Proceedings of ICAD2014
The Eighth International Conference on Axiomatic Design
Campus de Caparica – September 24-26, 2014
ICAD-2014-XX

AN AXIOMATIC DESIGN OF A MULTI-AGENT RECONFIGURABLE MANUFACTURING SYSTEM ARCHITECTURE

Amro M. Farid
afarid@masdar.ac.ae, amfarid@mit.edu
Engineering Systems & Management
Masdar Institute of Science & Technology
Abu Dhabi, United Arab Emirates
Mechatronics Research Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

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

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.

Keywords: multi-agent system, reconfigurability, reconfigurable manufacturing systems, axiomatic design.

1 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 products of increasingly short product lifecycle [Mehrabi et al. , 2002; Pine, 1999]. One particularly pertinent problem is the need to quickly and incrementally adjust production capacity and capability. To fulfil the needs of enterprises with extensive automation, reconfigurable manufacturing systems have been proposed as a set of possible solutions [Mehrabi et al. , 2000].

Definition 1. Reconfigurable Manufacturing System [Koren et al. , 1999]: [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 [Dashchenko, 2007; Setchi and Lagos, 2004]. These have included modular machine tools [Heilala and Voho, 2001; Ho and Ranky, 1995; Landers et al. , 2001; Shirinzadeh, 2002; Townsend, 2000] and distributed automation [Brennan and Norrie, 2001; Vyatkin, 2007] [Lepuschitz et al. , 2011]. Additionally, a wide set of IT-based paradigms such as Multi-Agent Systems [Leitão, 2009; Leitão et al. , 2012; Leitão and Restivo, 2006; Ribeiro and Barata, 2013a; b; Shen and Norrie, 1999; Shen et al. , 2003], Holonic Manufacturing Systems [Babiceanu and Chen, 2006; Ma□ ík et al. , 2002; McFarlane and Bussmann, 2000; 2003], Evolvable Assembly Systems [Ribeiro et al. , 2010], and Fractal Factories [Tharumarajah and Wells, 1997] have emerged. They include a number of notable reference architectures including PROSA [Van Brussel et al. , 1998], HCBA [Chirn, 2001] and ADACOR [Leitão and Restivo, 2006]. While these technological advances have demonstrated robust operation and been qualitatively successful in achieving reconfigurability, their ultimate industrial adoption remains limited [Marik and McFarlane, 2005].

Amongst the barriers to adoption has been the relative absence of 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 3.) how these designs may overcome their inherent design limitations to achieve greater reconfigurability in subsequent design iterations. In short, without a quantitative framework for assessing the design, potential industrial adopters are unconvinced by the technology's validation. Furthermore, and in addition to reconfigurability measurement, such methodologies must make a straightforward link between 1.) high level design principles (e.g. bionic, holonic, fractal) 2.) a reference architecture that is sufficiently general to apply to the scope of manufacturing systems it seeks to address 3.) the associated instantiation as a system-specific architecture 4.) and a detailed implementation with connected hardware. In short, even if a potential industrial adopter had confidence in a prototype implementation, there would be no certain to scale and transfer the knowledge of the prototype up to fullscale implementation. Rigorously derived reference architectures fill this gap because they provide they make all system-specific implementation simply instantiations of one design pattern.

This paper provides a multi-agent system reference architecture for reconfigurable manufacturing systems driven by a quantitative and formal design approach. In so doing, this paper makes four specific contributions. First, it roots itself in an established engineering design methodology: axiomatic design for large flexible engineering systems. Second, the architecture's design is directly informed by design principles distilled from prior works in reconfigurability measurement. Third, the resulting architecture is written in terms of the mathematical description used in reconfigurability measurement – thus facilitating its implementation. Finally, the reference architecture and its associated mathematical description straightforwardly address instantiation for system-specific application. These specific advantages have yet to be demonstrated in existing reference architectures such as PROSA [Van Brussel, Wyns, 1998], HCBA [Chirn, 2001] and ADACOR [Leitão and Restivo, 2006].

The remainder of the paper proceeds as follows. Section II qualitatively articulates 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. Finally, Section V brings the work to a conclusion.

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 [ISA, 2005] where

Definition 2. Reconfigurability [Farid 2007]: 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.

