AN ARCHITECTURE FOR FACTORY CONTROL AUTOMATION

Richard L. Campbell

Richard L. Campbell *is head of the Systems Integration Department at AT&T Engineering Research Center in Princeton, New Jersey. He is responsible for research and development to support AT&T manufacturing in robotics, machine vision, and controllers. Mr. Campbell, who joined the company in 1979, has a B.S. from the University of Maine and an M.S. from New York University, both in electrical engineering.*

The automation of manufacturing functions can be usefully categorized as physical, control, and information automation. Because control and information automation are functionally dedicated to processing data, they are founded in computing and data communications technology. But they characteristically differ in important ways, such as response-time requirements, variability of functionality, and volume and volatility of associated data. Thus, different computing architectures are appropriate for these automation arenas. This paper outlines the architectural considerations that derive from the functional and performance requirements typical in control automation. It also suggests an architectural approach to the automation of factory control functions.

Functional Architecture for Manufacturing

As a starting point for developing an approach to improving manufacturing effectiveness, it is useful to group relevant functions to facilitate understanding of manufacturing operations. While there is no broad industry agreement about the exact way to partition manufacturing functions, most serious attempts use five to eight different groupings, typically identified as functional *levels* in a roughly hierarchical *functional architecture.*

In AT&T, a seven-level functional architecture has been defined to provide the company with a standardized computer-integrated manufacturing architecture (CIMA). Figure 1 identifies these "CIMA levels" and characterizes the functions for each level.

Automation Domains

A manufacturing operation adds value by physically operating on materials and components to create products. These physical operations, which may be effected by machines or people, are characterized in our CIMA functional architecture as "sense physical conditions"

and "make physical transformations." They reside at the bottom of our functional hierarchy (lower portion of the "equipment level").

All functions above this domain of physical transformation functions operate on information—not physical objects—and do not directly add value to the product. Instead, these information-processing "overhead" functions generally aid the value-adding physical domain functions in a way that results in a financial profit.

A cursory review of these information processing functions shows that a subset is tightly coupled to the physical transformation functions. This subset directly controls and coordinates the operation of the machines (or people) that are sensing the environment and making the physical transformations. Thus, these *control functions* are distinguished from other *information functions* that are not concerned with the specifics of a production process—except in terms of productive capabilities, operational status, cost, and similar characteristics important to managing a production facility.

This (admittedly qualitative) partitioning of functions into *physical, control,* and *management* domains applies even if substantive automation is not implemented. In the simplest conceivable single-owner-operated manual manufacturing operation (the prototypical "garage shop"), the physical transformation of materials into products is accomplished by:

- Human sensory organs (sensing physical conditions) and manipulative organs (making physical transformations), which form a physical domain
- Human brain and nervous system, which effect direct control of the physical elements to make the desired physical transformations
- Human capabilities (primarily information processing done by brain and nervous system, but supplemented by abilities to accept and record information from and to external sources), which analyze, plan, and manage in such diverse areas as customer needs; product features and pricing; materials, tools, and facilities procurement; financing; accounting; etc.

This example is perhaps useful for clarifying the functional partitioning suggested above, but adds little of practical value for improving this simple manufacturing operation.

As a manual manufacturing enterprise becomes more complex, it involves many people who must work cooperatively to achieve the overall goal: to transform materials into products for acceptable profit. The CIMA structuring now becomes valuable as a guide for *organizing* the operation. Thus, when a garage shop grows to a multi-person enterprise, the physical transformation and related control functions typically become the responsibility of a production organization. The analysis, planning, and management functions then are assigned to engineering, administrative, and management organizations.

This analysis may imply that our CIMA is a guide to an appropriate organizational structure. But the reality is that the traditional organization of large manufacturing operations provides an understandable structural framework that establishes a foundation for a useful CIMA.

Let us shift our consideration from a manual to an *automated* manufacturing operation, one where some functions are done by machines, including computers. The partitioning suggested above now provides a useful framework for developing an *automation architecture.*

Computer technology has evolved to the point where complex physical processes usually can be efficiently controlled by:

1. *Extracting* (via sensors) information about the process in an abstracted electronic form (generally as *digital* electronic data)
2. *Manipulating* the abstracted information using algorithms that suitably model the process and incorporate information about the desired results
3. *Delivering* the new information to devices (effectors) that change the physical parameters of the process to achieve the desired results.

