What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC

H. Van Dyke Parunak
Industrial Technology Institute
PO Box 1485
Ann Arbor, MI 48106 USA
van@iti.org

Abstract

The Center for Electronic Commerce (CEC) embodies over fourteen years of experience in applying agents to industrial problems. We have found such a fit in three areas: coordination of industrial designers, simulation and modeling of complex products and processes, and scheduling and control of production systems. This presentation outlines several trends in modern manufacturing, describes how these trends affect the three problem areas, discusses the features of agents that make them attractive candidates for implementing such systems, and reviews example applications from CEC's portfolio in each of these areas.

1. Trends in Manufacturing

Three trends in modern manufacturing present challenges that agent technologies can address. Manufactured products and the systems that produce them have become more complex. The manufacturing process is increasingly spread out over a supply network rather than being concentrated in a single firm, and the variety of products that a firm must offer is increasing while the time to bring them to market is decreasing.

1.1 Increased Product Complexity

A comparison of the features available in a modern automobile with those in a Model T illustrates how the functionality of products has increased. A modern vehicle may include air conditioning, automatic transmission, power steering, power brakes, roll-up windows, turn signals, cruise control, seat belts, air bags, radio and tape player, adjustable seats, and a host of other features that did not exist or could not be installed in an automobile earlier in this century. Similar technical enhancements have enriched many other products, including appliances, machine tools, and aircraft.

1.2 Supply Networks

Modern industrial strategists are developing the vision of the "virtual enterprise," formed for a particular market opportunity from a collection of independent firms with well-defined core competencies [13]. The manufacturer of a complex product (the original equipment manufacturer, or "OEM") may purchase half or even more of the content in the product from other companies. For example, an automotive manufacturer might buy seats from one company, brake systems from another, air

conditioning from a third, and electrical systems from a fourth, and manufacture only the chassis, body, and powertrain in its own facilities. The suppliers of major subsystems (such as seats) in turn purchase much of their content from still other companies. As a result, the “production line” that turns raw materials into a vehicle is a “supply network” (more commonly though less precisely called a “supply chain”) of many different firms.

Figure 1 illustrates a simple supply network [1, 7]. Johnson Controls supplies seating systems to Ford, General Motors, and Chrysler, and purchases the components and subassemblies of seats either directly or indirectly from at least twelve smaller companies, some of which also supply one another. Issues of product design and production schedule must be managed across all of these firms in order to produce quality vehicles on time and at reasonable cost.

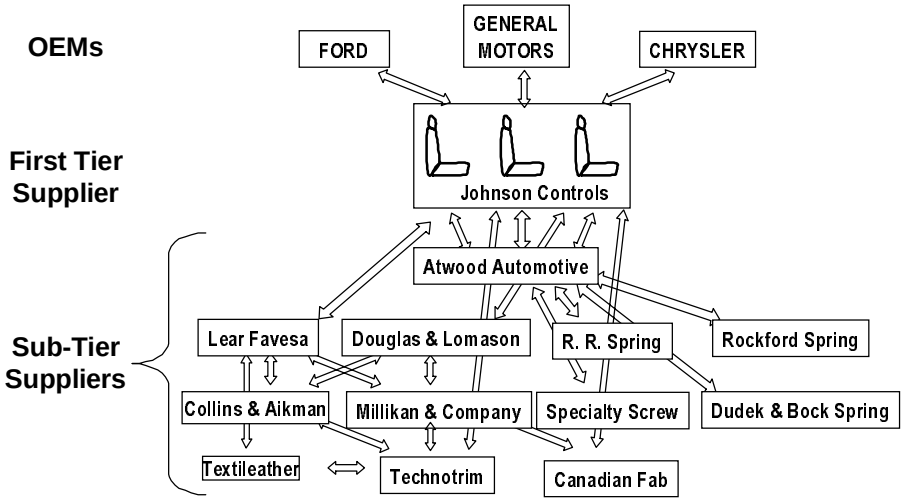

Figure 1: A Simple Automotive Supply Network

1.3 Increased Product Variety over Time

In manufacturing, the product that pleases the most customers has a tremendous advantage, and one of the most effective means known to determine what product features customer like is to turn out as many different product variations as quickly as possible. Customers tend to favor a company offering many product variations, because they can choose the model most closely meeting their desires. In addition, if a company can bring new products to market quickly, it can shift its offerings to reflect customer preferences and thus capture even more of the market. This strategy is responsible for the precipitous drop in the time-to-market for many products. The time from product concept to first production in automotive used to be 60 months. Now world-class performance requires 30 months, and some vehicles have been produced in even less time.

From an AI perspective, a product represents a particular point in the space of product characteristics, and its market performance measures the desirability of that point to customers. The more variations a company can offer and the faster it can

bring them to market, the more thoroughly it can sample the space of customer demand, and the more satisfaction it can deliver to its customers.

2. What Needs to be Done in Industry?

Industry creates wealth and enriches people’s lives by conceiving new products and services and bringing them from vision to reality. In manufacturing, two essential steps in this process are design (working out the details of the product and the systems needed to manufacture it) and operation (of the manufacturing system that produces the product). Increasingly, simulation and modeling tools support the design process. The CEC’s experience in applying agents to industry focuses in these three areas.

2.1 Design

Manufactured products do not just happen. They and the systems that make them must be designed, planned out in great detail to guide the expenditures of capital and effort necessary to realize them.

The movement toward supply chains means that each supplier shares in the design of the overall product by helping design the subsystems it supplies. In addition to product design, much of the design of the production system is routinely assigned to vendors of production equipment or third party “system integrators” responsible for assembling this equipment into a working factory. This strategy poses difficulties of communication and coordination. Design is difficult enough when the participants come from different technical disciplines. When they belong to different companies, they often have divergent corporate cultures and proprietary needs to keep certain information confidential, and information movement between designers can be much slower than between members of a single company. As a result, the cost of design can easily increase, while the coherence of the overall design suffers. Three challenges, difficult enough when design is done by a single-firm, become daunting in the distributed environment of a modern supply chain.