2 ARCHITECTURE DESIGN STRATEGY AND RATIONALE

The aim of this work is to develop an Axiomatic Design of a Multi-Agent Reconfigurable Manufacturing 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 behaviours described in the previous section. It is dependent upon the characteristics of the production's system structure [Farid, 2007; Farid, 2014b]. That said, Axiomatic Design states generally [Suh, 2001] and previous works on this subject [Farid, 2007; Farid, 2014b] 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.

The architecture design strategy and rationale is directly informed by recent work in the reconfigurability measurement of automated manufacturing systems [Baca et al. , 2013; Farid, 2007; 2008a; Farid, 2008b; Farid, 2013; 2014b; Farid and Covanich, 2008; Farid and McFarlane, 2008; McFarlane and Farid, 2007]. 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). Therefore, the architecture design strategy and rationale presented in this section is being described as a set of qualitative design principles which have been distilled from these prior works on quantitative reconfigurability measurement. The 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.

2.1 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) [Suh, 2001]: 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 [Suh, 2001].

In practice, this means that each functional requirement (i.e. production process) must be related mathematically to a design parameter (i.e. production resource). Ultimately, each match of production process to production resource is assigned an event (in the discrete-event sense) called a production degree of freedom [Farid and McFarlane, 2008] which individually and collectively have a number of interesting properties:

- Individually, they represent ***all*** of the available capabilities of the ***physical*** production system.
- Collectively, they represent the configuration of the production system.
- Their (countable) number represents the production system's reconfiguration potential.
- 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 [Farid and McFarlane, 2008].

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 [Farid and McFarlane, 2008]. 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 [Farid and McFarlane, 2008].

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 2. Existence: 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.

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.

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

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.

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

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).

Because the production of the entire production line can be described as sequences of individual production degrees of freedom, it is logical to describe the agents in terms of these same production degrees of freedom (Principle 1). In that regard, production degrees of freedom are the quantitative equivalent of agent semantic ontologies. The production degrees of freedom must also be necessary and sufficient; neither overstating nor understating the production system's capabilities (Principles 2 and 3). The agents must also have a level of aggregation that mimics that of the physical entities that they represent (Principle 4). Next, the agent architecture must distinguish between the existence and availability of its capabilities (Principle 5). The existence of sequence-dependent constraints on the physical shop floor suggests for the need for the same amongst the agents (Principle 6). For example, a material handling process must end where another material handling process begins. Finally, adding agent interactions beyond the ones on the physical shop floor is likely to introduce unnecessary constraints (Principle 7).

2.2 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 [Farid, 2008a; McFarlane and Farid, 2007]. 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. Physical Agents: Align agents' scope and boundaries with their corresponding physical resources and their associated production degrees of freedom.

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

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).

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 [Farid, 2008a]; thus hindering reconfiguration ease. 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. Finally, Principle 10 accounts for the potential for reconfiguration processes to be conducted manually or automatically.

3 MATHEMATICAL DESCRIPTION OF THE ARCHITECTURE

On the basis of the design principles described in the previous section, the Axiomatic Design Multi-Agent Reconfigurable Manufacturing System (ADMARMS) architecture is developed and is shown in Figure 1. It's high level structure is now discussed in terms of the mathematical treatment found in the reconfigurability measurement of automated manufacturing systems [Baca, Farid, 2013; Farid, 2007; 2008a; Farid, 2008b; Farid, 2013; Farid and Covanich, 2008; Farid and McFarlane, 2008; McFarlane and Farid, 2007].

3.1 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 [Farid and McFarlane, 2008]: A binary matrix J_S of size $\sigma(P)x\sigma(R)$ whose elements $J_S(w,v)\in\{0,1\}$ are equal to one when event e_{wv} exists as a production process p_w being executed by a resource r_v.

By Principle 2, these physical resources have their informatic counterpart in the resource agent (RA). It is decomposed into itself to allow a production system to be divided into a physical hierarchy of departments and cells (Principle 4).

