Another interpretation of stigmergy for product-driven systems architecture

Rémi Pannequin, André Thomas

Noname manuscript No.
(will be inserted by the editor)

Another interpretation of stigmergy for product-driven systems architecture

Rémi Pannequin, André Thomas

Research Centre for Automatic Control (CRAN)
CNRS (UMR 7029) – Nancy University,
27, rue du merle blanc, F-88000 Epinal, France
e-mail: remi.pannequin@cran.uhp-nancy.fr

The date of receipt and acceptance will be inserted by the editor

Abstract Product-driven systems (PDS) may enable manufacturing companies to meet business demands more quickly and effectively, by localizing decision as near as possible to the material flows. However, the actual architecture of a PDS is still not well defined. This paper proposes a PDS architecture based on a particular interpretation of the concept of stigmergy, where cooperation between production actors is achieved thanks to attributes (informational pheromones) attached to products. This stigmergic design pattern is presented. Agent-oriented components which implement it are presented, are firstly applied to a laboratory platform, and secondly on an industrial test-case.

Key words multi-agent systems; software architecture; manufacturing control; product-driven control; stigmergy

1 Introduction

High competition between enterprises and market volatility lead enterprises to be more *agile* [6]. Agility, from the point of view of production control, may be seen as the ability to operate with a high level of coordination and proactivity throughout the supply-chain, and *at the same time* to react efficiently to disturbances on the shop floor while taking into account the increasing process complexity (variabilities, high product variety, reconfiguration issues).

Historically, "centralized" approaches have been implemented thanks to MRP^2 systems and ERP, with tools and methods mainly based on operational research concerning production activity control (PAC). Facing the eighties' market challenges other decision making philosophies and strategies have emerged. Requirements for more and more reactivity and flexibility have led to the implementation of "distributed" approaches such as anthropocentric and visual management methods (kanban, operators empowerment , etc). Unfortunately these new ways to pilot and control the material flows have led to "black boxes" in management systems, and have highlighted the need to more and more real-time closed-loop information systems. On the other hand, it has been shown [14] that in such kanban systems, very short term priority management is always a key issue. Gradually, following information technology improvements, it seems obvious that to give to the physical system entities decision making capabilities could be a new way to face this ever unsolved issue. That's why holonic manufacturing [2] and agent-based manufacturing [16] systems try to bring intelligence as near as possible to the physical system.

The *product-driven* paradigm [20] is based on the assumption that the product is the core object of this kind of system. Indeed, the product is the common object shared by a vast majority of "actors" (i.e. by any company decision-making entities), each one having its own point-of-view on this product. For instance centralized decision system which focus on business aspects, and distributed decision systems which focus on operational ones. The main postulate in product-driven systems is that products are able to be active and are at the core of an enterprise architecture which integrates every actor in the company, from the central systems to the processes, products and also operators, into the same *ambient* information system. Therefore, the product become an active object in its environment [17].

Many researches and industrial works relative to the product-driven paradigm have been done. Amongst them, *product-driven systems*[21], *intelligent products* [17] [18] and *product-centric systems* [20] [25] can be cited by authors. All have been driven by the availability of new identification technologies such as radio-frequency identification, by the development of embedded systems such as wireless sensors networks, and by the spreading of distributed architecture like multi-agents systems. In fact, many points of view of this paradigm co-exists. Accord-

ing to [18] [17] these kind of systems can be implemented with (i) various levels of intelligence given to the product, (ii) different ways to locate intelligence (on the network or on the product) and (iii) different aggregation levels of intelligence. From these features, several classes of systems could be distinguished, leading to various interpretations of this concept.

Therefore, developing product-driven systems (PDS) raises numerous issues, ranging from conceptual to logical and technical ones. Example of architectural (logical) issues are for instance: how to structure the control system ? What entities should exist ? What kind of decisions should be allocated to the products, or to the other actors ? How should products interact with their environment ?

In this paper, we focus on architectural aspects of PDS: in particular, what decisions should (or shouldn't) be allocated to the product ? Indeed, if it is widely proposed to make the product active, its precise role inside the control system is seldom given, and the design patterns required to actually develop such a system are not always well-defined. This paper contributes to bridge this gap by proposing an architectural pattern based on a revisited interpretation of stigmergy to design product-driven systems.

First, section 2 will review the literature on intelligent products, state the architectural issues faced while designing product-driven systems, and remind the principle of stigmergy. Then, the proposition of a stigmergic architecture will be enunciated in section three, and its implementation will be demonstrated in section four. Finally, section five will present an application of our approach on an industrial test case.

2 Problem Statement

2.1 Application domains of product-driven systems

As stated in the introduction, in the eighties, people were (re)introduced in the reactive decision making processes. The main reason for that was that only operators had decision-making abilities and at the same time were in contact with the physical processes. In case of disturbing event, operators could perceive the event, gather accurate information and take decisions. In PDS, the main idea is to give these reaction capacities to the products (and to other parts of the physical system), because they are directly concerned by these events. However, one of the main issues to solve in the definition of a PDS architecture is what entities to define, and how to structure their interactions.

In holonic manufacturing systems, the structure of the control system is mapped from the physical entities. In the PROSA [26] reference architecture, three types of control entities (or agents) are associated to physical elements : *Product Holons*, that gather the knowledge about making products, *Resource Holons*, that control the devices used to transform products, and *Order Holons*, that represent and control the production of an item or a lot. Finally, *Staff Holons* help achieving coordination amongst the system.

Likewise, the ADACOR architecture [15] defines four manufacturing holons classes : product, task, operational and supervisor holons, the first three being quite similar to the product, order and resource holon of the PROSA architecture, while the supervisor holon introduces coordination and global optimization in decentralized control and is responsible for the formation and coordination of groups of holons.

While the holonic approach focuses on the manufacturing domain, other approaches try to consider more broadly the whole life-cycle of the products. The concept of the *Internet of Things* aims at allocating to every object a global identifier, that serves as a key to access to further product data, stored on the network. The *EPCGlobal* architecture proposes to store an identifier on the product, and introduces the concept of object name server (ONS) to locate the database server where the product data are, while the *Dialog* system [13] proposes to store this address on the product too, in the form of an universal resource identifier (URI).

Applications are mainly in supply-chain management. For instance, the product-centric approach [12], sees the product as a mean to easily integrate a changing network of enterprises. Classical integration technologies based on central databases fail to cope with frequent reconfigurations of supply chains, faced with interoperability, information structure mappings issues. Using product agents, the global interaction network between the partner of the supply chain can be discovered and updated continuously [23].

2.2 Architectural patterns

Object-oriented programming patterns have been applied to manage product life-cycle information throughout the supply chain [9]. Most often, the *Composite* and *Observer* (or Model-View-Controler) patterns are used, respectively to deals with aggregation of products' components, and to automatically notify many peers when the state of a product changes.

