Journal of Machine Engineering, Vol. 17, No. 3, 2017

Received: *06 February 2017* / Accepted: *02 June 2017* / Published online: *28 September 2017*

holonic manufacturing systems,
integration, control, reference architecture,
virtual execution

Hendrik Van BRUSSEL[1*]
Paul VALCKENAERS[2]

DESIGN OF HOLONIC MANUFACTURING SYSTEMS

The introduction of CIM (Computer Integrated Manufacturing) systems in the 1980s, aiming at integrating automatic workstations into fully automated factories, was not successful. The root causes of this failure were that the subsystems to be integrated were not suitably designed for easy integration into a larger system. This situation stimulated the authors to embark on a research programme on 'design for the unexpected'. It defined how subsystems have to be designed so that integration into larger systems becomes easier and how such an integrated system can be controlled so that it can cope with change and disturbances. In the paper, the design principles and salient features of holonic manufacturing systems (HMS) are outlined. The PROSA reference architecture, defining the basic structure of any HMS, is described. It is further explained how coordination and control of the HMS is achieved by a holonic manufacturing execution system (HMES), based on the combination of the PROSA reference architecture and a biologically inspired Delegate Multiagent System (DMAS). Finally, the power and universality of the PROSA/DMAS system is demonstrated by some case studies from manufacturing, robotics and open air engineering.

1. INTRODUCTION

Somewhere in the 1980s, computer-integrated manufacturing (CIM) systems came into existence. These were *systems of systems* aiming to integrate automated workstations into fully integrated factories. Unfortunately for the industrial world, the results were underwhelming. Why?

From 1985 onwards, the authors set out to find out what causes smaller systems when integrated into a larger system of systems, to collide. Subsequently, they developed a design methodology that leads to easily scalable, robust (manufacturing) systems. The methodology is described in a book, entitled: 'Design for the unexpected. From holonic manufacturing systems towards a humane mechatronics society' [1].

As the title of the book suggests, in order to avoid collisions, the designer of the smaller systems to be integrated must refrain from making assumptions about the system of systems in which the smaller systems are to be integrated. Indeed,

[1] Faculty of Engineering Science, KU Leuven, Belgium
[2] Faculty of Engineering Technology, KU Leuven, Belgium
[*] E-mail: hendrik.vanbrussel@kuleuven.be

the designer may not impose arbitrary constraints or at least must make it easy to revise them. As will be shown, low-and-late commitment is important to achieve that goal.

2. DESIGN OF COMPLEX-ADAPTIVE SYSTEMS

2.1. SIMPLE, COMPLICATED AND COMPLEX SYSTEMS

Systems can be simple, complicated or complex. The behaviour of simple and complicated systems is well predictable. The reasons are that the interconnections between the system components are well-defined and fixed, while the component interactions remain simple and predictable. A bicycle with some hundred components is a simple system, a car with some ten thousand components is a complicated system. Manufacturing systems are complex systems. *Complex systems* are systems whose behaviour cannot be inferred from the knowledge of their constituent subsystems in isolation. Their behaviour is dictated by the nature and the sequence of the interactions of the subsystems. They exhibit emergent behaviour, or even self-organisation, and this makes the system behaviour less predictable and their control difficult. Large complex systems, with a large number of components (called 'agents' or 'holons') that not only interact, but also adapt and/or learn, are called *complex adaptive systems*. Holonic manufacturing systems, the subject of this paper, are complex-adaptive systems.

Designers of complicated systems, e.g. cars, remain largely in control of what their artefacts will be. In contrast, many artefacts in a modern human and industrial society, like infrastructures and manufacturing systems, are simply too complex to be conceived explicitly by humans. They emerge by the combination and integration of simpler systems. The resulting emergent behaviour is too often characterised by poor performance and missed opportunities. In this paper, a design methodology is outlined that avoids collisions between subcomponents when combined into larger systems and that enables the system to cope with changing requirements or unexpected events.

2.2. FUNCTIONAL DESIGN VERSUS STRUCTURAL DESIGN

Simple or complicated systems use to be designed by a *top-down functional design* methodology, by which the design team remains largely in control of the system performance. *Axiomatic design* [2] is a representative example of this approach. The method starts from the functional requirements (FRs) defined by the users. These FRs can be satisfied by manipulating some design parameters (DPs). Axiomatic design is based on two basic axioms: (i) the independence axiom, and (ii) the information axiom. The independence axiom requires the FRs to be independent, meaning that each FR should preferably be controlled by only one DP. The information axiom states that the best design is the simplest design that still satisfies all the (independent) FRs.

Top-down functional designs are extremely vulnerable to changes in user requirements. Furthermore, they force designers to make important choices early in the design process when they have minimal knowledge about the future use of the designed artefact. In conclusion, top-down functional design remains the methodology of choice for developments that need to become operational as fast as possible, need to be efficient and can recover their costs by answering their initial requirements only. Such developments typically produce mechatronic (systems of) systems, like e.g. cars, robots, machines, chemical plants.

A design approach that is more suitable for the aims pursued here is based on a structural decomposition of the design problem, rather than the functional approach of the axiomatic design. It has been developed predominantly for software design in the 1980s and is called *object-oriented design*, pioneered by Jackson [3]. It is based on the insight that user requirements are highly unstable parts in a design problem, and further on the facts that in top-down design one is forced to take crucial decisions early in the design process, and that in the real world there is seldom a single hierarchical decomposition.

Jackson realised that the world of interest for the problem is the most stable over time. His methodology consisted of reflecting in software the entities of interest and their relationships. Next, measures are implemented to keep the information in the computer system synchronised with reality. Finally, the functionality needed to answer the user requirements is implemented on top of this reflection of the *world of interest* (WOI). This reflection of the world of interest is called *essential model.* It describes (i) the possible states the WOI can be in, (ii) which events cause which state transitions, and (iii) the possible sequences in which events can occur. In our design for the unexpected of HMS, essential models play a crucial role, as a *single source of truth.*

2.3. DESIGN PRINCIPLES - DESIGN FOR THE UNEXPECTED (D4U)

Design of subsystems for emergent solutions imposes its own requirements. Two design principles guarantee the easy integrability of subsystems into larger systems under varying operational conditions.