Planning. Design tasks cannot be sequenced in detail. Because of the complex dependencies among components and subsystems, early decisions often have to be changed in the light of later ones. Such backtracking is much easier for closely-linked designers in a single firm than for a widely distributed team.

Coupling. Designers think locally, focusing on their own subsystems, but the complexities of the product couple them with other designers. This challenge and the previous one can lead to infinite loops: designer A’s decision on one feature leads to a decision by designer B that invalidates designer A’s decision on another feature, but when designer A revises this feature, it invalidates B’s earlier decision, and so forth. Distributing the designers across multiple firms only increases the time lag in such loops and makes their resolution more difficult.

Prioritizing. Designers have no common language for comparing the importance of issues. For instance, every subsystem in a vehicle contributes mass to the overall vehicle, and if one weighs more, others must weigh less to keep the entire vehicle in a reasonable range. However, the decision of which components can have more mass and which must make do with less is typically resolved politically, on the basis of whose supervisor is more powerful. In a supply network, this dynamic requires the

OEM to adjudicate conflicts between the suppliers, forcing the OEM to make design decisions about purchased subsystems that it would rather defer to its suppliers.

2.2 Simulation and Modeling

A design for an industrial system or manufactured product is a plan, and like other plans, it can sometimes fail. The modern manufacturing trends discussed above make failure both more likely and more costly than in earlier days. Thus it is increasingly important to test the design before incurring the expense of implementation.

The increased complexity of modern products raises the risk of design failure through the possibility of unanticipated interactions among subsystems and components. Pressures toward distributed manufacturing and design result in a more heterogeneous product. Shorter product life cycles leave less time for design. All of these issues raise the risk that the physical product or system will not satisfy the expectations of the design. The increased scope of the venture also raises the investment necessary to bring the product to fulfillment. An automotive manufacturer must spend between $\$1*10^9$ and $\$3*10^9$ to bring a new vehicle to market, an investment that can be compromised at many points by design failure.

The increased risk and cost of manufactured products is leading to increased use of simulation and modeling technology to assess the performance of a design before its physical implementation. Product simulation (for example, to determine the crash-worthiness of a vehicle) usually relies heavily on finite element analysis, while process simulation (such as establishing the capacity of a production line or the behavior of an inventory system) depends traditionally on stock-and-flow models such as queueing systems or differential equations.

A number of requirements constrain the value of a simulation and modeling environment. A simulation is easier to develop and use if the modeling formalism maps naturally onto the problem domain both structurally (so that model elements correspond to domain entities) and semantically (so that the model language covers the ontology of the domain). The more accurately the model predicts the behavior of the real world, the more risk it will remove from the development process and thus the more value it contributes to the enterprise. The model is not an end in itself, but a step in a larger life cycle that begins with design and moves on to operation of the resulting product, and a model that can support operation will be more valuable than one that serves only the modeling activity.

2.3 Scheduling and Control

Design and simulation can be applied both to the consumer product being manufactured and to the tools and system that will manufacture it. Scheduling and control apply mostly to the manufacturing system. They monitor its trajectory through state space over time and adjust operating parameters to make that trajectory satisfy some overall criterion.

Scheduling and control differ in their characteristic time constants and the kind of information manipulated [19]. Scheduling is longer-term, usually on a scale to which humans can respond, and involves the manipulation of concepts through semantically-grounded symbols. Control manipulates scalar- and vector-valued physical variables, and usually happens too fast for direct human supervision.

In manufacturing, scheduling ranges from long-range plans that support delivery promises to customers, to shop-floor scheduling that determines what happens when at which resources on the factory floor. The objective is to produce the most product possible with the lowest capital investment, while satisfying customer requirements for quality and delivery time. Control deals with turning devices on and off to execute the processes that transform input materials into a finished product.

Increasing product variety challenges both scheduling and control. Scheduling is essentially a search through a combinatorially complex space (roughly, the Cartesian product of the set of parts being manipulated, the set of resources that manipulate them, and the time line). The more products or product variations one manufactures, the larger this space, and the more difficult the scheduling problem. Increased product variation also makes control more difficult, since it increases both the number of devices that must be coordinated and the range of behaviors that those devices must support.

Scheduling is also aggravated by the shift toward supply networks. By passing manufacturing of critical components to suppliers, an OEM relinquishes direct control on when they are produced, and thus increases the uncertainty in the timing of the processes by which it integrates these components into the overall product.

Traditionally, both control and scheduling are done with centralized processes. The combinatorial explosion of scheduling possibilities taxes the ability of a centralized program to compute an optimal schedule. The constant change of product mix needed to respond to customer orders against an increasingly varied set of product offerings means that the factory may never even reach a steady state in which an optimal schedule is well defined. The increased distribution that product complexity brings to control and that supply networks bring to scheduling makes centralization awkward. For both scheduling and control, increased product complexity and supply networks greatly increase the size and complexity of traditional monolithic systems, increasing the cost of software engineering and maintaintenance.

3. What Can Agents Do?

Agents are not a panacea for industrial software. Like any other technology, they have certain capabilities, and are best used for problems whose characteristics require those capabilities. Five such characteristics are particularly salient: agents are best suited for applications that are modular, decentralized, changeable, ill-structured, and complex.[1] The tasks of design, simulation, and scheduling and control manifest many of these characteristics, particularly under the impact of modern manufacturing trends, suggesting that agents are a natural way to address them.