On the other hand, some patterns have also been proposed to determine the relationships between entities in the system. The simplest pattern is a master/slave coordination, where agents receive requests, that are broken down into sub-requests sent to slave agents. Likewise, reports are aggregated and sent back to the original request initiator. This pattern define a hierarchical structure (as in [1]) that enable to achieve high performance levels but that is often rigid and therefore not able to adapt to changing operation conditions. To solve some of this issues, unconstrained, dynamic or partial hierarchies have been proposed [4], but remains hard to implement.

Another approach is to use negotiation or auction-based patterns. These approaches are based on the emergence of a complex global behavior from simple local entities that interacts. Because only elementary behaviors are defined, such emerging systems are able to adapt easily to changes. However, their global behavior is hard to predict and misbehaviors such as famine, deadlocks and livelocks can degrade performance. The design of such system is often inspired by human or animal behavior. For instance the contract-net protocol [22] is one of the most widely used negotiation pattern, and has been industrially applied, for instance at Daimler [5].

Finally, bio-inspired systems, like swarm intelligence [10] are a novel approach to achieve a completely decentralized control while using very simple agents.These approaches, which take their roots in the biological study of self-organized behaviors in social insects, have inspired emerging research in manufacturing control. Their decentralized nature and adaptability to uncertain performing conditions are some of the key drivers to their application in manufacturing control. In particular, a swarming approach seems to be able to cope with the huge number of products and their limited computing and memorization capacities.

2.3 Stigmergy

Stigmergy is a concept that comes from the study of animal behavior (ethology), where it is defined as a class of mechanism that mediate animal-animal interactions, in particular for social insects. It has been first observed in the case of nest-constructing termites by Grassé [11], who coined the word stigmergy from the Greek roots *stigma* (mark, sign) and *ergon* (work). He observed that coordination between the numerous individuals implied in nest building was achieved using pheromones. Pheromones are chemical substances released by an insect that causes another individual of the same species to react. Termites release pheromones on the building material (soil pellets), thus influencing the behavior of other termites. The result of the process are the columns, vaults, etc, which will finally constitute a very complex nest (fig. 1).

Other studies on bees or wasps have shown that the current configuration of the construction can be the stimulus for the next task [24]. Nest-building behaviors have been modeled using micro-rules of the form "if *configuration* then *building decision*", where the configuration depends of the combs directly around the builder insect.

Using stigmergy, a global coordination of the complex process involving many entities is done without any global design, and only through indirect interactions through cues deposited on the environment, and a stimulus/reaction sets of behavioral rules. So stigmergy can be defined as a coordination mechanism of many workers by the mean of pieces of information deposited on their common work.

Fig. 1 "Cathedral" termite mount in the Northern Territory of Australia. Researchers have been amazed during a long time that such small and nearly blind insects are able to erect these huge and complex structures (photo by Andrew Finegan, available under a Creative Commons Attribution license)

The key point of stigmergy is that there is no *direct* communication between actors, only indirect communication through cues deposited on the environment. Thanks to this indirect communication mechanism, the communication abilities of the workers can be very simple (for example there is no need to route data to its receiver); likewise, the workers don't have to memorize information for later use : indeed, the environment itself retains the information about its completion, and the rules that determine the reaction of the workers to the stimuli of the environment are far more simpler than the collective work achieved.

One of the most famous application of stigmergy is the ant colony optimization algorithm (ACO) [7]. This algorithm comes from the modeling of the behavior of foraging ants. To find the shortest path between their ant-hill and a food source, ants release pheromones to mark their tracks. It has been shown that the pheromone path converges to the shortest path between the anthill and the food. Moreover, if an obstacle is added on the track, or if the food is moved, the system is able to adapt to the new situation. ACO has been used to solve traveling-salesman kind of optimization problems. It has also been transposed in the domain of manufacturing control [25].

In this last application, orders generate various kind of ants, who search the better route for their orders in the network of manufacturing resources, and release pheromones on the resources which correspond to their intension of using this resource sometime in the future (thus providing proactivity).

From the study of this proposition, it appears that a major question is how to project the stigmergic pattern to the control of an artificial system. Indeed, if ACO is

the source of inspiration, it seems natural to map ants to products and ant-tracks to products-routes. However, if the wider concept of stigmergy is considered, it seems natural to map the manufacturing transformation and control processes to worker insects, and the products to their environment, too. This point of view has a deep impact on the underlying architecture, especially on the role of the products.

3 Proposition

3.1 Stigmergy-based pattern

In this paper, stigmergy is applied to the manufacturing context by interpreting products as the *common work*, and operators, decision systems and processes as the *workers*. That differs significantly from approaches based on ant colony optimization which identify resource reservation as the common work (the path) and products as workers (ants).

According to this interpretation, a *stigmergic product* is defined as a physical object able to carry data (that will called annotations and that are the computer equivalent of pheromones). Likewise, *actor* are defined as any system that contributes to the elaboration of the product, either directly by transforming its morphology or its position (physical resources), or indirectly by producing and consuming control annotation attached to the products. Actors, is a very generic concept, that can includes a great variety of systems, such as for instance, observation devices, operators, decision and information systems, processes, etc (fig. 2).

The stigmergic product is closely related to the concept of holonic product, in the sense that a holonic product enables to see the physical part and the informational part as a complete object. The key architectural point is that there is a relation between each product and its annotations (and vice versa). The actual location of the data i.e. directly on a device carried by the product, or on a network emplacement such as a remote database does not concern us in this paper.

The stigmergic pattern is summarized by an UML class diagram presented figure 3.

Actors can interact with products by reading or writing their attributes. Depending of the kind of actor, three categories of attribute can be defined, corresponding to the product's present, future, and past:

- what the product currently is;
- the requirements that it will have to meet, from the point of view of physics (customer specifications), or of the point of view of control, such as a centralized decisions (e.g. a scheduled processing date);
- the history of what was made: this includes physical transformations (morphological and spacial changes), and control decisions, such as the results of distributed decision processes (e. g. a routing decision).

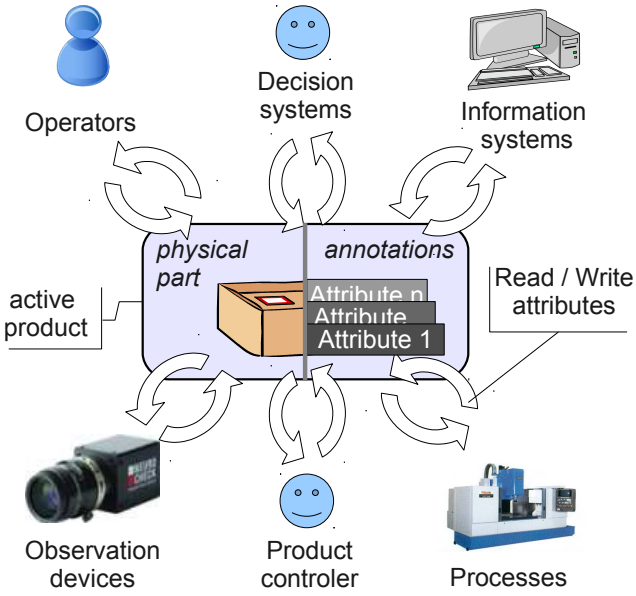

Fig. 2 The active product : a physical part carrying annotations, able to communicate with its environment. Various kind of actors interact with it by reading/writing its annotations