P1. Problem solvers must avoid introducing potentially harmful constraints

This principle calls for design decisions introducing *stable constraints* first and as much as possible, e.g. the universal use of 240V/50Hz in Europe. Maps in navigation systems and nonlinear process plans (essential models) are other examples. *Unstable constraints* must be avoided or introduced as little and as late as possible (*low and late commitment*). An example would be integer versus decimal representation of numbers in software.

P2. Problem solvers must avoid/reduce the inertia build-up for potentially harmful constraints

When designers cannot avoid introducing unstable constraints, they may not use them to justify subsequent unstable design decisions. Repeatedly making design choices introducing an unstable constraint will build up inertia; it will practically become impossible to undo this introduction when it reveals to be an unfortunate choice later. E.g. legacy software systems are commonly associated with this issue.

2.4. HOLONIC SYSTEMS – FLEXIBLE HIERARCHIES

Two observations of how social and biological systems are organised motivated Arthur Koestler [4] to propose the concepts of *holon* and *holonic systems*.
The first observation was that complex systems will evolve from simple systems much more rapidly if there are *stable intermediate forms* present. This observation was influenced by Herbert Simon's parable of the two watchmakers [5]. Simon's parable demonstrates how, in dynamic and demanding environments, the chances of emerging and surviving for systems composed of suitable subsystems are vastly superior to systems composed from basic elements without stable intermediate states or subsystems.

The second observation was that, although it is easy to identify sub-wholes or parts, *wholes* and *parts* in an absolute sense do not exist anywhere. The term *holon* was proposed to describe the hybrid nature of sub-wholes/parts in real-life systems. Holons simultaneously are self-contained wholes to their subordinated parts, and dependent parts when seen from the inverse direction. Or put more simply: a holon is something that is whole in itself as well as part of a greater whole. Koestler called this behaviour the *Janus effect*.

Koestler also points out that holons are autonomous self-reliant units, which have a degree of independence and handle contingencies without asking higher authorities for instructions. At the same time, these holons are subject to control from higher authorities. The first property emphasizes that holons are stable forms and can cope with disturbances. The second property highlights that holons are intermediate forms, providing the proper functionality for the larger whole.

According to Koestler, a holonic system or *holarchy* is then a hierarchy of self-regulating holons which function: (i) as autonomous wholes in supra-ordination to their parts; (ii) as dependent parts in subordination to control at higher levels; (iii) in coordination with their local environment.

2.5. HOLONIC MANUFACTURING SYSTEMS

Based on the concepts of Koestler, a new form of manufacturing systems, called *holonic manufacturing systems,* emerged. This new paradigm had the ambition to provide an answer to shortcomings of earlier factory control systems that led to the failure, and ultimate demise, of the then prevailing CIM (Computer Integrated Manufacturing) paradigm.

Holonic manufacturing execution systems (HMES) were put forward as it was realised that neither centralized, hierarchical nor heterarchical control systems could face the challenges the manufacturing world was confronted with. HMES try to combine the high and predictable performance promised by hierarchical systems with the robustness against disturbances and the agility of heterarchical systems by having characteristics of both architectures. To avoid the rigid structure of hierarchical systems, holonic manufacturing systems provide autonomy to the individual holons. This allows the control system to respond quickly to disturbances and to reconfigure itself to face new requirements. In order not to ban all hierarchy, which is essential to master complexity, holons work together in ‘loose’ hierarchies. Such a hierarchy is different from a traditional hierarchy in that: (i) holons can belong to various hierarchies, (ii) holons can form temporary hierarchies, and (iii)holons do not rely on the correct functioning of the other holons in order to perform their tasks.

The relationship between different levels is not a master-slave, but an advisory relationship.

3. THE PROSA REFERENCE ARCHITECTURE

This section addresses the reference architecture of a holonic manufacturing execution system (HMES). It describes the structure of the system of holons (*holarchy*), not the internal structure of individual holons.

A *reference architecture* describes the mapping from various functionalities, which cooperatively solve the problem, onto software components and the data flows between these components. A reference architecture is not an architecture in itself, but can be used as the basis for designing the system architecture for a particular system. Reference architectures are used in a specific (mature) domain and arise from experience. E.g. the reference architecture ‘Gothic cathedrals’ is the collection of knowledge and skills, acquired by the medieval guilds, to build Gothic cathedrals. The cathedral of Chartres, with its exquisite architecture, is an impressive instantiation of that reference architecture.

PROSA (Fig. 1) was originally developed for the manufacturing domain and, based on experience in this domain, special attention was paid to: (i) separating the essential elements, which are generic, from the optional elements, which can be domain specific; these latter are called *plugins* in the sequel; (ii) separating the structural aspects from the functional (algorithmic) aspects for resource allocation and process planning; (iii) separating resource allocation aspects and process specific aspects; (iv) enabling the incorporation of legacy systems, or the introduction of new technology.

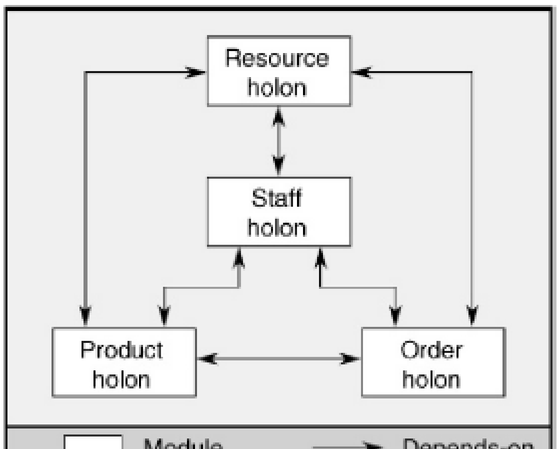

Fig. 1. Module view of the PROSA reference architecture

The PROSA reference architecture is developed in accordance with the holonic manufacturing paradigm. The basic components are holons and the architecture describes the responsibilities of the various holons and their interactions. The acronym PROSA stands for *Product-Resource-Order-Staff Architecture* and refers to the different types of holons. Three basic types of holons can be distinguished: *product holons*, *resource holons* and *order holons*. *Staff holons* complete the set of PROSA holons.

Each holon represents a separate concern in the application domain: process planning, resource allocation and logistics management, respectively. The basic holons can be aggregated into larger holons and specialization can be used to structure them. Staff holons are optional and can be added to provide the other holons with expert knowledge or to incorporate legacy systems. Fig. 1 shows a module decomposition view of the holonic reference architecture. The depends-on relationships between the holons indicate that the various holons share data with each other.