3.1 Modular

Agents are pro-active objects, and share the benefits of modularity that have led to the widespread adoption of object technology. They are best suited to applications that fall into natural modules. An agent has its own set of state variables, distinct from those of the environment. The agent’s input and output mechanisms couple its state

[1] These are an extension of the categories described by [8].

variables to some subset of the environment's state variables. An industrial entity is a good candidate for agent-hood if it has a well-defined set of state variables that are distinct from those of its environment, and if its interfaces with that environment can be clearly identified.

All three of the modern manufacturing trends we are discussing lend themselves to modular analysis. Engineers manage increased product complexity by refining the overall product into subsystems with well defined interfaces. Supply networks assign these subsystems and components to different companies, each with its own organizational identity, operational boundaries, and procedures for interfacing with its environment. The proliferation of product varieties leads to the definition of a product as a base platform and a set of modular options.

3.2 Decentralized

An agent is more than an object; it is a pro-active object, a bounded process. It does not need to be invoked externally, but autonomously monitors its own environment and takes action as it deems appropriate. This characteristic of agents makes them particularly suited for applications that can be decomposed into stand-alone processes, each capable of doing useful things without continuous direction by some other process.

Many industrial processes can be organized in either a centralized or a decentralized way. Centralized organizations go back to the governments of ancient Egypt, Assyria, China, and Babylon, with their focus on a central demigod and an elaborate bureaucracy to manage the flow of control down and information back up. The popularity of this structure can be traced through the army of Alexander the Great, the Roman legions, and the rival empires of pre-modern Europe down to the structure of modern Fortune 500 companies and industrial control architectures [2].

This approach is not the only alternative. The power of decentralization has been made clear in recent years in the contrast in performance between a centralized economic system (the former Soviet Union) and a decentralized one (free-market capitalism). Supply networks are an expression of the decentralized approach, and agent-based architectures are an ideal fit to such an organizational strategy. In fact, a European observer suggests that one of the forces leading to the growing popularity of multi-agent systems is “the rise of the American style of liberalism and individualism” [26]. In addition, the increase in product variation leads to a proliferation of manufacturing devices across increasingly larger factories, so that even within a single company problems of control and scheduling may be more naturally considered from a decentralized point of view.

3.3 Changeable

Agents are well suited to modular problems because they are objects. They are well suited to decentralized problems because they are pro-active objects. These two characteristics combine to make them especially valuable when a problem is likely to change frequently. Modularity permits the system to be modified one piece at a time. Decentralization minimizes the impact that changing one module has on the behavior of other modules.

Modularization alone is not sufficient to permit frequent changes. In a system with a single thread of control, changes to a single module can cause later modules, those it invokes, to malfunction. Decentralization decouples the individual modules from one another, so that errors in one module impact only those modules that interact with it, leaving the rest of the system unaffected.

The modern trend toward increased product variety and shorter time to market makes the manufacturing environment much more changeable than it was previously. Much of the cost of a new factory is in its software. Agent-based architectures permit reuse of much existing code and self-configuration of large portions of the system, reducing both the cost and the time needed to bring up a new factory.

3.4 Ill-structured

An early deliverable in traditional systems design is an architecture of the application, showing which entities interact with which other entities and specifying the interfaces among them. For example, installation of a conventional system for electronic data interchange (EDI) among trading partners requires that one know the providers and consumers of the various goods and services being traded, so that orders can be sent to the appropriate parties.

Such a structure could be defined in advance in the days when a company made most of each product itself and dealt only with its own internal divisions and selected suppliers. The structure of modern supply networks can be dynamic, changing many times over the lifetime of an information system. Consider an electronic system to support open trading, where orders are open to any qualified bidder. Requiring the system designer to specify the sender and recipient of each transaction would quickly lead to “paralysis by analysis.” From a traditional point of view, this application is ill-structured. That is, not all of the necessary structural information is available when the system is designed. Agents can discover this structure as the system operates, rather than requiring it to be designed into the system initially.

Some applications are intrinsically under-specified and thus ill-structured, and agents offer the only realistic approach to managing them. Even where more detailed structural information is available, the wiser course may be to pretend that it is not. A system that is designed to a specific domain structure will require modification if that structure changes. Agent technology permits the analyst to design a system to the classes that generate a given domain structure rather than to that structure itself, thus extending the useful life of the resulting system and reducing the cost of maintenance and reconfiguration.

3.5 Complex

All three of the modern manufacturing tendencies that we have identified increase the complexity of the manufacturing problem. Increased product complexity explodes the size of the space that designers must search and the number of machines that must be scheduled and controlled, and makes the task of mapping between design and simulation model more daunting. Supply networks and increased product variability push traditional optimal scheduling against the wall of computational complexity.

As is often the case, the complexity in this example is combinatorial in nature, resulting from the fact that the number of different interactions among a set of

elements increases much faster than does the number of elements in the set. By mapping individual agents to the interacting elements, agent architectures can replace explicit coding of this large set of interactions with generation of them at run-time. Consider 100 agents, each with ten behaviors, each behavior requiring 20 lines of code. The total amount of software that has to be produced to instantiate this system is 20,000 lines of code, an extremely modest undertaking. But the total number of behaviors in the repertoire of the resulting system is on the order of ten for the first agent, times ten for the second, times ten for the third, and so forth, or 10^{100}, an overwhelmingly large number. Naturally, not all of these will be useful behaviors, and one can imagine pathological agent designs in which none of the generated behaviors will be appropriate. However, appropriately designed agent architectures can move the generation of combinatorial behavior spaces from design-time to run-time, drastically reducing the amount of software that must be generated and thus the cost of the system to be constructed.