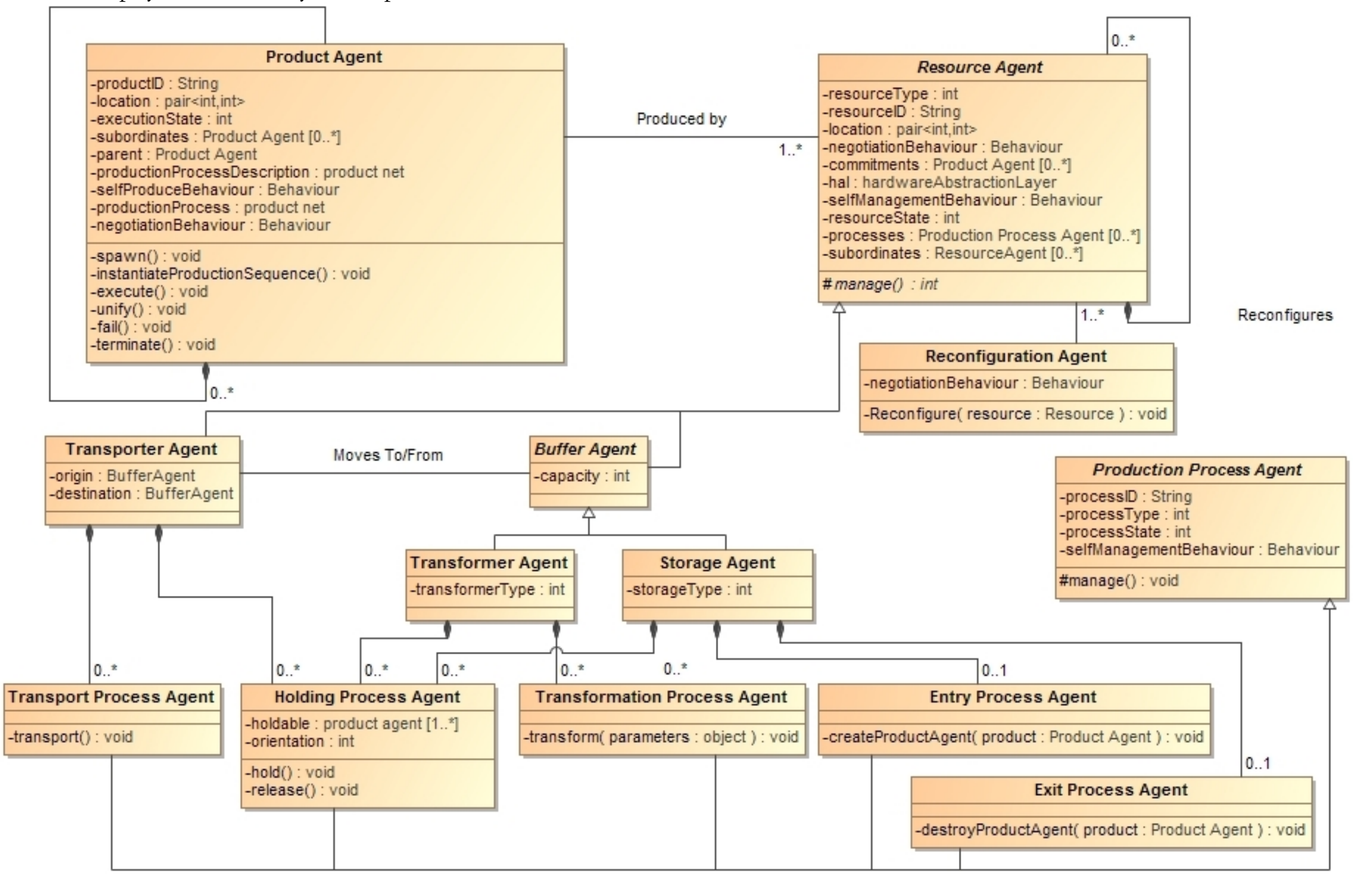
Product Agent
-productID : String
-location : pair<int,int>
-executionState : int
-subordinates : Product Agent [0..*]
-parent : Product Agent
-productionProcessDescription : product net
-selfProduceBehaviour : Behaviour
-productionProcess : product net
-negotiationBehaviour : Behaviour
-spawn() : void
-instantiateProductionSequence() : void
-execute() : void
-unify() : void
-fail() : void
-terminate() : void
Produced by
1..*
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
0..*
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

Figure 1. ADMARMS Architecture & Data Model

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. $R=M\cup B\cup H$. 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). $B_S=M\cup B$ *[Farid and McFarlane, 2008]*.

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 behavioural production processes they can perform.

