

B. Design Principles for Reconfiguration Ease

The aspects of reconfigurability measurement related to reconfiguration ease were founded upon the use of the production design structure matrix [41], [47]. It encourages the design of maximally cohesive and minimally coupled modules within the production system. To that end, three more design principles are added for reconfiguration ease.

Principle 8. Scope of Physical Agents: Align agents' scope and boundaries with their corresponding physical resources and their associated production degrees of freedom.

Principle 8 ensures that when a reconfiguration process occurs (i.e. addition, modification or removal of a production degree freedom), it does so simultaneously on the physical resource as well as on the corresponding agent. Previous reconfigurability measurement work has shown that in many cases misaligned informatic entities such as centralized controllers lead to greater coupling of production degrees of freedom [41]; thus hindering reconfiguration ease.

Principle 9. Encapsulation: Production system information should be placed in the agent corresponding to the physical entity that it describes.

Principle 9 recognizes that information is more often used locally rather than remotely and thus encourages greater encapsulation and modularity. Principle 7 also serves to support the modularity of the production system agents.

Principle 10. Reconfiguration Method: The same reconfiguration process can require significantly different effort (measured in time, cost, or energy) depending on the method used to conduct the reconfiguration (and not just the reconfigured resources).

Finally, Principle 10 accounts for the potential for reconfiguration processes to be conducted manually or automatically.

III. Mathematics of Reference Architecture

The discussion now continues to Stage II of Figure 1. On the basis of qualitative design principles in the previous section, and the mathematical treatment found in the reconfigurability measurement of discrete-part automated manufacturing systems [39]–[41], [44]–[48], the Axiomatic Design Multi-Agent Reconfigurable Mechatronic System (ADMARMS) architecture is developed and shown in Figure 2. It's high level structure is now discussed in terms of this mathematical treatment as a demonstration of the high level design principles. The reader is strongly encouraged to follow and verify the 1-to-1 link between the mathematics in this section and the architecture in Figure 2.

Note that the research scope defined in Section I-B directly aids in the design of a reference architecture. Discrete-part production systems, as a class, have a set of common mathematical characteristics which are exploited in the identification of the architecture's classes, attributes and associations. Therefore, a specific instantiated physical system need not occur prior to the development of a MAS agent architecture either at the reference of system-specific levels.

A. Production System Knowledge Base

The production system knowledge base is the key matrix for describing a system's production degrees of freedom. Its usage is a mathematically explicit adherence to Principle 1

Definition 4. Production System Knowledge Base [40]: A binary matrix J_S of size $\sigma(P) \times \sigma(R)$ whose element $J_S(w,v) \in \{0,1\}$ is equal to one when event $e_{wv} \in E_S$ exists as a production process $p_w \in P$ being executed by a resource $r_v \in R$ (where $\sigma()$ gives the size of a set).

By Principle 2, these physical resources have their informatic counterpart in the resource agent (RA). Resource agents are further classified into transformer agents (TFA) M, storage agents (SA) B, and transporter agents (TPA) H to differentiate between the inherently different types of production resources. $R = M \cup B \cup H$ [40]. The first is often considered "value-adding" while the other two are often intentionally minimized. The architecture also recognizes that the transformation and storage resources and their respective agents may be grouped into a set of buffer resources and their respective agents (BA); locations in which products remain stationary (Principle 3). $B_S = M \cup B$ [40].

While the resource agent effectively describes the system form, the production process agent (PPA) effectively describes the system function. While it is common that physical resources have their associated agents, a novel aspect of the ADMARMS architecture is the decision to assign agents to each production process. They act as component slaves of their resource agent masters. This serves to emphasize the distinction between the form of physical resources and the (potentially variable) set of behavioral production processes they can perform.

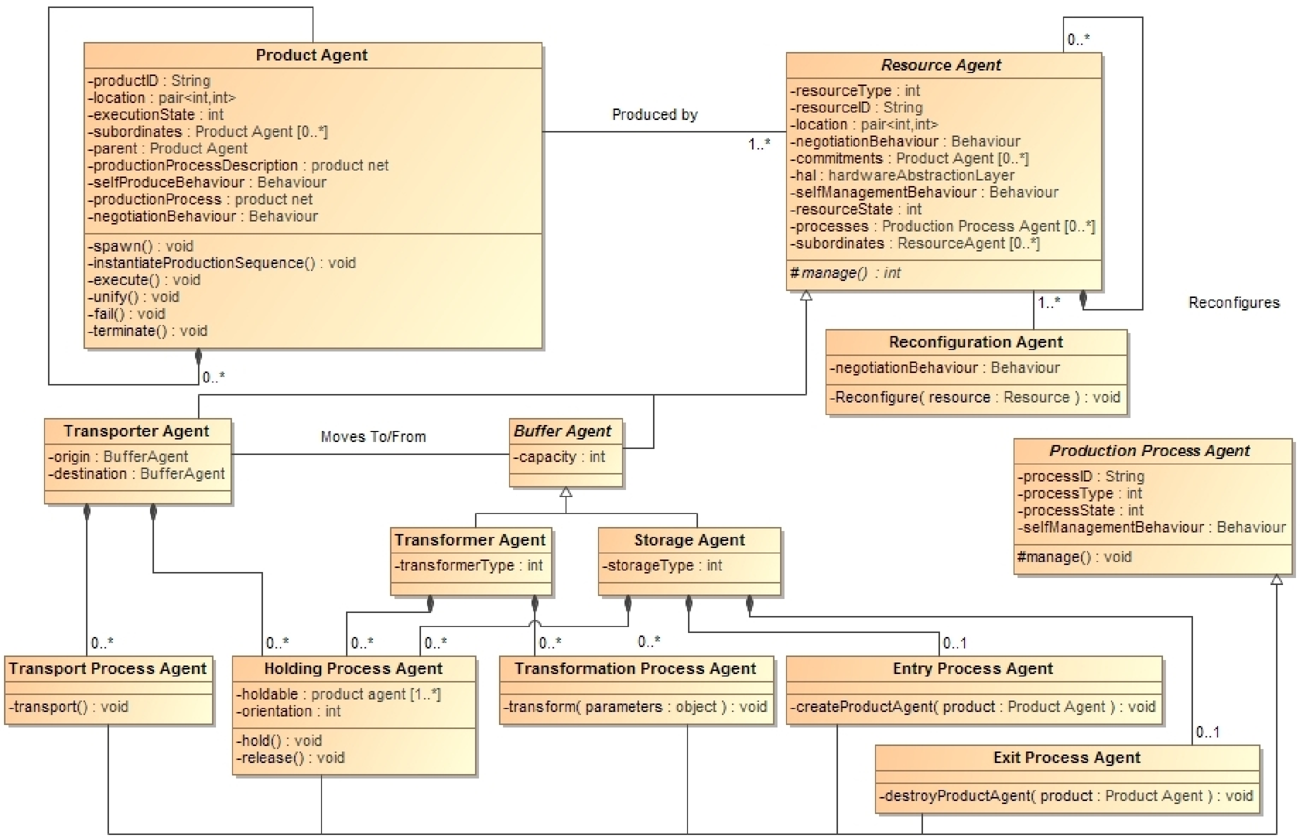

Product Agent
-productID : String
-location : pair<int,int>
-executionState : int
-subordinates : Product Agent [0..*]
-parent : Product Agent
-productionProcessDescription : product net
-selfProduceBehaviour : Behaviour
-productionProcess : product net
-negotiationBehaviour : Behaviour
-spawn() : void
-instantiateProductionSequence() : void
-execute() : void
-unify() : void
-fail() : void
-terminate() : void
Produced by
1..*
0..*
Resource Agent
-resourceType : int
-resourceID : String
-location : pair<int,int>
-negotiationBehaviour : Behaviour
-commitments : Product Agent [0..*]
-hal : hardwareAbstractionLayer
-selfManagementBehaviour : Behaviour
-resourceState : int
-processes : Production Process Agent [0..*]
-subordinates : ResourceAgent [0..*]
#manage() : int
1..*
Reconfigures
Reconfiguration Agent
-negotiationBehaviour : Behaviour
-Reconfigure(resource : Resource) : void
0..*
Transporter Agent
-origin : BufferAgent
-destination : BufferAgent
Moves To/From
Buffer Agent
-capacity : int
Production Process Agent
-processID : String
-processType : int
-processState : int
-selfManagementBehaviour : Behaviour
#manage() : void
Transformer Agent
-transformerType : int
Storage Agent
-storageType : int
0..*
0..*
0..*
0..*
0..*
0..*
0..1
Transport Process Agent
-transport() : void
Holding Process Agent
-holdable : product agent [1..*]
-orientation : int
-hold() : void
-release() : void
Transformation Process Agent
-transform(parameters : object) : void
Entry Process Agent
-createProductAgent(product : Product Agent) : void
0..1
Exit Process Agent
-destroyProductAgent(product : Product Agent) : void

