MULTI-AGENT MANUFACTURING CONTROL USING STIGMERGY

Paul Valckenaers, Hendrik Van Brussel, Hadeli, Martin Kollingbaum, and Olaf Bochmann

K.U.Leuven – P.M.A.,
Celestijnenlaan 300B, B-3001 Leuven, Belgium
Paul.Valckenaers@mech.kuleuven.ac.be

Abstract: This paper discusses a multi-agent coordination and control system design, inspired by the behaviour of social insects. This design makes desirable overall system behaviour emerge without exposing individual agents to the complexity and the dynamics of the overall system. This enables these individual agents to survive changes without maintenance, it allows individual agents to be re-usable across systems, and it allows having the emergent behaviour handle disturbances. The paper starts with the biological concept, stigmergy, that constitutes the basis of the coordination and control system. Next, it discusses the different steps in the development of a coordination and control system. Copyright © 2002 IFAC.

Keywords: Agents, control systems design, co-operation, co-ordination, intelligent manufacturing systems, manufacturing systems.

1. INTRODUCTION

Manufacturing control manages the internal logistics in a production system; it decides about the routings of product instances, the assignment of workers and raw material or components, and the starting of the processes on semi-finished products. Manufacturing control software developers face harsh challenges posed by the changes and disturbances occurring in manufacturing systems and their environment.

This paper presents the design of a multi-agent coordination and control system inspired by the behaviour of social insects. The design facilitates the emergence of desirable overall system behaviour. It achieves this without exposing the individual agents, which constitute the control system, to the complexity and the dynamics of the overall system. This enables these individual agents to survive changes without maintenance (= live longer), and it allows individual agents to be re-usable across systems. Moreover, the emergent behaviour handles disturbances by considering them business-as-usual.

The paper starts with a biological concept, stigmergy, that constitutes the basis of the coordination and control system. Next, it discusses the different steps in the development of a coordination and control system:

1. Agentify the environment to make it part of the solution.
2. Implement the control layers that make necessary and useful information available to the agents that take the decisions.
3. Implement the decision taking mechanisms at the highest level in the control system, such that changing and adapting these rules becomes very easy.

The research, which this paper discusses, focuses on manufacturing control. Nonetheless, the approach remains applicable to the coordination and control of other types of ironware systems (e.g. traffic systems).
Food foraging ants execute a simple procedure:

- In absence of any signs in the environment (consisting of scents from a pheromone), ants perform a randomised search for food.
- When an ant discovers a food source, it drops a smelling chemical substance — i.e. pheromone — on its way back to the nest while carrying some of the food. Thus, it creates a pheromone trail between nest and food source. An important property of such pheromone trail is that it will evaporate if none of the ants deposes fresh pheromones.
- When an ant senses a pheromone trail, it will be urged by its instinct to follow this trail to the food source. Note that a scent strength gradient indicates the direction toward the food. The ant’s behaviour is probabilistic: there is a high probability that it follows the trail, but no certainty.
- When this ant arrives at the food source, it will return with food, while deposing more pheromones. In this manner, the strength of the pheromone trail is maintained and even reinforced. When the ant finds an exhausted food source, it starts a randomised search for a new food source and the trail disappears because of the evaporation.

The above scheme results in an emergent overall behaviour for the colony that is highly ordered and very effective at foraging food. At the same time, it is robust against the uncertainty and complexity posed by the environment. An important capability of this type of stigmergy is that global information — about where to find food in a remote location — is made available locally — in which direction must the ant move to get to this food.

The main achievement is that individual ants are not exposed to the complexity and dynamics of the situation. Instead, the environment is incorporated into the solution and allows the overall system to cope with its complexity; none of the ants needs a mental map of the environment. Similarly, the evaporation and refreshing of the pheromone trails allows the ants to cope with the dynamics of the environment; there is no information in the head of the ants that must be kept in sync with reality.

The remainder of this paper discusses how the above advantages can be achieved in multi-agent manufacturing control. Note that it is important to apply the insights rather than to attempt to copy nature’s solution; history reveals that it was a bad idea to design an airplane flapping its wings.

3. AGENTIFY THE ENVIRONMENT

To make the environment part of the solution, the first step in the development of an ant-based control system is to agentify the environment. Indeed, computing agents have no direct access to the underlying production system. Therefore, agents representing the entities in the physical world, with appropriate connections, are required. To this end, the research applies the PROSA architecture (Van Brussel, 1998). The acronym stands for Product-Resource-Order-Staff Architecture. The next section gives an overview of this architecture. Then, the paper discusses how resource agents make the environment a part of the solution by means of an example.

3.1 PROSA

The PROSA architecture is built around three types of basic agents: order, product and resource agents. Each of them is responsible, respectively, for one aspect of manufacturing control: (i) internal logistics, (ii) recipes or process plans, and (iii) resource handling. These basic agents are structured using object-oriented concepts like aggregation and specialization. Staff agents can be added to assist the basic agents with expert knowledge.

