Development of a Structural Philosophy for an Automatic Network Control System

Terry M. Linden and James F. Thompson, Transportation Systems Division, General Motors Corporation
Frank E. Tillotson, Delco Electronics Division, General Motors Corporation

Control of a set of vehicles on an automated network can be divided into two parts: vehicle control, which deals with the control of an individual vehicle, and network control, which deals with the management of vehicle interaction on the network. This paper discusses a structural philosophy for network control and the development process that led to the philosophy. The process started with the development of the automated network system, which resulted in selecting a deterministic synchronous slot scheme as the fundamental control methodology, in partitioning computer control of the system into a three-level hierarchy, and in partitioning the automated network system into vehicle control and network control, or network management, subfunctions. Development of the management system structure involved an investigation of the relation between management system structure and direct system costs, system flexibility, potential level of technology increases, and functions the system was designated to perform. The resultant structure is characterized by three primary aspects: the organization and sequencing of system tasks, the level at which the management system perceives the real world, and the way the model set is used to respond to real-world anomalies to execute normal operational strategies related to system changes. Each task is performed by a discrete computer "job" that has an assigned priority or desired time of execution. A set of "primitive models" can be linked together to define the system or configure the system "image" or view of the real world. The system can reconfigure the image, including addition, deletion, or replacement of a model or model group. In the case of a real-world anomaly, the reconfiguration process includes detection and circumvention of any illegal real-world states that may have arisen as a result of the anomaly.

A system structure should facilitate attainment of goals set for the system. One major aspect of system structure is its inherent type of modularization. Modularization is a process in which a system is divided into small, stand-alone units that have well-defined boundaries. The boundary conditions encompass inputs, outputs, and an input-output transfer function. The individual modules may be interconnected in several ways. Interconnection organizations may be categorized into three fundamental types: hierarchical, central flow, and state block.

In a hierarchical organization, modules are composed of submodules at the next hierarchical level. Control flows from module to module by passing up and down through the hierarchical level. Decisions regarding flow between modules at the same level are made at the next higher hierarchical level. For example, assume module A is composed of submodules B and C and module B is composed of submodules D and E. Upon completion of its task, module D passes control to module B. Module B can pass control back to D, pass control to E, or relinquish control directly to A. In this scheme, decisions are local in nature; they relate to the particular module function and whether that function is complete. Also, much of the processing is often done in lower levels of the hierarchy, while the higher levels, which increasingly tend to become abstractions of the problem and functional groupings of modules, involve less processing.

In a central flow organization, the system consists of central flow logic and a set of modules. Module to module flow is via, and flow decisions are made in, the central flow logic. The central logic, in its decision making, has system cognizance as opposed to the local decision making done in a hierarchical structure.

A state block organization is similar to a hierarchical organization in that modules are composed of submodules at the next hierarchical level. However, control flows between modules at the same level directly instead of passing through the hierarchical levels; module flow decisions are made in the originating module.

Operating systems generally use all three interconnection organization types. The interconnection types are important because they lend a logical framework to a system. As such, although each should be used to its best advantage, care should be taken to avoid overmixing of the techniques, both in original design and subsequent maintenance functions. If this is not done, the logical framework becomes so complex as to be worthless.

The characteristics of a modularized system are related to the interconnection organization and to the existence and nature of the module boundaries. Modularization inherently facilitates system perception and comprehension. Since the boundary conditions represent a simplified statement of the internal workings of a module, modularization allows a user to perceive the system in terms of the boundary conditions of the component modules rather than in terms of the more complex collective internal workings of all the modules.

A still higher level of perception of the modules can be gained by defining the modules as functional blocks without addressing the exact input-output data and

transfer function required to actually achieve the function. This allows a user to deal with most of a system at a high level and to go into more detail, via the boundary conditions, in an area of immediate concern. Hence, a user may deal with different system areas at a level appropriate for the task, thereby enhancing his or her perception by eliminating unnecessary detail. The effects of modularization in this area can be enhanced by modularizing along functional lines that are easily related to real-world entities.

Modularization also, for the same reasons, allows a user to learn a system from the top down and at an appropriate level for his or her task. Unmodularized systems often have a "can't-see-the-forest-through-the-trees" effect, which requires learning an entire system before being able to fully comprehend the implications of any one part. Modularization minimizes this problem. In large systems, the central flow organization may be of limited value in this area. The hierarchical and state block organizations, by composing modules of submodules, allow a user to perceive a system from as many levels as desired.