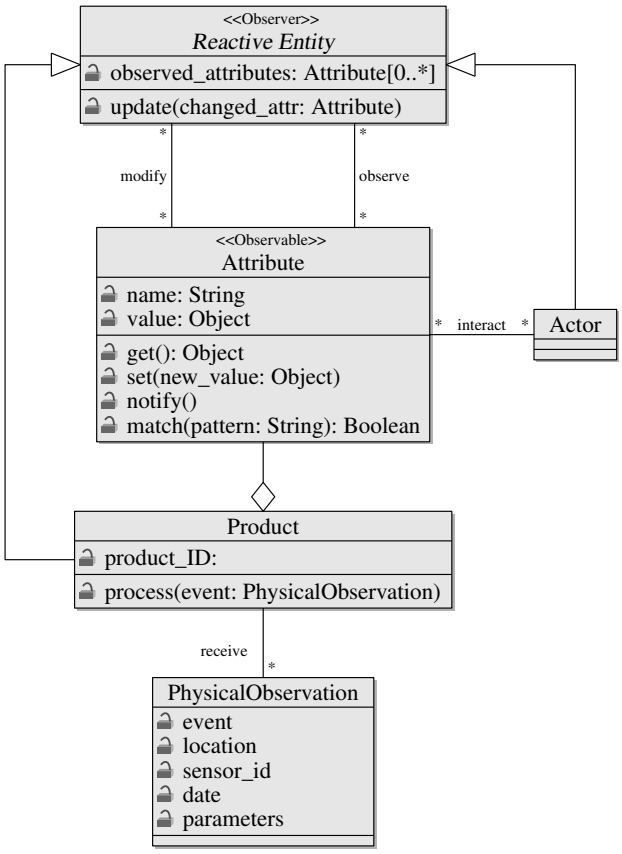

Fig. 3 UML Class Diagram of the proposed stigmergic pattern.

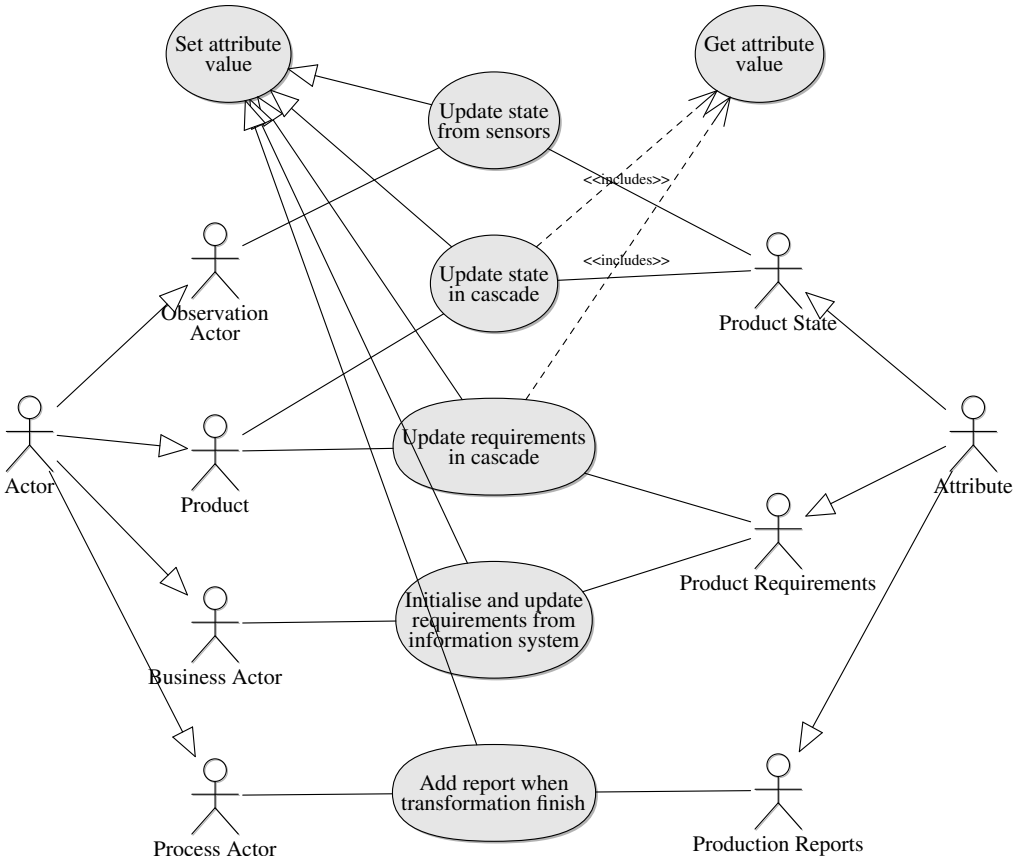

Fig. 4 UML use case diagram, focusing on attribute setting/updating functions. Interaction between actors and attribute types is highlighted in this diagram

Let these attribute classes be called respectively ProductState, ProductRequirement, and ProductionReport.

Moreover, it is possible to classify actors in two types : *Process Actors* that perform a physical transformation of the products (machining, transportation or assembly/disassembly stations), and *Business Actors* that focus on information- and decision-related activities. The role of business actors will be to initialize the product's requirements (what needs to be done) according to the customers requests, and to update them if needed. Process actors have to write on the product's report attributes, which may leads to changes in state attributes as explained below.

But beyond simple product-actor interaction, the key point of the proposed stigmergic architecture is to achieve coordination *between* actors, only by changing the attributes of the products and by reacting to such changes.

The specificity of our stigmergic approach lies (i) on the specific interpretation of actors and products attributes and (ii) on the way to achieve the coordination between actors only indirectly through products attributes. Two specific cases of this coordination can be highlighted:

- coordination between control actors. For instance, an operator can be informed of centrally-made schedule thanks to a priority attribute (as illustrated in section 5).
- coordination between a control actor and the observation system. Physical observation of the product (e. g. an event reporting that product p is arrived in station s) may trigger an attribute change, that can then be perceived by the corresponding control actor.

The product itself is able to observe its own attributes, and can react to changes of one of its attributes by modifying others. This feature enables the product to manage its own life-cycle, by setting the value of attributes describing its current state and requirements, according to the transformations made. For instance, a product may react to changes of several reports of physical transformation, by declaring itself to be quality compliant or not.

It is also possible for an aggregated product to react to notifications from its component by updating its own attribute. A simple instance of this cascading attribute update, is for instance updating the weight attribute of an assembly when the weight attribute of one of the assembly's component change.

All these interactions are summarized in figure 4.

3.2 Interactions with the stigmergic product

According to our interpretation of stigmergy, the product being the common work, actors interact with it by deposing their pheromones, that is to say by modifying their attributes. On the other hand, products emit these pheromones, that can be perceived by the actors in its environment.

