Computers in Industry 120 (2020) 103226

Contents lists available at ScienceDirect

Computers in Industry

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

Perspective on holonic manufacturing systems: PROSA becomes ARTI

Paul Valckenaers[a,b]

[a] UCLL, Digital Solutions, Proximus Campus, Leuven, Belgium
[b] KU Leuven, Faculty of Engineering Technology, Bruges Campus, Bruges, Belgium

ARTICLE INFO

Article history:
Received 28 February 2020
Received in revised form 9 March 2020
Accepted 11 March 2020
Available online 26 May 2020

Keywords:
Holonic manufacturing systems
Manufacturing execution systems
Design for the unexpected
Obedient digital twins
PROSA
ARTI
Sciences of the artificial
Bounded rationality
Complex-adaptive systems
Autocatalytic sets

ABSTRACT

Looking back at 30 years of research into holonic manufacturing systems, these explorations made a lasting scientific contribution to the overall architecture of intelligent manufacturing systems. Most notably, holonic architectures are defined in terms of their world-of-interest (Van Brussel et al., 1998). They do not have an information layer, a communication layer, etc. Instead, they have components that relate to real-world assets (e.g. machine tools) and activities (e.g. assembly). And, they mirror and track the structure of their world-of-interest, which allows them to scale and adapt accordingly.

This research has wandered around, at times learning from its mistakes, and progressively carved out an invariant structure while it translated and applied scientific insights from complex-adaptive systems theory (e.g. autocatalytic sets) and from bounded rationality (e.g. holons). This paper presents and discusses the outcome of these research efforts.

At the top level, the holonic structure distinguishes intelligent beings (or digital twins) from intelligent agents. These digital twins inherit the consistency from reality, which they mirror. They are intelligent beings when they reflect what exists in the world without imposing artificial limitations in this reality. Consequently, a conflict with a digital twin is a conflict with reality.

In contrast, intelligent agents typically transform NP-hard challenges into computations with low-polynomial complexity. Unavoidably, this involves arbitrariness (e.g. don't care choices). Likewise, relying on case-specific properties, to ensure an outcome in polynomial time, usually renders the validity of an agent's choices both short-lived and situation-dependent. Here, intelligent agents create conflicts by imposing limitations of their own making in their world-of-interest.

Real-world smart systems are aggregates comprising both intelligent beings and intelligent agents. They are performers. Inside these performers, digital twins may constitute the foundations, supporting walls, support beams and pillars because these intelligent beings are protected by their real-world counterpart. Further refining the top-level of this architecture, a holonic structure enables these digital twins to shadow their real-world counterpart whenever it changes, adapts and evolves.

In contrast, the artificial limitations, imposed by the intelligent agents, cannot be allowed to build up inertia, which would hamper the undoing of arbitrary or case-specific limitations. To this end, performers explicitly manage the rights over their assets. Revoking such rights from a limitation-imposing agent will free the assets. This will be at the cost of reduced services from the agent. When other service providers rely on this agent, their services may be affected as well; that's how the inertia builds up and how harmful legacy is created. Thus, the services of digital twins are to be preferred over the services of an intelligent agent by developers of holonic manufacturing systems.

Finally, digital twins corresponding to the decision making in the world-of-interest (a non-physical asset) allow to mirror the world-of-interest in a predictive mode (in addition to track and trace). It allows to generate short-term forecasts while preserving the benefits of intelligent beings. These twins are the intentions of the decision-making intelligent agents. Evidently, when intentions change, the forecasts needs to be regenerated (i.e. tracking the corresponding reality by the twin). This advanced feature can be deployed in a number of configurations (cf. annex).

E-mail addresses: Paul.Valckenaers@ucll.be, Paul.Valckenaers@kuleuven.be

https://doi.org/10.1016/j.compind.2020.103226
0166-3615/© 2020 Elsevier B.V. All rights reserved.

1. Introduction

In its 30 years of existence, Computers in Industry has offered an unrivaled forum for interdisciplinary research. Looking back, “Reference architecture for holonic manufacturing systems: PROSA” (Van Brussel et al., 1998) has been a landmark for Computers in Industry as is reflected in its citations (Times Cited on 16/01/2020: 600+ by Web of Science, 900+ by Scopus, 1000+ by ResearchGate, 1500+ by Google Scholar) and, more significantly, by the research that has adopted, adapted, imitated, extended, misunderstood, opposed and/or enhanced PROSA. Here, building on PROSA, this manuscript selects and covers research results with the objective to *highlight what separates holonic systems from other research contributions* in its domain. A survey of holonic manufacturing control can be found in (Derigent et al., 2020).

Holonic systems, which includes holons in manufacturing, originate from fundamental research on bounded rationality. Indeed, a human brain has a bounded information processing and communication capacity. Nobel Prize winner Herbert Simon has been a pioneer in demonstrating how this limited brain capacity has unavoidable implications. Simon’s work reveals that human-created artifacts (e.g. software systems) are subject to scientific laws, which he calls *the sciences of the artificial* (Simon, 1996).

These *scientific laws of the artificial* exhibit similarities to the *Principles of Carnot* when they state what is impossible. Carnot revealed the impossibility of a perpetuum mobile, where its analogy in Simon’s work shows why *monolithic designs rapidly become impossible when their complexity and size grows*.

Simon’s work shares similarities with Einstein’s work versus Newton’s work, where Newton’s work becomes ineffectual when velocities approach the speed of light. Conventional approaches to creating artifacts are fine until they become too large and complex in environments that are dynamic and competitive. Here, accounting for Simon’s heritage is rapidly becoming indispensable in our society.

Indeed, in a nutshell, Simon observes that the combination of (1) a finite time window (i.e. a dynamic environment) and (2) finite information processing (i.e. bounded rationality) and (3) competitive pressures (i.e. survival of the fittest/first/...) unavoidably implies that *flexible time-varying aggregation hierarchies are the only option*. Monolithic designs take too much time to build or adapt. Simon’s malleable aggregates are the so-called *holarchies*, which are *hierarchies of holons*.

Moreover, the holons in these holarchies exhibit some *autonomy allowing to cope with a range of challenges and conditions* and an *ability to cooperate*. This was the understanding during the early days of research on holonic manufacturing systems, which was not connected to any fundamental research. This paper addresses this by connecting it to complex-adaptive systems research and splitting this desirable property into more detailed and precise characteristics. Note: it takes more than autonomy to be a proper holon, and some design for the unexpected is needed for cooperation.

Overall, the contribution of this paper is to carve out what made and makes holonic systems different where this difference is the scientific contribution. This difference fits Computers in Industry as it is highly interdisciplinary. It is about software in relation to a corresponding reality. It is about computer-based intelligence that has a body, is situated and will evolve. And, it is about software that is mirroring a corresponding reality without imposing limitations of its own making.

First, the discussion looks at the past with the benefit of hindsight. It starts with PROSA as a *translation of the design of a computer operating system kernel*. In this early stage, PROSA is an heterarchical manufacturing execution system. These heterarchical designs are criticized for their inability to guarantee performance (i.e. system behavior is unpredictable) and/or to optimize globally (i.e. only local decision making by intelligent products or product carriers).

Hence, the discussion continues with *hybrid designs* – combining hierarchical and heterarchical control – to address these issues. The hybrid designs contribute on the performance aspect but inherit drawbacks of hierarchical control (e.g. they are limited as regards the properties of the underlying manufacturing systems and/or the operating conditions). Accordingly, *bio-inspired designs* enhance heterarchical designs and preserve the advantages of heterarchical designs over hierarchical control. These bio-inspired designs facilitate global performance optimization and prediction of performance; they avoid the issues with the hybrid designs but solely facilitate addressing the performance matter.

Further, *holonic manufacturing and multi-agent systems* (MAS) have been considered closely related research domains. However, after getting acquainted with both domains, they are fundamentally different as soon as the narrower views of the respective research communities are taken into account. The discussion focuses on where and how these two research domains differ.

Next, the discussion looks *beyond holonic manufacturing*. It starts with the application of PROSA-related research results to other application domains. This confrontation with new domains resulted in a more universally applicable reference architecture, an evolution of PROSA: ARTI or activity-resource-type-instance. This avoids manufacturing-only wordings and acknowledges that the true scope of holonic execution systems is “activities executing on resources”.

Applying holonic execution systems to new domains revealed the importance of lucid communication of the holonic concepts to researchers in those domains. ARTI is an improvement. The concept of digital twins for each of the ARTI components is another enhancement in this regard. Nonetheless, significant challenges remain; e.g. ARTI digital twins have a much broader scope than twins of physical assets.

Furthermore, encountering issues while translating (i.e. attempting, suffering and even failing to do so) the above-mentioned bio-inspired enhancements, the view on their position in the overall execution system was revised. This corrected view draws *a better-founded boundary between decision making elements and the remainder of a system*. This boundary respects the autonomy of researchers focusing on decision making (e.g. planning and scheduling in operations research, control engineering or AI).

Finally, as the guiding principle across the discussion is accounting for the scientific laws of the artificial and complex-adaptive systems, a better understanding of the autonomy in holons is linked to the concept of an autocatalytic set. These sets execute self-reinforcing cycles that allow holons to sustain and adapt. The discussion covers system properties and features that enable membership of such sets as well as facilitating the onset and sustaining of such self-reinforcing cycles.

2. Holonic manufacturing execution systems (HMES)

This section looks at holonic systems applied to manufacturing execution. Section 3 addresses the application of holonic execution systems in other domains.