Because of the stand-alone aspect of the modules, modularization enhances system flexibility with respect to system growth. When used to enhance system flexibility, the objective of modularization is to allow addition, deletion, or changing of part of the system with a minimum impact on the rest of the system, or, more briefly, to preserve the whole by dividing the system into replaceable components. The module boundaries provide the boundaries for system change. Since the boundary conditions are well defined, a module or group of modules may be replaced with another module with identical boundary conditions, but totally different internal workings, with no impact on the surrounding modules. Since the module and change boundaries are to correlate, effective modularization requires anticipation of areas of change in the system. If the functional application of the system will expand or if specific functional areas will change, then modularization should be along functional lines. If level of technology increases are likely to occur in certain areas of the system, modularization should isolate those areas so as to allow incorporation of the technological breakthroughs as they occur.

The existence of well-defined boundaries simplifies system testing. A module can be tested by supplying the inputs and verifying the outputs. As modules are completed, their interfaces with others can be checked. In a hierarchical structure in particular, testing can proceed up through the hierarchy levels until the entire system is checked out. In some systems, the module interconnection design can be tested by computer simulations in which the boundary conditions are simulated. This allows parallel check-out of system and individual module design.

By providing modules that can be time multiplexed among many users, modularization can reduce system size. If several users need to perform approximately the same function, it may be more cost effective to have several of them use a single module inefficiently than to have each of them do his or her own processing efficiently.

Modularization has some negative effects. The inherent well-defined boundary aspect requires that the boundaries be imposed on what might be an integrated system. Imposition of the boundaries costs time and money. Of necessity, the boundaries are often somewhat artificial. The artificiality is further extended because, in general, the signals and data need to be collected and formatted for presentation at the module boundary; data and signal buffering is also often required. The collection, formatting, and buffering logic costs money to design and mechanize; mechanization costs include the additional storage, be it hardware or software, costs associated with the additional logic. Also, the additional collection and formatting logic often consumes time, thereby making the module slower. Hence, an objective with regard to minimizing the negative effects of the "artificial module boundaries" is to modularize in a fashion that minimizes intermodule communication. Any use of modularization requires a trade-off between its positive and negative aspects.

System structure can have a minor impact on attainment of goals set for a system. Too often, however, insufficient attention is paid to system structure. One cause is the failure to recognize how widespread the effects of system structure are. For example, system structure will have a major impact on the partitioning of a system into subsystems for development. The development of the subsystems will require communication among personnel working on the various subsystems. The quantity of communication will affect system development costs. Hence, structure may affect cost by affecting intrapersonnel communication.

Another common cause of insufficient structure development is the failure to distinguish between system modularity and system structure. This is particularly true when the system is modularized to enhance system flexibility. In this case the system is partitioned into small, stand-alone units that can be replaced or rearranged with a minimum impact on the remainder of the system. Too often, the stand-alone attribute is the only criterion used to dictate what should constitute a particular module. One result is that, since a system approach is not used, the modules are difficult to interconnect and can be used by only a few. Another result is that the boundaries of modules and groups of modules and the boundaries of future changes will not correlate. A rigorous structure development would address future changes and would dictate module boundaries accordingly. Hence, system structure actually dictates the benefits that can be obtained from system modularity, and the lack of a system structure may negate any such benefits.

An important aspect of the development process is to halt the development process in any area when a point of diminishing returns has been reached. If this is not done, time will be wasted collecting and analyzing data that cannot directly be translated into rigorous guidelines for system structure. One cause of failure to halt the process is the desire to generate the appearance of having performed a "highly detailed study of the problem." Another cause is not recognizing the point of diminishing returns. At each step in the development process, the feasibility of generating any further meaningful conclusions should be analyzed before proceeding.

DEVELOPMENT

Prenetwork Management Development

The development of the management system was preceded by the development of the concept for the automated network system. The system concept evolved approximately as described below.

After consideration of various basic control methodologies, a deterministic synchronous slot scheme was selected. The synchronous slot scheme is a technique for moving vehicles through the network in a synchronized fashion. The location of each slot is defined by a set of mathematical equations in the system management computer. The slots move through the guideway network at synchronous speed. A slot that comes to a diverge divides into two slots, one proceeding down each branch

of the diverge. Similarly, slots that meet at a merge combine to form a single slot on the output link of the merge.

In the deterministic synchronous slot scheme, the management system routes a vehicle from origin to destination by simply reserving a slot on every link of the vehicle route. Since a slot can be occupied by only one vehicle, this enables the vehicles to move through the network without interference. In the deterministic scheme, all slot reservations are made before the vehicle is launched onto the guideway. Therefore, completion of the vehicle route, without conflict, is ensured before the vehicle enters the guideway network.

Since a slot conflict with another vehicle can occur at any merge along the vehicle route, the management system may have to try several different slots before it finds a slot that goes from the vehicle origin to its destination without conflicts on any network link. To increase the probability of finding a successful slot, the system can use a slot slip technique. In this technique a vehicle is scheduled to slip back one or more slots as it traverses a link. For example, a vehicle may be able to go through a merge without a conflict only to encounter a conflict at the next merge. If the slot behind the vehicle clears before the next merge and that slot is available at the next merge, the management system may elect to command that vehicle to slip into that slot before arriving at the merge.