The question is to specify the most efficient mechanism to implement these interactions. In social insects behavior, pheromones are deposed directly on the environment, at the location where they are needed (i.e. common work). Thus, an insect (worker) is instantly notified of any relevant information in its current location-context. This led appear two essential features of the interaction with pheromones. First, the worker get *only* information that are relevant in a particular context, second, the worker don't have the initiative of the communication.

In the proposed architecture, these features are provided by a subscription/notification mechanism. When a product is created, it broadcasts a list of its attributes to other actors. The actors can then respond by subscribing to some attributes. A subscription is a pair

$$(attribute\ name, pattern)$$

When the attribute's value changes and matches the pattern, the subscriber is notified. The pattern enables to filter out notifications that are not relevant for the actor.

This subscription-based mechanism can be implemented in various ways. For instance, in the object-oriented approach, the model-view-controller can provide a way to implement this notification mechanism. In this paper, the multi-agent paradigm have been used. This ensure a clear separation between entities (products and actors are modeled as separate agents) and impose to specify the content language and interaction protocols that are used for agents communication. Moreover, the multi-agent approach may be able to scale well to the size of industrial applications [19].

A content language has been defined to implement communication among agents. Figure 5 shows two examples of such messages, encoded in the FIPA-SL0[1] language [8]. Two main concepts have been used: the first one is the concept of 'attribute', seen as the association of a name and a value. Both name and value are strings. The second concept is 'holon', defined by its name (a string) and type (three types of holon have been defined: product, resource and staff).

Several predicates and actions can be built on these concepts. The predicates 'owns', 'has-value' and 'matches' describe respectively that a holon has a sequence of attributes, that a holon's attribute has a particular value, and that a holon attribute value matches a particular pattern (regular expression). The actions 'change-value' and 'query-value' are used to modify or query an attribute value.

```
((owns
   (holon
      :name p1-pla2
      :type product)
   (sequence
      epc color createBlocID state affectation assyDone)
))
```

a

```
((action
   (agent-identifier
      :name p1-pla2@tracilog)
   (change
      :attribute (
         attribute
            :name affectation
            :value p0))
))
```

b

Fig. 5 Example of an attribute declaration message (a) and of an attribute change request message (b) in the Fipa-SL0 language.

Several interaction protocols have also been used. The FIPA-subscribe implements the notification mechanism. First, product agents declare their attribute list, by broadcasting an INFORM message. Agents that want to be notified when an attribute value matches a pattern, initiate the subscription interaction with the product agent. Moreover, other request-like protocols (such as FIPA-Query and FIPA-Request) are used to query or modify products attributes.

3.3 Structure of a stigmergic product

A stigmergic product is represented as an agent, that is both connected to the physical product, and to other agents representing production *actors*. At the core of this agent is the product's attribute base (fig. 6).

Physical observations received by an agent result in modifications in the attribute base thanks to a sensor processing subsystem. Several processing approaches can be used: on the one hand, an attribute can take the direct value of the event name, or one of the event's parameters (e. g. the epc attribute is assigned after the epc parameter of a RFID-reading event). On the other hand, the attribute's value can be indirectly computed from the event. The most typical case is to use a finite-state machine for that task.

Product agents are able to communicate with other actors about attribute values, thanks to two subsystems. The first one notifies other agents of attribute changes and responds to queries about attribute value, while the

[1] FIPA (the Foundation for Intelligent Physical Agents) is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.

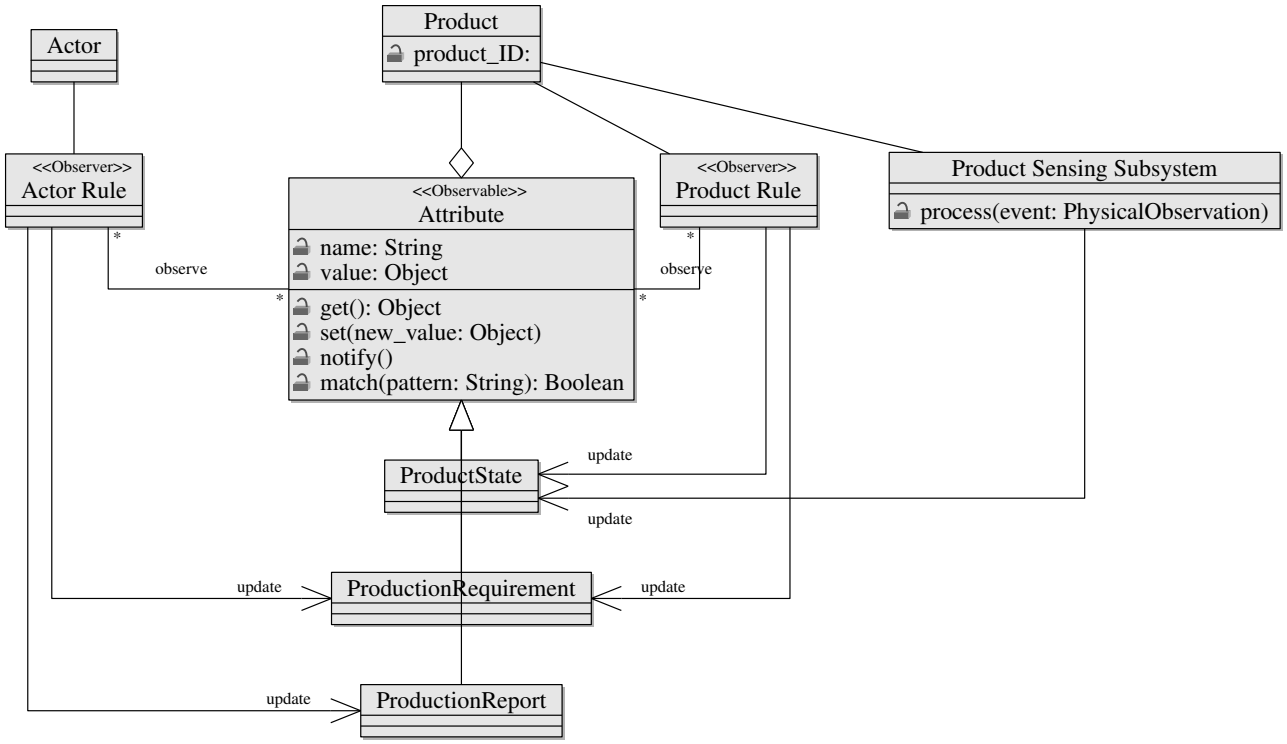

Fig. 7 UML class diagram showing the various types of attributes, and their interactions with various product's and actors' subsystems

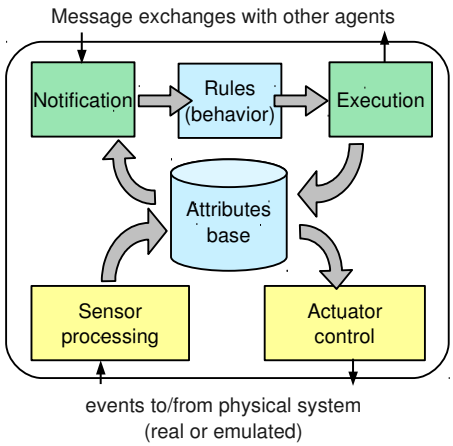

Fig. 6 Internal structure of a stigmergic physical agent

second one receives attribute change requests and implements them in the attribute base.