Just as well-structured systems can become ill-structured when viewed over their entire life span, so a system that appears to require only a few behaviors can become more complex as it is modified in response to changing user requirements. By adopting an agent approach at the outset, systems engineers can provide a much more robust and adaptable solution that will grow naturally to meet business needs.

4. What has the CEC done with Agents in Industry?

The Center for Electronic Commerce has been active in applying agents to industrial problems since the early 1980's. These applications cover all three of the tasks described above, and illustrate many of the benefits of agents in industry.

4.1 Design

The CEC's major initiative in applying agents to design is in the context of RAPPID (Responsible Agents for Product-Process Integrated Design) [2] [22-24].

A designer seeks to embed a set of *functions* (e.g., optical, electromechanical, control) in an artifact with specified *characteristics* (e.g., weight, color, complexity, materials, power consumption, physical size). The functional view drives most designs, since it distinguishes the disciplines in which engineers are trained and in support of which design tools are available. Conflicts arise when different teams disagree on the relation between the characteristics of their own functional pieces and the characteristics of the entire product. Some conflicts are within the design team: How much of a mechanism's power budget should be available to the sensor circuitry,

[2] RAPPID was sponsored by the Rapid Design Exploration and Optimization (RaDEO) program (formerly MADE) of DARPA, managed successively by Pradeep Khosla and Kevin Lyons, and administered through the AF ManTech program at Wright Laboratories under the direction of James Poindexter. The project team includes Steve Clark, Mike Davis, Mitch Fleischer, Bob Matthews, Van Parunak, and John Sauter (all ITI), Al Ward and Tzyy-Chuh Chang (Ward Synthesis), and Mike Wellman (University of Michigan). The RAPPID prototype is being tested on the design of military land vehicles at the U.S. Army's Tank and Automotive Command (TACOM) in Warren, MI, with the support of the Technology Integration Division.

and how much to the actuator? Others face design off against other manufacturing functions: How should we balance the functional desirability of an unusual machined shape against the increased manufacturing expense of creating that shape?

It is easy to represent how much a mechanism weighs or how much power it consumes, but there is seldom a disciplined way to trade off weight and power consumption against one another. The more characteristics are involved in a design compromise, the more difficult the trade-off becomes. The problem is the classic dilemma of multivariate optimization. Analytical solutions are available only in specialized and limited niches. In current practice such trade-offs are sometimes supported by processes such as QFD (Quality Functional Deployment) or resolved politically, rather than in a way that optimizes the overall design and its manufacturability. The problem is compounded when design teams are distributed across different companies.

RAPPID uses a marketplace to set prices on each characteristic of a design. Agents representing each component buy and sell units of these characteristics. A component that needs more latitude in a given characteristic (say, more weight) can purchase increments of that characteristic from another component, but may need to sell another characteristic to raise resources for this purchase. In some cases, analytical models of the dependencies between characteristics may help designers estimate their relative costs, but even where such models are clumsy or nonexistent, prices set in the marketplace define the coupling among characteristics.

Figure 2 shows a design decomposed into Component Agents (rounded rectangles), each with one Characteristic Agent (ovals) for each dimension in the design space. For example, the "SS.Weight" Characteristic might represent the constraint that the entire product weigh between 5 and 10 kg. The topmost

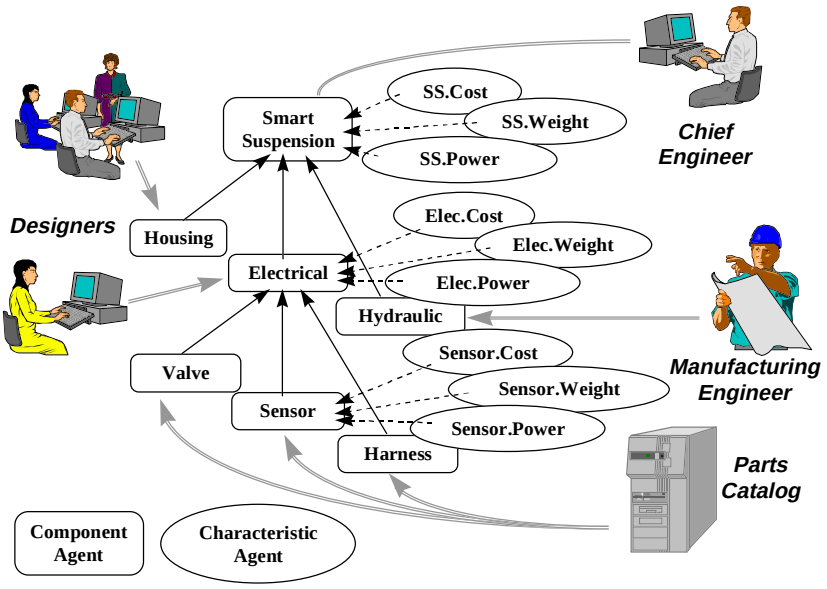

Figure 2: The RAPPID Design Ecosystem for a New Tank Suspension

Component represents the complete product (in this case, a new suspension for a tank), and is the concern of the Chief Engineer, who reflects the Customer's requirements in the initial allocation of design space. The bottom-most Components are either custom-manufactured or (in the Figure) selected from an on-line Parts Catalog. Designers, who typically have responsibility for intermediate levels of the product tree, propagate the constraints from the top and bottom of the tree toward each other. Each Component (either automatically or under guidance from its Designer) buys and sells allocations on its Characteristics to and from other Components.

RAPPID rests on three basic concepts: markets as a mechanism for coordinating distributed decision-making, set-based design, and the use of computer agents in partnership with human agents rather than as a replacement for them.

Market-Based Control.—Researchers addressing distributed problems in a wide range of domains have recently begun to turn to market-based mechanisms for coordination. [3] offers a convenient collection of applications to fields as diverse as computer network control, memory allocation, factory scheduling, pollution management, and air-conditioning load balancing. In all of these areas, competitors for scarce resources can efficiently express their needs in terms of a common currency, and the balance between supply and demand can set prices for those resources that rationalize the distribution of resources across competitors.

