

Automation and Control in Large-Scale Interactive Systems

Jay S. Bayne

jbayne@wi.rr.com

Abstract

Designing increasingly complex distributed systems is a challenging engineering problem. Computers, control strategies and communications networks are the tools we have to use. Their effective use should be directed by a science competent to handle the architecture and behavior of large-scale dynamic systems. This is systems science, the science of communications and control in federated systems. The application of cybernetic principles to the design of enterprise control systems yields architectures with specific functional, structural and performance characteristics.

Introduction

The central thesis of cybernetics is that there are natural laws that govern the behavior of large-scale interactive systems. These laws have to do with self-regulation and self-organization. They correspond to the internal management principles by which systems simultaneously grow and are stable, learn and self-regulate, adapt and evolve. This behavior is key to a system's ability to exhibit viability, to identify appropriate objectives, and with minimal resources to regulate internal processes to achieve dynamic stability.

Modern commercial (e.g., industrial) enterprises may be modeled as large-scale systems with dynamic structure embedded within local, national or global market environments with stochastic behavior. Many are enterprises are organizationally decentralized, acting as federations of domestic legal entities. As such, their coordinated and optimized economic behaviors must be organized and operated with communications and control structures that are simultaneously formal yet flexible, decentralized yet globally interconnected, functionally partitioned yet operationally integrated.

Flexibility requires operating policies to be distinct from the underlying mechanisms used to carry them out. This *separation principle* encourages generalization of structure [10] with effective management being

considered a dynamic multi-level control problem. Higher levels deal with policy—constructing and delivering knowledge about the world. Lower levels deal with mechanism—constructing and delivering information in the form of products and services. In an interactive system, the separation principle allows coupled systems elements to be adaptive, where new policies may be put into play through the use of established mechanisms, and new mechanisms may be invoked without affecting running policies.

Decentralization requires a communications infrastructure which is capable of supporting both signal processing and transactional interchanges among cooperating entities, both vertically within and horizontally across the enterprise's accountability hierarchy, and geographically among its distributed operations. Today's *internet* and related *intranet* communications technologies make distance technically irrelevant. The essential message types and communications protocols supported between the system and its context constitute a formal *metalanguage* for the system.

Figure 1 depicts a multilevel enterprise control hierarchy [1, 11]. The structure contains two manufacturing plants and a corporate control level. The plants are each organized into equipment, station, unit, area and plant control domains. At the corporate level two level structures includes operations and development functions. This structure supports both intra-plant and inter-plant management activities.

Automation and control services are present at each level. They provide sequence and modulating controls for motor, power, steam, start-up and shut-down, safety and production applications in the lowest levels; planning, scheduling, maintenance management, and supervisory systems at the plant level; and order processing, fulfillment, financial controls and transportation logistics at the interplant levels. Nodes in Figure 1 define the *spans of a control* of operational services that populate the enterprise infrastructure.

Integration requires that the automation environment, specifically its command, communications and control infrastructure be designed and implemented with concern for the mission and operating principles of the enterprise. In a recursive fashion, the automation environment must be modeled, organized and administered as a formal part of the technical infrastructure of the enterprise it serves. Being formal, it endows the enterprise with the integrative benefits of a formal operational context. It is the enterprise *intranet*, simultaneously providing the communications and computing support for the enterprise's subconscious (*reactive*) behaviors, and the knowledge bases and decision-support applications for its conscious (*proactive*) supervisory behaviors.

Figure 2 recognizes the fact that enterprises do not function in isolation. They are part of webs of alliances made up of participants in a value proposition, each member of which typically has its own infrastructure and associated suite of automation and control capabilities. In such a web, integration of applications among participants requires connectivity and interoperability among specific functions. Such internetworking defines the web's *extranet*, the operational fabric that binds the functional elements of the alliance.