Finally, the product agent can execute a rule, that represents its behavior. The rule is notified of attribute change and can then modify local attributes.

Many components of this structure are generic and can be re-used. However, when developing a new application of a product-driven system, the only task is to develop the rule and to define the product attributes.

3.4 Architecture of a control system integrating stigmergic products

Product and actors agents are the main component of the proposed architecture. However, other types of agents are needed to integrate them into a real shop floor : they need to be able to communicate with the shop floor's hardware (processes, data acquisition systems, etc) and also with the enterprise's information system.

Therefore, stigmergic products must be included in a control architecture that comprises other agents in order provide them with these services (fig. 8). The system can be broken into three subsystems:

- a hardware layer, with agents responsible of communication with shop floor devices (programmable logic controller, RFID readers, vision systems, etc...);
- a production control layer, with a set of product agents, communicating with other control agents (process and business agents) thanks to the notification mechanism presented above;
- an information system layer, with agents responsible with communication with enterprise applications such as enterprise resource planing, product life-cycle management, supply chain management, etc.

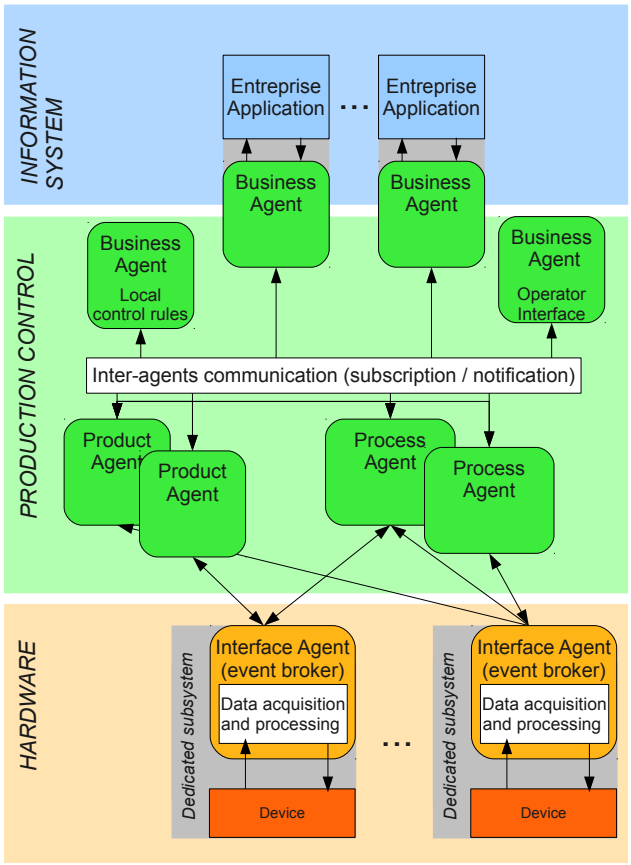

Fig. 8 Agent architecture that integrates stigmergic products

4 Implementation of the pattern

4.1 Material and methods

The proposed architecture have been implemented in java using the FIPA-compliant JADE agent environment [3]. The implementation is broken into two parts:

- a generic library that comprises the generic structure of agents,
- a set of configuration files and classes, specific to a case, that constitutes the agents' configuration and behavior rules.

The *Tracilogis*[2] platform has been used for this last point. Using this platform it is possible to illustrate how the stigmergic pattern can be applied on a concrete case. While being simple, this platform nevertheless includes an assembly station, one transformation and several routing stations (fig. 9). Therefore, it comprises several different situations that will be used to illustrate how the stigmergic products interact in various situations.

The products are composed of:

- a support;

[2] The Tracilogis platform is located on the "campus fiber" at Épinal, France.

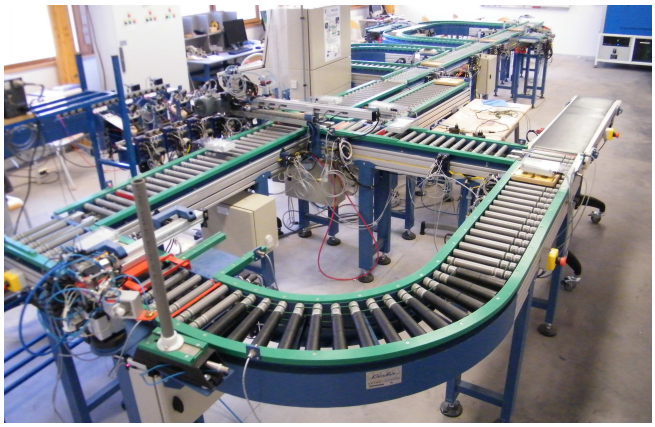

Fig. 9 Tracilogis platform, with the assembly cell in front

- up to four wooden square pieces, yellow or orange;
- up to four wooden chips.

This product structure generates more than one thousand different products, providing sufficient variety. Figure 10 shows some examples of assembled products.

Fig. 10 Examples of products on the Tracilogis platform

The experimental platform has four stations linked to a conveyor belt (fig. 11) : M_1 is an assembling station, where square pieces are assembled on support; M_2 is a transformation station, this transformation being materialized by putting some chips on the product; A_1 and A_2 are routing stations.

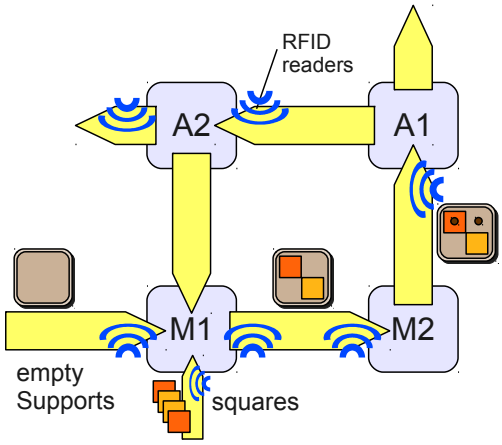

Fig. 11 A overview of the material flow

The product components are equipped with identification devices. The supports and square pieces carry a RFID tag. RFID readers are placed in front of each station, and additional readers enable to check products integrity. The color of the square pieces is deduced from reading their tag.

The production scenario is to respond to customer requests as fast as possible. Due to the high number of products configurations, the control system must be able to cope with mixed flows. Moreover, the control system must also deals with perturbations, like last minute order modification, resource breakdown or uncontrollable change of the flow (e.g. when a product is taken from the conveyor belt and put back later elsewhere on the belt).