Fig. 2. ADMARMS Architecture & Data Model

As with resource agents, the production process agents must be classified to account for the different types of activities on the shop floor (Principle 3). These include the entry and exit processes agents (ENPA and EXPA) for crossing the system boundary and transformation processes (TFPA) for value-adding processes [52], [53]. Mathematically, they have often been lumped into a single set of production processes P_μ. Transportation process agents (TPPA) P_η are defined between individual buffers. $\sigma(P_\eta) = \sigma^2(B_S)$. Notice that the only resource-agent-to-resource-agent interaction occurs here because a transporter agent needs to know the identity of the origin and destination buffers. Holding process agents (HPA) P_γ account for the ability to fixture/hold a product during storage or transportation.

The overall production system knowledge base J_S can then be reconstructed straightforwardly. The transformation system knowledge base J_M relates P_μ to M. The transportation system knowledge base J_H relates P_η to $R = M \cup B \cup H$. M and B are included here to account for their "null-transportation" or storage processes where no motion occurs. The holding system knowledge base J_γ relates P_γ to R. Then J_S becomes [40], [45]

$$J_S = \left[\begin{array}{c|c} J_M & \mathbf{0} \\ \hline \multicolumn{2}{c}{J_{\bar{H}}} \end{array} \right] \tag{1}$$

where [45]

$$J_{\bar{H}} = \left[J_\gamma \otimes \mathbf{1}^{\sigma(P_\eta)} \right] \cdot \left[\mathbf{1}^{\sigma(P_\gamma)} \otimes J_H \right] \tag{2}$$

and $\otimes$ is the kronecker product and $\mathbf{1}^n$ is a column ones vector of length n.

A resource agent maybe physically aggregated either permanently or temporarily into itself to allow a physical hierarchy as a logical aggregation (or coalition) of resources (Principle 4). In reconfigurability measurement, the set of resource R is physically aggregated into a set aggregated resources $\bar{R}$ by means of an aggregation matrix and operator $\circledast$ [39], [40].

$$\bar{R} = A \circledast R \tag{3}$$

At which point, the production system knowledge base would require a combination of the associated columns.

$$\bar{J_S} = J_S \circledast A^T \tag{4}$$

Hence, physical aggregation would neither violate the Independence Axiom, nor change the associated value of the reconfigurability measure. In contrast, functional aggregation of the resources is likely to violate the independence axiom most notably in the form of centralized control structuresCITE ME.

B. Production System Scleronomic Constraints Matrix

Each of the resource and process agents (and their specific types) can potentially be unavailable due a physical fault or some rigidity in the control system (Principle 5). To account for this, the agents have an "on-off" status. These are described mathematically in the production system scleronomic (i.e. sequence-independent) constraints matrix.

Definition 5. Production System Scleronomic Constraints Matrix [40]: A binary matrix K_S of size $\sigma(P) \times \sigma(R)$ whose element $K_S(w,v) \in \{0,1\}$ is equal to one when a constraint eliminates event e_{wv} from the event set.

It is calculated ***analogously*** to the production system knowledge base [40], [45]:

$$K_S = \left[\begin{array}{c|c} K_M & \mathbf{1} \\ \hline \multicolumn{2}{c}{K_{\bar{H}}} \end{array}\right] \tag{5}$$

where [45]

$$K_{\bar{H}} = \left[K_\varphi \otimes \mathbf{1}^{\sigma(P_\eta)}\right] \cdot \left[\mathbf{1}^{\sigma(P_\varphi)} \otimes K_H\right] \tag{6}$$

From these definitions of J_S and K_S, follows the definition of sequence-independent production degrees of freedom.

Definition 6. Sequence-Independent Production Degrees of Freedom [40]: The set of independent production events E_S that completely defines the available production processes in a production system. Their number is given by:

$$DOF_S = \sigma(\mathcal{E}_S) = \sum_{w}^{\sigma(P)} \sum_{v}^{\sigma(R)} \left[J_S \ominus K_S\right](w,v) \tag{7}$$

where the $A \ominus B = A \cdot not(B)$ operation is boolean subtraction.

C. Production System Rheonomic Constraints Matrix

Once the individual agents have been defined around production system degrees of freedom, the design of the architecture turns to defining the resource-agent-to-resource-agent interactions. In that regard and as mentioned previously, the only such interaction occurs because transporter agents need to know the identity of the origin and destination buffers (Principle 6). These minimal interactions are reflected in the production system rheonomic constraints matrix.

Definition 7. Rheonomic Production Constraints Matrix K_ρ [45], [46]: a square binary constraints matrix of size $\sigma(P)\sigma(R) \times \sigma(P)\sigma(R)$ whose elements $K(\psi_1, \psi_2) \in \{0,1\}$ are equal to one when string $z_{\psi1\psi2} = e_{w_1v_1}e_{w_2v_2} \in Z$ is eliminated and where $\psi = \sigma(P)(v-1)+w$.

Previous work has shown that K_ρ must be non-zero so as to account for basic rules of continuity; the destination of one production degree of freedom must occur at the same location as the origin of a subsequent one [40], [45]. This includes transformation degrees of freedom which explicitly state where the corresponding transformation process must occur. Aside from these minimal constraints, the architecture does not introduce any other resource-agent-to-resource-agent interactions on the resource side (Principle 7). The introduction of any further such interactions in the detailed design of the distributed control algorithms would constitute a violation of the Independence Axiom. Instead, the absence of any further interaction must be considered a high level design constraint on the implementation of such distributed algorithms.

D. Product Model

The design of the ADMARMS architecture then includes product agents (PA) as the 1-to-1 informatic counterpart of each of the physical products (Principle 2). A product agent maybe decomposed into itself to allow a physical hierarchy of subassembly and component products (Principle 4). It is also important to note that the product agent must have some awareness of how it should be produced. This has been achieved previously with a product net.

Definition 8. Product Net [39], [44]: Given product l_i, it may be described as:

$$N_{l_i} = \{S_{l_i}, E_{l_i}, F_{l_i}\} \tag{8}$$

where N_l is the product net, S_l is the set of product places, E_{l_i} is a set of product events, and F_l is set of product flow relations.

Definition 9. Product Event [44]: A specific transformation process that may be applied to a given product.

In addition to the events, places and flow relations all have physical meaning. Each of the places represents a product or component at a raw, work-in-progress, or final stage of production. Finally, the flow relations describe which products or components receive which product events.

The presence of an instantiated product agent in the production system is achieved by the entry and exit process agents found within a given storage agent (Principle 5).

E. Product Feasibility Matrices

Once the individual product agents have been defined, the design of the architecture turns to defining the product-agent-to-resource-agent interactions. These are absolutely necessary operator-to-operand relations (Principle 6). The relationship between product events to transformation degrees of freedom is achieved with the product transformation feasibility matrix.

Definition 10. Product Transformation Feasibility Matrix $\Lambda_{\mu i}$ [44], [45]: A binary matrix of size $\sigma(E_{l_i}) \times \sigma(P_\mu)$ whose value $\Lambda_{\mu i}(x,j) = 1$ if e_{xl_i} realizes transformation process $p_{\mu j}$.

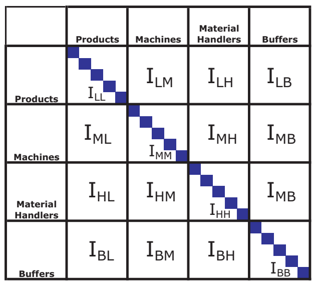
The relationship between products and the required holding/transportation processes is similarly defined.