Cybernetic System Principles

The science of cybernetics is 56 years old this year. Norbert Wiener is credited with its creation in 1946 as "the science of control and communication in the animal and in the machine" [13]. As both a science and a philosophy, it states that there are general laws governing control of large-scale dynamic, probabilistic systems, whatever the systems being governed, be they computers, mechanical servo-mechanisms, auto-

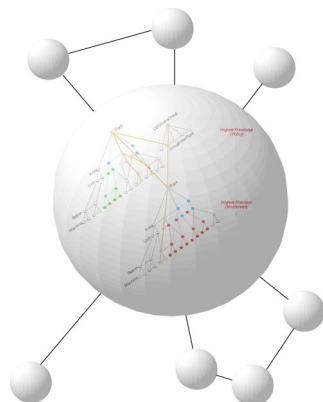


Figure 2 - Enterprise Alliance Web

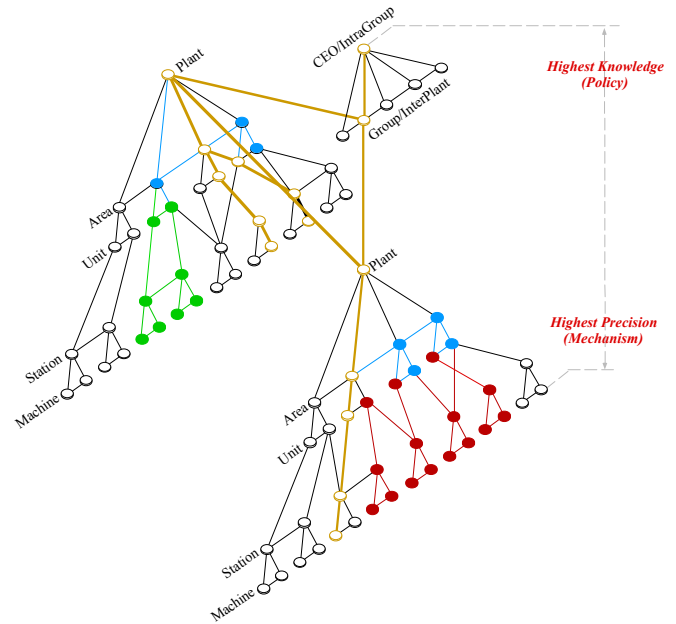


Figure 1 - Command & Control Hierarchy

replicating biology [8], the speed and trajectory of a missile, the temperature of a power boiler, populations of animals, stock prices, the economies of nations, or any complex yet quantifiable dynamic process. Commercial enterprises are examples of such systems, as are their internal manufacturing and service delivery subsystems.

Systems are *large-scale* when, at the resolutions of interest, the number of states the system may occupy (i.e., the *state-space*) is considerably larger than can be held in our three pound, slightly alkaline, 25-watt, electro-chemical cranial computers. A moderately sized manufacturing process may contain 100 analog inputs and 100 analog output signals, each resolved to twelve bits. In addition, it might contain 500 binary inputs and 500 binary outputs. The state space is too large to contemplate, let alone attempt to compute system behaviors analytically, either by first-principles or in a simulated manner [7].

Systems are *dynamic* when their behaviors unfold in time as functions of their present states, present inputs, and generated outputs [10]. Systems are *probabilistic* when, for example, measures of their inputs and outputs contain random errors (i.e., noise) or their internal states are uncertain due to statistical failures [2]. Systems are *interactive* when they are required to react to their environment's unpredictability, to adapt to an evolving context. And systems are *viable* when they survive in such evolving contexts. Viability is the primary objective of interactive systems.

Interactive systems are dynamic by nature, using subsets of their internal states to store and track the evolving context in which they are immersed. Such systems are, in a specific sense, *conscious* in that they exhibit an expanding awareness of the conditions of their own state of affairs. The resulting knowledge bases constitute the system's *world-model*. It is the basis of a coherent process-centric world-view, of the envelope within which viable dynamic behaviors are allowed to develop.

Throughout nature viable dynamic systems exhibit four principle characteristics [3]:

- Their behaviors are *coherent* within finite contexts;
- Their survival through time is associated with some well-defined *identity*; and
- Their viability is associated with operating within quantifiable *dynamic equilibrium* conditions (homeostasis);
- They are self-aware (reflexive), assimilating their own unfolding experience into *self-regulating* and *self-organizing* processes of learning, adaptation and evolution.

These essential characteristics provide a basis for the following assumptions about appropriate automation and effective enterprise management.

Effective Management

If cybernetics is the science of control, management is the profession of control [4]. Because interactive systems, both in structure and in interaction, are generally too complex to be held in the minds of any given manager or set of managers, the best we can hope for is to effectively partition the enterprise into logical chunks, or control domains. However, elementary system science recognizes that interacting sets of locally optimal processes may very well produce seriously sub-optimal global behaviors [6]. Under certain conditions, it is likely that good managers of enterprise subsystems will work at cross-purposes, causing oscillatory, divergent or deadlocked behaviors. In process systems this situation occurs routinely and its effects can be seen in fluctuations in throughput, energy use, financial performance, resource scheduling, inventory whip, and labor capacity or skills imbalances. In these situations there appears to be a deficient higher order logic on which local managers (process controllers) can rely for performance metrics, balanced shared objectives, rules for cooperative action, global resource sharing mechanisms, or conflict resolution.