Each resource agent corresponds to a production resource in the manufacturing system and contains an information processing part that controls the resource. In object-oriented terminology, each resource agent reflects a physical resource, is able to drive this resource, and keeps itself in sync with the resource status.

Each product agent owns a “product model” of a product type — not the “product state model” of one physical product instance being produced. A product agent acts as an information server to the other agents, delivering the right recipes in the right place.

Each order agent represents a task. It is responsible for performing the corresponding work correctly and on time. It manages the physical product(s) being produced, the product state model, and all logistic information processing related to the job. An order agent may represent rush orders, “first-of” orders, maintenance/repair orders resources, etc.

The staff agent mirrors the difference between line functions and staff functions in human organizations.
In a human organization one of the main goals for the introduction of staff functions is to reduce the workload and complexity of line functions (or operational processes) by providing them with expert knowledge. Accordingly, staff agents provide the basic agents with information such that they can take better decisions. The basic agents are responsible for taking the decisions; the staff agents are external experts giving advice.

3.2 Resource Agents

In an ant-based system, the environment is part of the solution: for every resource in the factory, a resource agent is created. This agent is knowledgeable about its corresponding resource and keeps itself in sync with reality — i.e. the resource agent observes and tracks the state of the corresponding resource. In addition, the resource agents offer to the other agents some data storage spaces on which these agents can put, modify and retrieve information (pheromones). Such information will have a time-bound lifespan (evaporation).

Importantly, the exposure of such resource agents is limited to the corresponding resource. As a consequence, the agent's software code can be reused wherever such a resource resides. It is exactly this property that makes the ant-based control system designs attractive and worthwhile. The following illustrates this development step with the discussion of a sample resource agent: a conveyor agent.

A conveyor agent reflects an accumulating conveyor belt on which products are transported in a single direction and are unable to overtake each other. It is a first-in, first-out device with a finite capacity. The following paragraphs discuss the functionality supported by this conveyor belt agent.

Attributes. The conveyor belt agent provides a collection of local blackboards. These blackboards correspond to the physical places on which the ants put the pheromones. Agents write, read and remove information from these blackboards. The infrastructure supports evaporation over time for information on the blackboards. In addition, the resource agent has attributes to model its state. The sample conveyor agent provides the following attributes:

- Blackboard connected to the belt entrance
- Blackboard connected to the belt exit
- Blackboard connected to the middle section
- State vector including references to the entities on the belt
- Graph supporting (virtual) navigation across the conveyor including references to the entities connected to respectively the conveyor entry and exit
- …

Life cycle support. It is important to cover the full life cycle if the manufacturing control system has to cope with changes; with incomplete support the system will have time windows in which it will be extremely vulnerable. Access to these life cycle support functions is subject to proper authorization. The sample conveyor belt agent supports the following:

- Installation in the factory (creation)
- Removal from the factory (destruction)
- Connecting the entrance to a resource
- Connecting the exit to a resource
- Disconnecting the entrance
- Disconnecting the exit point
- Updating the state vector to synchronize with reality
- …

State observers. Resource agents support methods to observe their state. As a rule, it is preferable to offer observation methods over direct access to attributes; the method can control access to the attributes more effectively and it encapsulates the implementation, which may need to change later. Sample observation methods offered by the sample conveyor agent are:

- Query the current load
- Query belt speed
- …

Event processing and notification. An agent is an active entity, capable of triggering actions. For instance, the conveyor agent will notify the corresponding order agent when a product instance reaches the conveyor exit. Conversely, the conveyor agent will process the arrival event of such a product at its entry, possibly informing the corresponding order agent that the conveyor is full and that the product is unable to enter.

3.3 Intelligent Resource Agents

The system development approach in this paper insists that agents have limited exposure to the complexity and the dynamics of the situation. This will prevent resource agents from taking up certain responsibilities. E.g. the conveyor agent must not manage the product flow on top of itself since it would become exposed to the specifics of the factory in which this conveyor resides (and worse). However, this does not prevent the development of resource agents with increasing levels of capabilities — intelligence — within their own scope. In the example, conveyor agents support what-if analysis; they mirror the behaviour of their physical resource in a virtual world.

What-If Analysis. The resource agent has a moderate level of intelligence, in contrast to the passive environment of the ants. The resource agent provides reflection functionality — self-knowledge and self-modelling — that enables other agents to cooperate with this agent during what-if analysis. To this end, the agent supports the following methods:
- **PropagateUpstream** will move an (mobile) agent from the exit to the entrance and adjust time information, which is presented to this conveyor agent, to reflect the expected transportation delay on the belt; time data will indicate earlier times at the entrance. The agent is then directed to the resource connected to the entrance (if any).