4.2 Interactions between customers, products and processes

The test-case system is controlled using a multi-agent system. Agents have been associated to resources (M_1, M_2, A_1 and A_2), and to products (both the support and the square pieces). Business agents are present in the system to enable customers to configure and generate production orders (i.e. product agents). Finally, the platform comprises additional agents, which are necessary to communicate with the physical equipments (the programmable logic controller and the RFID subsystems), or to offer common services (agent creation, directory facilitators, ...). The typical number of agents cooperating is between 10 and 50.

A typical interaction scenario corresponds to the following sequence of messages. First, products are configured from the business point of view and are then introduced on the multi-agent system (fig. 12).

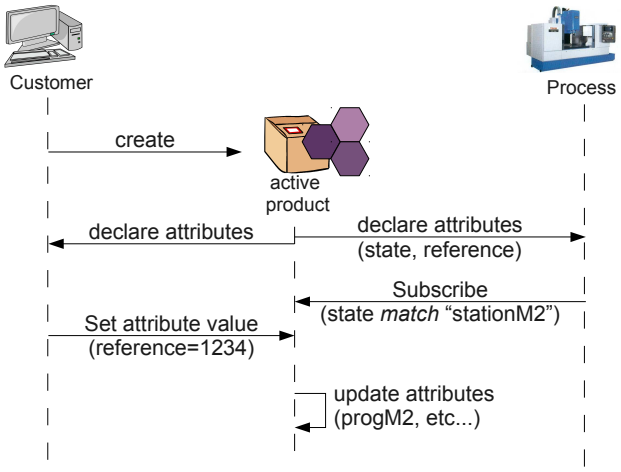

Fig. 12 Sequence diagram showing the initialization of a stigmergic product agent

1. The customer-agent creates a new product-agent and waits for agent initialization, then sets the 'reference' attribute of the agent.
2. The product agent declares its attributes. Actors subscribes to some to them. For instance, M_2 subscribes to the attribute 'state', with value 'stationM$_2$'.
3. The newly created product agent reacts to the change of its reference attribute, initializes its internal bill of operations and set it progM$_1$, progM$_2$,... attributes accordingly.
4. A physical product is associated with the product agent by scanning its RFID tag. An association between the product ID and the epc code of the tag is created, enabling event brokers to route RFID event to the right product agent.

Then, products agents interact with the process, until production finishes (fig. 13).

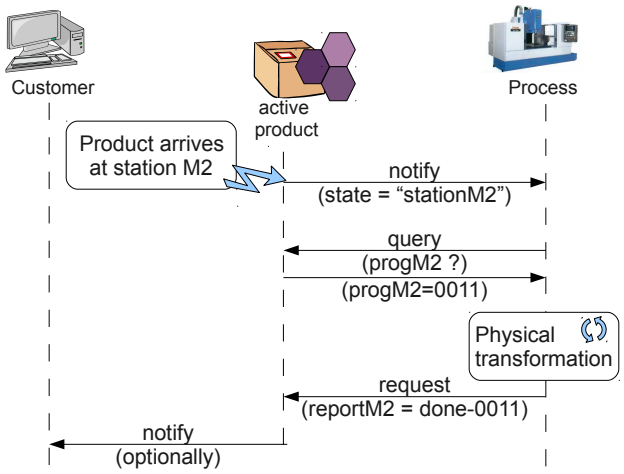

Fig. 13 Sequence diagram between the active product and the process, corresponding to the typical interaction occurring during transformation

5. When the physical product arrives in a station (e. g. in M_2), a RFID event is received by the product agent, and the 'state' attribute is consequently changed (e. g. 'stationM$_2$')
6. The station's control agent is notified of the new value of the product's state attribute. Consequently, it queries the product about its requirements (e. g. attribute progM$_2$), and transforms the physical product accordingly.
7. When the transformation ends, the station control agent releases the product, and requests to change a report attribute (e. g. reportM$_2$). The product agent reacts to this change by updating its internal bill of operation and consequently modifying attribute value.

This approach based on stigmergy enables a high product variety (the flow can mix any number of differ-

ent products), and also a high robustness (products are able to interact with any resource at any time, allowing them to find back their normal route). Comparatively, an existing traditional implementation, developed by an independent engineering company, allowed only to produce one type of product at a time, and was not able to cope with flow disturbances.

The implementation of this system is a relatively simple task, because only the behavior rules and the attribute base have to be implemented. Indeed, most of the components are generics and may be re-used.

The most important feature of the system is the high functional independence between products requirements definition, products life-cycle management and product transformation resources control. Since interactions are based on product attributes, the system can also cope with a completely asynchronous execution of all these tasks.

4.3 Special case of assembly

The assembly between a support an square pieces is a special case, because both types of pieces are controlled by a product-agent. Therefore, three agents must interact here: the product-agent associated with the support (S), the product agents associated with a square piece (Sq), and the resource agent controlling manipulator M_1. In this situation, the stigmergic pattern can be applied at a smaller scale (fig. 14).

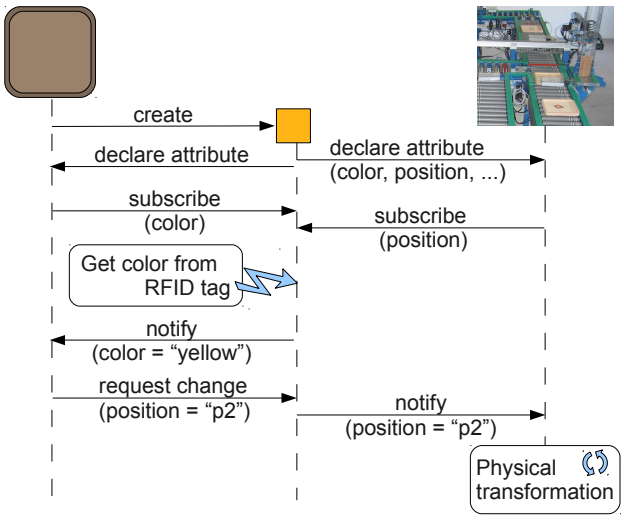

Fig. 14 Sequence diagram between a support agent ('p2'), a square piece agent ('p2-pla1') and assembly station M_1.

Agent S creates agent Sq, gets its 'color' attribute, decides to assemble it at some position or to discard it, and sets its 'destination' attribute accordingly. Agent M_1 is notified of the 'destination' attribute of Sq, moves the square piece, and finally reports this action. Cooperation between agents M_1 and S has been made through the attribute of agent Sq.

5 Application

5.1 Case presentation

In this section, the proposed stigmergic architecture is applied to an industrial case study. In this case-study, the key performance indicators pertain to production control : inventory levels, productivity, etc. Therefore, the main issue here is scheduling, and this section present an application of the proposed stigmergic pattern to this particular issue. More precisely, stigmergic products are used to solve the issue of a centralized vs. distributed control approach, which use algorithms already presented in [21].

However, the proposed pattern could have been used in other situations, where scheduling is not the key problem to solve.

The case study is an automotive-industry subcontractor shop floor. The assembly site can produce up to 10,000 products a day, with hundreds of different part numbers. The factory is actually divided into several production cells, each including every production step to make a finished product from raw material, and is dedicated to a particular client. One of these cells has been modeled, using emulation software.