3.1. RESOURCE HOLON

A *resource holon* corresponds to a resource in the underlying domain (equipment, infrastructure elements, personnel). In a logistic context, for instance, this means that all transport means (trucks, freight trains, cargo aircraft…) and material handling equipment (forklift trucks, conveyors, automated guided vehicles…) will be represented by a resource holon. There will also be resource holons for other entities that are scarce and have to be shared (e.g. docking doors, pallet racks, floor space, etc).

Each resource holon comprises the physical resource, together with a software part that controls this resource. It offers knowledge about processing capacity and processing functionality to the other holons and organizes and controls the usage of the physical resource. A resource holon has the following responsibilities:

(i) Reflection of reality: A resource holon reflects its corresponding physical resource, at all times, as a *single source of truth*, i.e. contains information about the current state of the resource and expected future states.

(ii) Information provision: A resource holon should be able to provide resource related information to the other holons. This includes process information (e.g. possible operations), information about the local topology (which other resource holons this holon is logically connected with) and about possible constraints (e.g. truck capacity, maximum cargo weight, etc.).

(iii) Maintaining a local schedule: Each resource holon owns an agenda in which its future tasks/operations are recorded, based on requests from order holons.

(iv) Managing its local schedule: The resource holons have local authority on how they organize (sequence or schedule) the various operations (from order holon requests), for instance by applying priority or batching rules.

(v) Virtual execution: This responsibility is a service for the order holons who can request information on the virtual outcome of an operation (e.g. quality and end time).

(vi) Controlling the resource: A resource holon controls the real-world resource by starting and stopping the (scheduled) operations and by monitoring the execution.

3.2. PRODUCT HOLON

A *product holon* corresponds to a task type or order type. To accomplish the task or to fulfil the order, a process (a sequence of operations) has to be executed. The product holon contains the knowledge on how instances of a specific task type (represented by order holons) can be executed by the resources, i.e. which operations are required to accomplish the task correctly and qualitatively. The product holon also has information about constraints on or process parameters of these operations. For instance, if the package contains refrigerated products, the holon knows the allowable temperature range to which the package can be exposed during transportation.

The main responsibilities of a product holon are:

(i) Maintaining process knowledge: The product holons hold the necessary process knowledge to realize instances of their type. This includes, amongst others, process plans, process parameters and quality requirements.

(ii) Determination of operation options: A product holon informs the order holons about all possibilities for their next operation.

(iii) Process information provision: Just before a selected operation should start on a resource, the resource holon needs to know the desired process parameters. The product holon is responsible for providing this process information to the resource holon.

3.3. ORDER HOLON

An *order holon* corresponds to a task (instance) or order (instance) that needs to be executed, e.g. the delivery of a package. The order holon is responsible for handling the required resource allocations in order to accomplish the correct execution of its task. In a manufacturing context, an order holon might correspond to a number of products that have to be produced by a certain due date.

An order holon has the following responsibilities:

(i) Reflection of reality: An order holon reflects the order instance, i.e. contains information about the current state of the order and the corresponding physical entity. This includes for instance the location of the order, the current operation being processed, the resource performing this operation, etc. The order holon is responsible for keeping the reflection of its state up-to-date with the actual state.

(ii) Searching solutions: The order holons search for solutions to execute their tasks. During their search, the order holons will consult their product holons to know the required operations and will virtually execute these operations (by using the virtual execution service of the resource holons) to check for resource availability.

(iii) Intention selection: Each order holon evaluates the solutions it has found and chooses the most attractive solution (according to its performance measure) to become its intention.

(iv) Reserving its intention: The order holon then informs the other holons about its intention by making the necessary reservations (future allocations) at the involved resource holons. As these reservations evaporate after a certain time, the holon has to confirm its reservation at regular time intervals.

3.4. STAFF HOLON

The three basic types of holons can be assisted by one or more staff holons. These holons can provide the other holons with expert knowledge about certain aspects of their decision making. Note that the staff holons only provide advice and that the basic holons are still responsible for taking the final decisions. This way, the concept of staff holons allows for the presence of centralized functionality in the architecture without introducing a hierarchical rigidity. This centralized functionality allows aiming for a good global performance, which is otherwise difficult to obtain as every holon tries to optimize its own (selfish) objective. To obtain its advice, a staff holon may rely on centralized scheduling algorithms, human input, artificial intelligence methods, etc. The various order holons will attempt to execute (the relevant part of) the provided schedule. They will deviate from the original schedule only if they find a significantly better solution or the provided advice appears to be (or has become) infeasible.

3.5. INTERACTIONS BETWEEN THE HOLONS

As indicated in Fig. 1, the various holons interact and share data with each other. Important to stress here is that these interactions do not describe the dynamics of the holonic system described by the PROSA reference architecture. These dynamics are taken care of by the Delegate MAS (DMAS) coordination system, described hereafter. They form the basis of the holonic manufacturing execution system (HMES).

Product-order interaction:

The order holons interact with their corresponding product holon on how to correctly execute their task by using certain resources. After (virtual) execution of an operation, the order holon passes information about the resulting state and about next possible resources to the product holon. Based on this information, the product holon provides the order holon with all possible next operations. For instance, after loading a container onto a trailer, the corresponding order holon consults its product holon to know the following operation that should be executed. Usually, multiple options are available, e.g. direct transportation to the final destination, transportation to an intermodal hub to be loaded onto a train or ship, etc.

Product-resource interaction:

Product and resource holons share process related information. When generating a list of possible operations for an order holon, the product holoninteracts with resource holons to know which operations the resources can perform. The other way around, the product holon provides the resource holon with technological aspects to correctly process an order, i.e. the necessary process parameters to perform an operation. For instance, if refrigerated products have to be transferred between two trucks in a non-cooled terminal, the product holon will indicate that this transfer should happen as fast as possible and impose a maximum transfer time.

Resource-order interaction:

The resource and order holons mainly interact to reserve operations on the resources. To this end, the resource holons provide the order holons with the results of virtually executed operations and reserve capacity when requested. Once an operation is started, the resource holon also informs the order holons about the execution result and progress. The desired coordination and control then emerges in a self-organizing way from the interactions between the various holons.