Rather than direct intervention and control,

cybernetics argues that individual and cooperative management action should be more an act of catalyzing the metabolic processes inherent in the enterprise, processes which are carried along with the large-scale behavior of the system [3]. Routine behavior should ultimately become automatic, operating subconsciously and relegated to the system's automation and control infrastructure. Management's attention can and should be free to rise to the metasystemic level, to address the higher-level knowledge and policy problems of self-organization, value judgment, adaptation, and evolution.

Given this view, effective management is not about direct intervention in the self-regulating affairs of a system, but rather engaging in directed learning and self-organizing activities required, in the face of external forces and uncertainty, to invent new elements and restructure old ones to allow a system to "servo" itself into a new, higher state of dynamic equilibrium, a higher state of awareness, *homeostasis* [8]. In this view, management is better focused on *metaprogramming* the system rather than the microprogramming of its subsystems.

Communication and control hierarchies are a natural result of establishing accountability structures among subsystem elements [5, 10]. Hierarchies are organizational relationships in which clear objectives can be defined and compartmentalized, processes can be defined and invoked, performance can be measured, and conclusions can be reached about effectiveness. Examples of natural communication and control hierarchies may be found in biological *wetware* systems, software systems, mechanical and electronic hardware systems, social and political systems, and military-industrial systems. Not all such hierarchies are vital, effective, viable or adaptive. Where they are, cybernetic laws can be shown at work.

The highest-level tasks of enterprise management are diagramed as in Figure 3. There are two fundamental aspects: one dealing with operational issues of the here and now, and the other dealing with developmental issues of anticipated future behaviors. Overseeing these two components is a superior authority, or *management board*, capable of simultaneously dealing with the pragmatics of the present while facilitating development of effective change. This is a meta-linguistic level, requiring the balanced support of discourse outside and above the language of operations while addressing the pragmatics of business financials. At this management level, decisions establish envelopes for future trajectories, result in operational policy changes, and establish internal metasystems competent to deal with

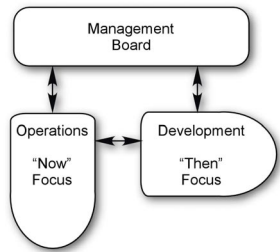


Figure 3 - General Management Functions

complexity.

The operational elements of an enterprise focus on current objectives within a preset and reasonably well-defined and static context. The developmental elements look forward in time, defining new objectives within new, less certain, and evolving contexts. Enterprise adaptation and evolution takes place at the interfaces between these two directorates, at the points in time when operational processes can accept “reprogramming” and avoid serious instability. During these evolutionary spurts, selected and appropriately tuned conscious development initiatives are instantiated as operational subconscious processes, and the enterprise structure is changed. Effective discourse and compromise at this interface tests even the very best management teams.

Effective Enterprise Structure

In this now classic model an enterprise realizes its operational and developmental imperatives through the collective behavior of a *federation* of semi-autonomous *agents*. In the vernacular of contemporary computer science these agents, when automated, correspond to active or passive computational *objects*. An elementary object exhibits both state and behavior, encapsulating a single process and its local data in order to provide specified services. Each such service translates a specific input message (service demand) into a specific output message (service response). This behavior is called the *client-server model* of information processing. It has proved to be a very effective vehicle for defining and organizing systems of semi-autonomous distributed agents engaged in cooperative activities.

The *object-oriented* version of the client-server model recognizes two types of objects. *Active objects* encapsulate single-threaded services that, once invoked by a client process, run to completion. Their activity is synchronous with respect to their invocation (and thus, their client), yet they may be continuously,

aperiodically, or periodically re-invoked. *Passive objects* encapsulate multi-threaded services that, once commissioned, are engaged in a never-ending task that may be continuously or periodically interrupted by client requests. These processes tend to run concurrently with the client’s activities. These two information-processing models provide essentially all of the semantics necessary to construct the operational and developmental processes of Figure 3. In short, these object models are sufficiently robust linguistic frameworks in which to define enterprise communications and control [10].