As shown on figure 15 the production process is divided into two stages. A first set of operations on the first line results in semi-finished products, which are further assembled on three independent assembly cells. It is assumed that all assembly cells cannot work simultaneously on the same reference. The production module therefore includes, in addition to the four cells, an inventory storing semi-finished products.

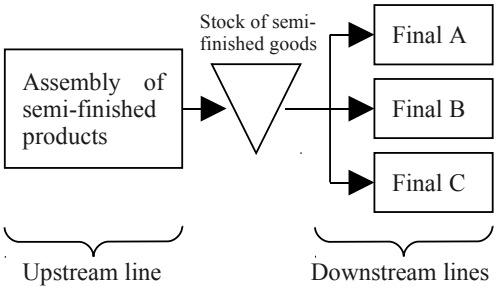

Fig. 15 Material flow in a production cell

As downstream lines can consume three kinds of products simultaneously, the major decision problem is to choose whether and when to change the setup on the upstream line. This decision aims at minimizing the number of setups (each setup takes 30 minutes), while also

minimizing work-in-process levels and avoiding starvation on downstream lines. Moreover, business goals such as due dates must also be taken into account.

To solve this control problem, several strategies can be used. In the centralized approach, the problem is modeled and solved off-line. The optimal solution is then implemented. In the distributed approach, local control entities can decide to make a setup when needed. Their behavior is modeled as a set of heuristic rules. The proposed product-driven-system tries to integrate these two relatively antagonistic approaches.

5.2 Decision-making procedures

In the developed system (fig. 6) the centralized control is represented by an agent, as well as the control entities for both upstream and downstream lines. Products and resources are controlled by their respective agents and are interfaced with the physical system thanks to event brokers agents. Figure 16 summarize this architecture.

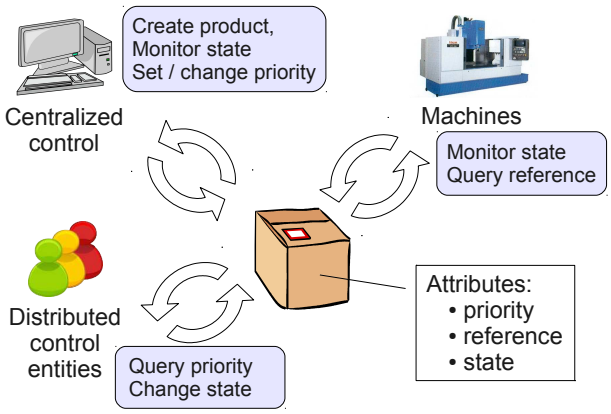

Fig. 16 An overview of the control architecture. There is no direct interactions between the actors, every interaction is done through products attributes.

One of the main product attributes is *priority*. At system initialization, the centralized control system schedules jobs according to their critical ratio, defined as the ratio between the processing time required to complete it and the available time before its due date. Products priorities are then assigned according to the schedule.

The distributed control centers are notified by the products of the value of their priority attribute. According to these notifications, they maintain the ordered sequence of products to process. Two modes of product-driven control have been programmed, according to the way this ordering is done.

In the simpler one, local control decisions are based on products priority *only*: When a cell is ready to operate, it scans through the products waiting and selects the one with the highest priority. This control algorithm is called "product-driven constrained" because it depends on the centrally made schedule.

In the other one, a more complicated decision algorithm is used, allowing autonomy with respect to product data. This algorithm tries to mimic the behavior of an operator, taking into account not only product priorities, but also other local parameters, such as levels of semi-finished inventory in order to avoid starvation downstream, or the amount of products of the same reference that have been done, to minimize setups. This control algorithm is called "product-driven relaxed".

Other attributes of products are their reference, used to setup the transformation resources, and their state, that tracks the execution level of the routing.

5.3 Results and discussion

The first experiments have been run without introducing perturbations. The performance of the system under the control of the various control system is shown figure 17. The main result here is that the stigmergic approach can reproduce exactly the behavior of the centralized system. Indeed, under the constrained product-driven control system, the centralized schedule carried by the products have been precisely followed. One of the slight differences that can be observed are the setup times: the product driven control system was not able to anticipate the setups, because it was triggered by the arrival of a product with a new reference.

The relaxed product-driven system, where products priorities were not the only decision factor, display a different behaviors, but yield to comparable performances in terms of make-span.

Both product-driven control modes have been tested in various perturbation situations. Moreover, the centralized control mode (where the centralized schedule is applied unmodified) has been added, as a reference for comparison. Figure 18 give some results of the experimental study.

These results show that the stigmergic approach enables combining the good performances of centralized control in nominal situations with the adaptability and robustness of distributed control.

It is noteworthy that these experiments have demonstrated that local decision-making and its accuracy are crucial in a product-driven environment. Indeed, without autonomous decisions, performance is similar to that of centralized control, when there is no perturbation, but performance tends to plunge in a highly perturbed environment. In contrast, with autonomous decisions, product-driven control was slightly less successful in non-perturbed situations, but showed more robustness.

However, the questions related to design the stimulus/reaction behavior of the decision systems require still a lot of investigation.

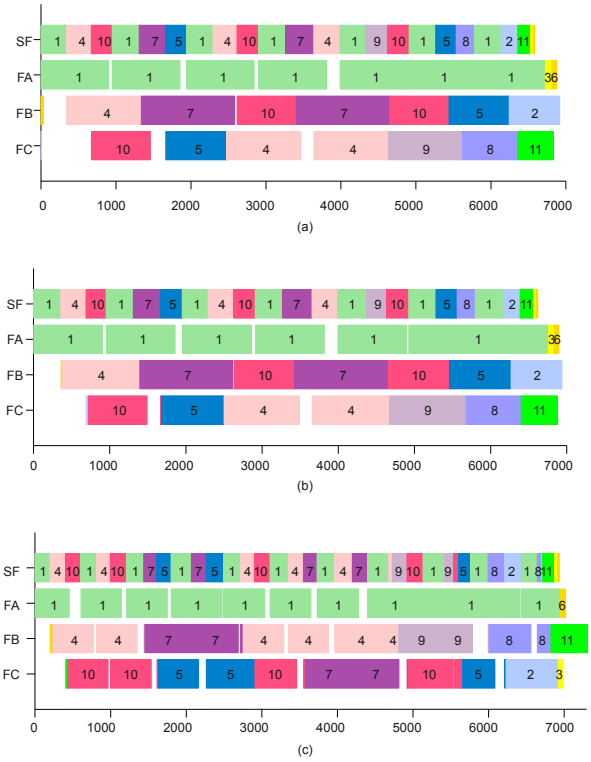

Fig. 17 Gantt charts showing occupation of semi-finished, final A, B and C lines (labeled SF, FA, FB and FC) for each decision modes, in an unperturbed situation: the centralized reference control (a), the constrained product-driven control (b) and the relaxed product-driven control (c) [21]