Thus, the separation of physical and information domain functions, as suggested by our model, is consistent with a functional division that current automation technology can

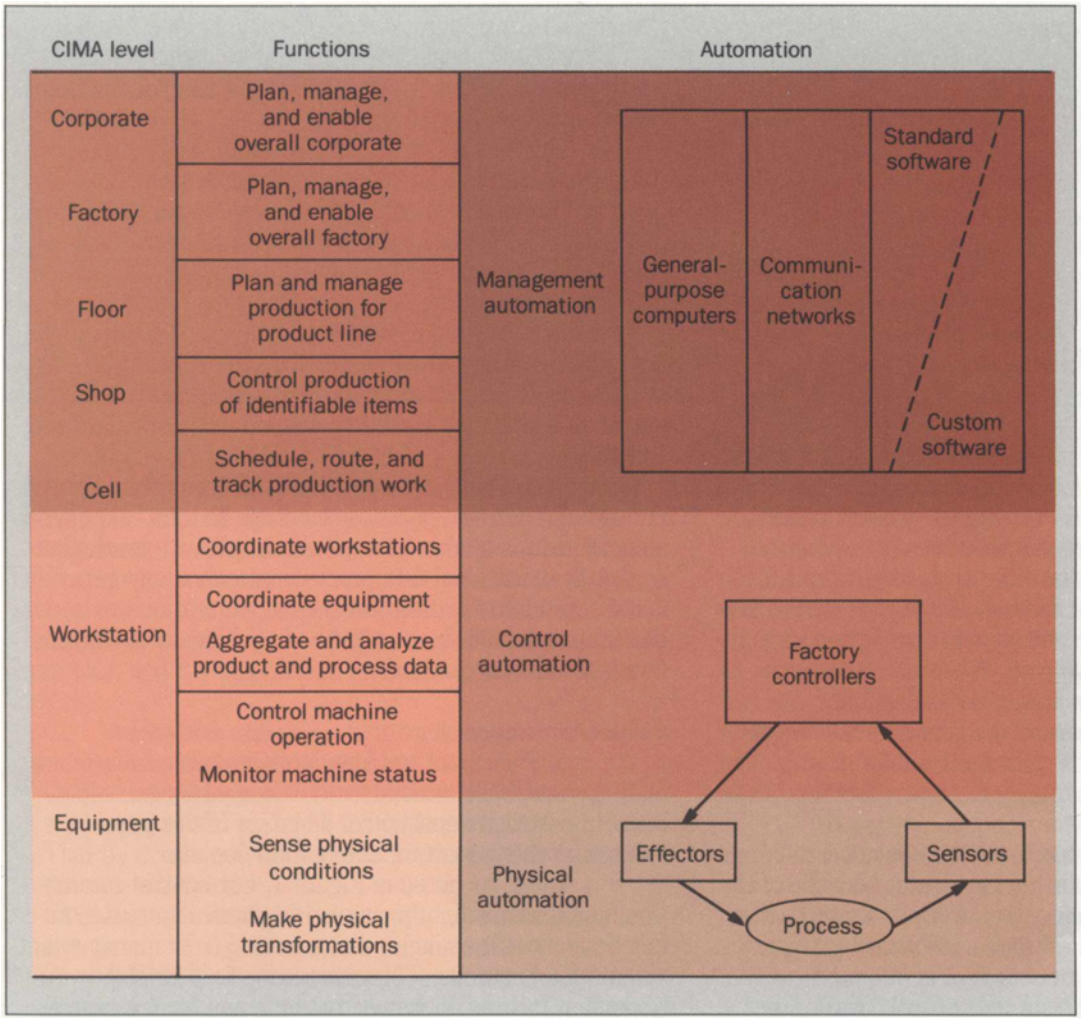

Figure 1. CIMA (computer integrated manufacturing architecture) provides a functional framework for organizing manufacturing operations and helps establish functional domains that are useful for structuring automation within factories.

efficiently support.