Figure 4 diagrams an enterprise sub-system service as an idealized process *object*. The figure identifies the object’s essential communications interfaces, or message *ports*. Port *d* is the *demand* port to which clients of the service subscribe and submit requests. Port *s* is the *service* port on which client demands are met. Port *e* is the feedback, or *error*, port on which the object determines within its service sector any mismatch between demand levels and service capacity, allowing it to adjust its internal organization and mechanisms to at least partially compensate for over or under capacity conditions. Port *p* is the *policy* or supervisory control port on which a higher authority may place incentives or disincentives to adjust the object’s essential behaviors. Port *c* is a general *open communications* port on which the object is free to interact with other typically domain-specific, peer-level objects for the purpose of information sharing, consensus, redundancy, adaptation, and replication.

In this model all objects of the federation are semi-autonomous and self-regulating. Their designs, following Jeffersonian principles, are required to 1) be viable and identifiable members of a community, 2) be governed by its laws, and 3) provide their individual contributions to coherent ensemble behaviors which characterize the outcomes of the mission of the enterprise as a whole. These obligations require that the context in which the federation functions define basic

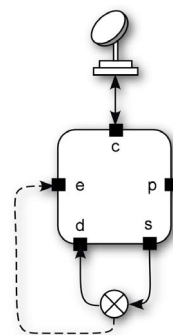


Figure 4 - A System Service Object

standards of behavior and to provide a coherent interconnect structure. At minimum, behavioral standards must provide protocols for the creation, invocation, modification and graceful termination of process objects within the environment. Side effects, or aberrant behavior, must be contained.

In a federated management environment, a global naming convention, a set of message formats and related symbol set must support formal communications protocols, and a set of specific messages with agreed upon meanings. Naming establishes unique identity, and is required for accountability. Message formats establish an alphabet and syntactic rules for discourse. And specific messages are the selectors of system behavior, establishing the semantic framework in which a federated community may define and pursue shared and coordinated objectives.

Therefore, in a decentralized real-time object-oriented environment, understanding “the system” requires an understanding of the behaviors exhibited at the communications interfaces of its contained objects. These interface specifications are sufficient to describe the system’s dynamic structure, the services it may provide, and its relevant performance metrics. Such a specification is referred to as the *architecture* of the system. A system’s architecture is specified in a metalanguage, and its specific request-response behaviors are defined in a programming language.

For example, a software object within a given system may be implemented in C++, but the behavior witnessed at its interfaces may be defined in an *interface definition language* (IDL). This is the separation principle at work – the isolation of mechanism from policy, the independence of implementation from architecture, the distinction between an object’s internal organization and its external behavior, the difference between a system and its encapsulating metasytem.

Systems provide two essential classes of service, one among internal objects of the system, and one between special *gateway services* (i.e., foreign system interfaces) and external environment elements. Internal objects share a common naming, access control, and accounting framework, share common communications protocols, and are supported by a uniform execution environment. External clients and services may require protocol conversion, address resolution, security services, and special execution services. Internal system objects assigned to such gateway functions are called *proxy*

agents, individuals who represent foreign objects in the affairs of the system.

Figure 5 depicts the traditional behavior model for *intelligent objects* capable of operating within a federated control system [2, 11]. Each object is capable of sensing the state of its local context as measured through its d-port sensors. This input is presented to an internally maintained *world-model* through sensory *signal processing* logic. The world-model qualifies the signal processing activity and accepts updates to the database(s) that represent the context state. *Behavior generation* logic produces appropriate outputs that are delivered to the environment through the object’s s-port actuators.

The upper loop depicted in Figure 5 provides the object’s internal management function, supporting all adaptability and intelligence exhibited by the object. In addition to having access to d-port input, this *value-judgment* function communicates directly on the open c-port channel. It provides a function capable of making qualitative decisions and appropriately biasing the activities of both sensory perception and behavior generation [5, 12].

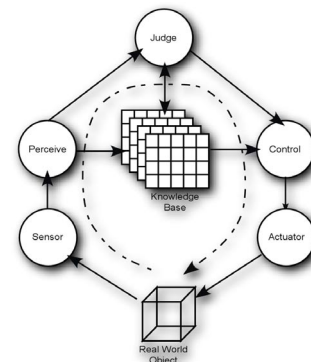


Figure 5 - A Control Server

Figure 6 is a schematic of an activity in an industrial plant-level communications and control intranet. It combines Figures 1, 4 and 5 to express the notion of vertical computation shared among concurrent processes cooperating on production objectives. This execution model is bi-directional, with *reactive threads* of execution flowing from the lowest level upwards. In the opposite direction, *proactive threads* flow from the policy levels downward towards the machinery on the plant floor.