2.1. Operating systems for manufacturing systems

Industrial automation has developed information technologies with fundamentally different designs. More or less chronologically, this includes:

- *Printer attached to a computer*: a CNC program is as a document that is sent to a printer. But instead of documents, it are products or product parts that are made by a machine tool. 3D printers commonly employ the same design. It originates from the 1940's and 1950's when computers were too expensive and too bulky to be attached/dedicated to single machine tool. Instead, the computer punched holes in a paper tape and pianola-like electromechanical devices would instruct the machine tools to produce the proper parts (repeatedly). Justified at the time, this is an overly rigid design in today's context. Indeed, companies have developed mechatronic control systems to address this but are unable to market it; indeed, a lock-in into this rigid 'printing architecture' suppresses the spreading of these improvements (cf. Valckenaers and Van Brussel, 2015, pp. 33–34).

- *Software replacing hardwired relays, timers and sequencers*: a PLC or programmable logic controller adopts the design from hardwired relays, timers and sequencers. This is well-suited for real-time decision making based on input values and unsophisticated state data. It struggles with more elaborate state representations and advanced information processing of those states. Here, IEC-61499 is an effort to remedy this and brings PLC programming up to the level of modern programming languages (Zoitl et al., 2007). IEC-61499 is struggling to achieve critical user mass, which is imperative for a technology of its complexity. Likewise, it is unlikely to become part of mainstream programming technology. It is however a unique effort as it is rooted in the industrial automation community and allows for programmers to tell techno-ignorant bosses that they are using function blocks, omitting that it are not IEC-61131 function blocks.
- *Domain-specific programming languages*: a typical industrial robot control system brings its own programming language. This struggles with the impact of (a lacking) user mass, which prevents these from being state-of-the-art (typically PASCAL-like) and e.g. renders them ill-suited for Industry 4.0 applications. In addition, the design of these languages was a compromise to fit computer hardware in the 1980s (typically 16-bit microprocessors with kilobytes of RAM memory and a few megabytes of hard disk storage). This effectively prevented to incorporate the knowhow from mechanical engineering such as accounting for the dynamics of the robot arm. E.g. the robot motion primitives reflect a rather naïve geometry-only view on the world, which creates issues when the robot is to move fast.

Looking back, industrial automation has developed its own information technologies adapting to the conditions at the time. Regrettably, this has resulted in design choices that are ill-adapted to the present time and those choices have built up considerable inertia: it has proven impossible in practice to undo or reverse them (i.e. lock-in as in (Valckenaers and Van Brussel, 2015) pp.33–34).

There is however another starting point in mainstream IT to address the challenges and concerns in industrial automation: the computer operating systems (OS):

- *Computer operating systems–device drivers:* in mainstream computer systems, peripheral devices are controlled and operated through a device driver. A device driver is a computer program that operates or controls a particular type of device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions. The actual applications that are using the devices can be programmed in any programming technology (or mix and combination thereof) supported on the given computer and its operating system. Typically, industrial applications may require the operating system to support hard real-time processes inside the device driver and, often, in the application software as well. ***Mechatronic control systems adopt this design***. This approach allows to use state-of-the-art programming technologies enjoying proper user masses. But, it requires multi-disciplinary skills bridging mainstream IT with industrial automation.
- *Computer operating systems - kernel:* modern computer operating systems manage a pool of software processes. It is a resource management system where *processor cores* are the resources, the *code segments* are the recipes and *data segments* are the state of a recipe that is being executed. PROSA translates this to manufacturing execution systems (MES). The resource holons are workstations and production equipment, the product holons are process plans or recipes and order holons are the parts and products that are being made by executing their recipe/process plan. ***The holonic MES becomes an operating system for a factory***.

The device driver brings freedom to industrial automation. It has however a missing link: hard real-time is a balkanized area within mainstream computing. There is no leading real-time OS with a dominating share of the overall user community. Similarly, the dominating operating systems (Linux, Windows, iOS, Android) are lacking hard real-time services. Moreover, mechatronics merely delivers this freedom but does not offer a widely adopted structure beyond that (i.e. designers apply control theory, mechanics, etc. directly into their software written in C/C++, Rust, etc.) as concerns the system architecture.

Holonic systems bring structure and guidance in their subdomain (manufacturing execution). PROSA identifies how an application is to be subdivided into smaller components, what the scope and services of the respective components are, and how they interact and organize themselves (Van Brussel et al., 1998):

- Resource holons – production equipment (i.e. capable of executing processing steps)
- Product holons – non-deterministic process plans (i.e. all known/validated step sequences)
- Order holons – executing a valid step sequence on resources
- Aggregation – possibly varying over time mirroring a world of interest
- Specialization – provide abstraction and reuse

PROSA originated as a response, in a meeting, to a functional decomposition proposition. The key motivation to prefer PROSA was structural decomposition: the design scales with its world of interest. A functional decomposition rapidly becomes unable to decompose further. In addition, PROSA provides suitable abstraction allowing to service wide ranges of components with a single solution.

Moreover, a PROSA subdivision increases the (upper bound on the) user mass for the holons drastically: management (routing, resource allocation) and technology (valid processing sequences) are provided by different software components that can be combined as needed. Job shops and flow shops need different managers whereas assembly and machining need different technology experts. Also, the simplicity and clarity of the reference architecture contribute to its applicability and recognition.

Evidently, computer operating systems and holonic manufacturing execution systems enjoy different conditions. Operating systems cannot spend a lot of computing power to manage their pool of computing processes as better decisions need to compensate this spending; this computing power, which the OS consumes, is lost for the actual computing processes for the users. A holonic MES has a nicer trade-off: computing power versus manufacturing equipment utilization. The former is cheaper and keeps getting more affordable in the foreseeable future. Conversely, microprocessors have been designed to assist the OS; its low-effort management of processing cores is aided by their design and renders complex decision making superfluous. In contrast, the MES faces a wide range of possible underlying manufacturing systems and poor decisions may cause significant losses (relative to the cost of computing).

Clearly, the translation of an OS Kernel toward MES represents an enduring research agenda rather than a short-lasting single effort. It provides a start of an improvement and enhancement process in the MES domain. The next section dives into this matter further. Note that, in the future, the computer OS is likely to share more challenges with the manufacturing OS as it will be managing a more heterogeneous set of resources when general-purpose cores are combined with increasing numbers of special-purpose cores (e.g. tensor cores). Also, when high service levels are to be guaranteed, such common ground may grow when resource capacity needs to be pre-assigned (scheduled?) to guarantee a service.

2.2. Pressures on heterarchical designs – hybrid system architectures

Basic PROSA implementations – and many other heterarchical designs – operate *as cars in traffic*. Each individual product (order holon) routes itself in order to get itself processed properly. These systems are robust and self-configuring when they opportunistically find free processing slots on suitable resources (Duffie and Prahbu, 1994). The more advanced designs use non-deterministic process plans, which increases the robustness even more (Detand, 1993).

Here, simplicity and robustness have a price: these heterarchical systems are myopic and performance optimization is virtually nonexistent. Worse, predictability as regards performance is absent as well. In contrast, industrial practice delivers on predictability and good performance at the expense of robustness and genericity. It requires a much tighter control over the operating conditions of the manufacturing system. And, specific control systems are needed for specific production systems. This situation results into pressure on the holonic systems researchers, especially from research communities focusing on performance while disregarding these other issues with industrial hierarchical control.

This community pressure to deliver ‘optimized performance’ and ‘performance guarantees’ triggered research efforts that are hybrids (from a pure heterarchical perspective). Some examples, impacting the system architecture, are discussed below (nonexhaustively).

2.2.1. ADACOR – www.ipb.pt/∼pleitao/index.php/adacor

ADACOR – inspired by PROSA – has its product holon contribute to the integration of all the manufacturing control functions (e.g. planning and scheduling). It is a bridge between the shop floor and the planning level. As such, it is designed to fit the prevailing organization within manufacturing enterprises. In contrast, PROSA confines its product holon to being an expert on (all possible alternatives of) how an order may be produced. PROSA also encapsulates (hides) the details of the process planning activities. The bridging in PROSA is facilitated by non-interference; its product holon will not impose demands beyond what is technically infeasible, uncertified. . .

Much more significantly, ADACOR introduces the supervisor holon, bringing hierarchy into its architecture (Leitão, 2004; Barbosa et al., 2015). *The supervisor holon brings coordination and global optimization, answering precisely to the pressure from the research community and industry.* In normal operation, this supervisor holon oversees and regulates the activity of the holons under its domain while, when a disturbance occurs, these holons may have to find their way without the help of the supervisor holon.

2.2.2. ORCA-FMS

ORCA is an hybrid control architecture (Pach, 2013; Pach et al., 2014). This hybrid architecture is able to dynamically and partially switch between a hierarchical predictive mode and a heterarchical reactive mode. If an event ‘forbidding the planned behavior to be followed’ occurs, the heterarchical mode takes over.

ORCA-FMS answers the pressure to address performance in its entirety. It uses mixed integer linear programming in the hierarchical mode for global control, and it has local control to detect perturbations and, in case of perturbation, to switch to disrupted mode when indicated. The disrupted mode uses potential fields to guide the self-routing.

2.2.3. Staff holons in PROSA

The PROSA reference architecture introduces a staff holon to address this demand for predictable and globally optimized performance. However, although the team investigated the collaboration of reactive schedulers with holonic manufacturing execution, the emphasis is on the relationship of staff holons with the other holons (product, resource and order). *To qualify as a staff holon (e.g. a scheduler), the staff holon must be no more than an advisor.*

Removal or disabling of a staff holon must not break the execution system composed of order holons, product holons and resource holons. Performance may degrade, performance predictability may suffer, but the heterarchical operating mode remains operational. Worst case would be a malfunctioning staff holon that transmits deceptive advice to the execution system, but even that needs to be survivable.