4. BIO-INSPIRED COORDTION AND CONTROL IN HOLONIC EXECUINATION SYSTEMS

Initial PROSA implementations were heterarchical control systems. Their order holons would steer product carriers through a manufacturing system, visiting processing stations and having production steps performed as allowed by the product holons. The system would be myopic, lacking global optimization or coordination and operating much like automobiles on the road.

In order to cope with this myopic nature of a pure heterarchical system, an HMES based on stigmergy was developed. Stigmergy describes the mechanism by which ants and other social insects are foraging for food. It reveals how to incorporate non-local information in a solution while employing only local reality-mirroring components. This property fits D4U perfectly.

Food foraging ants execute a simple procedure:

- In absence of any signs in the environment, ants perform a randomized search for food.
- When an ant discovers a food source, it drops a smelling substance, called *pheromone*, on its way back to the nest while carrying some of the food. This pheromone trail evaporates if no other ant deposits fresh pheromone.
- When an ant senses a pheromone trail it will be urged by its instinct to follow this trail to the food source. When the ant finds the food source, it will return with food, while depositing pheromone itself. When the ant discovers that the food source is exhausted, it starts a randomized search for food and the trail disappears because of the evaporation.

This simple behavioural pattern results in an emergent behaviour of the ant colony that is highly ordered and effective at foraging food while being robust against the uncertainty and the complexity of the environment.

An important capability of this type of stigmergy can be observed: global information about where to find food in a remote location is made available locally, indicating in which direction the ant must move to get to this food. Also, the complexity of the environment is handled in an elegant way by making the environment part of the solution (i.e. the complex shape of the pheromone trails), effectively shielding the ant colony solution from this complexity.

For the design and development of coordination and control systems, based on stigmergy, following principles are recognized:

- Make the environment part of the solution to handle a complex environment without being exposed to its complexity. This complies with the *essential modelling* approach of object-oriented design.

- Place relevant information (pheromones) as signs in this environment ensuring that locally available data informs about remote system properties, supporting system-wide coordination.
- Limit the lifetime of this information (*evaporation*) and refresh the information as long as it remains valid. This allows the system to cope with changes and disturbances.

The combination of these sources of inspiration resulted, ultimately, in the *architectural patterns* that are discussed next. The signal became the local schedule of the resource holon. The deposition of pheromones was translated into order holons reserving time slots in these local schedules. The ability to answer what-if questions allowed *ant agents* to travel virtually and execute virtually what an order holon might 'do for real'.

Our research translated this food-foraging in ant colonies into a solution that remedies short-sightedness (*myopy*) in decentralised coordination and control systems. The solution makes non-local information – using local reality-mirroring software components – locally available. Remark that 'non-local' is to be understood both in the geographical/spatial sense and in the temporal sense. It is about something both elsewhere and in the future.

4.1. DELEGATE MAS (DMAS)

The DMAS pattern allows an agent – i.e. a holon in the present discussion – to delegate a responsibility to a swarm of *lightweight agents*. These lightweight agents perform particular activities to support the issuing holon in fulfilling its functions. A holon can simultaneously delegate multiple responsibilities, applying the delegate MAS pattern for each of them. The holon may also use a combination of DMASs to handle just a single responsibility.

These lightweight agents are called *ant agents* or simply *ants*, after their biological source of inspiration. They are lightweight in the sense that each ant may only perform a bounded computational effort within its bounded lifetime and has a bounded footprint (memory). They are responsible for executing a task that serves a responsibility of the issuing agent/holon.

Each ant is created and initialized by its issuing holon. It (virtually) travels autonomously through the (virtual) environment. The ants start from a location selected by their issuing holon. Typically, this location is where this issuing holon resides (virtually), e.g. the location of a product carrier. But, an issuing holon may create ant agents at a location from where finished products are shipped to their customer. From there, the ants travel (virtually) in opposite directions, typically toward the location of the issuing holon. Ants may even (virtually) traverse their journey twice, collecting information first and depositing information (digital pheromones) during the return journey.

Corresponding to the description used before, the *environment* is a software representation of the world-of-interest. To support navigation of the ants, resource holons know their neighbours (note that this is local information). This effectively provides a directed graph, possibly augmented by relevant information (e.g. maximum height), allowing ants to discover their world-of-interest starting from their initial location. Note that the evaporate-and-refresh mechanisms – copied and translated from the real-world ant colony behaviour – ensures that reconfigurations and other changes will be mastered by the DMAS in a holonic MES.

A holon delegating a responsibility to a swarm of ants is responsible for maintaining the population size and the diversity of this swarm. It chooses the creation frequency and initialization for every ant type. The individual ants are not aware of these swarm properties. The holons observe and interpret the (digital) pheromones in the environment and adapt their behaviour accordingly.

Three types of delegate MASs are distinguished in the research prototypes: *feasibility, exploring and intention delegate MAS* [9].

4.2. FEASIBILITY ANTS

A resource holon delegates part of its 'information providing' responsibility to a swarm of so-called *feasibility ants*. These ants make global feasibility information (about the capabilities of the resource) locally available for the other holons. They put a kind of digital signposts on the blackboards of resource holons. They enable order holons to decide locally which routing options are available to them.

4.3. EXPLORING ANTS

Every order agent generates explorer ant agents at a given frequency. These explorer agents are scouts each of which searches for an attractive route through the underlying production system that is to accomplish the given task (Fig. 2). Depending on the performance criterion, these explorer agents search forward from the current state of the task on (e.g. lead time minimization) or backwards from the final delivery point (e.g. due date accuracy). Note that different order agents can have different performance concerns; rush orders, normal orders, low priority orders, maintenance orders have different objectives. The objective of a given order may even suddenly change (e.g. when a work piece gets damaged and needs a speedy replacement).

These scouts use the same method as the order agent, managing the actual execution of the task, to ensure that a proper sequence of processing steps gets executed, but virtually. The feasibility concern is handled by the feasibility ant agents. As explained, these ants deposit information on the information spaces (blackboards) attached to entries and exits of the resources that allows the product agents to discern valid and invalid routings locally. This information also evaporates and is refreshed to account for changes in the production system.

The search strategy employed by the explorer agents is a *plug-in* of the control system. Not every explorer ant uses the same strategy. Typically, some percentage looks for the promising routes whilst other ant agents look for solutions that aim to avoid critical resources. The key point is that the emergent forecasting (see further) does not rely on which strategy is employed by these scouting agents.