Figure 6 is not meant to imply a static “stove-pipe” arrangement among cooperating objects. On the contrary, the figure is meant to represent the dynamic structure of the system at given instant in time. It depicts a specific thread of execution flowing through objects engaged in carrying out a particular control strategy. The dynamic configuration comes about by the invocation pattern resulting from message flows among the federated participants. Indeed, cybernetics requires that these associations be dynamic and allowed to selectively reorganize under evolving conditions, pressured both from externally and internally derived situations specified by policy.

To emphasize the real-time (timeliness) properties of vertically integrated enterprise controls, Figure 7 provides a view of the event horizons for each of the levels depicted in Figures 1 and 6. Plant level activities have time horizons that span days or weeks. Area level activities operate on horizons of hours, and so on down to equipment controls that may be in the millisecond range. Yet each of these levels affects the ones above and below: command threads flow downward, response threads flow upwards.

Note that the figure identifies “present time”. Preceding events are history and are plotted to the left. Activities planned for the future are shown to the right. The shading emphasizes the depth of the historical record retained and the number of future milestones planned at each level. Both the historical event record and the forward plans are contained in the distributed world-model databases of each object at each level that is participating in a given activity.

Figure 7 also indicates the nature of control strategies at various levels of the enterprise. At the

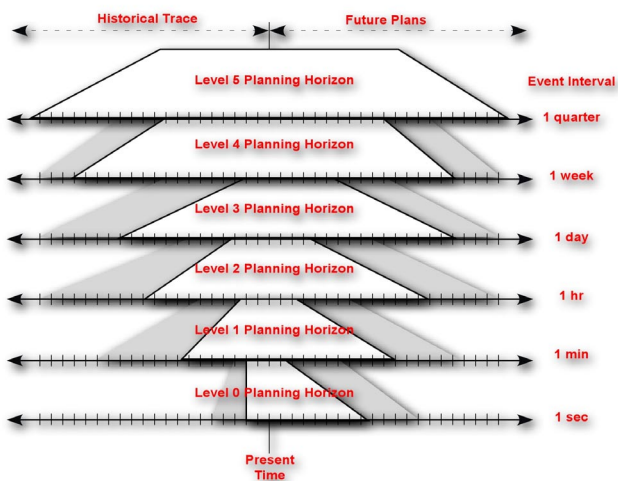


Figure 7 - Planning & Event Horizons

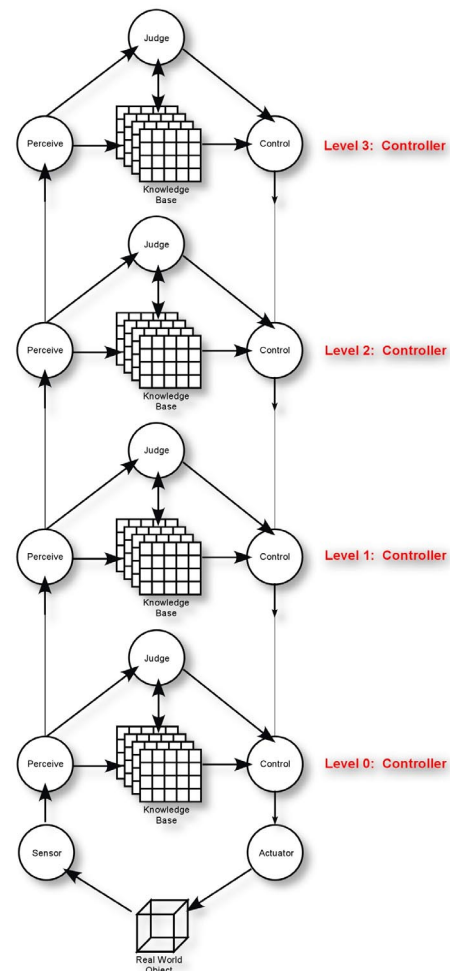


Figure 6 - Vertically Integrated Applications

lowest level activities are synchronous, operating on periodic clocks. At the higher levels, activities are typically asynchronous, and triggered by supervisory interventions by humans that are driven either by external or unplanned events. Policy change is the result of [presumably] intelligent developmental activities that serve to integrate the dynamics and uncertainties of complex operational situations. These command interventions can force dynamic replanning and reprogramming up and down the operational fabric.