Definition 11. Product Transportation Feasibility Matrix $\Lambda_{\gamma i}$ [44], [45]: A binary row vector of size $1 \times \sigma(P_\gamma)$ whose value $\Lambda_{\gamma i}(g) = 1$ if product l_i can be held by holding process $p_{\gamma g}$.

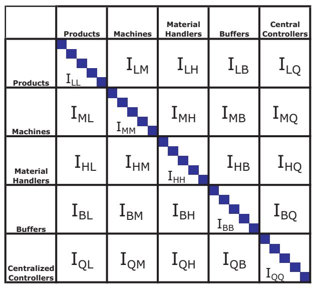
The product-agent-to-resource-agent interaction becomes the primary interaction by which distributed coordination of the production can be achieved without introducing any further agent interactions.

F. Production Design Structure Matrix

The subsections above collectively address the reconfiguration potential of the production system and so the attention now turns to reconfiguration ease. In that regard, the production system design structure matrix [41] has been previously



(a) Without Centralized Controllers



(b) With Centralized Controllers

Fig. 3. Production System Design Structure Matrix

used for measurement. It captures the physical, energy, and informatic couplings between production system entities. In the case of multi-agent systems in production systems, the scope of each agent aligns with the underlying production system resource or product (Principle 8). In such a case, the coupling between production degrees of freedom can be minimized. In contrast, centralized controllers introduce new blocks to the production system design matrix which appear as off-diagonal coupling between resources and their associated production degrees of freedom. As this architecture is developed to include the specific behaviors of agent-to-agent interactions and negotiations, care will be taken to demonstrate that each agent is maximally cohesive and minimally coupled (Principle 9).

G. Reconfiguration Processes

The final element of the ADMARMS architecture is to specify the method of conducting reconfiguration processes (Principle 10). In that regard, the reconfiguration agent (RCA) is an automatic way of changing the production degrees of freedom of a given resource agent. Mathematically, it is responsible for conducting the reconfiguration process [40], [48]:

$$(J_S, K_S, K_\rho) \rightarrow (J'_S, K'_S, K'_\rho) \tag{9}$$

Examples of reconfiguration agents are automatic tool changers (change of transformation degree of freedom), automatic fixture changers (change of holding degree of freedom), conveyor gates (change of transportation degree of freedom). Reconfiguration agents can also act *between* several resources at which point it would be reflected in Equation 9 with a change in the rheonomic constraints matrix K_ρ.

In summary, each of the agents and interactions in the ADMARMS Architecture shown in Figure 2 follows directly from its mathematical description. The architecture defines an agent for each of the types of resources and production processes defined in the mathematics. Consequently, the production system knowledge base relates which resources can realize which processes. This relationship is described by the composition relations between resource and production process agents. The aggregation matrix allows the logical aggregation of resources. The rheonomic constraints matrix causes the interaction between transporter agent and buffer agent. The product feasibility matrices causes the interaction between the product and resource agents. The product nets can be self-decomposed whenever assembly is required. The reconfiguration agent arises from the definition of a reconfiguration process for one or more resources.

IV. Architecture Data Model Implementation

As shown in Figure 2, the previous section provided a high level mathematical description of the ADMARMS so as to identify its member agents. This section now discusses the contents of these agents from an implementation point of view. It is understood that these agents would be implemented in a multi-threaded programming language such as JAVA and adhere to the latest multi-agent standard platforms (e.g. FIPA, JADE) [54].

A. Resource Agent (RA)

The RA is an abstraction for all the potential mechatronic entities in the system. It encapsulates the common features of the specialized classes. Its parameters are described as follows:

-resourceType : int – a type that identifies the implementation class.
-resourceID : String – a unique identifier in the form of a serial number.
-location : pair<int,int> – a coordinate to keep track of its location.
-negotiationBehaviour : Behaviour – a negotiation behavior to mediate the interaction between the RA and a PA.
-commitments : Product Agent [0..*] – a list of commitments to various production agents.
-hal : hardwareAbstractionLayer – a hardware abstraction layer (HAL) that mediates low-level execution.
-selfManagementBehaviour : Behaviour – a persistent behavior that generically allows for self-management.
-resourceState : int – a state of the resource linked to its availability in the scleronomic constraints matrix.
-processes : Production Process Agent [0..*] – a list of production process agents that are associated with resource agent.
-subordinates : ResourceAgent [0..*] – a list of resources that are logically aggregated within the scope of the parent resource. No command & control or master-slave relation is exercised.

Additionally, the resource agent has a single method:

# manage() : int – an abstract management method invoked by the **selfManagementBehavior** parameter that monitors and updates the resource state.

Furthermore, the commitment parameter gives a measurement of the anticipated load on a resource and is fundamental to robust system operation when making further negotiations. It is necessarily updated after each successful execution of a production process. Also, the HAL serves as a generic interface that enables resources of the same type to operate similar physical devices regardless of the specific low level implementation details. Therefore, the HAL decouples the agent environment from controller specific implementations [24].

B. Specializations of the RA

As shown in Figure 2, the RA is specialized as a BA, TFA, SA and a TPA. These are discussed in turn.

1) Buffer Agent (BA): The BA is a specialized resource that denotes a resource with the ability of storing PAs. It has a single additional parameter:

-capacity : int – a finite capacity of PAs.

The TFA and SA stand as specializations of the BA.

2) Transformer Agent (TFA): The transformer agent abstracts shop-floor entities with transformation capabilities and therefore hosts both transforming and holding process agents (inherited from the BA). It has a single additional parameter.

-transformerType : int – a parameter that identifies the various types TFAs be they assembly, additive and subtractive in nature.

3) Storage Agent (SA): The Storage Agent is the implementation of the buffer concept. It is responsible for the storage, introduction, and removal of a PA within, into and from the system. Subsequently, it hosts the corresponding production processes. It also has a single additional parameter:

-storageType : int – a parameter that identifies the various types of SAs be they passive or active.

C. Transporter Agent (TA)

The TA is responsible for moving a PA between buffer agents. Consequently, it has two parameters:

-origin : BufferAgent – the identity of the origin BA.
-destination : BufferAgent – the identity of the destination BA.

It hosts transportation process and holding process agents as it executes the motion between these buffers.

D. Production Process Agent (PPA) and its Specializations

The PPA abstracts the different production processes hosted by the system's resources. It has the following parameters:

-processID : String – a unique identifier that identifies the process and its instance.
-processType : int – a type that defines the specialized class to which the PPA belongs.
-processState : int – a state of the PPA linked to its availability in the scleronomic constraints matrix.
-selfManagementBehaviour : Behaviour – a persistent behavior that generically allows for self-management.

Additionally, a production process has a single method:

# manage() : int – a management method invoked by the selfManagementBehaviour.

This method, whose implementation must be provided by the specializing classes, is responsible for updating the state of the PPA and embodies the PPA's proactive behavior towards the emergence of constraints.

The PPA class has five specializations: the transformation process agent (TFPA), the transportation process agent (TPPA), the holding process agent (HPA), the entry process agent (EPA) and the exit process agent (EXPA). Collectively PPAs, Resource Agents and their associated specializations define the production system knowledge base.

1) Transportation Process Agent (TPA): The TPPA has a single method that is hosted by the TPA:

-transport() : void – a method responsible for executing the displacement of parts between buffers.

2) Transformation Process Agent (TFPA): The TFPA has a single method that is hosted by the TFA:

-transform(parameters : objective) : void – a method responsible for the transformation of a part.

TFPAs execute differently depending on their parametrization to accommodate the distinct transformation requirements.

3) Holding Process Agent (HPA): The HPA has two parameters that relate to the product agent:

-holdable : product agent [1..*] – a parameter to define which parts can be held.
-orientation : int – a parameter to define the orientation of that part.

In addition, the holding process agent has two methods:

# hold() : void – a method to allow the PA to be held.
# release() : void – a method to allow the PA to be released.

The HPA is hosted conjointly with either a TFPA or a TPPA by a TFA or TPA respectively.