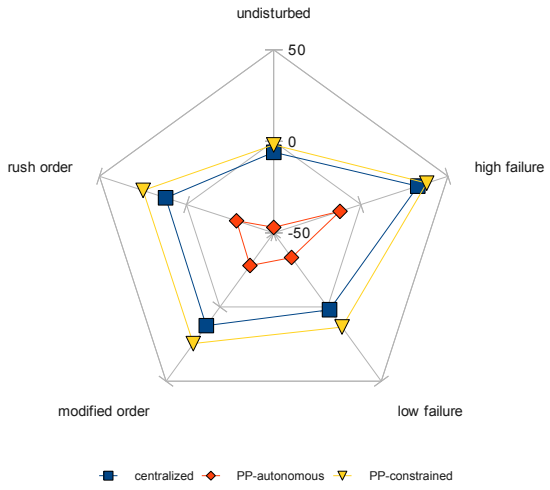

Fig. 18 Comparison of the various control scenarios: effect on WIP level of the centralized reference control (blue square), the constrained product-driven control (yellow triangle) and the relaxed product-driven control (red diamond)

6 Conclusion

In this article, an architectural pattern based on stigmergy have been proposed to be applied to the design of product-driven systems. According to our interpretation of stigmergy, products able to carry information (in the forms of a set of attributes) and able to take actors of their environment informed about these attributes values, play a central role. The set of stigmergic products constitute the basis of an ambient control architecture, where information carried by the products can be shared by all the actors of production, thus creating an implicit integration, while at the same time, preserving a kind of autonomy. For instance, this approach enable to mix different decision-making approaches, which enable to take advantage of both centralized and distributed decision. In the future, this approach might also be used in a larger scope, to implement cooperation amongst all the product life-cycle actors, such as design, sales and maintenance.

The originality of the stigmergic architecture proposed in this paper, is the role given to the active product: It is a trade-of between a completely passive role, where every information is stored elsewhere than on the product, and a "super-decisional" product, which is responsible of the control of the whole operational system. If the first approach is often very complex and rigid, the second one can be technically infeasible, for instance because of the need of negotiation between products. The stigmergic product can take decision only about its own life-cycle, and must notify its subscribers of change in its attribute base.

This approach have been implemented in an experimental platform, and applied on an industrial test case. It is planed to apply it to larger industrial-scale emulated test-cases (with ten of thousand of agents) to see how the system perform in very large scale systems.

In another perspective, it is planed to develop and study new features, such as interactions between products or an evaporation/reinforcement system for product attributes. Another important perspective is the integration of our approach in software-engineering tools, in order to apply a model-driven design of products attributes and behaviors.

Acknowledgements The authors gratefully acknowledge the financial support of the CPER 2007-2013 "Structuration du Pôle de Compétitivité Fibres Grand'Est" (Competitiveness Fibre Cluster), through local (Conseil Général des Vosges), regional (Région Lorraine), national (DRRT and FNADT) and European (FEDER) funds.

References

1. J. Albus and A. Barbera. Rcs: A cognitive architecture for intelligent multi-agent systems. *Annual Reviews in Control*, 29(1):87–99, 2005.

2. R. Babiceanu and F. Chen. Development and applications of holonic manufacturing systems: A survey. *Journal of Intelligent Manufacturing*, 17(1):111–131, February 2006.
3. F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, April 2007.
4. R. Brennan and D. Norrie. Evaluating the performance of reactive control architectures for manufacturing production control. *Computers in Industry*, 46(3):235–245, October 2001.
5. S. Bussmann and K. Schild. An agent-based approach to the control of flexible production systems. In *Proc. of the 8th IEEE Int. Conf. on Emergent Technologies and Factory Automation (ETFA 2001)*, pages 481–488, Antibes Juan-les-pins, France, 2001.
6. M. Christopher. The agile supply chain : Competing in volatile markets. *Industrial Marketing Management*, 29(1):37–44, 2000.
7. M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.
8. FIPA. The foundation for intelligent physical agents. http:/www.fipa.org/repository/index.html, 2005.
9. K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström. Design patterns for managing product life cycle information. *Commun. ACM*, 50:75–79, June 2007.
10. S. Garnier, J. Gautrais, and G. Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31, 2007.
11. P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation des termites constructeurs. *Insectes Sociaux*, 6:41–84, 1959.
12. J. Holmstrom and K. Framling. Product centric integration:exploring the impact of rfid and agent technology on supply chain management. In *Network-Centric Collaboration and Supporting Frameworks*, volume 224 of *IFIP International Federation for Information Processing*, pages 565–572. Springer Boston, 2006. 10.1007/978-0-387-38269-2_59.
13. M. Kärkkäinen, T. Ala-Risku, and K. Främling. The product centric approach: a solution to supply network information management problems? *Computers in Industry*, 52(2):147–159, 2003.
14. T. Klein, A. Thomas, G. Morel, and H. El Haouzi. A simulation-based decision support system: The example of a furniture manufacturer. In *17th IFAC World Congress 17th IFAC World Congress*, pages 562–568, Seoul Corée, République De, 07 2008.
15. P. Leitao and F. Restivo. Adacor: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, 57(2):121–130, February 2006.
16. F. Maturana and D. Norrie. Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*, 7:257–270, 1996. 10.1007/BF00124828.
17. D. Mcfarlane, S. Sarma, J. L. Chirn, C. Y. Wong, and K. Ashton. Auto id systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence*, 16(4):365–376, June 2003.
18. G. Meyer, K. Främling, and Holmström; J. Intelligent products: A survey. *Computers in Industry*, 60(3):137 – 148, 2009.
19. B. Montreuil, E. Brotherton, Ouazzani N., and M. Nourelfath. Antzon layout metaheuristic: coupling zone-based layout optimization, ant colony system and domain knowledge. In *Progress in Material Handling Research*, pages 301–331. Material Handling Industry of America, 2004.
20. G. Morel, P. Valckenaers, J.-M. Faure, C. E. Pereira, and C. Dietrich. Manufacturing plan control challenges and isssues. *Control Engineering Pratice*, 15(11):1321–1331, 2007.
21. R. Pannequin, G. Morel, and A. Thomas. The performance of product-driven manufacturing control: An emulation-based benchmarking study. *Computers in Industry*, 60(3):195–203, 2009.
22. R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.
23. M. Stuit and G. Meyer. Agent interaction modeling based on product-centric data: A formal method to improve enterprise interoperability. In Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Klaus Fischer, Jörg P. Müller, James Odell, and Arne Jørgen Berre, editors, *Agent-Based Technologies and Applications for Enterprise Interoperability*, volume 25 of *Lecture Notes in Business Information Processing*, pages 197–219. Springer Berlin Heidelberg, 2009. 10.1007/978-3-642-01668-4_11.
24. G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5(2):97–116, 1999.
25. P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, and H. Van Brussel. Emergent short-term forecasting through ant colony engineering in coordination and control systems. *Advanced Engineering Informatics*, 20(3):261–278, July 2006.
26. H. van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems: Prosa. *Computers in Industry*, 37(3):255–274, December 1998.