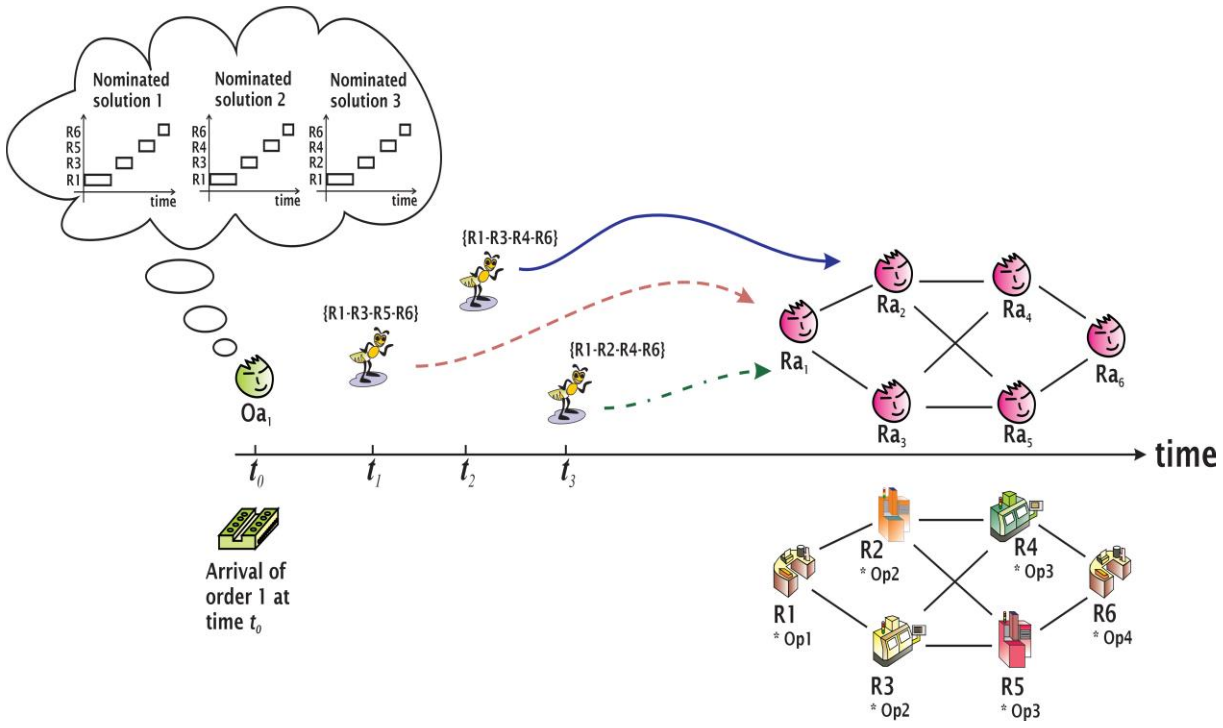

Fig. 2. Ant agents explore possible routes

During its exploration journey, ant agents delegate the information processing to the product holon and resource holons. Their product holon provides the set of legal routing options that are open to the scout at each routing point. It makes sure that the product recipe is obeyed. The resource agents provide the necessary performance estimates.

When an explorer ant agent has virtually executed the task, it reports back to the order holon. The report includes the journey and the performance estimates of that journey. Based on the results of its exploring ants, the order holon keeps a set of candidate routes. These candidates get refreshed regularly, either explicitly by specialized exploring ants that simply follow a given route, or by ensuring that the normal exploring ants will rediscover these currently attractive candidates with a high probability. The set of candidate routes is selected based on the performance estimates and on their complementary nature (i.e. limit the number of candidates that have very similar routings). The candidates that have become too old are eliminated from the set of candidates by evaporation.

4.4. INTENTION ANTS

The above mentioned exploration requires the resource holons to possess an adequate estimate of their future workload. The order holons generate intention ant agents, at a given frequency, to serve this purpose. When a suitable set of candidate solutions has been constructed (see above) and the estimated starting time for the processing of the product instance(s) approaches, the order holon selects one of the candidate solutions to become its intention. Then, the order holon generates intention ant agents to notify the holons of the affected resources of its intentions (Fig. 3).

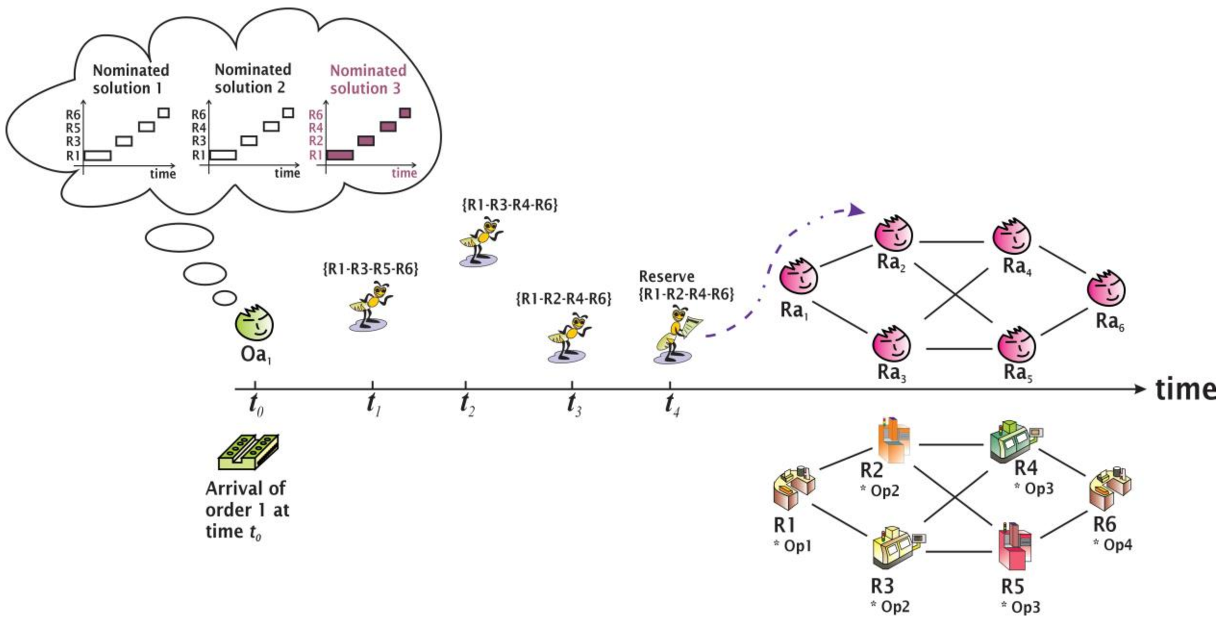

Fig. 3. Ant agents propagate the order intentions

4.5. SHORT-TERM FORECASTING – PREDICTING THE UNEXPECTED

The combination of exploring and intention delegate MAS provides a view on the short-term future of the system, which is based on an estimation constructed through a decentralised virtual execution (i.e. a simulation embedded in the holonic MES). Both resource and order holons have short-term forecasts about their predicted execution. The order holons know the expected routings and resource allocations for their orders. The resource holons the predicted loads for the corresponding resources.