4) Entry & Exit Process Agents (ENPA & EFPA): The ENPA and the EXPA create and destroy product agents with their respective methods.

-createProductAgent(product : Product Agent) : void) – allows for a new order to spawn a new product agent.
-destroyProductAgent(product : Product Agent) – allows a completed product to be registered as a fulfilled order.

E. Product Agent (PA)

The PA abstracts each active product under production in the system and has the following parameters:

-productID : String – a serial number as a string.
-location : pair<int,int> – a location coordinate.
-executionState : int – a state variable that captures the condition of the overall process.
-subordinates : Product Agent [0..*] – a list that is used when the PA has subordinate PAs. From a structural point of view, each product may be composed of other subordinated products.
-parent : Product Agent – a string that is used when the PA is subordinate to another PA. This string mirrors the composition relationship with the aggregation relationship. Parent product agents can proceed to complete their product events only after the completion of the subordinate PA's product events.
-productionProcessDescription : product net – captures the flow of product events. Its hierarchical decomposition

encourages a similar decomposition of the resulting product nets hence creating different views, with levels of abstraction, over the entire set of product events that define the production of a product. Each PA establishes an identity with a final form of an assembly, sub-assembly, part or material.

The autonomy of the PA is defined by its main persistent behaviour:

-selfProduceBehaviour : Behaviour – a behaviour that controls the production process of the PA. It has two main phases: configuration and runtime.

The configuration phase is responsible for initialization routines, evaluation and instantiation of the PA's production process. These are carried out by different methods:

-spawn() : void – a method that spawns all the PA's subordinates.

-instantiateProductionSequence() : void – a method that evaluates and instantiates the production process description and subsequently populates the product net's associated production processes.

-productionProcess : product net – a net that stores the association between product events and production processes. It provides the required information to devise the path connecting all the resources allocated in this process.

-negotiationBehaviour : Behaviour – a behaviour that encapsulates the communication and negotiation logic between the PA and the system's resources. It is activated in the configuration phase to ensure the initial association of processes and resources and later, in runtime, as a response to disturbances.

The runtime phase controls and monitors the production process and disturbances. These actions are implemented in the execute method.

-execute() : void – this method implements a supervisory state machine that governs: resource activation, agent unification/termination and re-negotiation. It describes all the PA to PA and PA to RA interaction logic. This state machine is therefore supported by three methods:

-unify() : void – a method that signals the parent PA that this subordinate PA has successfully terminated its process and the resulting sub-product can be integrated in the parent's process.

-fail() : void – a method that signals the parent PA that this subordinate has encountered an unrecoverable fault and cannot be integrated in the parent's process.

-terminate() : void – a method that removes a subordinate PA from the system in a clean way or removes the top level PA at a resource hosting an exit process agent.

F. Reconfiguration Agent (RCA)

The RCA's behaviour is mainly defined by one function:

-Reconfigure(resource : Resource) : void – reconfigures the process agents in a specific resource and ensures that the instantiation of new process is conflict free.

The reconfiguration occurs on request from a RA and through a negotiation procedure whereby the RA asks the RCA to enable new processes or re-parametrize existing ones. The RCA will then assess the availability and potential reconfiguration conflicts on the desired processes and reconfigure the RA accordingly. In the case of reconfigurations between resources, the RCA decouples the reconfiguration actions from the physical aggregates. From a design perspective this enables the resources to participate in any aggregate that makes physical sense and allows the development of custom reconfiguration agents for very specific cases without the need for reprogramming the resources, as would happen otherwise.

V. Illustrative Example

A practical explanation of the proposed reference architecture is now provided through an illustrative example. Again, the provided discussion is set as a demonstration of the design principles in Section II and the mathematical description in Section III. In all, the process of instantiation carries the discussion through the first three design stages in Figure 1. The Startling III manufacturing system, firstly introduced in [40], and depicted in (Figure 4) is a conceptual example of a flexible production line for wooden bird feeders. The bird feeders enter the system at the input buffer as wood stock and proceed to one of the available milling stations where they are shaped to their final form. They are subsequently assembled and painted in the corresponding stations before leaving the system as the final product at the output buffer. Figure 5 shows a model of the birdfeeder with its corresponding product net.

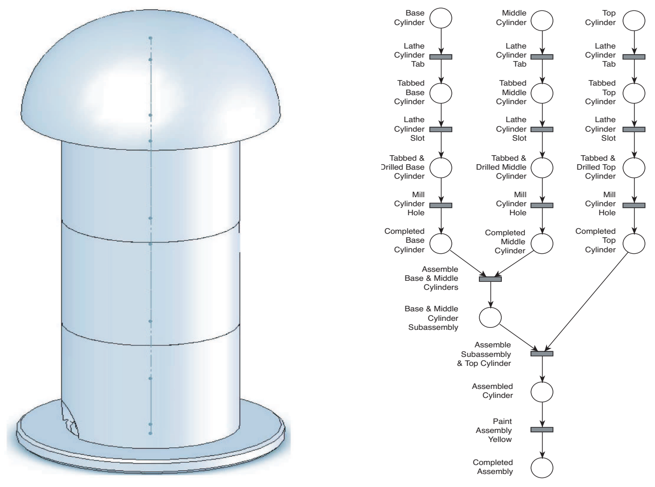


Fig. 5. a.) CAD Model of a Starling Birdfeeder (on left) b.) Associated Product Net (on right)

The system (Figure 4) also features a conveyor network capable of driving the product, during its different production stages, across multiple paths depending on the system conditions (i.e. product mix, load, resource availability, product's process plans etc.). At the beginning and at the end of each conveyor stretch lies a gate that acts as a routing element in the system.

Principle 1 requires an explicit description of the production degrees of freedom. In that regard, there are six transformation resources: {milling stations 1 & 2 , assembly stations 1 & 2, and painting station 1 & 2}. There are 20 independent buffers corresponding to the 18 gates and the input and output buffers. These combine to account for 26 buffers in total. There are

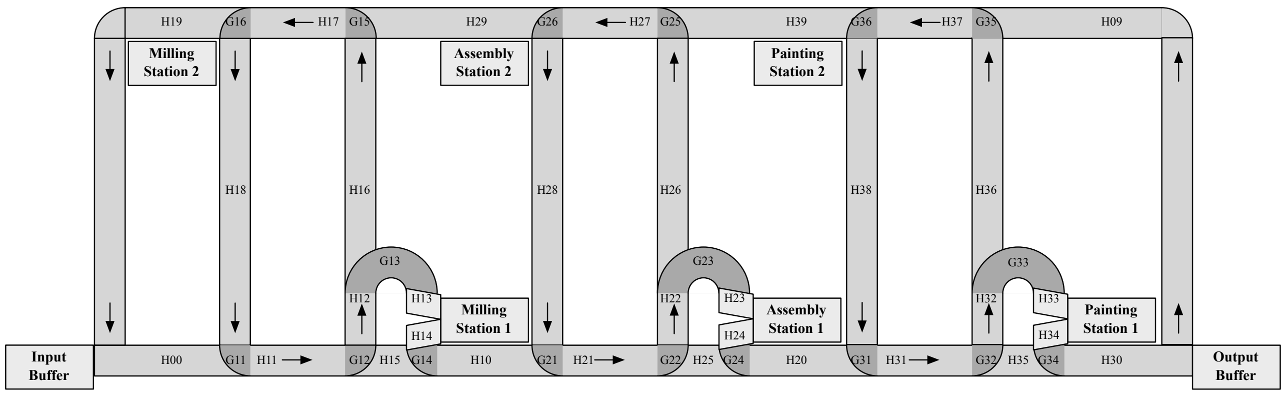


Fig. 4. Example system - the Starling III manufacturing line

32 transporting resources which are indicated by HXY on the figure. The transformation processes are {lathe tab, lathe slot, mill hole, assemble, paint red, paint yellow, and paint green}. The holding processes are: {axially, for small radius, and for large radius}, and there are $\sigma(P_\eta) = \sigma^2(B) = 26^2 = 676$ transportation processes between the previously identified buffers. Consequently, the transformation, transportation, and holding system knowledge bases can be obtained by definition. As these matrices are large and sparse, they are best mined automatically from a database. They are presented in Figure 6 as monochrome images. The production system knowledge base is then straightforwardly calculated by Equation 1 which then yields 78 production degrees of freedom.