In the deterministic scheme, all slips along a vehicle route are also planned before the vehicle is launched onto the guideway; hence, all vehicle slot assignments are known before vehicle launch. The slot length is transparent to the synchronous slot scheme. This means a system can use any slot length without affecting the concept of the synchronous slot scheme. The slot length for a particular system is designed to satisfy predefined performance criteria. A common criterion is the "brick-wall" stop. Here, the slot length is such that, if a vehicle had a brick-wall stop, the system could detect the blockage and stop the next vehicle on the guideway before it hit the first vehicle.

Next, the problem of control hierarchy was addressed. For vehicle control, an on-board logic set was required that could control the propulsion system and brakes, sense position and velocity, and detect vehicle and communication link anomalies. The vehicle data sampling rate and control loop times and the desire to avoid heavy dependence on ground-vehicle communication links dictated that a computer be placed on board the vehicle. If vehicles always behaved ideally and if station dwell times could be exactly predicted, vehicle on-board computers would be adequate for real-time control. However, to control vehicle interactions in a stream of contiguous vehicles under anomalous conditions and to compensate for inexact station dwell times that might result in a vehicle not leaving its berth at its predicted time, an off-vehicle computer was added to the vehicle's real-time control loop. The data sample rates were designed and two-way, fail-safe signals were used to ensure system safety in the event of a communication failure between the vehicle and the off-vehicle computers.

It was determined that a single off-vehicle computer would not be fast enough to perform real-time control of all vehicles in systems of the anticipated sizes. Therefore, the system was geographically partitioned into sectors, each of which had its own sector computer. To distribute the system management function among the sector computers is not cost effective or technically practical, and a system management computer was, therefore, added as the third level of hierarchy in the computer control scheme. The bulk of the management function is performed in the management computer; the sector computers provide a real-time data acquisition system for the management computer, translate management level commands from the management computer into vehicle level control commands for the vehicle computer, and provide an initial reflexive response to system anomalies. The reflexive response involves control of the interaction of adjacent vehicles and the notification of the management system of the anomaly and of the initial reflexive response.

The automated network system can be partitioned into two subfunctions: vehicle control and network management. The three-level computer control hierarchy, which had evolved independently of the functional partition, inherently contains the same vehicle control-network management partition. Therefore, the exact nature of the two subfunctions was clarified as follows: Vehicle control is, except for a reflexive response to anomalies affecting the interaction of adjacent vehicles, concerned with the control of a single vehicle with respect to a single control point; it is not concerned with the networkwide interaction of vehicles. Network management is concerned with the interaction of all vehicles in the system; it is not concerned with the microrelation of a single vehicle to a single control point. The level of perception required to perform these two functions is totally different. The vehicle control system must deal with real-world entities such as distance, velocity, acceleration, and jerk. However, the management system can perceive the system in terms of high-level models that inherently contain the requirements imposed on, but do not explicitly reflect, the aforementioned entities.

Network Management Structure Development

The first step in the development of the management system was to list the parameters that might have a significant effect on the management system structure. Ultimately all parameters relate to either cost, feasibility of completion within an alotted time period, or system function. So that analysis would not be extended beyond the point of diminishing returns, the initial list was generated in terms of the more direct effects of a parameter. For example, system flexibility may ultimately relate to the cost of replacing part of the system or totally regenerating the system. Rather than considering the cost of regenerating the system or reconfiguring the system, we considered flexibility as an entity in itself in the initial parameter list. The next step in the process was to define the relation between parameters and system structure attributes. An output of this step was a statement of desirable structure-related attributes as a function of each parameter.

One set of parameters considered consisted of the projected time period, the project staff, and the system development costs. The more staff a project has and the longer it is, the more highly partitioned it must be in order to allocate work among the project personnel. A heavily staffed or long project may, independent of cost, require special structural techniques for completion. To analyze these parameters, we used an ongoing simulation to determine the human effort required for the proposed system. A project time period was estimated, and an average staff size was calculated from the time period and human effort. The conclusion was that, although the staff size would require some system partitioning, these parameters would not have a significant effect on system structure.

Next, the time period was considered as a separate entity. In addition to constraints imposed by a long time period, a short time period might constrain the system structure so as to ensure completion of the project.

Based on the estimated time period, the conclusion was that this parameter would not affect system structure.