The resource holons receive the necessary information to calculate a short-term forecast of their utilisation via the intention delegate MAS. Based on these forecasts (and their what-if functionality), they are able to give accurate answers to the queries from the exploring ants. This in turn allows the order holons to have a precise view on their short-term future. Note that the order holons create exploring and intention ants at regular time intervals, even after they have selected an intention. This allows them to react to disturbances and new opportunities and keeps the short-term forecasts up-to-date.

All short-term forecasts together can be seen as a dynamic schedule.

5. CASE STUDIES

A few case studies taken from manufacturing, robotics and open-air engineering, are discussed here to illustrate the application of the PROSA/DMAS scheme to achieve *design for the unexpected* of holonic execution systems. Execution systems manage operations in real-time. They only trigger system-specific activities, leaving detailed control to other systems. For instance, a manufacturing execution system (MES) leaves the pick-and-place of a dashboard into a car body to the robot controller and/or human workers. In contrast, the MES manages the routings and processing sequences for these products (e.g. car bodies, doors and dashboards). Next to the mentioned three application areas, the PROSA/DMAS scheme can be equally applied to logistics, urban mobility, railway operations, smart grids, smart homes, e-health, showing its universality [1].

Recently, when trying to apply PROSA to the non-manufacturing world, e.g. robotics, the need was felt to rephrase the nomenclature of the manufacturing-specific PROSA holons (product, resource, order, staff). This resulted in the ARTI (Activity-Resource-Type-Instance) reference architecture, containing following holons: *activity type, activity instance, resource type, resource instance*. More details can be found in [1].

Example 1: Networked manufacturing [6]

This manufacturing case study addresses a highly automated heat treatment multi-plant facility. This facility performs heat treatment on metallic parts and includes several processes: case hardening, vacuum hardening, induction heating, etc. The products demand a certain temperature trajectory inside the furnaces in order to reach the required quality. The time between different processes (for instance between case-hardening and tempering) should not be too long for some products. The various furnaces differ from each other in the range of working temperature and environmental condition (e.g. carbon level). The facility is organized as a job shop in which the baskets containing the metallic parts are transported automatically.

The HMES system is responsible for the routing of the to-be-treated metallic parts through the facility, ensuring that these parts receive a correct treatment. The resource holons correspond to the transportation and heat treatment equipment (e.g. furnaces, washing stations and cooling beds). The services offered by these resources are used by the product holons, corresponding to the metallic parts that have to be treated.Specific for this application is that parts with compatible process temperature trajectories and environmental conditions can be batched. When properly executed, this batching has a significant impact on the performance of the capital intensive production system. Indeed, a fully loaded furnace and a partially loaded one operate almost at identical cost whereas the output differs significantly. The product holons can make use of a delegate MAS to discover batching opportunities or, alternatively, to trigger the build-up of such batches.

This case study also investigated the scalability of the HMES by coordinating manufacturing and transportation activities within networked production. A virtual enterprise was considered, consisting of a network of heat treatment factories. New companies can dynamically join or leave the network and new processes and equipment are introduced as needed. Now the products have to route their corresponding parts at two levels: the network level and the factory level. At the network level, the product searches for transportation services between the different factories and heat treatment services (offered by aggregated resource holons, offering all services of the resources at a factory). As such a virtual enterprise is a semi-open system, lacking a single command and control centre, the operations have to be organized without the disclosure of sensitive information to other members of the network. Also, a mechanism is required to deal with trust and reputation issues [1].

Example 2: Robot fleets [7]

One marked difference with manufacturing applications, for holonic task execution control of multi-mobile-robot systems, the environment of the robots has to be included explicitly as a (mostly aggregated) resource holon. The explicit representation and allocation of environment resources facilitates the execution of coordination tasks.

Without coordination, the mobile robots risk deadlock and live-lock situations. Indeed, failure to explicitly manage resource allocation creates the need to enhance this coordination-less system with a deadlock detection and roll-back functionality, which is far from trivial even for a specific system configuration. Without a DMAS mechanism generating predictions concerning the time and duration of resource (narrow corridor) allocations, the mobile robots lack information to balance a longer route against waiting for the resource to become available.

Consider the case with an environment consisting of two rooms, upper room 1 and lower room 2, connected by a narrow corridor (Fig. 4). Two autonomous wheelchairs, SARA and INGA, are involved; SARA wants to go from 1 to 2, and INGA wants to go simultaneously from 2 to 1.