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 [Farid, 2014a]. 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. $\sigma(P_\eta)=\sigma^2(B_S)$ where $\sigma()$ gives the size of a set. 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 [Farid, 2013; Farid and McFarlane, 2008]

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

where [Farid, 2013]

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

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

3.2 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 [Farid and McFarlane, 2008]: A binary matrix K_S of size $\sigma(P)x\sigma(R)$ 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 [Farid, 2013; Farid and McFarlane, 2008]:

$$K_S = \begin{bmatrix} K_M & | & \mathbf{1} \\ & K_{\bar{H}} & \end{bmatrix} \quad (3)$$

where [Farid, 2013]

$$K_{\bar{H}} = \left[K_\gamma \otimes \mathbf{1}^{\sigma(P_\eta)} \right] \cdot \left[\mathbf{1}^{\sigma(P_\gamma)} \otimes K_H \right] \quad (4)$$

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 [Farid and McFarlane, 2008]: The set of independent production events E_S that completely defines the available production processes in a production system. Their number is given by:

$$DOF_S = \sigma(E_S) = \sum_{w}^{\sigma(P)} \sum_{v}^{\sigma(R)} \left[J_S \ominus K_S \right](w,v) \quad (5)$$

where the $A \ominus B$ operation is boolean subtraction.

3.3 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 [Farid, 2013; Viswanath et al. , 2013]: a square binary constraints matrix of size $\sigma(P)\sigma(R)x\sigma(P)\sigma(R)$ whose elements $K_\rho(\varphi_1,\psi_1) \in \{0,1\}$ are equal to one when string $z_{\varphi 1 \psi 2} = e_{w1v1}e_{w2v2}$ 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 [Farid, 2013; Farid and McFarlane, 2008]. 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).

3.4 Production Model

The design of the ADMARMS architecture then includes product agents (PA) as the informatic counterpart 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[Farid, 2008b; McFarlane and Farid, 2007]:

Given product l_i, it may be described as:

$$N_{li} = \{S_{li}, E_{li}, F_{li}\} \quad (6)$$

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

Definition 9. Product Event [Farid, 2008b]: 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).

3.5 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 $\Lambda_{\mu i}$ [Farid, 2008b; Farid, 2013]: A binary matrix of size $\sigma(E_{li})x\sigma(P_\mu)$ whose value $\Lambda_{\mu i}(x,j) = 1$ if e_{xli} realizes transformation process $p_{\mu j}$.

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

Definition 10. Product Transportation Feasibility Matrix $\Lambda_{\gamma i}$ [Farid, 2008b; Farid, 2013]: A binary row vector of size $1x\sigma(P_\gamma)$ whose value $\Lambda_{\gamma i}(g) = 1$ if product l_i can be held by holding process $p_{\gamma g}$.

3.6 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 (PDSM) shown in Figures 2 &3 [Farid, 2008a] has been previously used for measurement. It captures the physical, energy, and informatic couplings between production system entities. While the PDSM is highly sparse, it does have heavy coupling on both sides of the main diagonal[Farid, 2008a]. As a result, this paper advocates heuristic-based approaches to usage rather than optimization. The latter being extremely computationally intensive (e.g. NP-hard job-shop NP-hard scheduling problems.)

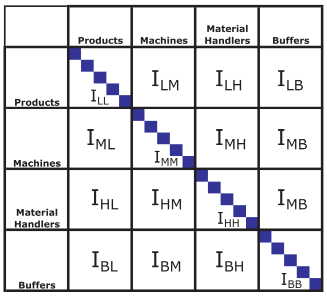

Figure 2. Production System Design Structure Matrix

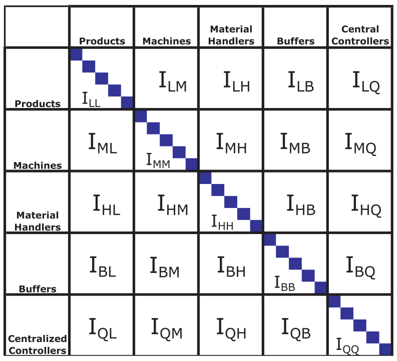

Figure 3. Production System Design Structure Matrix (with centralized controllers)

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 behaviours of agent-to-agent interactions and negotiations, care will be taken to demonstrate that each agent is maximally cohesive and minimally coupled (Principle 9).