2.2.4. Discussion

The emphasis in PROSA of confining schedulers and planners to an advisory role is motivated by their inability to preserve the robustness and broad applicability of the heterarchical-only designs. Indeed, manufacturing planning and/or scheduling problems are NP-Hard (or worse). Any optimizing intrinsically and unavoidably requires a reduction of the solution space to (low) polynomial size.

This reduction will be partially random where most of the nonarbitrary part depends on the specifics of the underlying production system and its environment. This dependency means that a hybrid design breaks when confronted with changes in its current target and/or when applied to other manufacturing systems.

In fact, these hybrid designs are attempts to preserve the range of manufacturing systems, products and market conditions under which they remain effective or even just usable, but they do not succeed in that respect. E.g. connecting a number of such hybrid designs will not be easy; without being allowed to modify them at code level, it often will prove to be impossible. In other words, hybrid designs achieve their goals concerning manufacturing control: they combine *optimization and predictability whenever the operating conditions allow it* with *robustness when the operating conditions are bad*. But, hybrid designs inherit the drawbacks of more conventional manufacturing control as regards the ability to serve a wide range of underlying manufacturing systems/environments, which includes struggling with changes in underlying system as well as co-existence with other systems.

On the other hand, this inability to deliver a perfect outcome does not make the industrial concern (predictability and high performance) disappear and these research efforts into hybrid designs are justified. The research efforts are especially justified where the cooperation between a scheduling system and heterarchical systems has been investigated by means of research prototypes. Important to highlight, some challenges and issues emerged and warrant further investigation:

- What does it mean to follow/execute a given schedule? When to disregard which parts/aspects of a given schedule? How to fill in the missing parts/details in a schedule?

- How to reconnect to the (updated) schedule? How to start following a given schedule (closer) anew when a disruption or disturbance made the system engage heterarchical mode(s)?
- How to collaborate with a reactive scheduler? What must be the initial state from which the rescheduling starts (when rescheduling requires non-negligible amounts of time)?
- How to include autonomy or margins in a schedule for the execution system? How may the scheduler provide the execution system with information to handle (necessary) deviations in optimized ways (e.g. for predictability)?
- How to have multiple schedulers, planners, execution systems collaborate without having to re-design and re-implement them?
- How does this fit in manufacturing supply networks and chains, comprising also logistic execution systems, planned and unplanned maintenance, etc.?

Overall, there is the issue of unavoidable arbitrariness when delivering a manufacturing control that handles all the decisions that are to be made. This arbitrariness is the root cause of integration failures, deficient in-depth interoperability and underwhelming performance, especially when scaling beyond organizational boundaries. As this arbitrariness needs 'undoing' when confronted with these larger-scale challenges, a reference architecture is to minimize its inertia (i.e. minimize the effort needed by avoiding a cascade of system components that need to change when some arbitrariness needs undoing).

2.3. Pressures on heterarchical designs – bio-inspired forecasting

The hybrid architecture delivers answers to the pressure concerning performance and predictability, but the associated drawbacks remain significant. In many ways, the rigidity of hierarchical control as regards software development and software maintenance remained. The decision making and optimization in the control systems still need adapting to a specific manufacturing system, characteristics of processing equipment, transportation systems and production environment. Effectively, the hybrid designs handle disruptions and disturbances on an operational level but not concerning the software itself.

Accordingly, research continued to enhance heterarchical control with forward-looking intelligence in which the effect of future interactions is accounted for. In a basic heterarchical control, a smart product (carrier) ensures that a proper trajectory along suitable processing units is executed. It uses locally available information and some heuristics but no system-level information processing. More advanced designs started to use coordination fields, assisting e.g. a smart product to find a processing unit that is available rather than queueing before a busy unit. But such enhancements still fail to incorporate future interactions in manners accounting for the specifics of process plans, resource capabilities and availability. Such coordination fields need regular tuning and do not guarantee performance.

Here, food foraging in ant colonies provided the inspiration for a superior design. When ants deposit a pheromone trail, from their nest to a food source, they use their environment as part of their solution. As a consequence and strategic bonus, the algorithm in the ant's brain does not need a model of this environment. A simple procedure copes with a huge variety of situations and accounts for all the details in these situations. This created the 'eureka-moment' enabling the design of forward-looking heterarchical coordination accounting for the specifics of the interactions.

For obvious reasons, it is uneconomical to have the actual products perform the exploration and (digital) pheromone depositing task as proposed in fundamental research (e.g. by Brooks). This had to be virtual. To this end, resource holons provided a self-model allowing a visitor, representing an order holon, to have its steps executed virtually. In other words, every smart product is able to virtually execute a valid trajectory (much) faster than real time.

This information dissemination and collection design was called a delegate multi-agent system (Holvoet et al., 2012). Order holons would create at regular time intervals a lightweight computer process (called an "ant agent") that virtually executes the routing and processing for the order through the factory. The order holons may do this in (at least) two manners:

- *Intention propagation*. While virtually executing a given trajectory (indicating which resources will be visited and which processing steps are to be executed), an ant agent informs all the resource holons on its route; it reserves the required processing capacity with these resource holons. This generates a short-term forecast for the overall production system. This is repeated regularly to account for any changes. Failure to reconfirm a reservation makes it disappear. *Intention propagation generates an innovative kind of coordination field.*
- *Exploration*. Exploration involves the same virtual execution but does not reserve any processing capacity with the resource holons on route. *It observes the coordination field generated by the intention propagation.* The virtual execution is needed for the order holon to extract proper information from the coordination field as it is relevant for its execution. Exploration creates ant agents, at regular time intervals, that typically executes different trajectories to discover which trajectories are best or preferable.

The ant agents use the product holons as they are; here, PROSA's strict boundaries on holon responsibilities pays off as a product holon cannot distinguish virtual from actual execution. The order holons and their ant agents both execute the same NEU protocol (Valckenaers and De Mazière, 2015). Most importantly, the resource holons and the product holons only need self-knowledge to support this virtual execution. This makes the design plug-and-produce and addresses the software maintenance issue in the hybrid architectures.

Now, ***resource holons have an agenda*** and ***order holons have a routing***. This facilitates addressing predictability, provided order holons don't change their intentions too easily (Hadeli, 2006). This provides a basis for performance optimization when congestions and opportunities are detected early. At the same time, it avoids the "NP-hard $\rightarrow$ low-polynomial" problem by leaving it to the application-specific software parts. In other words, the order, product and resource holons become reusable and long-lived but also their swarms of ant agents. The result gets a significant part of the benefits of a hybrid design without the drawbacks.

Moreover, the design offers a cooperation mode to schedulers when exploration dedicates a significant effort to exploring the scheduled routes, when intention selection prefers the scheduled route on condition that the predicted performance matches the scheduled performance, and when resource holons give priority to allocations according to the schedule. Finally, note that the generated forecast is not constrained by the presence of controls or schedulers. It generates across any (organizational) boundary as it scales with a corresponding reality (world of interest).

2.4. MAS in manufacturing execution systems

Holonic systems and multi-agent systems (MAS) frequently are seen to be similar concepts by outsiders. Similarly, holonic system research often considers multi-agent systems as a manner to implement holonic manufacturing execution systems. Indeed, it sees a MAS as a collection of communicating computing processes, which fits holonic system designs quite naturally. In the industrial automation community, this relaxed view on the definition of MAS and intelligent agents continues to be used (cf. https://rd.springer.com/conference/holomas).

In contrast, after getting acquainted with the multi-agent community and their understanding of a MAS, this relaxed view on multi-agent systems no longer holds. Intelligent agents are goal-seeking computing processes (e.g. in Váncza and Márkus, 2000). E.g. the widely-known BDI architecture explicitly has objectives as a top-level element inside each agent. Here, all the ingredients for a never-ending discussion are present, particularly when involving the numerous research communities that are involved.

Fortunately, looking back at the research, it suffices to be aware of these contrasting views. A key achievement of holonic systems research is the absence of goal-seeking or, alternatively, postponing the goal-seeking to last when designing and implementing a manufacturing execution system. Indeed, a holonic MES delivers (predictive) situation awareness without imposing its own planning or controls. This minimizes the inertia of the decision making elements in the system. Note that this also sets holonic systems apart from operations research and most manufacturing control research. The discussion below further addresses this distinction.

2.4.1. Intelligent agents in manufacturing execution

Research on MAS in industry has a broader scope than holonic systems. Moreover, most research results focus on the agent technology and utilize manufacturing execution as a case study. As a consequence, it proved exceedingly difficult to identify a long-running research activity addressing MAS manufacturing execution systems (in contrast to PROSA, ADACOR, ORCA. . .). Long-running MAS research focuses on the agent technology, not on manufacturing execution. Nonetheless, it is possible to highlight some representative work to support the discussion.

- *P2000+ or Production 2000+.* Addressing manufacturing execution for an engine plant of Mercedes-Benz, the P2000+ consortium implemented and assessed a MAS design. This effort resulted in a doctoral thesis published in a book (Bussmann et al., 2004).

In a nutshell, the lessons learned by P2000+ called for identifying the decision points – e.g. routing of engine parts – within the execution system at the start of the design process. These decision points become the (responsibility of the) agents (Bussmann et al., 2001). This reflects the goal-seeking nature of the intelligent agents in a MAS.

Next, the research focuses on how to design and implement a MAS – taking the decisions – as lean and as fast as possible. This reflects the fact that the decision making is subject to frequent changes; P2000+ agents are no long-lived multi-purpose software entities.

As goals and goal-seeking are the sole top-level system elements, fetching information from the factory floor and sending instructions to the equipment is to be implemented without much support from the MAS technology. The main re-use concerns interaction protocols for the agents, not getting to know the manufacturing system itself (Bussmann et al., 2003).