Set-Based Ideas.—A product's characteristics can be thought of as dimensions of a Cartesian space within which the product is defined. Traditionally, designers seek to *build* a design to fit a predefined subspace of characteristics, without knowing in advance whether any acceptable design fits. Toyota has pioneered a more promising vision that begins with a design space much larger than necessary, and then shrinks it incrementally to *discover* the subspace occupied by the desired design [29].

Current design tools do not support this vision. With a market in design characteristics, low prices identify slack characteristics (dimensions of the space) that the chief engineer can collapse by buying up allocations of that characteristic. This action simultaneously reduces the amount of that characteristic available for use by designers and increases the funds in the system available to purchase other characteristics to compensate for the decrease in the given characteristic. As designers buy and sell, the relative prices of the various characteristics change, identifying a new slack dimension that can further shrink the design space.

Silicon and Carbon Agents.—RAPPID does not automate the entire design problem. Its market mechanisms function alongside conventional interactions, which we describe as SLOWH (Standard Legacy-Oriented Work Habits). The market mechanisms themselves are not entirely automated, but are implemented partly in computer algorithms and partly in human behaviors, a mixture that we describe as “hybrid carbon-silicon systems.”

4.2 Simulation

The CEC’s research in agent-based simulation and modeling includes the development of XSpec (a modeling tool specifically equipped for industrial controls) and the DASCh project (applying the SWARM modeling environment [11] to industrial supply networks).

4.2.1 XSpec

XSpec ("eXecutable Specification")[3] [28] is a simulation environment designed specifically for modeling industrial control systems. It develops three techniques that are critical for success in this domain: pairing of physical and control models, the use of wrappers to integrate existing applications, and a technique for synchronizing time across such integrated applications.

Pairing of Physical and Control Models.—Design of a manufacturing mechanism typically requires close coordination between a mechanical engineer who designs the physical components and an electrical engineer who designs the control algorithms that manipulate it. Conventionally, these distinct aspects are often developed in isolation from one another, using separate engineering tools, and then brought together in an integration task when inevitable inconsistencies are discovered. XSpec promotes the integrated development of physical and control models with a class of agents, "components." Each component includes both physical and control models and the interfaces between them for a specific element of the system, and defines the interfaces that this combined model presents to other components. These external interfaces include not only control signals but also physical interactions. The discipline of defining both physical and control sides of each component before tying these components together into a system forces engineers to work closely together, resolving inconsistencies at the level of an individual device rather than after devices have been elaborated into two complete but incompatible systems.

Wrapping of Existing Applications.—Integrating physical and control models device-by-device runs contrary to the grain of existing modeling tools, which focus either on the physical behavior of the system (e.g., a kinematic modeler such as Robcad) or on its control behavior (e.g., a DEDS modeler such as Flexis). Because these tools divide the world by discipline (mechanics vs. control) rather than by device, they discourage engineers from the close integration of disciplines on a device-by-device basis. To overcome this pressure, XSpec enables individual tools to be wrapped so that they can automatically exchange their results with one another, much along the lines of ARCHON [4, 9, 25]. In the applications we have modeled, we use XSpec to integrate Robcad (for the kinematic behavior of individual mechanisms), SIMAN/Cinema (for queueing behavior) and Flexis (for control algorithms).

Synchronizing Time.—A major challenge in integrating different models of real-time behavior is maintaining a common view of time across the various applications. The XSpec wrapper for modeling packages includes a special synchronization protocol and a time server, or "Syncer." These mechanisms coordinate the temporal execution of models constructed in different tools so that their integrated behavior is causally meaningful and useful as an overall model of the system [10]. The Syncer supports the integration of different time models, including discrete events, continuous time, and the special case of zero time between two events on different tools.

The power of XSpec was demonstrated in its application to a gear weld cell for a major automotive manufacturer. The cell, designed and constructed with conventional

[3] The XSpec team includes Mark Brown, John Sauter, Ray VanderBok, and Jack White of ITI, and Bob Judd of the University of Ohio.

technologies, was only producing 75% of its rated capacity. The XSpec team modeled the line, and then tested various modifications on the model without the constraint of working around daily production use of the cell. When the modifications recommended by the model were installed in the cell, its production rose to 125% of its rated capacity.

4.2.2 DASCh

Supply networks, like most systems composed of interacting components, support a wide range of dynamical behavior that can interfere with scheduling and control at the enterprise level. Data analytic approaches are not effective in understanding these dynamics, because the commercial environment changes too rapidly to permit the collection of consistent data series long enough to support statistical requirements. The DASCh project (Dynamical Analysis of Supply Chains)[4] [16, 17] explores the dynamical behavior of takes the approach of constructing and experimenting with an agent-based emulation model of the system that can maintain a given set of conditions as long as desired.

Following the pioneering work of Jay Forrester and the Systems Dynamics movement [6], virtually all simulation work to date models the supply chain as a set of partial differential equations and then integrates these equations over time. DASCh uses agent-based emulation, representing the various components of the supply chain by software agents that emulate their actual behaviors. The DASCh approach is more faithful than numerical simulation, better supports the increasingly decentralized nature of supply chains and the need to protect proprietary information, and provides a much closer interaction between model and real system [17].

DASCh includes three species of agents. *Company agents* represent the different firms that trade with one another in a supply network. They consume inputs from their suppliers and transform them into outputs that they send to their customers. *PPIC agents* model the production planning and inventory control algorithms used by company agents, and currently support a simple MRP model. *Shipping agents* model the delay and uncertainty involved in the movement of both material and information between trading partners.