Perhaps less obvious is the advisability of distinguishing between control and management functions within the information domain in our automation model. However, given the current state of digital computing technology, such a division is useful because of substantive differences in these two domains—differences that are relevant to the selection of a computing approach. In particular,

- *Data volume*—In the control domain, the extent and nature of techniques for process sensing greatly influence data volume. As quality requirements and process

Panel 1. Acronyms in This Paper

CIMA	computer-integrated manufacturing architecture
COMASE	common manufacturing application support environment
LEC	logical equipment controller
LSC	logical station controller
LCC	logical cell controller
POMASE	problem-oriented manufacturing-application support environments
SVID	System V interface definition
VME	Versa Module European (computer bus standard)

sophistication drive us to ever-tighter process control, the volume of sensor data that needs to be assimilated grows apace. (Vision system sensing is a good example.) In the management domain, the overall size of the manufacturing operation and complexity of products largely determine data volume. While these data volumes may be very large, they are more stable.

- *Data volatility*—In the control domain, data volatility is very high compared to the management domain because the value of sensor information for controlling a process decreases rapidly with time.
- *Computational responsiveness*—In the control domain, these requirements may often be severe, because of the need to provide effective control over processes that operate in time frames that range downward to microseconds.
- *Variability of functionality*—In the control domain, the variability needed is at least as great as in the management domain and often cannot be accommodated fully by application software.
- *Accommodation of technology change*—For the control domain, this is a key consideration. If we cannot rapidly take advantage of new technology for physical automation, we compromise the fundamental ability of a manufacturing operation to add value competitively.

These considerations lead us to adopt different automation models for management and control functional domains, as Figure 1 shows.

The management domain can be well accommodated by "traditional" data-processing approaches. They generally presume a standardized, general-purpose computing and communications capability and try to adapt to an application environment by suitably organizing application functions. Given some basic standardization edicts (e.g., programming language, database manager), the architectural issues in this domain depend on the specifics of the required end-user functionality and revolve around the extent to which they can be addressed using standardized software.

In contrast, the control domain requires a computing approach that is less susceptible to total standardization below the application software level. Consequently, traditional data processing technology generally is not appropriate to this domain and, for various reasons, the control domain has not yet been subjected to similar forces for standardization.

Control Automation

Architectural and standardization issues for automating control functions are increasingly recognized as being of practical importance. The rest of this paper focuses on this important automation domain.

Model. As noted in Figure 1, our control automation functions are tightly coupled to physical automation functions, but (seemingly) disconnected from management automation functions. This characteristic is helpful, because it focuses on issues that emanate from a process control model. As we discuss our control automation architecture, it will become clear that we are simply applying some technical judgements on how to gain the benefits of standardization while accommodating rapid technological change and, thereby, establishing guidelines for useful implementations of closed-loop, process-control automation facilities.

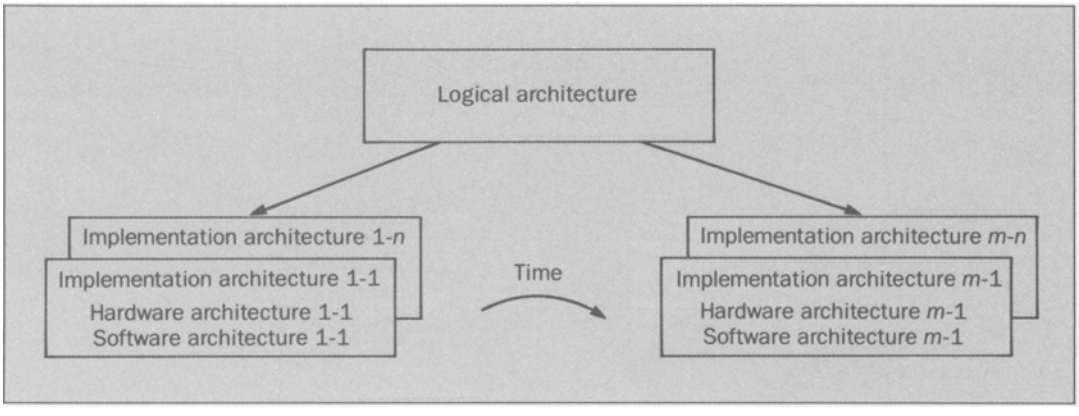

Figure 2. A logical control architecture provides a stable overarching structure wherein variable implementation architectures can evolve. A standardization strategy is imposed to constrain variability and provide manageability.