Note that this also reflects an industrial state-of-practice in which PLC programs implement decision making where the reasoning behind the decision happens in the brains of the programmers. Neither the P2000+ agents or the PLC program will have an explicit representation of the underlying production system. As an analogy or metaphor, a MAS only knows routings (without the benefit of knowing maps) of the production system.

- *Market based control.* Market based economies have been successful in human society. For that reason, MAS research has investigated digital markets to coordinate, among others, manufacturing operations. The widely-known contract net protocol has been used frequently. Relevant for the discussion are the limitations encountered, especially the hard-to-remedy ones.

These can be summarized by Parunak stating (during the break at a conference) that “a price for a service” is a one-dimensional number, which implies a tremendous loss of information. This has implications. A lot of information processing and communication outside the market itself is necessary (to repeat the success of market based economies for humans). E.g. deciding what to offer on the market is not supported by the market (protocol) itself. It also means that a “price” is only valid for a narrow set of conditions, which mostly includes a narrow time window.

A one-dimensional price implies decision myopia. It is hard to coordinate multi-step activities, omnipresent in manufacturing. Naïve designs suffer from combinatorial/exponential explosions. To remedy this, the required tooling is considerable (Csáji et al., 2006), regrettably rendering it unworkable for most research teams and industries. Moreover, even such an advanced scheme shares its limitations with real-world economies (i.e. relying on past experience to construct bids for multi-step tasks requires significant stability in the world-of-interest).

Overall, MAS contribute in the decision making. Especially when agent technologies allow to design, implement and deploy suitable decision making rapidly with low effort, MAS contributes by lowering the inertia of the decision making. This is valuable when a system needs adapting in case of integration or interoperability conflicts as well as modifications of an underlying production system.

On the other hand, the decision making needs information (input) and actuation (output). The starting points for the provision of this input and output remain out-of-sight. This starting point has an important impact. It influences the development and deployment time/efforts for the decision making. More importantly, it influences what the decisions can be about and it influences how robust the decision making can become. As an analogy, a decision maker with a map may re-route around congestions and blockages whereas a decision maker with a only given route description may suffer and fail.

2.4.2. Intelligent beings

The more restrictive views on MAS research call for improving the communication among researchers. It made sense to give the distinction its own name. Hence, the *intelligent being* as opposed to the *intelligent agent* was introduced (Valckenaers et al., 2009). It emphasizes that this software process corresponds to what exists (to be) rather than a software process that takes decisions (to act).

An old technology implementing intelligent beings are maps. They simply mirror relevant facts about a world-of-interest. An organization delivering coastal maps to ship captains, updating these maps twice a year, is an example of an intelligent being that is no longer static (as maps are on their own).

Holonic systems along the PROSA reference architecture now start with the intelligent beings for the product, order and resource holons and add intelligent decision makers (agents or otherwise) last. Overall, intelligent beings create a software platform on which decision makers are the applications.

Compared to decision making agents, intelligent beings are long-lived and compose-able. As revealed by M. Jackson, during a series of classes for an IBM chair in Leuven, a problem domain is significantly more stable over time than the functionality requirements for a software development. Jackson’s example was personnel management. Today, tomorrow and in the foreseeable future, organizations will recruit people, have them go on vacation, promote them, . . . while personnel will retire or leave themselves, etc. In contrast, the information demanded by the personnel managers, government administrations, etc. may change every week.

Fig. 1. In conflicts, in-depth interoperability enjoys the protection of reality. Ignoring is at your own risk.
By nolativ (flickr) [CC BY 2.0 (https://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons.

Moreover, ***intelligent beings exhibit in-depth interoperability*** where in-depth goes beyond syntactic (data formats, interaction protocols) and semantic (understanding of the data) interoperability. Intelligent beings mirror a real-world counterpart, even when this relevant reality is undesirable. This reality is coherent and consistent. Intelligent beings inherit this (cf. Fig. 1). When two maps overlap, a conflict implies that at least one map is wrong where the corresponding reality decides what needs correcting. When a (decision making) application conflicts with an intelligent being (e.g. needs a bridge to be higher), changing the intelligent being will not solve the conflict (with reality).

Furthermore, the disruptions caused by upgrading, enhancing or expanding an intelligent being are easy to handle. E.g. upgrading a tourist map into a full-fledged traffic map provides new and better services while the existing users of the tourist maps continue to receive the services they need. These existing users only need changing if they want to benefit from the improved map services.

Overall, the contrast between MAS and Holonic systems allowed to become aware of a key contribution from holonic systems. It reveals how to design a software platform that scales with a corresponding reality and, as an operating system for its world of interest, provides a basis on top of which decision making applications can be developed with less effort, faster, with better and more robust services. Importantly, those decision mechanisms will have a lower inertia (as a lot is done for them by intelligent beings) and handle a wider scope of conditions (e.g. because they have maps in addition to route descriptions).

3. Moving beyond holonic manufacturing execution systems

In the above, the contributions from holonic systems are not necessarily manufacturing-specific. This is especially true for the heterarchical designs and also for the bio-inspired enhancements that preserve their advantages over hierarchical designs. Hence, this section covers holonic systems in non-manufacturing application domains and the insights that emerged from these developments.

3.1. ARTI & Digital Twins

Holonic systems research produced results that address a wider scope than manufacturing execution. Here, the hybrid architectures

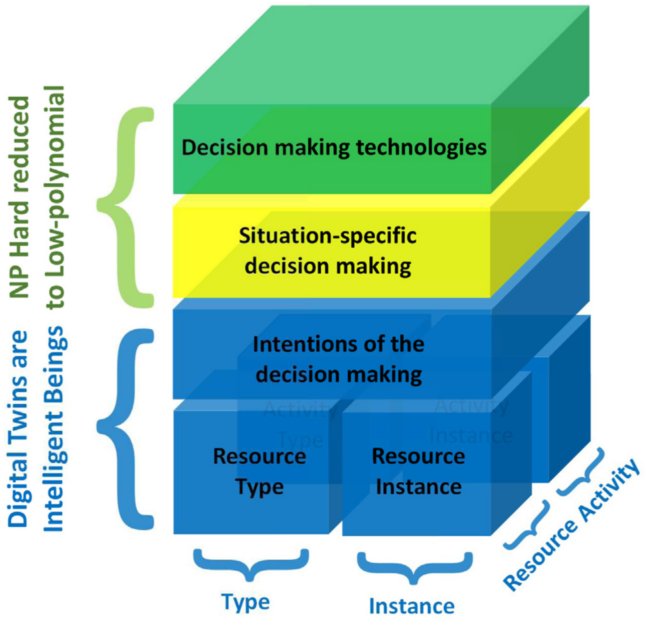

Digital Twins are Intelligent Beings
NP Hard reduced to Low-polynomial
Decision making technologies
Situation-specific decision making
Intentions of the decision making
Activity Type
Activity Instance
Resource Type
Resource Instance
Type
Instance
Resource Activity

Fig. 2. ARTI cube.

impose the structure that is found in manufacturing and closely related domains such as logistic execution systems. Unfortunately, this limits their applicability while they deliver more complete solutions in their narrower scope. It are the heterarchical architectures that enjoy a wider applicability while they deliver incomplete functionality.

The intelligent beings in an heterarchical architecture cover the domain of "***activities executing on resources***" where the cost of intelligent coordination is more than recuperated by improved resource utilization and service levels. Accordingly, application domains beyond manufacturing have been investigated in varying depths:

- Logistic execution systems (Van Belle, 2013).
- Networked manufacturing (Saint Germain et al., 2007; Saint Germain et al., 2011; Saint Germain, 2010).
- Robotic fleets (Ali et al., 2013; Ali, 2010; Philips, 2012).
- Intelligent transport (modum-project.eu).
- Smart energy (horizon2020-story.eu)
- Healthcare – integrated care
- Healthcare – multi-disease and polypharmacy
- Human avatar or e-Butler

This broadening has strengthened the scientific basis for holonic execution systems. E.g. it has integrated traffic models to account for the back-propagation of congestion in car traffic (Philips et al., 2013). It has looked into trust and reputation in networked manufacturing where multiple factory owners have selfish goals (Saint Germain et al., 2012). Most importantly, it improves the communication by refining and improving on the PROSA reference architecture: ***ARTI (Activity-Resource-Type-Instance)*** as shown in Fig. 2.

The refined reference architecture avoids manufacturing-specific wordings and has more symmetry. Relative to PROSA, ARTI renames the *order holons* and *product holons* into *activity instances* and *activity types* respectively. The PROSA resource holons are split into *resource instances* and *resource types*, rendering the architecture symmetric (i.e. it introduces orthogonality among the concepts of resource and activity versus instance and type). In addition, the ARTI reference architecture distinguishes – at top level – the intel-

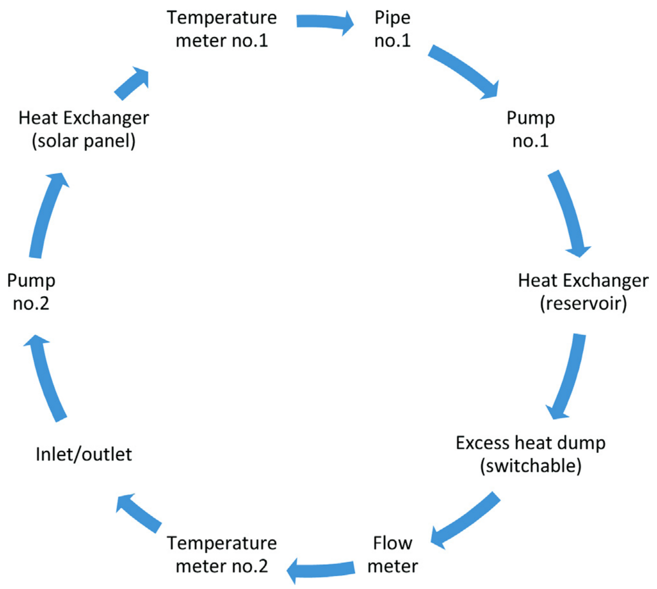


Fig. 3. Simple Thermal Circuit.

ligent beings from the decision making components in a given system. It preserves the aggregation and specialization from PROSA.