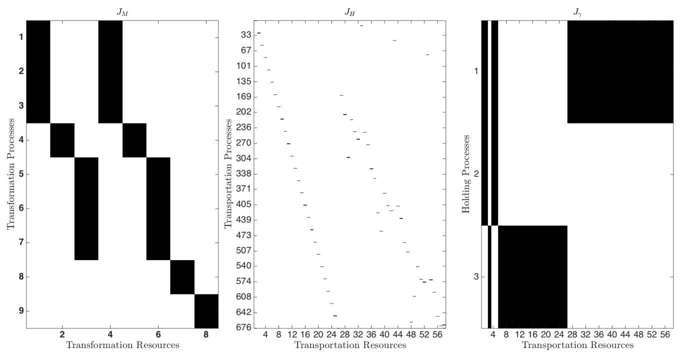


Fig. 6. Transformation, Transportation, & Holding System Knowledge Bases for the Starling III Manufacturing System

The production degrees of freedom are implemented as follows in the instantiated agent architecture. Each transporter agent represents a conveyor. It is associated with one instance of a transport process agent that monitors and controls the flow of products inside the conveyor and a holding process agent that will simultaneously control the gripping of the product. Both agents directly manipulate the resource's hardware abstraction layer. Similarly, the milling, assembly, and painting stations are abstracted to a transformer agent which has its respective transformation and holding process agents. The milling stations, for example, have three transformation processes and two holding processes as shown in their respective knowledge bases. The input and output buffers as well as the gates are abstracted as storage agents. They are associated with one entry and exit process agent respectively in addition to a holding process agent. The gates are associated with a holding process agent and a reconfiguration agent.

With each production degree of freedom accounted for, Principle 2 promotes a one-to-one relation between the agents described in the architecture and physical entities in the shop-floor. Furthermore, Principle 3 ensures that the architecture respects different physical behaviors of the physical resources. Together with Principle 1, the first three principles establish a direct link between the system capabilities and the mathematical framework supporting the architecture.

Principle 4 addresses the aggregates of both products and resources. On the product side, the (parent) product net in Figure 6 can intuitively spawn three product nets prior to the two assembly processes. On the resource side, the self-decomposition of the resource agent allows resources to operate together as aggregates especially when such temporary aggregates are operationally more efficient. As mentioned before, such physical aggregation does not necessarily imply command and control within the aggregates. Instead, ADMARMS allows the processes, as independent agents, to become associated with different resources via reconfiguration agents.

Principle 5 is, therefore, directly related with the presence or absence of resources and their processes. Since the agents maintain a 1-to-1 relationship with the physical resources, faults, failures or different system reconfigurations are automatically considered from a reconfigurability point of view.

Principles 6 and 7 address the interactions between the various agents. The axiomatic design of the reference architecture has lead to four main interactions: Product-Resource, Resource-Reconfiguration, Transporter-Buffer and Resource-Process. As is in current automation practice, no additional interactions would be introduced upon instantiation or detailed design of distributed control algorithms. Indeed, Axiomatic Design Theory imposes the constraint that the implementation of these algorithms not introduce any further interactions than those already specified in the high level design of the architecture [50]. Furthermore, a given reference architecture can (and should) permit several and not just one potential control algorithm. In the context of this specific work, it is sufficient to demonstrate that the result of the axiomatic design can be feasibly implemented in a self-consistent manner using only the four previously identified interactions. The Transporter-Buffer interaction occurs in (re)configuration phase where nearest-neighbor interactions are established. Similarly, the Resource-Process interaction configures their respective mapping. The remaining two interactions are detailed in Figure 7 as the main interaction pattern of the ADMARMS agents. All four interactions use request-reply protocols which are omitted for brevity.

The product-resource interactions in Figure 7 assume an "intelligent product" paradigm [55]–[57] which is supported by the product net described in Definition 8. Thus, the PA concentrates within itself the decision on where to go in the system and where to execute. PAs are created by storage agents associated with entry process agents. Product agents requiring assembly spawn a PA for each sub-product each with its subset product net. Each PA subsequently enters the execution stage where the different system resources are successively activated (Steps 3-16 in Figure 7). Upon completion of the PA process plan, it either unites with the parent PA if it is a sub-product or travel to the output buffer where the storage agent call's for its termination as a now finished good.

Resource level execution implies the verification of constraints and may include resource-reconfiguration agent interaction. If there are pending constraints the required reconfiguration is verified by the reconfiguration agent which may accept and reconfigure the resource or may suggest an alternative configuration which much be accepted by the resource before becoming effective. The resource then proceed with execution. In the Starling III manufacturing system, reconfiguration action may appear as either a routing reconfiguration at gate level or a station reconfiguration such as tooling or fixture change. Which reconfiguration is required depends on the negotiated commitments made between the resource and product agents.

Returning to the remaining design principles, this set of behaviors strictly adheres to Principles 8 and 9 by containing the information where it belongs, from a decision making point of view, and promoting an overall behavior whereby each product agent acts a client of the system's resources and can dynamically adjust to explore them, under different conditions, independently of other products. It also alleviates the PA from the potential micro-management burden of transport and execution actions – which are better left to the specialized resources.

VI. Discussion

The introduction to this paper mentioned that one of the barriers to the adoption of multi-agent system technology in production systems has been the lack of quantitative multi-agent system design methodologies. In this regard, the ADMARMS architecture presents a partial solution beyond the existing literature. In this section, the merits of the ADMARMS architecture are discussed from three perspectives: 1.) Its role in a facilitated design methodology, 2.) its features relative to the existing literature 3.) its suitability for quantitative reconfigurability measurement.

A. Facilitated Design Methodology

As a reference architecture, the ADMARMS architecture can greatly facilitate a systematic design methodology such as the one shown in Figure 1. The design principles identified in Section II effectively address Stage I. As part of Stage II, these principles were shown in Section III to correspond 1-to-1 to the features of the ADMARMS reference architecture. Furthermore, because the reference architecture addresses the full heterogeneity of production system processes, resources and their interactions, the process of instantiating this reference architecture into a system-specific architecture in Stage III can be fairly straightforward. The designer is then left with the task of designing a formal MAS control algorithm in Stage IV to achieve the desired manufacturing system behavioral performance. Finally, the hardware-in-the-loop implementation in Stage V is mitigated by the agent communication with the hardware abstraction layer.

This relatively straightforward design process may be contrasted to the state of the existing literature. Stage I design principles such as bionic, holonic and fractal manufacturing systems are relatively abstract and do not suggest an obvious reference architecture. Perhaps, this is one reason for the relative plethora of contributed reference and system-specific architectures. Furthermore, as will be discussed in the next subsection, these reference architectures do not necessarily account for the full heterogeneity of production system processes, resources and their interactions. As a result, when it comes time to instantiate the reference architecture into a system-specific design in Stage III, many agent-to-agent interactions ***must*** be added in an ad-hoc fashion. One of the most commonly neglected interactions is the coordination of transportation activities. An example of this can be found in the discussed limitations of the HCBA architecture which the designers themselves identified [35]. With such limitations early in the design process at the level of the reference and system architectures, the MAS control algorithm and the interfaces to the hardware abstraction layer naturally become very system-specific. Thus, reconfigurations involving the addition or removal of production system processes, resources or products; particularly those of completely different type, becomes far from assured in a practical industrial context.

B. Comparison to Existing MAS Reference Architectures

As mentioned in the introduction, the MAS literature has produced several reference architectures for production system control [21], [28], [58], [59]; most notably PROSA [32]–[34], HCBA [35], and ADACOR [22], [36]. These works were developed qualitatively based upon the experience of expert MAS designers. This caused several researchers to call for quantitative ways to measure the reconfigurability of MAS architectures [21], [29], [60]–[63]. Now that such techniques have been contributed, a natural evolution of the literature is to review these prior works in the context of reconfigurability measurement.