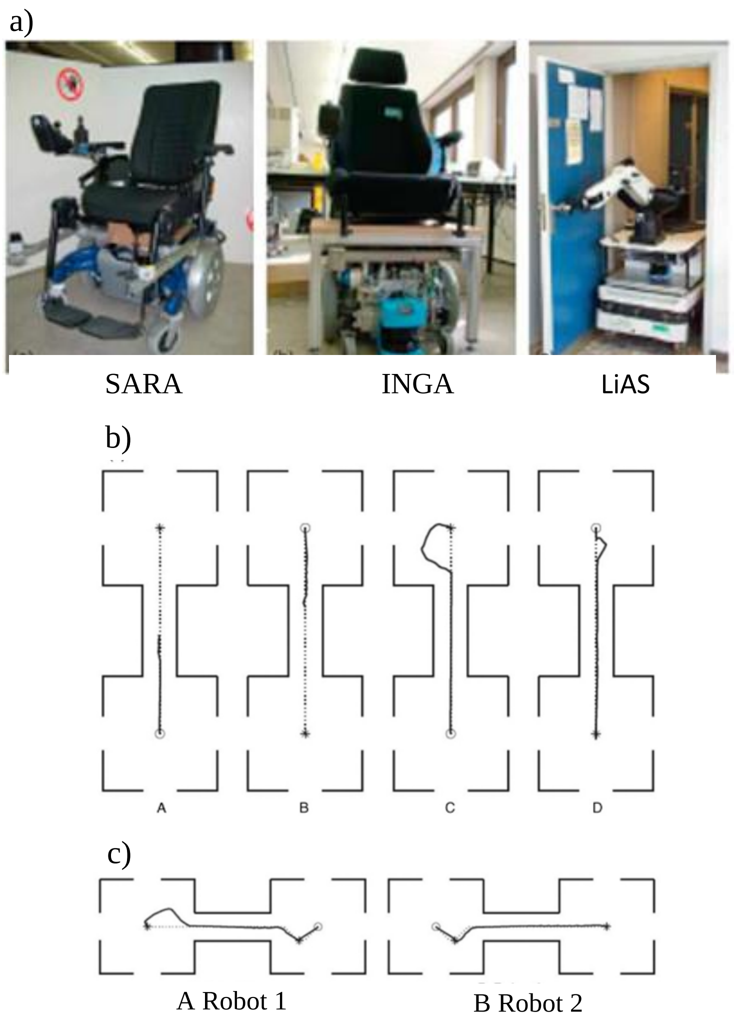

Fig. 4. The robots used in the experiments (a), examples of negotiating a common corridor without (b), with coordination (c)

The narrow corridor allows only one robot to pass at a single time, hence coordination is required. Figure 4a shows a failed (A- B) and a successful run (C – D). In the failed run, both wheelchairs entered the corridor almost simultaneously causing livelock. The deviations in the successful run are caused by one robot avoiding the other by the functioning of a built-in obstacle avoidance algorithm. In the experiment shown in Figure 4b, the robots are aware of each other's intentions by which they only enter the corridor after the scheduler of the particular corridor resource has allocated to them a valid slot. This way livelock or backtracking is avoided. Whenever a robot is not able to enter a corridor yet, it waits in a transit zone in front of the corridor entrance to give way to the arriving robot.

Figure 5 shows a slightly more complicated situation. Two wheelchairs, SARA (S) and LiAS (G), are to move in opposite directions between two labs in the authors' laboratory. There is a mutually exclusive region where coordination is required. Fig. 5b shows the 3D plot of the trajectories. The solid lines are the executed trajectories with respect to the time and the dashed lines indicate the corresponding trajectory projected onto the environment. The time dimension shows that LiAS, starting at the left, waits until SARA, at the right, passes through the area where their trajectories overlap. The fact that the solid trajectories do not intersect in Fig. 5b indicates that no collisions occurred.

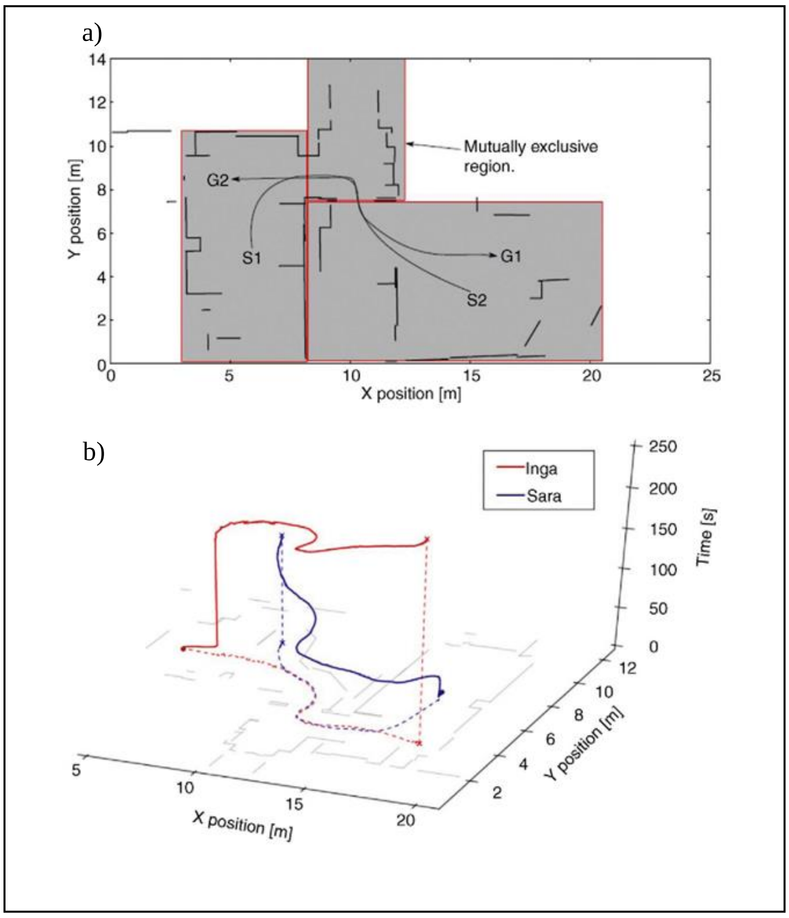

Fig. 5. Two mobile robots coordinate their behaviour to avoid collision in the mutually exclusive region
a) top planar view, b) 3D view

Example 3: Open Air Engineering [8]

Open air engineering processes, such as open-pit mining, road construction and farming are mostly carried out with high-tech mobile equipment, including self-propelled work vehicles such as excavators, dump trucks, asphalt layers, road graders, combine harvesters, etc. that are specifically designed to carry out these processes (Fig. 6).

Fig. 6. Open air engineering processes

Open air engineering applications have some typical characteristics: (i) the mobile equipment interacts with a (non-flat) 2D surface, the shape and properties of which are modified by the equipment, (ii) material is either removed (harvesting, mining) or deposited (asphalt depositing), (iii) some local storage might be available (e.g. on-board combine harvester), (iv) material needs to be supplied/shipped from/to a production/ processing site (ore, corn).

Since the operations of this mobile equipment are expensive, profitability depends directly on how effectively this equipment is utilised. Loss of production capacity is to be avoided and minimised by proper coordination. Idling of the bottleneck resource(s) is the key concern. Here, the coordination faces significant levels of uncertainty and variations in working conditions, possibly shifting the bottleneck from the mobile equipment to the transporters or vice versa.