Furthermore in ARTI, ***the intelligent beings are Digital Twins*** of a corresponding element in the world-of-interest. The underlying theory (Ch.3, Valckenaers and Van Brussel, 2015) draws a more permissive boundary around the intelligent beings but, in the prototypes, ARTI digital twins constitute almost all opportunities to identify and develop intelligent beings.

A key advantage of the wording digital twin is communicating that a *digital twin is mirroring a corresponding part in reality*. *Twins are 'intuitively understood to be the same'* and the wording thus stops a reader from adding something extra. Indeed, dissemination of the research into society requires smooth communications and a correct transfer of knowledge. The confusing situation with agents, planning, scheduling, control, etc. next to execution systems and intelligent beings justifies this quest to find such effective wordings.

3.2. Digital Twins of decision making elements

Thanks to exploring application domains beyond manufacturing, the boundary between the decision making elements (yellow and green in Fig. 2) and the digital twins (blue in Fig. 2) has been drawn more precisely and, actually, has been corrected. Blinded by its achievement (i.e. generating short-term forecasts while avoiding the rigidities of hierarchical control), research considered the delegate MAS to be blue within ARTI, erroneously.

Moving into other application domains, the necessity to develop case-specific enhancements (of the basic D-MAS design) indicated that a delegate MAS might be green rather than blue. Note that green elements are decision-making tools (e.g. linear programming) that require yellow elements to connect them to blue elements (e.g. linearize the problem). The application of ARTI to smart energy delivered the decisive information to classify the delegate MAS to be green within ARTI.

Because of the commodity nature of energy, a delegate MAS is ill-suited to contribute. Indeed, not all energy is equal but e.g. a petrol engine does not care from which oil well or refinery its fuel comes. Likewise, a heat pump does not care where its electricity originates from. This rendered intention-propagation or exploration by activity instances ineffective.

The correct boundary is to provide ***digital twins of the decision making*** within the blue part of ARTI (cf. Fig. 3). These twins (of a non-physical reality) allow the twins of activity instances to virtually execute (faster than real-time) the intentions of the decisions makers (e.g. control settings, planning...) on the twins of the resource instances. These twins are the ***intentions*** of the decision-making intelligent agents.

These intentions can be control settings, schedules and plans. However, they need to be sufficiently detailed (also called grounded) and complete to allow for virtual execution within the blue part of the ARTI system implementation. In particular, e.g. when multiple independent decision making components are present, the digital twins of the decision making must include conflict handling. When planners, scheduler or controllers disagree (e.g. have overbookings), *intention completeness* requires an executable model of what will happen.

This corrected view (on the position of a delegate MAS) improves the interactions among research communities as it respects the autonomy of research and researchers looking into performance optimization and/or controllability. Conversely, it places a spotlight on *the requirement to ground their decision making contributions* (i.e. the link to operational reality).

Furthermore, during research prototyping, these digital twins of the decision making revealed to belong to the following categories. When the decision making is automated and computationally undemanding (e.g. a dispatching rule), it can be its own twin. With computationally demanding mechanisms, its outcome (control settings, trajectories, schedules...) can be used but augmented with computationally simple mechanism to handle conflicts (with each other and/or with reality), incompleteness, etc. With human decision makers, some machine learning may indicated where it is advisable to exploit the narrow context in which the decisions are taken. Fortunately, these twins only serve to generate forecasts and are not responsible for the actual decision making. Thus, accuracy requirements can be relaxed in function of the situation and conditions that prevail.

Appendix A presents and discusses a number of system configurations that are supported when digital twins of the decision making are available.

3.3. Digital twins for ARTI & autocatalytic sets (aka critical user mass)

Nobel Prize winner Herbert Simon reveals how monolithic designs cannot exist beyond a certain complexity in a demanding and dynamic environment. Any non-trivial system will exhibit a flexible aggregation hierarchy, which will vary over time to adapt swiftly to a dynamic environment. Research on holonic systems has expanded on this, discovering that participation in Simon's aggregates requires membership of suitable autocatalytic sets (except for the simplest elements at the bottom of the aggregation hierarchy). In practice, such autocatalytic set comprises the artefact (e.g. a digital twin of a pump), its developers and supporters, and its users. Membership equals enjoying a user mass that is large enough for the complexity of the artefact (i.e. critical user mass).

Digital twins along the ARTI architecture maximize the potential for this user mass. First, as an intelligent being, twins do not add limitations of their own. This preserves the user mass of the real-world counterpart; the twin of a pump is able to serve wherever such a pump is installed or even considered to be installed (e.g. in a virtual mockup of a chemical plant).

Second, ARTI generalizes/broadens the common understanding of a digital twin significantly. Real-world counterparts of ARTI twins are not only physical assets. They can be resources (assets), but also activities and even intentions (mental states?). Moreover, ARTI distinguishes types from instances. All ARTI varieties of twins are so-called first-class citizens. E.g. a resource instance will not be confined to some data owned by its resource type; this avoids data model lock-in (a major cause of legacy issues).

Thus, ARTI provides significantly more starting points (real-world counterparts) for digital twins, which eases avoiding 'adding non-existing stuff' to a real-world counterpart, where 'stuff' includes limitations and simplifications. This is needed to preserve user mass potential as mentioned above. It also renders these digital twins simpler and smaller, which lowers the threshold for achieving critical user mass.

Furthermore, ARTI accounts for how user masses in the industrial world are structured. E.g. instance twins are largely agnostic about 'technology' (e.g. the temperature profile in a furnace) but fully knowledgeable about 'management' (e.g. the first-in first-out property of the tunnel furnace). Conversely, type twins are experts on technology (e.g. ensuring parts experience a suitable temperature profile in a furnace) but remain agnostic about how production is managed. Imagine what happens without such separation of concerns: the user mass of digital twins rapidly shrinks to the users of a specific factory layout and configuration.

Likewise, the twins of aggregates delegate to the twins of their constituents. This reduces their complexity and responsibilities, which lowers their threshold for achieving critical user mass. At the same time, it increases their potential user mass. When the aggregate twin is able to remain agnostic about its constituents, it serves its users regardless of the specifics of those constituents.

Unfortunately, concerning the concept of a digital twin, ARTI is facing a communication issue with some research communities and industrial societies. The wording digital twin intuitively communicates that it are intelligent beings, not adding non-existing stuff, but a large industrial community already has hijacked the wording in a more restrictive interpretation: twins of physical assets. Digital twins for processes are mentioned but lack visibility in a discussion that places twins for physical assets in the spotlight and has strong links with the internet of things (forgetting the activities on those things). An explicit distinction between activity instances and activity types remains unseen. On the positive side, there are researchers envisaging already an internet of activities executing on the internet of things as the next step in that domain.

Here, the challenge remains significant but the reward from adopting ARTI balances this. Having more starting points allows to develop clean digital twins first and add the constraint-adding decision makers last. It allows to have a clean twin platform contribute more with less effort. And, having twins for all the ARTI elements increases the (potential for) user mass of a digital twin. It gives the twin a narrower focus which fits a larger number of users. It makes the twin simpler and smaller, which lowers the required number of users. Aggregation and specialization, inherited from PROSA, also contribute.

3.4. Performers (triggering and sustaining autocatalytic cycles)

ARTI enlarges and maximizes the upper bound of the potential user mass, and it lowers the threshold for critical user mass. However, the self-reinforcing cycle – characteristic for autocatalytic sets – requires more than ARTI digital twins. Users call for aggregates that provide actual benefits and services. And, when aggregating digital twins into a working system or installation, it generally is unavoidable to include decision making elements, which will be human, digital or both (Valckenaers and Van Brussel, 2015, ch3). Such an aggregation of digital twins and decision making is called a ***performer***. Users are willing to spend their money, time and public support to enjoy its services. This "***activates the autocatalysis***" whereas the digital twins only create the favorable conditions for this to happen.

In other words, the ARTI digital twins provide the potential to have (sufficiently) large autocatalytic sets. The performer allows for the "autocatalytic processes" in these sets to execute. The performer provides value to the users, which has the users to provide the means for the performer to flourish. Thus, the performer receives the economic means to support itself by recruiting and rewarding developers, trainers, installers, operators, distributors, sales persons, etc. It also receives information on how to improve and adapt in order to remain competitive. Note that the autocatalytic sets comprise the artifact itself and its human users as well as human developers, sellers, trainers. . .

The challenge is to prevent the decision making in performers from accumulating too much inertia. This inertia would prevent us from repurposing the digital twins inside the performer when its decision making causes problems. The negative connotation of legacy software and systems hints at the gravity of this issue. Likewise, the enormous interoperability efforts – with underwhelming results when they remain stuck at syntactic interoperability – provide evidence that this inertia is harmful.

The generic answer is ***explicit management of the rights concerning the resources and activities embedded in the performer.*** A 'super user' will be able to de-allocate resources, denying the performer some control over some embedded resources and re-allocate it to another (external) performer. Likewise, a performer explicitly knows the resource capabilities and capacity it uses/needs to provide its services, and this performer may make the remainder available to outsiders as part of its explicit management of rights. Similarly, the embedded activities may be applied to external resources, may cooperate, etc. Notice that such explicit management is also a service offered by state-of-the-art computer operating systems in their domain.