An industrial system designed around a well-defined manufacturing process that produces a long-lived product or service will not suffer major upsets other than planned or unplanned start-up, maintenance, and upgrade sequences. However, in today’s innovative high technology driven market places, product half-lives are short and manufacturing systems undergo constant alteration and redesign. These conditions dynamically couple plant level operating policies to complex corporate level development decisions with increasingly shortened planning horizons.

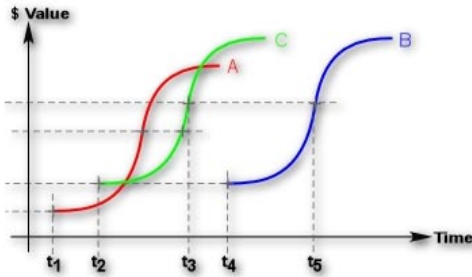


Figure 8 - Innovation Cycles

Figure 8 describes this increasing tension between the operational and developmental directorates in an enterprise. The graph plots “S” curves indicative of technical innovation, describing the lifecycle economic performance of product or technology. During the development phase of a product (e.g., t_1 to t_2) the asset “value” of the investment is low. Following introduction to the market (t_2 to t_3) the value increases as volumes and market share increase. At some point (t_3) the initial investment in product A is paid off, and the product begins to generate net profit. At a later point (t_4), product A’s value levels off and at maturity typically begins to deteriorate. Generally, only then is new development investment warranted. In the mean time, stable manufacturing systems can be developed and efficiently operated.

When innovation is low, or the product enjoys a proprietary advantage in the market, new investment (products B and C) may be postponed until well into the lifecycle of product A. However, when innovation or competitive pressures are high, developments in new technology B must begin prior to payback on A, and product C must be in the research stages. This continuous, overlapping situation represents a complex financial problem, and places the operational objectives of manufacturing plants in direct conflict with those of the business development directorate. The plants wish to have stable production regimes, yet continuous innovation requires constant adjustments in the scheduling of various manufacturing processes. In addition, R&D investments in skills and capacity are increased and require careful resource allocation strategies to minimize both costs and risks. It is precisely this type of complexity and conflict that requisite enterprise metasystems serve to address.

Figure 9 depicts a federated system, a control domain embedded within an interactive environment. The domain contains three cooperating processes identified as A, B and C. By way of example, they might represent three separate manufacturing areas within a single plant, with the

shared objective of maximizing total plant production while minimizing raw material usage. Each area independently progresses its own localized production agenda while interacting on the shared objectives. Each is free to self-organize, but must cooperate through dialogs on their respective c-ports. They serve their local markets (up-stream servers and down-stream clients) through their d, s, and e-ports. They each accept plant operating policy on their p-ports.

Figure 10 depicts a metasystem by embedding the federated ensemble of Figure 9 under a superior authority (SA) comprising the higher-order perception, value judgment and control functions. The shaded area highlights the core function at this level—adjusting system resources to meet overall market demands within specified levels of performance. This is the essential role of the management board of Figure 3. The feedback arrangement governs the mapping of global objectives onto local behaviors. The SA itself is composed of two parts, its own world-model containing the definition of the system under its authority, and a high-level value-judgment process. Through these two resources the SA provides “enlightened” supervision of the system [4].

The feedback coupling to the SA value-judgment process contains four elements: 1) a model of the metasystem itself, providing an interpreter for messages expressed in the metalanguage; 2) a set of filters that can identify and process signals of interest in measuring environment behavior and system performance; 3) a specification of the planned capacity of the system; and 4) a specification of the objectives (behaviors or outcomes) expected of the system. Through SA’s behavior-generator, incentives are sent to the system and distributed through locally controlled allocations to the p-ports of the objects.