Thus, the merits of the ADMARMS architecture may also be contrasted to existing multi-agent system reference architectures of similar scope and function. Because the ten

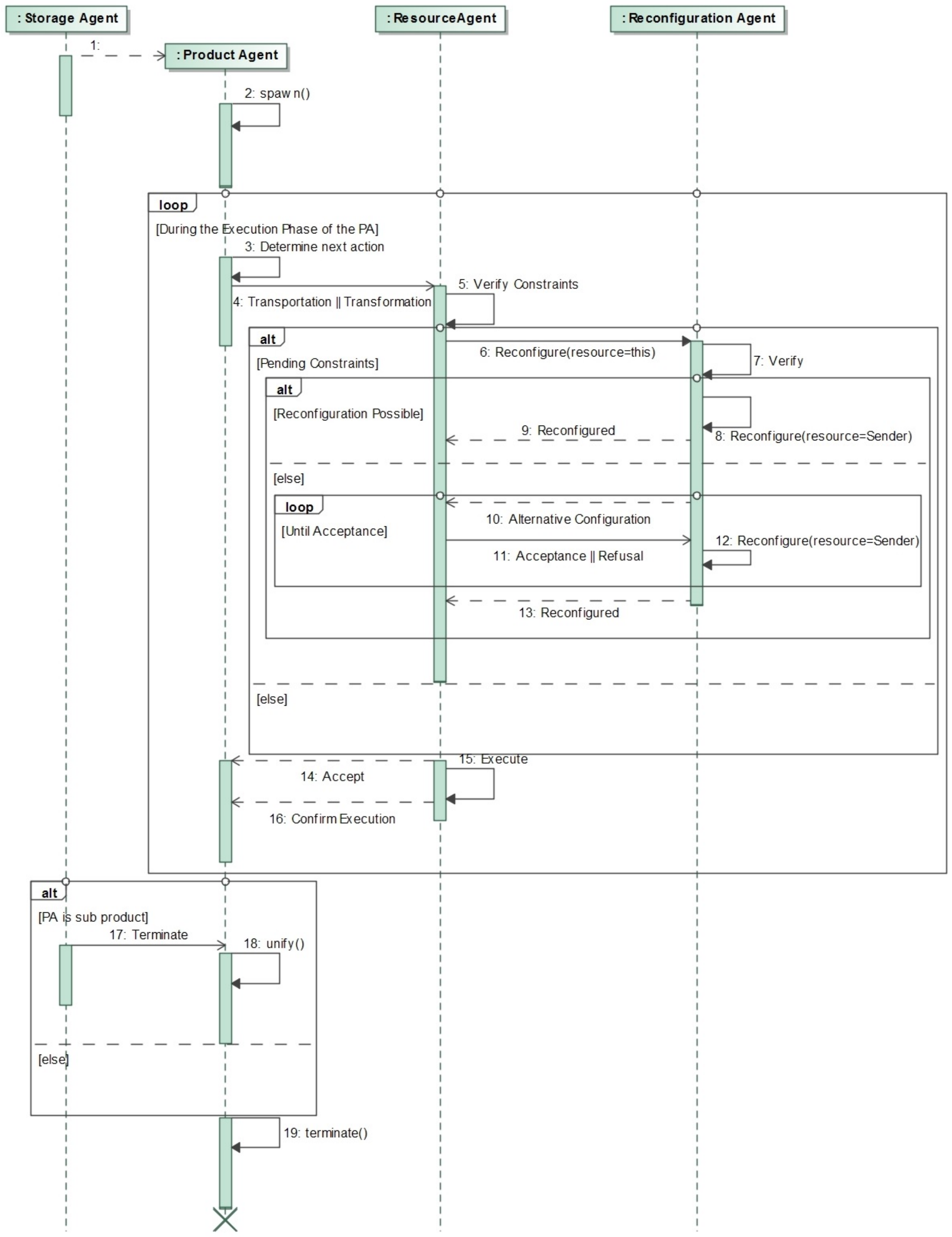
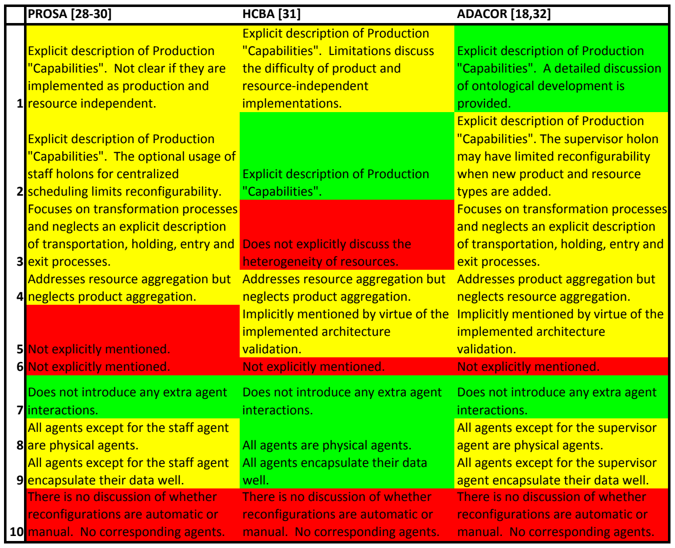


Fig. 7. Main agent interactions

design principles described in Section II were distilled from the existing literature on reconfigurability measurement [39]–[42], [44]–[48], they are taken as requirements upon which to draw an objective comparison. To that effect, Figure 8 assesses the PROSA [32]–[34], HCBA [35], and ADACOR [22], [36] architectures based upon their explicit written descriptions in their corresponding journal papers and doctoral theses. While other MAS architectures can be found in the literature [21], they were excluded from this discussion because they were either system-specific instantiations or of fundamentally different scope.

Figure 8 shows only a partial adherence to the reconfigurability measurement design principles. Of the three architectures, only ADACOR developed an ontology to facilitate product and resource independent descriptions of production (Principle 1). Meanwhile, HCBA was the only architecture that strictly adhered to a 1-to-1 relationship with shop-floor activities (Principle 2,8 and 9). All three architectures lacked a complete description of the heterogeneity of production system processes and resources (Principle 3). Similarly, each architecture either addressed product aggregation or resource aggregation but not both (Principle 4). HCBA and ADACOR provided some functionality for unavailable resources but did not explicitly describe it in the reference architecture itself (Principle 5). Because all three reference architectures did not address the necessary heterogeneity, they tended to understate rather overstate the necessary agent-to-agent interactions (Principle 6 and 7). Finally, none of the reference architectures specifically addressed the method of reconfiguration.

It is important to note that while this assessment is as objective as possible it is also very much anachronistic. The literature on reconfigurability measurement of automated manufacturing systems emerged not only after the development of these reference architectures but also because of a recognized need to introduce quantitative methods in the design of such architectures. The ADMARMS architecture, therefore, can be viewed as a quantitively driven design iteration over the existing literature where no such quantitative reconfigurability measurement approaches existed at the time.



	PROSA [28-30]	HCBA [31]	ADACOR [18,32]
1	🟨 Explicit description of Production "Capabilities". Not clear if they are implemented as production and resource independent.	🟨 Explicit description of Production "Capabilities". Limitations discuss the difficulty of product and resource-independent implementations.	🟩 Explicit description of Production "Capabilities". A detailed discussion of ontological development is provided.
2	🟨 Explicit description of Production "Capabilities". The optional usage of staff holons for centralized scheduling limits reconfigurability.	🟩 Explicit description of Production "Capabilities".	🟨 Explicit description of Production "Capabilities". The supervisor holon may have limited reconfigurability when new product and resource types are added.
3	🟨 Focuses on transformation processes and neglects an explicit description of transportation, holding, entry and exit processes.	🟥 Does not explicitly discuss the heterogeneity of resources.	🟨 Focuses on transformation processes and neglects an explicit description of transportation, holding, entry and exit processes.
4	🟨 Addresses resource aggregation but neglects product aggregation.	🟨 Addresses resource aggregation but neglects product aggregation.	🟨 Addresses product aggregation but neglects resource aggregation.
5	🟥 Not explicitly mentioned.	🟨 Implicitly mentioned by virtue of the implemented architecture validation.	🟨 Implicitly mentioned by virtue of the implemented architecture validation.
6	🟥 Not explicitly mentioned.	🟥 Not explicitly mentioned.	🟥 Not explicitly mentioned.
7	🟩 Does not introduce any extra agent interactions.	🟩 Does not introduce any extra agent interactions.	🟩 Does not introduce any extra agent interactions.
8	🟨 All agents except for the staff agent are physical agents.	🟩 All agents are physical agents.	🟨 All agents except for the supervisor agent are physical agents.
9	🟨 All agents except for the staff agent encapsulate their data well.	🟩 All agents encapsulate their data well.	🟨 All agents except for the supervisor agent encapsulate their data well.
10	🟥 There is no discussion of whether reconfigurations are automatic or manual. No corresponding agents.	🟥 There is no discussion of whether reconfigurations are automatic or manual. No corresponding agents.	🟥 There is no discussion of whether reconfigurations are automatic or manual. No corresponding agents.