In addition, ***publish-subscribe services*** provide access to the internal data/information. When relevant, performers support time stamping. The precision of these time stamps can ensured by means of the network timing protocol, the precision timing protocol, embedded GPS receivers, etc. The choice of precision depends on what is possible (e.g. wired network connection) and the nature of the real-world counterparts of the embedded twins. E.g. precision timing for a robot allows a subscriber to match manipulator positions against a video recording and measure with the *encoder resolution*. Conversely, a thermal system in which little changes in less than 1 minute, the system clocks can be used with minimal measures to keep them synchronized across the performers involved.

Such an explicit handling and managing of embedded resources and/or activities involves design and development, which accounts for the specifics of the performer. De-allocation or re-allocation of a robot arm while it is moving deserves/requires a research project. Sharing control over the robot arm between the own performer activity (pick and place) and an external diagnostics performer (wanting some wobble to be able to measure) is another challenge. De-allocating and re-allocating resources in a manufacturing plant without shutting it down also requires some thought. Here, future research will find a nice agenda of challenges to address.

The explicit management of rights and publish-subscribe are part of the performer itself. In addition, performers have to minimize, even avoid altogether, depending on the specifics of other performers. Indeed, when a super user removes rights, a performer may no longer provide some of its services and will *break* the users of those services. Hence, performers should be small clusters that interact through digital twins with the remainder of the world. Likewise, dependencies among performers may be weak (e.g. as in the PROSA staff holon) resulting in a more graceful degradation. Here, the design of hybrid reference architectures may find a way to deliver services without getting tied to rather specific manufacturing plant configurations.

3.5. Uncertainty and lazy non-determinism (bootstrapping autocatalytic cycles)

The elements that can become parts of an aggregate, as intended by Simon, cannot exist simply because a not-yet-existing aggregate may use/need it. A non-trivial element needs an autonomous existence based on the value it already provides to its users. Therefore, it must provide utility to its users under the conditions that prevail before the aggregate exists.

Alternatively, an expectation of utility may suffice, but only if the development effort is sufficiently low. Typically, this development aggregates itself elements that already exist where this development suffices to trigger a self-reinforcing autocatalytic cycle. This bootstrapping cycle, subsequently, creates the conditions for enhancement of the aggregate and the elements therein.

Note, performers rarely emerge from the design studio and deliver what is required without teething problems. Version 1.0 from a software product is notorious for being buggy and offering poor services. Indeed, non-trivial products and services need to mature by facing reality in full strength. This is yet another "scientific law of the artificial" when bounded rationality implies that a laboratory setting cannot deliver the information processing needed. Real-world deployment is required.

Above, two contributing factors to real-world exposure have been discussed. ARTI provides the potential for (more than adequately) large autocatalytic sets. Performers allow the autocatalytic process to happen. However, when discussing the development of an ARTI implementation, prospective stakeholders perceive an unsurmountable obstacle, erroneously. In particular in healthcare, prospective professional users argue that the digital twins require non-existing models, which would require unrealistic efforts to obtain and validate. Luckily, an ARTI implementation may start with whatever is available (e.g. tourist maps) and is able/designed to upgrade to better models (e.g. road maps) later.

Here, explicit support for uncertainty and lazy non-determinism allows for the autocatalytic process to start early and evolve steadily into a battle-proven full-fledged self-reinforcing result. The adopted approach is to represent explicitly how much is (not) known and gradually improve the models when the more knowhow becomes available. ARTI and, especially, the NEU protocol (Valckenaers and De Mazière, 2015) allow to upgrade twins without creating an avalanche of software maintenance issues.

Non-determinism allows twins to avoid decisions. E.g. when a process plan, in an activity type, is able to represent multiple manners to produce a given part correctly, it leaves the selection of a specific manner to the activity instances, who decide based on the available capacity with suitable resource instances (Detand, 1993). Lazy non-determinism allows to expand this set of possible manners to produce a part at later instances, e.g. when it makes sense to unload a scarce resource. The NEU protocol ensures that activity instances will detect any new options (or option removal) at the earliest opportunity.

Uncertainty representation allows to use the models and data that happen to be available. The first version may simply state that nothing much is known. E.g. the effect of taking medication (an activity instance) according to a medication scheme (an activity type) on a kidney (a resource instance) may be marked as unknown in a very first version. Gradually, the kidney expert twin (a resource type) may use the kidney state representation from a kidney instance twin (which may include a medical history) and consult with the medication scheme (using the planned dosage/timing from the intentions of the activity instance) to perform a triage: 'ok' or 'not ok' or 'unknown'. Note that the triage is likely to involve a decision maker (probably human or supervised by humans).

4. Concluding remarks

Looking back at 30 years of research into holonic manufacturing systems, the specific nature and contribution of holonic systems to manufacturing can be recognized, regardless of the confounded state in which research communities continue to disagree (or ignore each other) as regards terminology and wordings as well as evaluation or performance criteria. Indeed, the communication among the human stakeholders remains challenging, especially when it requires crossing a boundary between comfort zones of research communities. Here, Computers in Industry provides a precious and unique forum.

Specific for holonic systems is their rooting into complex-adaptive systems theory (e.g. self-reinforcing cycles by autocatalytic sets), the sciences of the artificial (i.e. scientific laws for human-created artifacts derived from bounded rationality) and design for the unexpected (e.g. digital twins protected by the consistency of a corresponding reality). Originating from this roots, holonic manufacturing separates a (predictive) situation awareness from the decision-making.

Whereas mainstream manufacturing control – e.g. operations research, A.I. planning, control optimization – addresses the formidable challenges of converting NP-hard problems into low-polynomial solutions, holonic systems bring computer-based intelligence to a world-of-interest mainly through executable models. These models cover the assets (resources), activities on those assets, and in advanced versions, models of the decision-making (intentions). It provides a body for computer-based intelligence (e.g. digital twins and their real-world counterpart) as well as situated intelligence (e.g. instance twins know the twins of their surroundings) and evolutionary improvement (i.e. by the autocatalytic sets).

Finally, the decision making elements are aggregated into larger system while measures to minimize inertia are provided. Indeed, converting NP-hard into low-polynomial unavoidably introduces limitations that are situation-specific, valid in narrow time windows and even arbitrary. When creating, flexible aggregation hierarchies of holons (also called holarchies), these limitations need to be easily undone. Generic services and designs for performers – i.e. aggregations of reality-mirroring digital twins and decision-making elements – facilitate the containment of such inertia commonly associated with legacy in its negative connotation.

Overall, this manuscript aims to share this insight, a top-level view, on intelligent manufacturing. It contributes by depicting a world in which lock-in into legacy systems and mono-disciplinary comfort zones need to be exposed to understand how the future challenges may be addressed. Hopefully, it initiates the understanding and development efforts to create and deploy an *operating system for the real world*.

Also note how these developments, aimed at intelligent manufacturing, resulted in a much more widely applicable result: the coordination of activities on resources scaling with a corresponding reality. Here, this scaling goes beyond current practices and research in decision making technologies, precisely because it refrains from decision making.

Declaration of Competing Interest

None.

Acknowledgements

Essential contributions to the insights, discussed in this paper, originate from informal interactions.

Parunak notes – benefiting from e.g. (Parunak et al., 1999) – that prices in markets are one-dimensional numbers and adds that this implies a very significant loss of information. If market designs are to be effective, a lot of additional information processing and communication is needed outside the market per se. Without proper support, market mechanisms are myopic and struggle with determining what the products/service for sale will be. Successful applications of market mechanisms in automated control typically target underlying systems that return to a reference state after every processing step, which renders multi-step optimization pointless.

Olivier Cardin participated in an innovative paper review process in which the reviews are co-published with the publication under review (Borangiu et al., 2019). During this non-anonymous interaction, Olivier pointed out that the delegate MAS disrespects the autonomy of researchers investigating decision making and performance optimization. This resulted, in combination with applying ARTI to smart energy, in a correct positioning of D-MAS as a decision making technology (green) rather than part of some digital twin (blue). He also suggested modifying the original ARTI cube (Valckenaers et al., 2019), which has been further enhanced in this paper.

Michael Jackson presented an eye-opener in Leuven in the 1980's: problem domains are stable over time whereas user requirements are not stable at all.

The BEAM community (cf. erlef.org, www.codesync.global, www.erlang.org and elixir-lang.org) has been creating and sustaining the software technology that renders the scientific results in this paper workable in real-world applications. The BEAM is bringing real-world implementations in reach of industrial development. As imitation is the best compliment, mainstream technologies have been embracing it through software tools (e.g. akka.io, getakka.net and actor-framework.org for Java, . NET and C++ respectively).

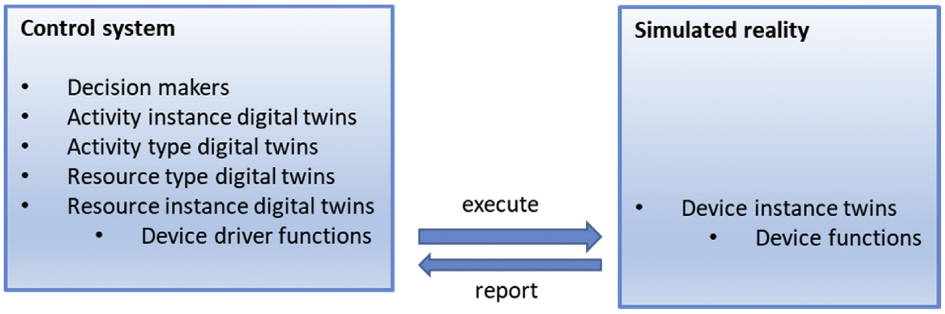
Appendix A. ARTI system configurations

The discussion uses examples based on the system in Fig. 3. It is a simple thermal circuit for which digital twins have been elaborated along ARTI.