3.7 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 [Farid and Covanich, 2008; Farid and McFarlane, 2008]:

$$(J_s, K_s, K_p) \rightarrow (J_s', K_s', K_p') \qquad (7)$$

Examples of reconfiguration agents are automatic tool changers (change of transformation degree of freedom), automatic fixture changers (change of holding degree of freedom) and conveyor gates (change of transportation degree of freedom).

4 ARCHITECTURE DATA MODEL IMPLEMENTATION

As shown in Figure 1, 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) [Bellifemine et al. , 2007].

4.1 RESOURCE AGENT (RA)

The RA is an abstraction for all the potential manufacturing 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 under the control of a specific resource in a master-slave relation.
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 [Ribeiro and Barata, 2013a].

4.2 SPECIALIZATIONS OF THE RA

As shown in Figure 1, the RA is specialized as a BA, TFA, SA and a TPA. These are discussed in turn.

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.3 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.

4.4 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.

4.4.1 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.

4.4.2 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 parameterization to accommodate the distinct transformation requirements.

4.4.3 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.

4.4.4 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.

4.5 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. This composition models sub-assemblies and component parts.
-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.

4.6 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-parameterize existing ones. The RCA will then assess the availability and potential reconfiguration conflicts on the desired processes and reconfigure the RA accordingly.

5 CONCLUSION

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.

6 REFERENCES

[1] Babiceanu RF, Chen FF. Development and applications of holonic manufacturing systems: a survey. Journal of Intelligent Manufacturing. 2006;17:111-31.
[2] Baca EES, Farid AM, Tsai I-T. An Axiomatic Design Approach to Passenger Itinerary Enumeration in Reconfigurable Transportation Systems. Proceedings of ICAD2013 The Seventh International Conference on Axiomatic Design2013. p. 8.
[3] Bellifemine FL, Caire G, Greenwood D. Developing multi-agent systems with JADE: John Wiley & Sons; 2007.
[4] Brennan RW, Norrie DH. Agents, holons and function blocks: distributed intelligent control in manufacturing. Journal of Applied Systems Studies. 2001;2:1-19.
[5] Chirn J-L. Developing a reconfigurable manufacturing control system: a holonic component-based approach: University of Cambridge; 2001.
[6] Dashchenko AI. Reconfigurable manufacturing systems and transformable factories: Springer; 2007.
[7] Farid A. Reconfigurability measurement in automated manufacturing systems. University of Cambridge, Cambridge. 2007.
[8] Farid A. Facilitating ease of system reconfiguration through measures of manufacturing modularity. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. 2008a;222:1275-88.
[9] Farid A. Product Degrees of Freedom as Manufacturing System Reconfiguration Potential Measures. International Transactions on System Science and Applications. 2008b;4:227-42.
[10] Farid A. Static Resilience of Large Flexible Engineering Systems : Part I - Axiomatic Design Model. 4th International Engineering Systems Symposium2014a.
[11] Farid AM. An Axiomatic Design Approach to Production Path Enumeration in Reconfigurable Manufacturing Systems. Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on: IEEE; 2013. p. 3862-9.
[12] Farid AM. Axiomatic Design & Design Structure Matrix Measures for Reconfigurability & Its Key Charactersitics in Automated Manufacturing Systems. International

Conference on Axiomatic Design. Caparica, Portugal2014b. p. 1-8.

[13] Farid AM, Covanich W. Measuring the effort of a reconfiguration process. Emerging Technologies and Factory Automation, 2008 ETFA 2008 IEEE International Conference on: IEEE; 2008. p. 1137-44.

[14] Farid AM, McFarlane D. Production degrees of freedom as manufacturing system reconfiguration potential measures. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture. 2008;222:1301-14.

[15] Heilala J, Voho P. Modular reconfigurable flexible final assembly systems. Assembly Automation. 2001;21:20-30.

[16] Ho JK, Ranky PG. An object-oriented and flexible material handling system. Assembly automation. 1995;15:15-20.

[17] ISA. Enterprise Control System Integration Part 3: Activity Models of Manufacturing Operations Management. Tech. Rep.2005.

[18] Koren Y, Heisel U, Jovane F, Moriwaki T, Pritschow G, Ulsoy G, et al. Reconfigurable manufacturing systems. CIRP Annals-Manufacturing Technology. 1999;48:527-40.