- **PropagateDownstream** will move an agent from the entrance to the exit and adjust time information, which is presented to the conveyor agent, to reflect the expected transportation delay on the belt; time data will indicate later times at the exit. The agent is then directed to the resource connected to the exit (if any).

- **GiveLoadForecast** will generate a (short-term) forecast of the resource usage or loading based on intentions that have been communicated by prospective users; these users communicate their intentions by means of the two previous functions, indicating their level of commitment through the parameter settings.

**Remark.** Note how a resource agent remains fully functional as long as he reflects the corresponding physical resource and as long as his internal representation of natural laws (time, space, probability…) stays valid. Beyond that, the agent has minimal exposure — with the exception of superficial matters (e.g. syntax issues) for which technical solutions exist. Consequently, resource agents are likely to be long-lived, capable of evolving with the corresponding resource, maximize their number of possible users, capable of becoming more intelligent in their own domain, and capable of surviving with little maintenance.

Other resource agents mirror switching elements in the transport system, storage and retrieval subsystems, processing units… Together, these agents provide an infrastructure on which the order agents place and observe information to coordinate their activities and optimise overall behaviour. The next sections discuss how the order agents use this infrastructure.

4. **CONTROL LAYERS**

The multi-agent control systems, on which this paper reports, have a layered design. In each layer, information is made available and/or exchanged to enable the order agents to make proper choices regarding the internal logistics of the manufacturing system. The order agents combine the information provided by the layers to make their decisions. In principle, the information, which the layers provide, consists of facts rather than decisions; note however that intentions of agents are considered to be facts. The layers are independent. The order agents select information from a subset of the available ones, in function of their situation, goals and capabilities. The presence of an unused layer only consumes processing power and communication bandwidth. The absence of a layer will degrade the performance of the agents that need it. This loose coupling is typical for a design based on stigmergy.

These control layers belong to one of three different categories. The first type of layers addresses feasibility. They deliver information that cannot be ignored because it reflects hard constraints. These layers reflect, among others, where which processing steps can be performed. The second type of layers provides information to operate the system smoothly and with good performance. These layers provide, among others, information about resource loads and availability. This information is needed to operate the system well. The third type of layers corresponds to the staff agents in the PROSA architecture. These layers typically provide advisory information to guide the search process of the order agents.

Due to space limitations, only one layer can be discussed. The paper presents a layer that makes global information locally available. It shows how the flexibility of local dispatching mechanisms can be combined with system-wide coordination. The following paragraphs discuss the design of an emergent load-forecasting control layer.

4.1 **Intention-based Forecasting**

The agents forecast the behaviour of the system by combining the intentions of the order agents with the self-modelling of the resource agents.

**Propagation of intentions.** Figure 1 shows how an order agent propagates his intentions downstream. First, the order agent, who stays together with the physical semi-finished product, creates a mobile agent who will represent the order in a what-if mode. This mobile agent — the ant in figure 1 — will move ahead through the production system in a virtual manner, and will make the order agent's intentions known wherever it visits a resource. The mobile agent uses the **PropagateDownstream** function of the resources on which he travels. This enables the mobile agent to predict arrival times with minimal knowledge about the system through which he travels (minimal exposure). The mobile agent informs the resource agent about the commitment level for the order agent's intentions.

Whenever the mobile agent reaches a decision point, it executes the order agent's decision mechanism. During this execution, the mobile agent uses the forecast information at a time corresponding to his expected arrival. Since the mechanism presented in this paragraph constructs short-term forecasts and makes them available locally, it is safe to assume that forecasts are available on the local blackboards of the resource agents. Because of its encapsulation, no software maintenance is necessary for the mobile agent if this decision mechanism changes.
The system evaporates the information in a binary manner: after a timeout, it is discarded completely. This means that an agent can change his intentions without worrying about leaving stale data about previous intentions around. In practice, the mobile agent may explicitly wipe out stale data to make the new situation known faster; however, it suffices to use a conservative wiping mechanism that covers most of the ground most of the time. Order agents create enough mobile agents to ensure the refresh.

Eventually, the mobile agent reaches a processing unit and makes the order agent's intentions known. Figure 1 shows how the intentions change when they move through the system. The order agent intends to start the next processing step as soon as possible (i.e. now). When the mobile agent propagates this, the expected transportation times are added. At arrival on the processing unit, the corresponding resource agent knows when processing can start. In addition, the mobile agent establishes a link between this resource agent and its product agent such that the expected processing time can be calculated. This information allows the resource agent to update the local load profile forecast (step 4 in figure 1). Indeed, by combining the intentions, which various mobile agents communicate to the resource agent, a local short-term load forecast for the processing unit can be constructed.