Configuration 1 – Simulation with software in the loop

The digital twins in the control/planning/scheduling/. . . are re-used *as they are* to simulate the reality of interest. The actual control system, a performer comprising digital twins and decision making, is used in this simulation configuration *as it will be deployed*. Only one version of the software is used and needs to be maintained.

The only affected software, which differs from the deployed version, are the *device driver functions*. Note that these also differ when the connection to the physical device differs (i.e. opening and closing the relay for the power supply to a pump may occur via Modbus, Ethernet, a PLC. . .). In other words, this minor variable part is not only affected by simulation versus real-world deployment. The various manners in which devices are connected in real-world deployment creates (at least) similar variability.


Control system
• Decision makers
• Activity instance digital twins
• Activity type digital twins
• Resource type digital twins
• Resource instance digital twins
• Device driver functions
execute
report
Simulated reality
• Device instance twins
• Device functions

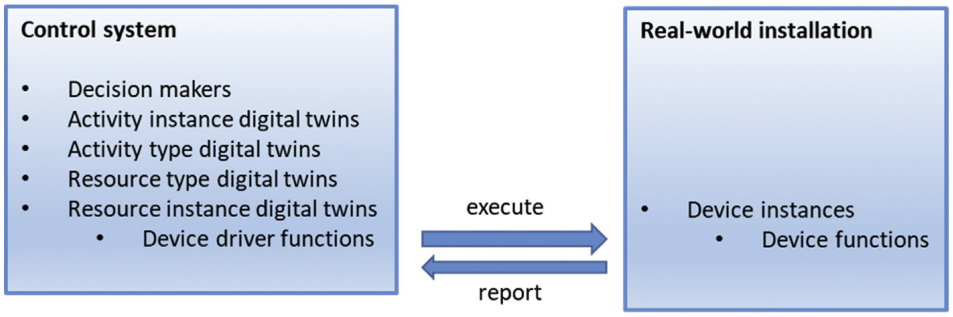
Configuration 1

This configuration is used to design installations, both greenfield and brownfield, to perform the early phases of ramp-up and tuning, to experiment under extreme conditions, to compare design alternatives, compare control alternatives, etc. In addition, it may serve to prepare the installation deployment activities and, among others, identify issues on beforehand (e.g. when a pump is too close to a flow meter and turbulence would prevent accurate measurements). In advanced settings, it becomes part of the co-design of products and production systems.

Configuration 2 – Simple deployment

In a straightforward deployment, the digital twins provide the control with situation awareness. This allows for, among others, health monitoring when the (executable) twin models and measurements disagree. E.g. this may happen if a pump is damaged or a measuring device is faulty.

Likewise, this situation awareness facilitates diagnostics, allows replacing a faulty sensor by a model-based estimate (a virtual sensor), etc. Moreover, the design supports separation of concerns. E.g. a computer process may monitor the temperature and take measures to prevent overheating when indicated. Likewise, another process prevents the backup pump from idling too long (several weeks) to ensure it will work when needed to prevent overheating and the main pump fails. Here, mistakes only need to occur once for the affected digital twins to learn the corresponding lesson everywhere and forever.

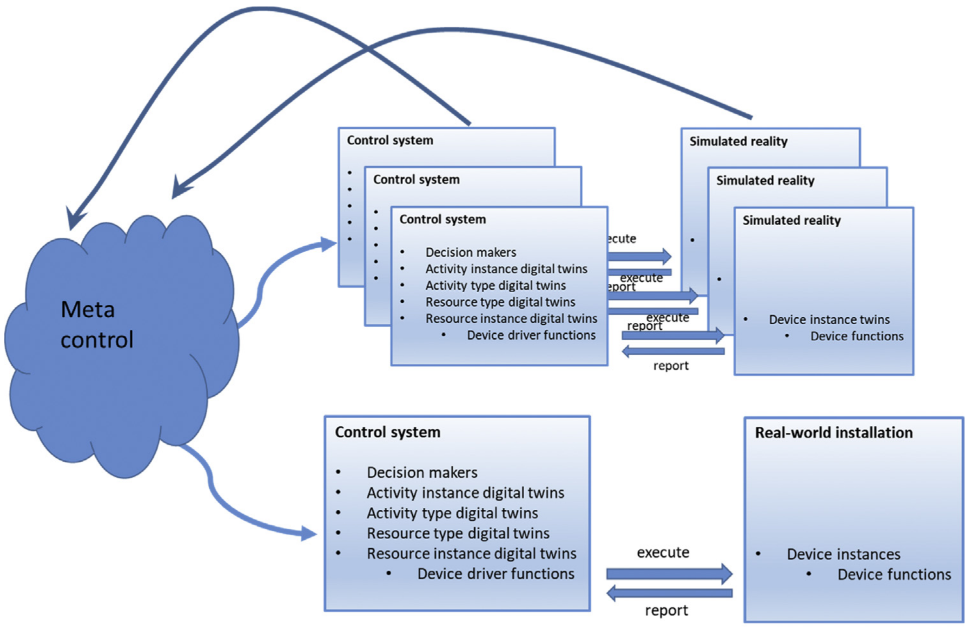


Configuration 2

If preceded by configuration 1 (simulation), the deployment activity itself can be supported. It will indicate to installers/workers what needs to be done, and with suitable IT support (e.g. smart glasses and QR codes on the pipes, pumps, etc.), the design allows for ensuring a correct correspondence between the digital twins and their real-world counterparts. It is possible to ensure the simulation configuration is actually build, but also it becomes possible to update the simulation if a deviating configuration gets installed. In the latter case, configuration 1 may check and validate such field modification. Note that it is not uncommon to have such changes when e.g. a supply of equipment delayed or replaced by an alternative model. Note that only device drivers are affected by the transition from simulation to deployed as far as the control system software is concerned.

Configuration 3 – Embedded simulation faster than real-time

When configuration 1 is able to execute (much) faster than real-time, a meta-level control may combine it with configuration 2. At a given time instant, the actual control has certain intentions (control settings, schedules, plans) in a configuration 2 (actually deployed). Then, the meta-level initiates a configuration 1 (faster than real-time simulation) and subscribes to the appropriate publishing services of twins and performers within this configuration 1, which brings *predictive situation awareness*.



Configuration 3

This situation awareness allows the meta-level to observe problems (long) before they occur as well as missed opportunities. Then, the meta-level may modify its intentions to prevent these problems from happening and to grasp worthwhile opportunities. As its intentions have changed, the digital twin of those intentions needs to be informed. Ordinarily, the embedded configuration 1 is terminated and restarted with the latest settings (intentions).

Moreover, the meta-level may initiate a finite number of embedded simulations. Each configuration 1 may have different intentions (i.e. they differ from the time foreseen for the next update of the intentions). Predictive situation awareness for each of the alternatives, explored in their own simulation run, allows the meta-level to make an informed decision at the next update of settings/intentions.

The most famous among these alternatives ideas is “the big boss enters with an important visitor and wants to show off with a *brilliant* idea”. Then, a battle-proven shop floor manager may want to compare this brilliant idea against the established way of working in his section of the factory in simulation first. Note that the terminating and restarting is still needed when intentions change.

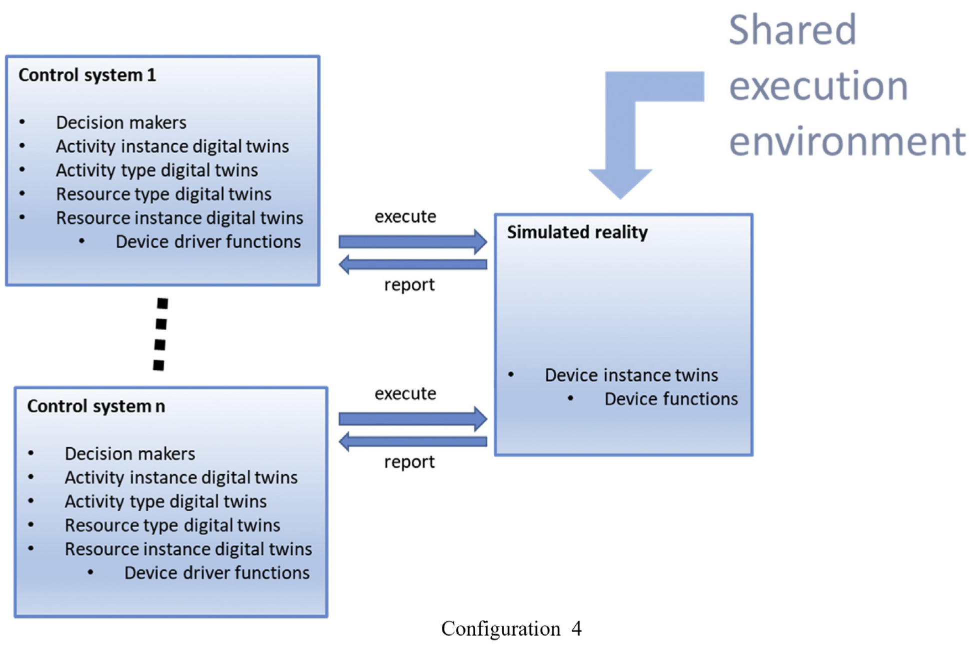
Also, a number of configurations 1 (simulations) running in parallel may utilize the same intentions but differing simulation settings to assess what will happen in a varying number of conditions (e.g. weather, process yield or processing duration).

When using an intention propagating delegate MAS, the embedded simulation does not need to restart. Indeed, the digital twins of activities and resources are updated in a fine-grained manner whenever an activity instance creates an ant agent to propagate its latest intention.