[19] Landers RG, Min B-K, Koren Y. Reconfigurable machine tools. CIRP Annals-Manufacturing Technology. 2001;50:269-74.

[20] Leitão P. Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence. 2009;22:979-91.

[21] Leitão P, Barbosa J, Trentesaux D. Bio-inspired multi-agent systems for reconfigurable manufacturing systems. Engineering Applications of Artificial Intelligence. 2012;25:934-44.

[22] Leitão P, Restivo F. ADACOR: A holonic architecture for agile and adaptive manufacturing control. Computers in industry. 2006;57:121-30.

[23] Lepuschitz W, Zoitl A, Vallée M, Merdan M. Toward self-reconfiguration of manufacturing systems using automation agents. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on. 2011;41:52-69.

[24] Marík V, Fletcher M, Pechoucek M. Holons & agents: Recent developments and mutual impacts. Multi-Agent Systems and Applications II: Springer; 2002. p. 233-67.

[25] Marik V, McFarlane D. Industrial adoption of agent-based technologies. IEEE Intelligent Systems. 2005;20:27-35.

[26] McFarlane D, Farid A. A design structure matrix based method for reconfigurability measurement of distributed manufacturing systems. International Journal of Intelligent Control and Systems. 2007;12:118-29.

[27] McFarlane DC, Bussmann S. Developments in holonic production planning and control. Production Planning & Control. 2000;11:522-36.

[28] McFarlane DC, Bussmann S. Holonic manufacturing control: Rationales, developments and open issues. Agent-based manufacturing: Springer; 2003. p. 303-26.

[29] Mehrabi MG, Ulsoy AG, Koren Y. Reconfigurable manufacturing systems and their enabling technologies. International Journal of Manufacturing Technology and Management. 2000;1:114-31.

[30] Mehrabi MG, Ulsoy AG, Koren Y, Heytler P. Trends and perspectives in flexible and reconfigurable manufacturing systems. Journal of Intelligent manufacturing. 2002;13:135-46.

[31] Pine BJ. Mass customization: the new frontier in business competition: Harvard Business Press; 1999.

[32] Ribeiro L, Barata J. Deployment of Multiagent Mechatronic Systems. Industrial Applications of Holonic and Multi-Agent Systems: Springer; 2013a. p. 71-82.

[33] Ribeiro L, Barata J. Self-organizing multiagent mechatronic systems in perspective. Industrial Informatics (INDIN), 2013 11th IEEE International Conference on: IEEE; 2013b. p. 392-7.

[34] Ribeiro L, Barata J, Cândido G, Onori M. Evolvable production systems: an integrated view on recent developments. Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology: Springer; 2010. p. 841-54.

[35] Setchi RM, Lagos N. Reconfigurability and reconfigurable manufacturing systems: state-of-the-art review. Industrial Informatics, 2004 INDIN'04 2004 2nd IEEE International Conference on: IEEE; 2004. p. 529-35.

[36] Shen W, Norrie DH. Agent-based systems for intelligent manufacturing: a state-of-the-art survey. Knowledge and information systems. 1999;1:129-56.

[37] Shen W, Norrie DH, Barthès J-P. Multi-agent systems for concurrent intelligent design and manufacturing: CRC press; 2003.

[38] Shirinzadeh B. Flexible fixturing for workpiece positioning and constraining. Assembly Automation. 2002;22:112-20.

[39] Suh NP. Axiomatic Design: Advances and Applications (The Oxford Series on Advanced Manufacturing). 2001.

[40] Tharumarajah A, Wells A. A behavior-based approach to scheduling in distributed manufacturing systems. Integrated Computer-Aided Engineering. 1997;4:235-49.

[41] Townsend W. The BarrettHand grasper–programmably flexible part handling and assembly. Industrial Robot: An International Journal. 2000;27:181-8.

[42] Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P. Reference architecture for holonic manufacturing systems: PROSA. Computers in industry. 1998;37:255-74.

[43] Viswanath A, Baca EES, Farid AM. An Axiomatic Design Approach to Passenger Itinerary Enumeration in Reconfigurable Transportation Systems. 2013.

[44] Vyatkin V. IEC 61499 function blocks for embedded and distributed control systems design: ISA-Instrumentation, Systems, and Automation Society; 2007.