The initial DASCh experiments involve a supply chain with four company agents (a boundary supplier, a boundary consumer, and two intermediate firms producing a product with neither assembly nor disassembly), illustrated in Figure 3. The two intermediate company agents each have PPIC agents to convert incoming orders to orders for their inputs, and shipping agents manage all movement of both material and information among company agents.

This simple structure was intended as a starting point. It was expected to yield relatively uninteresting behavior, on which the impact of successive modifications could be studied. In fact, it shows a range of interesting behaviors in terms of the variability in orders and inventories at the various company agents.

---

[4] DASCh was funded by DARPA, and administered through the AF ManTech program at Wright Laboratories under the direction of James Poindexter. The DASCh team includes Steve Clark and Van Parunak of ITI, and Prof. Bob Savit and Rick Riolo of the University of Michigan's Program for the Study of Complex Systems.

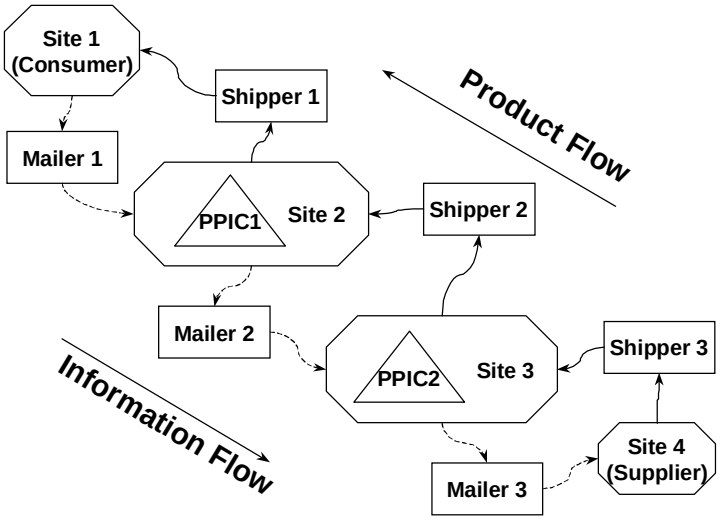

Figure 3: The DASCh Supply Chain

- As the demand generated by the top-level consumer propagates to lower levels, its variance increases, so that lower-level suppliers experience much more variability than higher-level ones. This phenomenon is widely discussed in the literature.
- Not as well recognized is the correlation imposed on an originally uncorrelated series of random orders by the PPIC algorithms in the supply network.
- A single modest change at the top of the chain generates disturbances in the order sequences of lower tier suppliers that persist long after the original change.
- Even when top-level demand is constant and bottom-level supply is completely reliable, intermediate sites can generate complex oscillations in inventory levels, including period doubling, as a result of capacity limitations.

The unexpectedly complex behavior of the DASCh model holds a lesson not only for industrialists with supply chains but for researchers in multi-agent systems generally. The interactions of agents with one another can generate complex behaviors, even when the agents and their organization seem simple and straightforward. This complex emergent behavior is both a challenge and a benefit. It is a benefit because it means that complex problems can in principle be addressed with sets of relatively simple agents, if their complex interactions can be harnessed. It is a challenge because it requires agent engineers to pay close attention to system behaviors as well as agent behaviors, rather than assuming that the whole will equal the sum of the parts.

4.3 Scheduling and Control

Shop-floor scheduling and control is one of the more mature application areas for agent technology, and CEC has executed several projects in this area. This summary will highlight major lessons from each of these.

YAMS (“Yet Another Manufacturing System”) [14] applied a contract net to a hierarchical model of manufacturing system, including agents representing the overall

shop, each of several workstations in the shop, and each of several devices in each workstation. The hierarchical model followed the widely-circulated NASREM architecture promoted by the US National Institute for Standards and Technology [2]. Our experiments using this system to control an actual shop in our laboratory led to two major conclusions. First, hierarchies are in general a poor way to organize agents. While most physical interactions were between peers in the hierarchy, the use of a rigidly hierarchical model forced messages to travel up to the lowest shared node and then back down. The resulting flood of messages rapidly overwhelmed the communication network. Second, messages can be physical as well as electronic. In spite of careful analysis to prove that the protocols were deadlock-free, the system deadlocked. Further analysis showed that the movement of a physical part from one station to another, and the signal generated when it encountered a sensor at its destination, provided a "message" from the sending station to the receiving station that had not been included in the protocol analysis and that compromised the system's liveness.

CASCADE[5] (not an acronym, although usually spelled with upper-case letters) [20, 21] focused agent methods on the specific domain of material handling. Our objective was to develop a distributed control mechanism for a Weldun-Bosch modular conveyor system, so that configuring the control software could be as straightforward as plugging together the physical components of the conveyor. We began by modeling the protocols as a contract net between conveyor segments. Because of the simple semantics involved ("Give me a pallet." "Here, take a pallet."), we were able to streamline the ontology and protocol considerably, using reinforcement learning in the individual segments to yield a self-routing system. During our work, the work of the Parallel Distributed Processing group [12, 27] came to our attention, and we noticed that our reduction of the contract net had led us to a structure isomorphic to a back-propagation neural network. We articulated the experience as the "marching-band syndrome," in which agents of considerable sophistication (such as the members of a marching band) sometimes achieve their objectives best by adopting a greatly reduced interaction protocol based on the requirements of the domain (in the case of the marching band, match one's velocity and separation to one's neighbors, a mechanism successfully executed by flocks of birds and schools of fish). CASCADE drew our attention to the power of fine-grained agent solutions and led to our advocacy of synthetic ecosystems, agent architectures that depend for their functionality on the emergent dynamics of interaction among agents as well as the explicitly defined behaviors of individual agents.