Less obvious from Figure 1, yet of considerable significance, is that the production domain (i.e., physical domain plus control domain) must be connected to the management domain to achieve a useful operation. We will not further examine the (considerable) architectural and standardization issues related to this important interface. Clearly, this interface is needed (consider our garage shop example), and it's a nontrivial interface in our automation model.

Architecture Objectives. In formulating an architecture for control automation, we have two objectives:

- Outline a structure of control functionality that will serve as a useful guideline for system implementors. That is, it will lead to cost-effective solutions that use current technology.
- Establish a framework that can accommodate the inherent variety of implementation requirements and inevitable technological change over time in a way that is graceful and conserves resources. That is, the framework leads to global optimization in a highly variable, changing environment.

Our overall model appears in Figure 2. We strive to establish a *logical architecture* that ideally remains stable indefinitely and provides a unifying framework for implementations. Further, we strive to establish a *standardization strategy* that constrains the variability and changeability of implementations, without unduly impeding our ability to introduce beneficial new technology as it becomes available.

Logical Architecture. The architecture that we propose for control automation consists of three "logical" controllers: a logical equipment controller (LEC), logical station controller (LSC), and logical cell controller (LCC). (A *logical controller* is an element in a control architecture that provides designated functionality, but whose physical realization is unspecified.) Each controller performs the functions of the corresponding CIMA level, functions that we have roughly characterized in Figure 1.

We further propose to constrain the interaction allowed among these logical control elements. We stipulate a hierarchical control regime, allowing peer-to-peer information transfer only when it is explicitly enabled by the supervisory controllers that "own" the communicating elements. (Exceptions to this strict hierarchical control scheme would be admitted for manual overrides, such as **emergency stop**.)

This highly simple logical architecture (depicted in Figure 3) appears to be usable at the lower levels of our CIMA structure. Implementations can be created that follow these constraints, are cost-effective, and can be efficiently maintained. However, this does not appear to be the case at higher levels, where more elaborate logical

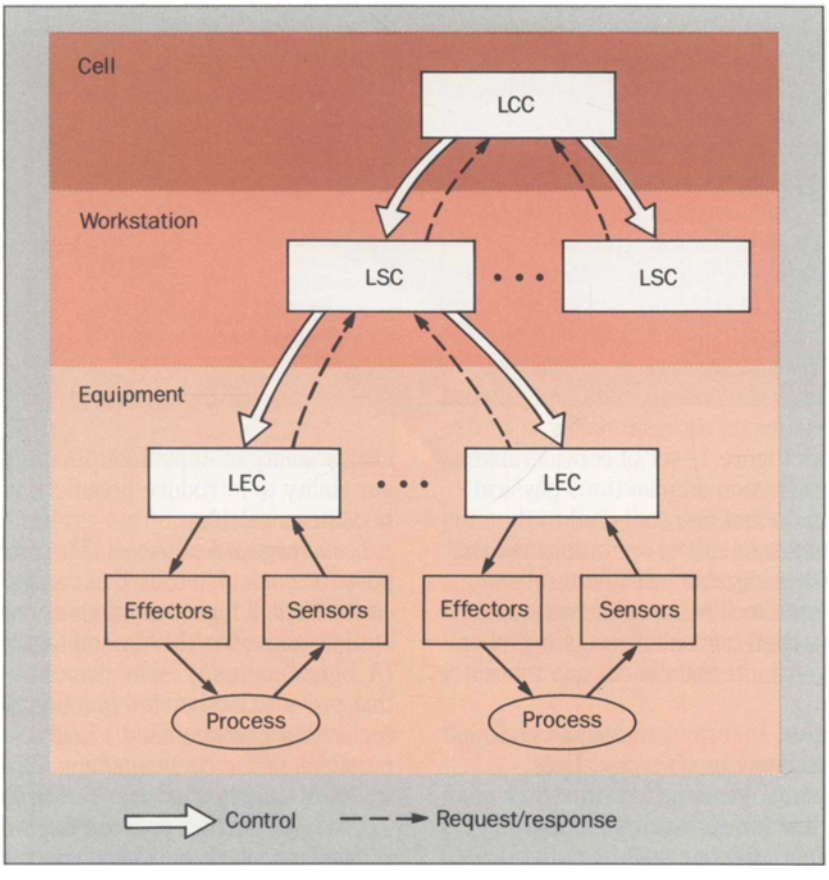

Figure 3. The suggested logical architecture provides "logical controllers" that supply each CIMA level's functionality. At cell level and below, a strict control hierarchy is imposed, where peer-to-peer data exchanges can occur only if directed by a higher level controller. LCC = logical cell controller; LEC = logical equipment controller; LSC = logical station controller.

linkages are required among the various information-processing domains (e.g., resource planning, product and process engineering, production management).