The yield of the surface processing by the mobile equipment varies and is subject to uncertainty in function of (i) surface properties (ore density, crop density, accessibility, …), (ii) trajectory executed (accessibility and allowed trajectories of transporters).

Coordination needs to adapt to these varying and uncertain working conditions. But, the coordination also impacts on these working conditions. The trajectory executed affects how transporters may cross the surface to service the mobile equipment (e.g. trucks cannot drive across a 1 meter steep slope created by the excavator, tractors must not drive over still-to-be-harvested crops). Using information about the surface, the trajectory selection affects the expected yield (e.g. crop density will vary across fields where historical data or aerial photography may allow for good estimates).

Overall, coordination by HMES can make a difference. When the truck is delayed in traffic, the mobile equipment shall be operated in an energy-saving mode, minimising losses on the surface, tackling tricky parts of the surface, or perhaps performing small maintenance tasks. When the site is near the processing unit or depot, the mobile equipment utilisation is a priority and truck may have to wait when an optimised trajectory keeps the mobile equipment out of reach until its reservoir is full/empty.

The novelty in addressing this application domain was twofold. Chronologically, it was among the first to require multi-resource allocation and, especially, multi-resource allocation for which a leader-follower approach was ill-suited. In particular, the assignment of transporters to activities on mobile equipment prompted the choice for a resource pool holon, managing a collection of very similar resources.

Secondly, this application domain involved modelling surfaces as resources where activities modify the surface properties and where surface properties determine resource capabilities. For instance, a corn field surface will have initial location-dependent properties based on historical information and measurements. When a combine starts to harvest, properties of the processed parts of the surface need updating (e.g. indicating that tractors may cross). Likewise, measurements by the harvesting equipment can be used to improve the estimates for the unprocessed parts of the surface. DMAS mechanisms explore the processing of these surface and, by propagating intension, predict the surface properties in function of time. E.g. the exploring ants for a tractor may see which paths to the harvesting equipment will be available when it arrives as well as the estimated position of the mobile equipment.

Overall, coordinating open-air engineering processes involves multi-resource allocation and trajectory determination. Using the DMAS mechanism, this coordination aims to optimize one or more performance objectives, for example, bottle-neck utilization, energy consumption, etc. In an open-air engineering process, the work vehicles perform operations at geographically distributed locations (mine site, storage depot, grain fields, etc.). Because of the open and distributed nature of open-air engineering processes, disturbances and variations are highly prevalent in their operating environments.

In practice, plans are generated before the process starts, based on approximate resource performance and predicted operating conditions. Although these plans provide a good starting reference for execution, they are unable to provide the necessary visibility for continued execution of the processes, which are subject to uncertainty and variations. For effective execution, gaining visibility at runtime hence is imperative. With more runtime information, it becomes easier to identify sources of problems or opportunities and take effective decisions. This is what holonic execution systems are designed to provide.

6. CONCLUDING REMARKS

Developers who want to ensure that their design is suited for integration without control over what needs to be integrated with, are bound to *design for the unexpected*. This can be achieved by developing, as much as possible, elements of a solution without the need to rely on expectations. In order to avoid or minimize unstable constraints, designers have to apply *low-and-late-commitment* in their design decisions.

The described *reference architecture,* called PROSA (and its generalisation ARTI), enables the description of the structure of holonic (manufacturing) systems, and their interactions. The formalism is scalable by its *rigorous separation of concerns*.

The control dynamics (task execution) are ruled by the DMAS, with as most important feature the *ability of short-term forecasting,* obtained by a *decentralised virtual execution*, based on a *digital mirror image of the world of interest* that reflects reality at all times, as a *single source of truth*. DMAS is scalable and provides robustness to the controlled system in terms of adaptivity and scalability.

The universality and general applicability of the PROSA/ARTI/DMAS framework are demonstrated with three case studies from manufacturing, robotics and open air engineering.

To assess the unique power of the here developed framework, the reader is referred to other parallel or similar research, the most representative of which are PRODUCTION 2000+ [10], XPRESS [11], ADACOR [12] and to the book by the authors 'Design of the unexpected' [1].

ACKNOWLEDGEMENTS

The research described here spans a period of almost 30 years [1]. The authors are indebted to a large group of dedicated researchers who participated and to the several European, national and regional research agencies for their financial support.

REFERENCES

[1] VALCKENAERS P., VAN BRUSSEL H., 2016, *Design for the unexpected. From holonic manufacturing systems towards a humane mechatronics society*, Elsevier.

[2] SUH N.P., 1997, *Design of systems*, Annals of the CIRP, 46/1, 75-80.

[3] JACKSON M., 1995. *Software requirements and specifications*, Addison-Wesley, Amsterdam.

[4] KOESTLER A., 1967, *The ghost in the machine*, The Macmillan Company.

[5] SIMON H.A., 1990, *The sciences of the artificial*, MIT Press, Cambridge, MA.

[6] SAINT GERMAIN B., VALCKENAERS P., VAN BRUSSEL H., VAN BELLE J., 2011, *Networked manufacturing control: an industrial case*, CIRP Journal of Manufacturing Science and Technology, 4/3, 324-326.

[7] PHILIPS J., VALCKENAERS P., BRUYNINCKX H., VAN BRUSSEL H., 2012, *Scalable and robust coordination of multiple mobile robots using PROSA and delegate MAS*. Proc. International Symposium on Robotics (ISR), 527-532.

[8] ALI O., VALCKENAERS P., VAN BELLE J., SAINT GERMAIN B., VERSTRAETE P., VAN OUDHEUSDEN D., 2013, *Towards online planning for open-air engineering processes*, Computers in Industry, 64/3, 242-251.

[9] HADELI, VALCKENAERS P., KOLLINGBAUM M., VAN BRUSSEL H., 2004, *Multiagent coordination and control using stigmergy*, Comput. Ind., 53/1, 75-96.

[10] BUSSMANN S., JENNINGS N.R., Wooldridge, M., 2004, *Multi-agent systems for manufacturing control: a design methodology*, In: Series on Agent Technology, Springer Verlag, Berlin, Germany.

[11] XPRESS, *Flexible production experts for reconfigurable assembly technology*, FP6-NMP Project n°26674, 2007-2011.

[12] LEITÃO P., RESTIVO F., 2006, *ADACOR: a holonic architecture for agile and adaptive manufacturing control,* Comput. Ind., 57/2, 121-130.