Fig. 8. An Assessment of Existing MAS Reference Architectures with Respect to Reconfigurability Measurement Design Principles

C. Suitability for Quantitative Reconfigurability Measurement

Finally, because the ADMARMS architecture was formally developed using the same data structures necessary for reconfigurability measurement (i.e. the production system knowledge base, and the production design structure matrix), these data structures may be directly embedded into the agent architecture for online reconfigurability measurement; either in design or real-time operation. In such a way, the manufacturing system MAS designer can determine the degree of reconfigurability with each successive design decision. Furthermore, such a quantitative approach can highlight which aspects of the design lead to limited reconfigurability. For example, it would become clear when the agent architecture has not well aligned the production degrees of freedom with the physical shop floor entities. Additionally, the introduction of unnecessary agent-to-agent interactions would have an immediate quantitative impact on reconfigurability. Finally, the reconfigurability of an ADMARMS design can be quantitatively compared against other systems. As discussed in the literature [39], such a comparison does have stringent requirements. These include a common ontological basis for describing production system processes, resources and products [39], [42] as well as equivalent methods for conducting the measurement either manually [47] or automatically [42]. Nevertheless, in such conditions, the ADMARMS architecture is easily compared because it embeds the production system knowledge base and design structure matrix within its implementation.

VII. Conclusions & Future Work

To our knowledge, this paper is the first multi-agent system reference architecture for reconfigurable manufacturing systems driven by a quantitative and formal design approach. This is in contrast to existing reference architectures (e.g. PROSA, HCBA, ADACOR) which were qualitatively developed on the basis of experienced design intuition. The ADMARMS architecture is rooted in an established engineering design methodology called axiomatic design for large flexible engineering systems and draws upon design principles distilled from prior works on reconfigurability measurement. The resulting architecture is written in terms of the mathematical description used in reconfigurability measurement which straightforwardly allows instantiation for system-specific application. Future work will seek to 1.) implement this architecture in virtual and hardware testbeds and measure the consequent reconfigurability and 2.) benchmark this reference architecture with respect to the existing alternatives.

References

[1] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, and P. Heytler, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 13, no. 2, pp. 135–146, 2002.

[2] J. B. Pine, *Mass Customization: The New Frontier in Business Competition*. Cambridge, MA: Harvard Business School Press, 1993.

[3] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing systems and their enabling technologies," *International Journal of Manufacturing Technology and Management*, vol. 1, no. 1, pp. 113–130, 2000.

[4] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable manufacturing systems," *CIRP Annals - Manufacturing Technology*, vol. 48, no. 2, pp. 527–540, 1999.

[5] A. I. Dashchenko, *Reconfigurable Manufacturing Systems and Transformable Factories*. Berlin, Germany: Springer, 2006.

[6] R. M. Setchi and N. Lagos, "Reconfigurability and Reconfigurable Manufacturing Systems – State of the Art Review," in *Innovative Production Machines and Systems: Production Automation and Control State of the Art Review*. Cardiff, UK: Innovative Production Machines and Systems: Nework of Excellence, 2005, pp. 131–143.

[7] P. Leitão, V. Marik, and P. Vrba, "Past, present, and future of industrial agent applications," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 4, pp. 2360–2372, 2013.

[8] U. Frank, J. Papenfort, and D. Schütz, "Real-time capable software agents on iec 61131 systems–developing a tool supported method," in *Proceedings of the 18th IFAC World Congress, Milan, Italy*, 2011.

[9] D. Schutz, A. Wannagat, C. Legat, and B. Vogel-Heuser, "Development of plc-based software for increasing the dependability of production automation systems," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 4, pp. 2397–2406, 2013.

[10] V. Vyatkin, "Iec 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 768–781, 2011.

[11] J. Heilala and P. Voho, "Modular reconfigurable flexible final assembly systems," *Assembly Automation*, vol. 21, no. 1, pp. 20–28, 2001.

[12] J. K. L. Ho and P. G. Ranky, "An object-oriented and flexible material handling system," *Assembly Automation*, vol. 15, no. 3, pp. 15–20, 1995.

[13] R. G. Landers, B. K. Min, and Y. Koren, "Reconfigurable machine tools," *CIRP Annals - Manufacturing Technology*, vol. 50, pp. 269–274, 2001.

[14] B. Shirinzadeh, "Flexible fixturing for workpiece positioning and constraining," *Assembly Automation*, vol. 22, no. 2, pp. 112–120, 2002.

[15] W. Townsend, "The BarrettHand grasper-programmably flexible part handling and assembly," *Industrial Robot*, vol. 27, no. 3, pp. 181–188, 2000.

[16] R. Brennan and D. H. Norrie, "Agents, Holons and Function Blocks: Distributed Intelligent Control in Manufacturing," *Journal of Applied Systems Science: Special Issue*, vol. 2, no. 1, pp. 1–19, 2001.

[17] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems*. Research Triangle Park, NC, USA: Instrumentation Society of America, 2007.

[18] W. Lepuschitz, A. Zoitl, M. Vallee, and M. Merdan, "Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 1, pp. 52–69, 2010.

[19] W. Shen and D. Norrie, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey," *Knowledge and Information Systems: An International Journal*, vol. 1, no. 2, pp. 129–156, 1999.

[20] W. Shen, D. Norrie, and J. P. Barthes, *MultiAgent Systems for Concurrent Intelligent Design and Manufacturing*. London, UK: Taylor and Francis, 2000.

[21] P. Leitao, "Agent-based distributed manufacturing control: a state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, Oct. 2009.

[22] P. Leitao and F. Restivo, "ADACOR: A holonic architecture for agile and adaptive manufacturing control," *Computers in Industry*, vol. 57, no. 2, pp. 121–130, Feb. 2006.

[23] P. Leitao, J. Barbosa, and D. Trentesaux, "Bio-inspired multi-agent systems for reconfigurable manufacturing systems," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 5, pp. 934–944, Aug. 2012.

[24] L. Ribeiro and J. Barata, "Deployment of Multiagent Mechatronic Systems," in *Industrial Applications of Holonic and Multi-Agent Systems*, ser. 6th International Conference, HoloMAS 2013. Proceedings: LNCS 8062. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 71–82.

[25] ——, "Self-organizing multiagent mechatronic systems in perspective," in *IEEE International Conference on Industrial Informatics*. Bochum, Germany: IEEE, 2013, pp. 392–397.

[26] R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: A survey," *Journal of Intelligent Manufacturing*, vol. 17, pp. 111–131, 2006.

[27] V. Marik, M. Fletcher, M. Pechoucek, O. Stepankova, H. Krautwurmova, and M. Luck, "Holons and Agents: Recent Developments and Mutual Impacts," in *Multi-Agent Systems and Applications II: Lecture Notes in Artificial Intelligence*. Springer Verlag, 2002, pp. 233–267.

[28] D. McFarlane and S. Bussmann, "Developments in holonic production planning and control," *Production Planning and Control*, vol. 11, no. 6, pp. 522–536, 2000.

[29] D. McFarlane, S. Bussmann, and S. M. Deen, "Holonic Manufacturing Control: Rationales, Developments and Open Issues," in *Agent-Based Manufacturing*. Berlin: Springer-Verlag, 2003, pp. 303–326.