A feedback process provides the supervisory, or

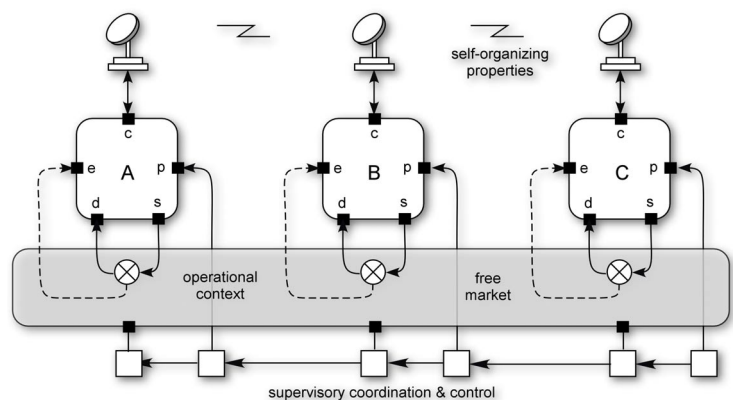


Figure 9 - A Federated System

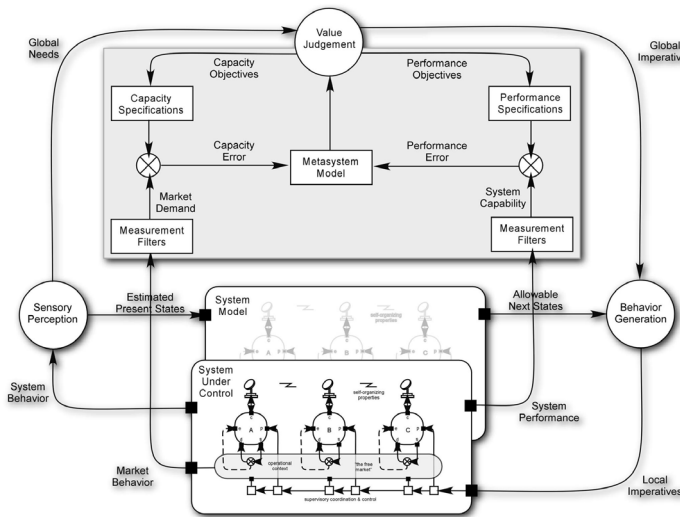


Figure 10 – Metasystem: Encapsulated System

metacontrols. The SA value-judgment loop sets capacity and performance objectives. Capacity and outcome specifications are derived from these objectives by comparison to actual (measured) system performance. These specifications are compared, with the “performance error” being sent to the meta-model controller. This difference is identified as the “performance error”, and when compared with the actual “free market” demand coming from the system, provides the modulating controls to raise or lower capacity, and to reset performance objectives.

Variations of the model presented above have found application since the early 1980’s in such diverse fields as public health, military command and control, economics, and neurobiology. Its application to industrial management and control problems, especially real-time control of distributed industrial process plants, is relatively new. This paper provides the framework for implementation of both distributed computing infrastructure and control software applications.

Summary

The intersection of the fields of systems science, computer science, and enterprise management provides a rich environment in which to address the issues of automation and control in large-scale interactive systems. The development of intelligent automation systems, including the effective management of populations of adaptive semi-autonomous agents (so-called “autonomic systems”) is a fertile area for further research and development.

References

- [1] Albus, J.S., *Outline for a Theory of Intelligence*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, No. 3, May/June 1991.
- [2] Antsaklis, P.J., Passino, K.M. and Wang, S.J. *Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues*, Journal of Intelligent and Robotic Systems 1:315-342, 1989.
- [3] Beer, S., *Platform for Change*, John Wiley & Son, 1975.
- [4] Beer, S., *The Brain of the Firm*, 2nd Edition, John Wiley & Son, 1995.
- [5] Brown, R. A., *Machines That Learn*, Oxford University Press, 1994.
- [6] Chu, K-C., *Team Decision Theory and Information Structures in Optimal Control Problems—Parts I & II*, IEEE Transactions on Automatic Control, Vol. AC-17, No. 1, February 1972.
- [7] Conant, R.C., *Laws of Information which Govern Systems*, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-6, No 4, April 1976.
- [8] Kauffman, S., *The Origins of Order*, Oxford University Press, 1993.
- [9] Mesarovic, M., Macko, D. and Takahara, Y., *Theory of Hierarchical, Multilevel Systems*, Academic Press, 1970.
- [10] Selic, B., Gullekson, G., and Ward, P., *Real-time Object-oriented Modeling*, Wiley, 1994
- [11] Singh, M.G. and Hindi, K., *A Multilevel Multilayer Framework for Manufacturing Control*, Journal of Intelligent and Robotic Systems, 4: 75-93, 1991.
- [12] Stephanou, H.E., and Erkman, A.M., *Model-based Sensory Pattern Fusion*, Journal of Robotic Systems, Vol. 7, No. 3, 1990.
- [13] Wiener, N., *Cybernetics*, MIT Press, 1948.