SFA (Shop Floor Agents)[6] [15] is a collaborative project in which several industrial firms are sharing their experiences in constructing agent-based shop floor systems. The project has highlighted the difference between a research environment (in which every developer is an agent researcher, capable of devising new architectural solutions in mid-stream to fit emerging challenges) and an industrial one

[5] YAMS and CASCADE were funded in part by the Kellogg Foundation. Participants included Bruce Irish, Jim Kindrick, Peter Lozo, and Van Parunak.

[6] Sponsored by the National Center for Manufacturing Sciences, directed by Tony Haynes, and executed at ITI by Steve Clark, Mike Davis, Jorge Goic, Van Parunak, and John Sauter.

(in which developers are general software engineers without any deep knowledge of agent theory). This experience shows that effective deployment of agents in industrial settings requires the articulation of a teachable methodology for specifying and designing multi-agent systems and the availability of development tools that package best practice for implementation. The project’s deliverables address these needs with extensions to standard object-oriented development methods and a sophisticated implementation environment within the Gensym G2 system.

AARIA (Autonomous Agents at Rock Island Arsenal)[7] [18] developed an architecture for manufacturing scheduling in which resources, processes, and parts are peer agents. In most previous work on agent-based scheduling, resources (such as machining workcenters) are agents, negotiating with one another for access to input parts and the disposition of output parts. This architecture is acceptable in a classical job shop in which parts compete for access to resources, but does not fit as well in block-build environments such as airframe fabrication or ship drydocking, in which the product is stationary and resources must compete for access to it. The basic idea of a multi-species approach to agent-based scheduling and control emerged from a conversation with Jean-Pierre Müller in Neuchâtel in September of 1992. The resulting ecosystem (Figure 4) includes three main species: resources (including machines, tools, and operators), part types, and unit processes (the basic production steps into which manufacturing is divided [5]). The dynamic of this system can be viewed as two flows, one of resources and one of parts, intersecting at the unit process.

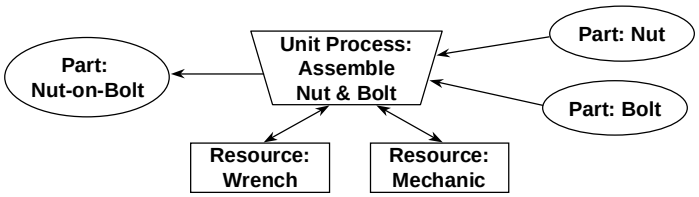

Figure 4: A Shop-Floor Ecosystem

By maintaining the orthogonality of these three species, the framework can support industries such as aerospace and shipbuilding that require integration of job-shop and block-build environments. In addition, these flows support mechanisms that permit different scheduling modalities to emerge dynamically at different resources as the system runs. Conventional shops are scheduled either by dispatching based on local equipment constraints, dispatching against an advance schedule, or Kanban (response to a demand pull). These techniques can be distinguished on the basis of the

[7] Initial research on AARIA was funded by the DARPA Agile program under contract F33615-95-C-5524, managed by USAF ManTech. The prime contractor is Intelligent Automation, Inc. (Len Haynes, Kutluhan Erol, and Renato Levy). Other contractors are the Industrial Technology Institute (Van Parunak, Steve Clark, Jorge Goic), the University of Cincinnati (Albert Baker, Bradley Matthews, Ben Moore, Brian Birtle, Veena Pandiri), Flavors Technology (Howell Mitchell), and Rock Island Army Arsenal (Greg Peters, Don Tice).

degree to which they are constrained by advance commitment and by customer demand (Figure 5). Dispatch is the least constrained modality. Advance scheduling is highly constrained by earlier promises, but unresponsive to changes in demand while the system operates. Kanban requires both advance commitment and current demand.

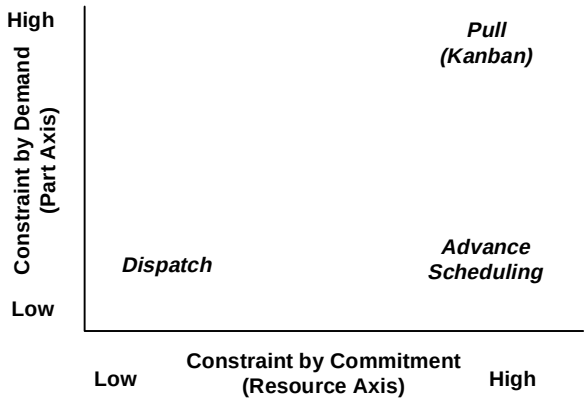

Figure 5: Emergent Scheduling Modalities

In a conventional shop, every workstation is driven by the same modality. In the three-species ecosystem, different portions of the shop can behave according to different modalities, and can change their modality over time as conditions change. The degree to which a given unit process is constrained by demand is determined by the behavior of part type agents, while the degree of constraint by commitment is determined by the behavior of resources. The result is a highly flexible architecture that is readily adaptable to a wide range of manufacturing applications.

Growing out of this long experience, the CEC is currently a subcontractor in three different industrial projects in the domain of manufacturing scheduling and control. One applies agents to scheduling problems in the domain of semiconductor manufacturing, another deals with scheduling in shipbuilding, and a third addresses dynamic reconfiguration of the equipment in a manufacturing line in the event of device failure. For proprietary reasons, details of these projects cannot be revealed at this time.

5. Summary

Manufacturing is a rich domain for application of software technology. Modern pressures toward increased product complexity and diversity and the use of supply networks to produce these products produce problems characterized by modularity, decentralization, changeability, poor structure, and high complexity, a set of characteristics that agents are uniquely suited to address. These problems impact the design of products and the systems that manufacture them, the simulation and modeling of these systems, and their scheduling and control during operation. The CEC’s extensive experience in applying agents to such problems confirms the importance and appropriateness of agent technology in modern industrial applications, and demonstrates that this technology is mature enough for wide-spread deployment.