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

[31] A. Tharumarajah and A. Wells, "Behaviour-Based Approach to Scheduling in Distributed Manufacturing Systems," *Integrated Computer-Aided Engineering*, no. Special Issue on Intelligent Manufacturing Systems, 1996.

[32] H. V. Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: PROSA," *Computers in Industry*, vol. 37, pp. 255–274, 1998.

[33] J. Wyns, "Reference Architecture for Holonic Manufacturing Systems," Ph.D., Catholic University of Leuven, 1999.

[34] H. Van Brussel, L. Bongaerts, J. Wyns, P. Valckenaers, and T. Vangin-derachter, "A conceptual framework for holonic manufacturing: Identification of manufacturing holons," *Journal of Manufacturing Systems*, vol. 18, no. 1, pp. 35–52, 1999.

[35] J.-L. L. Chirn, "Developing a Reconfigurable Manuacturing Control System - A Holonic Component Based Approach." Ph.D. Thesis, Univeristy of Cambridge, 2001.

[36] P. Leitao, "An Agile and Adaptive Holonic Architecture for Manufacturing Control," Ph.D. Thesis, University of Porto, 2004.

[37] V. Marik and D. McFarlane, "Industrial Adoption of Agent-Based Technologies," *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 20, no. 1, pp. 27–35, 2005.

[38] A. Zoitl and H. Praehofer, "Guidelines and patterns for building hierarchical automation solutions in the iec 61499 modeling language," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, vol. 9, no. 4, pp. 2387–2396, 2013.

[39] A. M. Farid, "Reconfigurability Measurement in Automated Manufacturing Systems," Ph.D. Dissertation, University of Cambridge Engineering Department Institute for Manufacturing, 2007.

[40] A. M. Farid and D. C. McFarlane, "Production degrees of freedom as manufacturing system reconfiguration potential measures," *Proceedings of the Institution of Mechanical Engineers, Part B (Journal of Engineering Manufacture)*, vol. 222, no. B10, pp. 1301–1314, Oct. 2008.

[41] A. M. Farid, "Facilitating ease of system reconfiguration through measures of manufacturing modularity," *Proceedings of the Institution of Mechanical Engineers, Part B (Journal of Engineering Manufacture)*, vol. 222, no. B10, pp. 1275–1288, 2008.

[42] ——, "Axiomatic Design & Design Structure Matrix Measures for Reconfigurability & Its Key Characteristics in Automated Manufacturing Systems," in *International Conference on Axiomatic Design*, Campus de Caparica, Portugal, 2014, pp. 1–8.

[43] ——, "Measures of Reconfigurability & Its Key Characteristics in Intelligent Manufacturing Systems," *Journal of Intelligent Manufacturing*, vol. 1, no. 1, pp. 1–26, 2014.

[44] ——, "Product Degrees of Freedom as Manufacturing System Reconfiguration Potential Measures," *International Transactions on Systems Science and Application*, vol. 4, no. 3, pp. 227–242, 2008.

[45] ——, "An Axiomatic Design Approach to Non-Assembled Production Path Enumeration in Reconfigurable Manufacturing Systems," in *2013 IEEE International Conference on Systems Man and Cybernetics*, Manchester, UK, 2013, pp. 1–8.

[46] A. Viswanath, E. E. S. Baca, and A. M. Farid, "An Axiomatic Design Approach to Passenger Itinerary Enumeration in Reconfigurable Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–10, 2013.

[47] A. M. Farid and D. C. McFarlane, "A Design Structure Matrix Based Method for Reconfigurability Measurement of Distributed Manufacturing Systems," *International Journal of Intelligent Control and Systems Special Issue*, vol. 12, no. 2, pp. 118–129, 2007.

[48] A. M. Farid and W. Covanich, "Measuring the Effort of a Reconfiguration Process," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, Hamburg, Germany, 2008, pp. 1137–1144.

[49] ANSI-ISA, "Enterprise Control System Integration Part 3: Activity Models of Manufacturing Operations Management," The International Society of Automation, Tech. Rep., 2005.

[50] N. P. Suh, *Axiomatic Design: Advances and Applications*. Oxford University Press, 2001.

[51] D. Gasevic, D. Djuric, and V. Devedzic, *Model driven engineering and ontology development*, 2nd ed. Dordrecht: Springer, 2009.

[52] A. M. Farid, "Static Resilience of Large Flexible Engineering Systems : Axiomatic Design Model and Measures," *IEEE Systems Journal (in press)*, vol. PP, no. 99, pp. 1–12, 2015. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/IES-J19.pdf

[53] ——, "Static Resilience of Large Flexible Engineering Systems : Part I – Axiomatic Design Model," in *4th International Engineering Systems Symposium*, Hoboken, N.J., 2014, pp. 1–8.

[54] F. Bellifemine, G. Caire, and D. Greenwood, *DEVELOPING MULTI-AGENT SYSTEMS WITH JADE*. John Wiley & Sons, Ltd, 2007.

[55] D. Mcfarlane, V. Agarwal, A. A. Zaharudin, C. Y. Wong, R. Koh, and Y. Kang, "The Intelligent Product Driven Supply Chain," in *Systems, Man and Cybernetics, 2002 IEEE International Conference On*, vol. 4, no. February 4, 2002, pp. 393–398.

[56] G. G. Meyer, K. Främling, and J. Holmström, "Intelligent products: A survey," *Computers in Industry*, vol. 60, no. 3, pp. 137–148, 2009.

[57] D. McFarlane, V. Giannikas, A. C. Wong, and M. Harrison, "Product intelligence in industrial control: Theory and practice," *Annual Reviews in Control*, vol. 37, no. 1, pp. 69–88, 2013.

[58] R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: A survey," *Journal of Intelligent Manufacturing*, vol. 17, pp. 111–131, 2006.

[59] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415 – 431, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034606000292

[60] R. W. Brennan and D. H. Norrie, "Metrics for evaluating distributed manufacturing control systems," *Computers in Industry*, vol. 51, no. 2, pp. 225–235, 2003.

[61] J. L. Chirn and D. C. McFarlane, "Evaluating Holonic Control Systems: A Case Study," in *Proceedings of the 16th IFAC World Confernece*, Prague, Czech Republic, 2005.

[62] H. Elmaraghy, "Flexible and reconfigurable manufacturing systems paradigms," *International Journal of Flexible Manufacturing Systems*, vol. 17, pp. 261–276, 2005.

[63] R. Brennan, "Towards the assessment of holonic manufacturing systems," in *INCOM 2006: 12th IFAC Symposium on Information Control Problems in Manufacturing*, Saint-Etienne, France, 2006.

Amro M. Farid (M'11) received the Sc.B. and the Sc.M. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA in 2000 and 2002 respectively, and then the Ph.D. degree in engineering from the Institute of Manufacturing, University of Cambridge, Cambridge, U.K., in 2007.

He is currently an Associate Professor of Engineering, Thayer School of Engineering Dartmouth College Hannover, New Hampshire 03755 2. He is also a Research Affiliate with the MIT Technology and Development Program. He maintains active contributions in the MIT Future of the Electricity Grid study, the IEEE Control Systems Society, and the MITMI initiative on the large-scale penetration of renewable energy and electric vehicles. His research interests generally include the system architecture, dynamics, and control of power, water, transportation, and manufacturing systems.

Luis Ribeiro (M'07) received the M.Sc., in Electrical and Computer Engineering, in 2007 and the Ph.D., in Electrical and Computer Engineering with specialization in Robotics and Computer Integrated Manufacturing, in 2012 from the Universidade Nova de Lisboa (UNL), Lisbon, Portugal.

He is currently a lecturer at the Division of Manufacturing Engineering, Department of Management and Engineering from the Linköping University. Linköping, Sweden. His research interests include the design, development and implementation of the next generation of production systems and its associated challenges namely: dynamic product routing under plug and produce conditions, runtime deployment of components, reconfigurability measurement, simulation and simulation in the loop, distributed system-level monitoring and diagnosis, advanced shop-floor logistics (focusing in product flow) and intelligent and self-organizing industrial systems.