Finally, the real-world installation in this configuration can be replaced by its simulation. This results in a configuration 1 in which the control comprises the meta-control and the embedded simulations.

Configuration 4 – Simulation with multiple decision makers and shared decision execution

In practice, many systems are too large and complex for a single decision making system to handle. Multiple control systems are present. They all affect a shared reality in which the actual activities execute on the resources. The execution reality is coherent and consistent. Thus, configuration 1 can be constructed with multiple controls, as they will be deployed, and a single simulation provided by digital twins. The simulation result will reveal how the controls coexist, fight, cooperate, etc.


Shared execution environment
Control system 1
Decision makers
Activity instance digital twins
Activity type digital twins
Resource type digital twins
Resource instance digital twins
Device driver functions
execute
report
Simulated reality
Device instance twins
Device functions
Control system n
Decision makers
Activity instance digital twins
Activity type digital twins
Resource type digital twins
Resource instance digital twins
Device driver functions

Configuration 4

Configuration 5 – Embedded simulation with multiple decision makers

Although perfectly possible, a deployed version of configuration 4 brings only minor benefits over configuration 2 (i.e. situation awareness covering the past and present). The combination of configurations 3 and 4, however, changes the situation radically (no picture as it would be too complicated and not contributing to a better understanding).

Combining configurations 3 and 4 allows all the controllers to observe their collective impact on the execution environment on beforehand. E.g. automobile drivers see congestion before it actually occurs and they may subscribe to services that allow to coordinate and agree on intentions preventing this congestion from happening (i.e. move the congestion into offices, homes and bistros in which time can be spend more agreeably or usefully).

This configuration 5 allows multiple decision makers to interoperate and cooperate without unworkable modification or adaptation efforts. Moderate nudging, or even simply observing the impact of the others, may suffice to bring coordination and control to another level.

Obviously, much of this still needs to be investigated. E.g. how will the automobile users ensure that niceness (i.e. leaving road capacity to others when demand is high) gets rewarded (i.e. get some priority rights in return) and is not punished (e.g. have no convenient parking space when arriving later).

Importantly, even this most advanced configuration does not require to develop or maintain multiple software versions of digital twin or performers. Ultimately and ideally, real-world assets, resources and activities only need one digital twin (in the world). In practice, a small number of twin implementations are likely to exist, which will have no issue to co-exist precisely because they are intelligent beings.

Appendix B. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.compind.2020.103226.

References

Ali, O., PhD thesis 2010. Operational Planning for Outdoor Engineering Processes. KU Leuven.

Ali, O., Valckenaers, P., Van Belle, J., Saint Germain, B., Verstraete, P., Van Oudheusden, D., 2013. Towards online planning for open-air engineering processes. Comput. Ind. 64 (3), 242–251.

Barbosa, J., Leitão, Paulo, Adam, Emmanuel, Trentesaux, Damien, 2015. Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution. Comput. Ind. 66 (January), 99–111, http://dx.doi.org/10.1016/j.compind.2014.10.011.

Borangiu, T., Cardin, Olivier, Babiceanu, Radu F., Giret, Adriana, Kruger, Karel, Răileanu, Silviu, Weichhart, Georg, et al., 2019. In: Borangiu, T. (Ed.), Scientific Discussion: Open Reviews of "ARTI Reference Architecture – PROSA Revisited. Springer, pp. 20–37, http://dx.doi.org/10.1007/978-3-030-03003-2_2, SOHOMA 2018, SCI 803, 2019.

Bussmann, S., Jennings, N.R., Wooldridge, M., 2001]. On the identification of agents in the design of production control systems. In: Ciancarini, P., Wooldridge, M.J. (Eds.), Agent-Oriented Software Engineering, LNCS 1957. Springer-Verlag, Berlin, Germany, pp. 141–162.

Bussmann, S., Jennings, N.R., Wooldridge, M., 2003. Re-use of interaction protocols for agent-based control applications. In: Giunchiglia, F., Odell, J., Weiß, G. (Eds.), Agent-Oriented Software Engineering III, LNCS 2585. Springer-Verlag, Berlin, Germany, pp. 73–87.

Bussmann, S., Jennings, N.R., Wooldridge, M., 2004. Multiagent Systems for Manufacturing Control, ISBN: 3-540-20924-7.

Csáji, B., Monostori, L., Kádár, B., 2006]. Reinforcement learning in a distributed market-based production control system. Adv. Eng. Inform. 20 (3), 279–288.

Derigent, W., Cardin, Olivier, Trentesaux, Damien, 2020]. Industry 4.0: Contributions of Holonic Manufacturing Control Architectures and Future Challenges. Journal of Intelligent Manufacturing. Springer Verlag, Germany), http://dx.doi.org/10.1007/s10845-020-01532-x, In press, hal-02455705.

Detand, J., PhD thesis 1993]. A Computer Aided Process Planning System Generating Non-linear Process Plans. KU Leuven.

Duffie, N., Prahbu, V., 1994]. Real-time distributed scheduling of heterarchical manufacturing systems. J. Manuf. Syst. 13 (2), 94–107.

Hadeli, PhD thesis 2006. Bio-Inspired Multi-Agent Manufacturing Control Systems with Social Behaviour. KU Leuven.

Holvoet, T., Weyns, D., Valckenaers, P., 2012]. Delegate MAS patterns for large-scale distributed coordination and control applications. In: Avgeriou, P., Weiss, M. (Eds.), Proceedings of the 15th European Conference on Pattern Languages of Programs, (1–16). , ISBN: 978-1-4503-0259-3.

Leitão, P., PhD Thesis 2004. An Agile and Adaptive Holonic Architecture for Manufacturing Control. University of Porto.

Pach, C., Berger, Thierry, Bonte, Thérèse, Trentesaux, Damien, 2014]. ORCA-FMS: a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. Comput. Ind. 65 (May 4), 706–720, http://dx.doi.org/10.1016/j.compind.2014.02.005.

Pach, C., PhD thesis 2013]. ORCA: Architecture Hybride Pour Le Contrôle De La Myopie Dans Le Cadre Du Pilotage Des Systèmes Flexibles De Production. University of Valenciennes.

Parunak, H., Ward, A., Sauter, J., 1999. The MarCon Algorithm:A Systematic Market Approach to Distributed Constraint Problems, Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM), vol. 13., pp. 217–234, 1999.

Philips, J., PhD thesis 2012. Holonic Task Execution Control of Multi-Mobile-Robot Systems. KU Leuven.

Philips, J., Saint Germain, B., Van Belle, J., Valckenaers, P., 2013]. Traffic radar: a holonic traffic coordination system using PROSA++ and D-MAS. Springer, Berlin Heidelberg, 2013 In: Industrial Applications of Holonic and Multi-Agent Systems Book - 6th International Conference HoloMAS 2013, Prague, Czech Republic, Proceedings Series: Lecture Notes in Computer Science, 8062., http://dx.doi.org/10.1007/978-3-642-40090-2_15, Print ISBN 978-3-642-40089-6.

Saint Germain, B., Valckenaers, P., Verstraete, P., Hadeli, P., Van Brussel, H., 2007]. A multi-agent supply network control framework. Control Eng. Pract. 15 (11), 1394–1402.

Saint Germain, B., PhD thesis 2010. Distributed Coordination and Control for Networked Production Systems. KU Leuven.

Saint Germain, B., Valckenaers, P., Van Brussel, H., Van Belle, J., 2011]. Networked manufacturing control: an industrial case. CIRP J. Manuf. Sci. Technol. 4 (3), 324–326.

Saint Germain, B., Valckenaers, P., Van Belle, J., Verstraete, P., Van Brussel, H., 2012]. Incorporating trust in networked production systems. J. Intell. Manuf. 23 (6), 2635–2646.

Simon, H.A., 1996]. The Sciences of the Artificial, 3rd ed. MIT Press, ISBN: 9780262537537, 256pp.

Valckenaers, P., Saint Germain, B., Verstraete, P., Van Belle, J., Van Brussel, H., Hadeli, 2009]. Intelligent products: agere versus essere. Comput. Ind. 60 (3), 217–228.

Valckenaers, P., Van Brussel, H., 2015]. Design for the Unexpected, from Holonic Manufacturing Systems towards a Humane Mechatronics Society. Butterworth-Heinemann, 18th November 2015, 234pp. ISBN: 9780128036624.

Valckenaers, P., De Mazière, P., et al., 2015. In: Mařík, V. (Ed.), Interacting Holons in Evolvable Execution Systems: The NEU Protocol. Springer International Publishing, Switzerland, pp. 120–129, http://dx.doi.org/10.1007/978-3-319-22867-9_11, HoloMAS 2015, LNAI 9266, 2015.

Valckenaers, P., et al., 2019. In: Borangiu, T. (Ed.), ARTI Reference Architecture – PROSA Revisited. Springer, pp. 1–19, http://dx.doi.org/10.1007/978-3-030-03003-2_1, SOHOMA 2018, SCI 803, 2019.

Van Belle, J., PhD thesis 2013. A Holonic Logistics Execution System for Cross-docking. KU Leuven.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P., 1998]. Reference architecture for holonic manufacturing systems: PROSA. Comput. Ind. 37 (3), 255–274.

Váncza, J., Márkus, A., 2000]. An agent model for incentive-based production scheduling. Comput. Ind. 43 (2), 173–187.

Zoitl, A., Strasser, Thomas, Hall, Ken, Staron, Ray, Sünder, Christoph, Favre-Bulle, Bernard, 2007]. The past, present, and future of IEC 61499. In: Proceedings of the 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems for Manufacturing (HoloMAS' 07), Springer-Verlag, Berlin, Heidelberg, pp. 1–14, http://dx.doi.org/10.1007/978-3-540-74481-8_1.