6. References

[1] AIAG. Manufacturing Assembly Pilot (MAP) Project Final Report. M-4, Automotive Industry Action Group, Southfield, MI, 1997.

[2] J. S. Albus, H. G. McCain, and R. Lumia. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). NBS Technical Note 1235, National Bureau of Standards, Gaithersburg, MD, 1987.

[3] S. H. Clearwater, Editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation.* Singapore, World Scientific, 1996.

[4] J. M. Corera, I. Laresgoiti, and N. R. Jennings. Using ARCHON, Part 2: Electricity Transportation Management. *IEEE Expert*, 11(6):71-79, 1996.

[5] I. Finnie, Editor. *Unit Manufacturing Processes: Issues and Opportunities in Research.* Washington, DC, National Academy Press, 1995.

[6] J. W. Forrester. *Industrial Dynamics.* Cambridge, MA, MIT Press, 1961.

[7] T. Hoy. The Manufacturing Assembly Pilot (MAP): A Breakthrough in Information System Design. *EDI Forum*, 10(1):26-28, 1996.

[8] N. Jennings. Applying Agent Technology. Plenary presentation at PAAM'96. 1996.

[9] N. R. Jennings, E. H. Mamdani, J. M. Corera, I. Laresgoiti, F. Perriolat, P. Skarek, and L. Z. Varga. Using ARCHON to develop real-word DAI applications, Part 1. *IEEE Expert*, 11(6):64-70, 1996.

[10] R. P. Judd, J. F. White, P. K. Hickman, M. E. Brown, and J. A. Sauter. System for combining originally software incompatible control, kinematic, and discrete event simulation systems into a single integrated simulation system. , Industrial Technology Institute, U.S., 1993.

[11] C. Langton, R. Burkhart, and G. Ropella. The Swarm Simulation System. http://www.santafe.edu/projects/swarm/, 1997.

[12] J. L. McClelland and D. E. Rumelhart, Editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models.* Cambridge, MA, MIT Press, 1986.

[13] R. N. Nagel and R. Dove. *21st Century Manufacturing Enterprise Strategy*. Bethlehem, PA, Agility Forum, 1991.

[14] H. V. D. Parunak. Manufacturing Experience with the Contract Net. In M. N. Huhns, Editor, *Distributed Artificial Intelligence*, pages 285-310. Pitman, London, 1987.

[15] H. V. D. Parunak. Workshop Report: Implementing Manufacturing Agents. http://www.iti.org/~van/paamncms.ps, Industrial Technology Institute, 1996.

[16] H. V. D. Parunak. DASCh: Dynamic Analysis of Supply Chains. http://www.iti.org/~van/dasch, 1997.

[17] H. V. D. Parunak. Agent-Based Behavioral Emulation vs. Numerical Simulation: A Users' Guide. In *Proceedings of ICMAS '98 Workshop on Multi-Agent Systems and Agent-Based Simulation*, Springer, 1998.

[18] H. V. D. Parunak, A. D. Baker, and S. J. Clark. The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design. In *Proceedings of First International Conference on Autonomous Agents (ICAA-97)*, 1997.

[19] H. V. D. Parunak and R. Judd. Sharpening the focus on intelligent control. *International Journal of Computer Integrated Manufacturing*, 3(1):1-5, 1990.

[20] H. V. D. Parunak, J. Kindrick, and B. Irish. Material Handling: A Conservative Domain for Neural Connectivity and Propagation. In *Proceedings of Sixth National Conference on Artificial Intelligence*, pages 307-311, American Association for Artificial Intelligence, 1987.

[21] H. V. D. Parunak, J. Kindrick, and B. W. Irish. A Connectionist Model for Material Handling. *Robotics & Computer-Integrated Manufacturing*, 4(3/4):643-654, 1988.

[22] H. V. D. Parunak, A. Ward, M. Fleischer, and J. Sauter. A Marketplace of Design Agents for Distributed Concurrent Set-Based Design. In *Proceedings of ISPE/CE97: Fourth ISPE International Conference on Concurrent Engineering: Research and Applications*, 1997.

[23] H. V. D. Parunak, A. Ward, M. Fleischer, J. Sauter, and T.-C. Chang. Distributed Component-Centered Design as Agent-Based Distributed Constraint Optimization. In *Proceedings of AAAI Workshop on Constraints and Agents*, pages 93-99, American Association for Artificial Intelligence, 1997.

[24] H. V. D. Parunak, A. Ward, and J. Sauter. A Systematic Market Approach To Distributed Constraint Problems. In *Proceedings of International Conference on Multi-Agent Systems*, pages (submitted), AAAI, 1998.

[25] F. Perriolat, P. Skarek, L. Z. Varga, and N. R. Jennings. Using ARCHON, Part 3: Particle Accelerator Control. *IEEE Expert*, 11(6):80-86, 1996.

[26] J.-F. Perrot. Preface. In J. Ferber, Editor, *Les Systèmes Multi-Agents: Vers une intelligence collective*, pages xiii-xiv. InterEditions, Paris, 1995.

[27] D. E. Rumelhart and J. L. McClelland, Editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridge, MA, MIT Press, 1986.

[28] J. A. Sauter and R. P. Judd. XSpec: The Modeling and Design of Complex Systems. In *Proceedings of DARPA Manufacturing, Engineering, Design, Automation Workshop*, pages 125-132, DARPA SISTO and DSO, 1992.

[29] A. Ward, J. K. Liker, J. J. Cristiano, and D. K. S. II. The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster. *Sloan Management Review*, (Spring):43-61, 1995.