**Back-propagation of load forecasts.** Similar to the order agents, resource agents create mobile agents; normally, this function is only activated for the more critical resources. These mobile agents propagate the local load forecasts upstream through the system. The mobile agents, created by resource agents, leave the corresponding resource through the entrance (in a virtual manner) and use the PropagateUpstream method from resource agents to travel backwards through the factory. The mobile agents take the local load forecast for their resource with them and deposit copies of it on the appropriate blackboards.

These local forecasts have their time information adapted when they are propagated upstream. Figure 2 shows how time information is made earlier in the load forecast by the amount $\Delta t_{B4}$. When the local load forecast is propagated upstream, the adapted load profile shows an occupation/availability forecast for times at the location of the blackboard taking into account the remaining transportation time to the resource. This changes a local load forecast into an opportunity forecast at the location of the blackboard on which the information is posted. An order agent at the entrance of conveyor B4 only has to look at the position with time equal to zero, in this opportunity forecast, to see what the expected load is when he will arrive at the resource. Mobile agents created by order agents use their expected arrival time at the entrance of B4.

**Fig. 1. Propagation of intentions.**

**Fig. 2. Propagation of a load forecast.**

At entrance of a conveyor, this information is not very useful. However, when an agent arrives at the
entrance of Crossing X2 (figure 2), the agent can look at the blackboards at the exits of X2 and compare opportunities. The routing decision across X2 can depend on these forecasts. Importantly, the order agents and their mobile agents do not need to know the route toward a resource when they make their routing decisions. The load forecasts on the local blackboards are sufficient. Similarly, the resource agents and their mobile agents do not need to know about routing through the system, because they use the PropagateUpstream method of the transport resources. Evaporation and refresh handle the dynamic aspects in a robust manner.

Summarizing, the downstream propagation of intentions enables the construction of local load forecasts, whereas the upstream propagation of these local forecasts creates opportunity forecasts for remote resources that can be used locally to guide routing decisions. A difference between the two propagations is that intentions follow a single route through the system; the mobile agents from order agents choose a route at every decision point. In contrast, forecasts spread out across the system when information arriving at an exit is propagated to all entrances; the mobile agents clone themselves when there is more than one entrance.

5. DECISION TAKING

The order agents take decisions based on local information. First of all, the agents respect all the hard constraints and eliminate options that are closed to them. Next, they use a decision rule that takes into account the information on the local blackboards. These rules are randomised to ensure exploration and to avoid getting stuck in a pathological pattern. Evidently, the higher is the perceived benefit, the more likely will an order agent select an option. These decision rules are likely to remain the most maintenance-prone part of the manufacturing control software. Indeed, this is the part where the user may incorporate system-specific and situation-specific knowledge. Fortunately, it is the top layer and there will be no cascade of software maintenance tasks caused by this. Note that the services of staff agents can be used in these decision rules.

The decision rules of the agents must account for the fact that other agents base their decisions on declared intentions while the propagated information only becomes available with some delay. Because of this, proper precautions are needed to avoid that the system behaves too nervously. Indeed, if every order agent (or his mobile agent) changes his intentions immediately whenever he perceives a better opportunity, the system is unlikely to achieve its goal (smooth and optimised production). For the forecasts to have any value, the intentions of only a small percentage of the order agents may change before the other agents are able to observe this within the updated forecast — recall that the agents regularly refresh their information while the old information evaporates. Therefore, order agents and their mobile agents must have a build-in tendency to stick to their intentions, which they declared earlier. For instance, the perceived utility increase has to pass a threshold value before an agent will change his intentions.

6. CONCLUSIONS

This paper discusses the development of multi-agent manufacturing control systems based on techniques inspired by biological system. It identifies the key achievement of the biological example: limited exposure of the individuals combined with the emergence of robust and optimised overall system behaviour.

Furthermore, the paper pinpoints essential properties of the ant solution: (1) the environment is a part of the solution, shielding the remainder of the system from its complexity and dynamics; (2) global information is made available locally. On its way through the system, this information is transformed, in appropriate manners, to enable the agents to make local decisions based on locally available information while being aimed at global goals (e.g. reaching a processing unit along an optimised path). Evaporation and refresh cope with the changes. Randomised decisions ensure exploration.

In the development process, only the last step addresses decision taking. The remainder reflects, as much as possible, facts about the system. Note that the novelty and contribution of the agent-based design comes foremost from the system engineering properties — desirable overall behaviour emerges from putting together agents with limited exposure — rather than from clever decision methods. The approach presented in this paper helps to cope with the need to rapidly develop adequately performing control systems that can adapt to disturbances and changes while being operational, rather than developing the optimal solution for a static situation.

ACKNOWLEDGEMENTS

This paper presents work funded by the Research Fund of the K.U.Leuven and the European Commission.

REFERENCES