Implementation Architectures. Our generic model (Figure 2) depicts a multitude of implementation architectures and shows their changeability over time. To manage this variable, changeable situation, we impose a standardization strategy: We identify certain key system interfaces and require that they be well defined and kept stable over time.

This standardization leads to a *template* for allowable implementation architectures. Figure 4 is such a template. It establishes a standardization model for control automation implementations that, we suggest, appropriately constrains variability to achieve manageability. Still it is flexible enough to allow new technology to be assimilated with minimum disruption.

The template in Figure 4 incorporates the following elements that we believe are important for meeting our control architecture objectives:

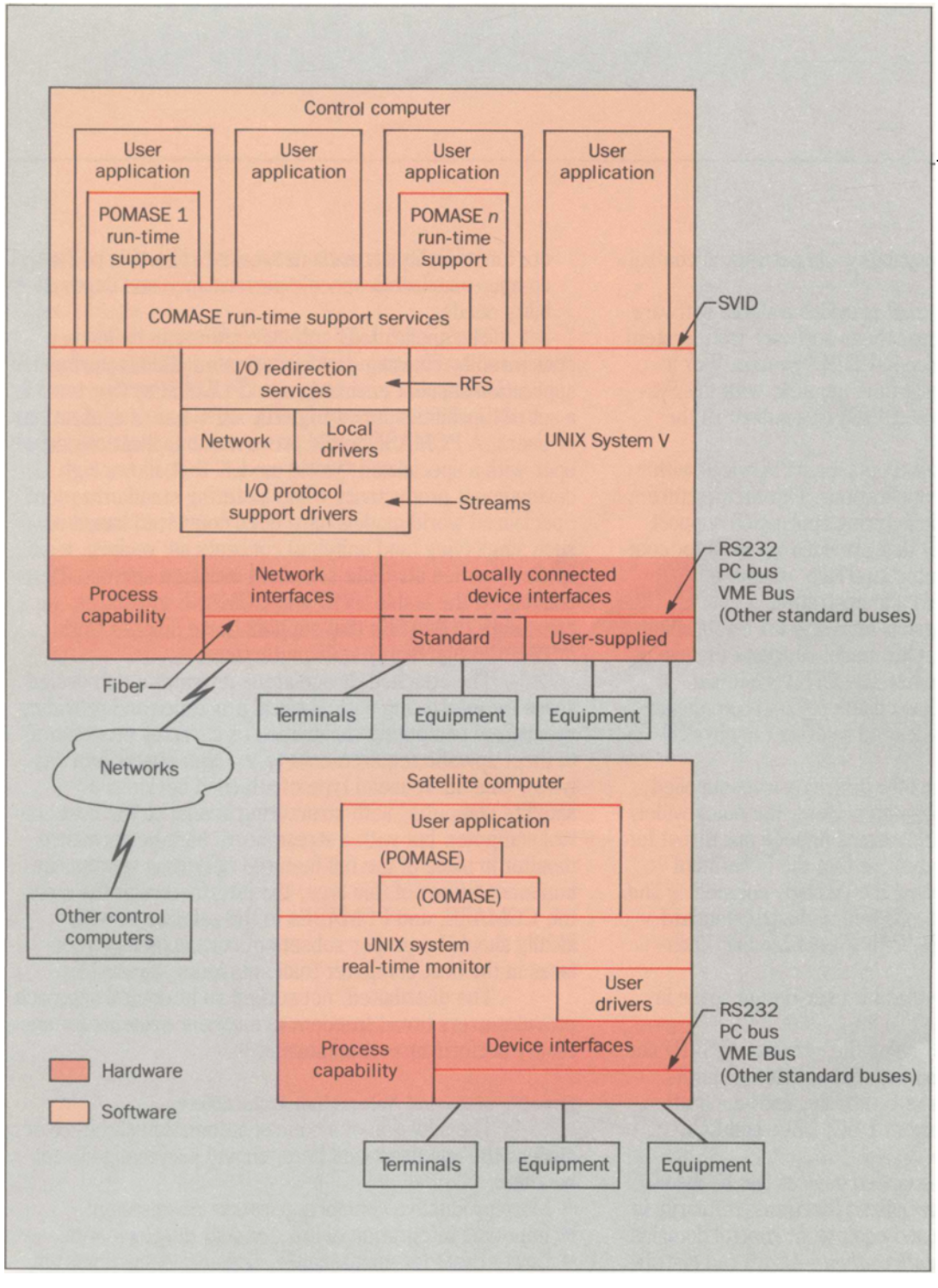

Figure 4. This template for implementation architectures stabilizes certain key hardware and software interfaces, identified by heavy (red), horizontal lines in this diagram. The UNIX System V software provides task scheduling and resource management. COMASE = common manufacturing application support environment; POMASE = problem-oriented manufacturing-application support environment; RFS = remote file sharing; SVID = System V interface definition; VME = Versa Module European, a computer bus standard.

- Distributed processing capability, via networked control computers.
- Single operating system that provides a stable software interface to decouple applications software from system underpinnings. Our choice is UNIX® System V or a UNIX system-derived real-time monitor, with the System V interface definition (SVID) or a subset as the interface.
- A standardized set of run-time support services with broad utility for control applications. Our architecture includes a common manufacturing application support environment (COMASE) that provides services for command and control, operator interface, database management, and system administration.
- Standardized, stable physical interface for establishing networked connectivity. Our model suggests the use of fiber as the standard media. The UNIX system's `streams` capability allows users to build communications protocol stacks as desired to drive the physical media.
- Standardized mechanisms for attaching user-supplied devices to the control computer. Here, the many widely used interconnection mechanisms impede our thrust for standardization. But we believe that the benefits of standardization will become increasingly compelling and increase the use of cost-effective, industry standard approaches—such as the VME (Versa Module European) Bus.

Our template provides for user-design space in three important arenas:

- *Application software*—By using the services of SVID and COMASE and the capabilities that suitable general-purpose languages such as C provide, user-application programmers can implement LEC, LSC, and LCC functionality.
- *Attached devices*—User-supplied devices can be incorporated that perform specialized functions, primarily in the physical automation and equipment control domains.
- *Distributed computing configuration*—Users can distribute functionality across a networked collection of control computers to meet specific performance and dependability needs.

The applications software arena can be further structured by creating problem-oriented manufacturing-application support environments (POMASEs) that provide a set of capabilities tuned to particular types of applications or users. A POMASE would present the applications developer with a specialized "world model" that allows high development productivity while fostering standardization. Specialized world models have been conceived based on such simplifying (and unifying) concepts as: recipes, logic ladders, named attribute sets, and message servers. By building on the stable SVID and COMASE interfaces, we can create POMASEs that support these diverse world models through highly stable interfaces.

The attached-device arena provides a rich design space for interfacing with physical processes and providing specialized computing capabilities (e.g., array processors) to meet specific requirements (e.g., high-speed data analysis). One often useful type of attached device is a satellite computer, with structuring similar to the host control computer, but with a streamlined, high-performance monitor in place of the full-featured operating system. In implementations of this type, the interfaces that the monitor, COMASE, and POMASEs in the satellite provide ideally should be proper subsets of corresponding interfaces in the host computer (null sets being admissible).

The distributed, networked architectural approach provides users broad freedom to engineer systems for necessary performance at minimal cost.

Benefits of Control Automation Architecture

The adoption of a control automation architecture, such as the one described here, should lead to significant benefits:

- More productive control application development
- Improved integration of independent developments
- Lower costs for maintenance, training, documentation,

and support of factory automation systems
- Reduced costs for assimilating new beneficial technology.

While these benefits are difficult to quantify, evidence suggests that major manufacturing operations—such as those within AT&T—can realize very substantial savings on an ongoing basis through systematic adoption of such a standardized control architecture.

(Manuscript received May 28, 1987)