Finally, system development costs were considered. A major variable in system development cost is the quantity and efficiency of interpersonnel communication. Since it is difficult for two people to communicate about interfacing subsystems if they do not understand each other's areas and the system implications of the areas, communication efficiency is dependent on system comprehension. Based on estimated system size, the conclusion was that the comprehension afforded by the multiple perception levels of a hierarchical structure was desirable, but not necessary. To minimize the quantity and maximize the efficiency of interpersonnel communication, we further concluded that the system should be modularized along natural functional lines that can be related to real-world entities.

Another parameter considered was potential level of technology increases in the computing area. Major technological improvements in computers are related primarily to storage costs and computer processing speed. Programming techniques often involve a trade-off between implicit data structures that use little core, but require a great deal of processing to obtain the desired data, and explicit data structures that use much more core, but require little processing to obtain the desired data. The management system is characterized by several large data bases. Therefore, the preliminary conclusion was that functions that deal with large data bases and that may cause the memory and speed trade-off to become significant should be offset as discrete modules.

Several levels of technology increases, such as bubble memories, are on the horizon. Therefore, although we realized that it is difficult to estimate when these increases will become reality and that specific conclusions and a high degree of faith in extrapolating the data would be dangerous, we investigated past history to attempt to predict what the future might offer. In the area of minicomputers and microcomputers, computing costs have continued to plummet. The lower costs have made computer use feasible in areas that were not previously cost effective. Costs for larger system computing power have also been dropping quite steadily. For example, in the last 15 years, the cost per byte for large capacity disk storage has decreased by more than an order of magnitude. When data access time is figured into the ratio, the drop approaches two orders of magnitude. The cost effectiveness of central processing units and associated main frame storage, measured in cost per number of storage cycles per second per byte of storage, has been improving by an order of magnitude every 4 or 5 years for the past 20 years. This ratio does not take into effect the increases that have been made in fault tolerant computer design and instruction repertories during the same period. In both these examples, the improvement was due primarily to economies available with larger systems. In general, the improvements were so large in magnitude and scope that most data collected provided only "apples-to-oranges" comparisons. The conclusion was that, although valid comparisons might be made by collecting more data and performing more detailed analyses, any structural conclusions beyond the original "modularize along memory and speed technological lines" are highly subjective and limited in scope. Therefore, investigation in this area was halted.

The development also considered the nature of the management problem. Since, in the deterministic synchronous slot scheme the management system can preprogram the vehicle management system for the entire vehicle route before the vehicle is launched, the management system need not execute any real-time responses under normal operating conditions. In the event of an anomaly, the vehicle control system initially detects the anomaly and can stop the vehicles. This means that even under anomalous conditions the management system need not perform real-time responses. However, under this situation, vehicle stoppages quickly propagate throughout the network in a "vehicle-follower" fashion, thereby drastically reducing the system availability.

Therefore, if the vehicle-follower propagation problem is to be avoided, the management system should execute strategies that control the propagation of a blockage, thereby increasing system availability by keeping the majority of the vehicles en route to their destinations. This, in general, requires rerouting of vehicles heading for the blockage and temporary suspension of routing vehicles near the blockage area. Since vehicles on the guideway are en route to the blockage, this requires that the management system respond in real time. Therefore, the system must be able to quickly alter its normal task sequencing and execute those tasks that are necessary to respond to the faults. Furthermore, the system must be capable of responding to different anomalies in different portions of the system. Because of the large number of anomalies that could occur, a system that deals with specific combinations of anomalies will quickly get out of hand. The system either becomes large or has to degrade quickly in the face of multiple anomalies. A generalized system is desirable, therefore, that can respond to generalized multiple anomalies. Such a system must recognize when it is in a transition mode and responding to a current anomaly or set of anomalies. Since the management system is required to execute real-time responses to anomalies and since these responses may encompass a wide variety of combinations of responses, the system is virtually required to use an automated task-sequencing scheme. Rapid and safe response to anomalies could be ensured by the use of "canned" response sequences that are selected as a function of the anomaly and the current system state. Further analysis revealed that management system speed would be dictated by the anomaly response strategies designed to maintain system availability. This meant that, except in the malfunction response area, any reasonable system structure need not consider the effect of speed on system structure. This was important because the central processor and main-frame storage cost penalty for a change to a faster computer system is significant.

The general use of the system was also considered. The initial use of the system will be for prototype systems. This type of use implies that initially the program will be used at only one or two facilities and that the management system will likely undergo changes in functional areas. Since these prototype systems are likely to span several years, applications beyond that point will be affected by level of technology changes. Subsequent use will involve installation in a few cities to test system use in a real environment; this will again increase the impact of technological changes. These factors indicated that it would be necessary to have an extremely flexible and maintainable system to be cost effective.

A system might expand in the magnitude of the geographical area served or in the density of service offered. It might be installed in functional increments. For example, some systems may ultimately wish to accommodate mixed-vehicle operation. As the control technology is demonstrated in a system, it might be desirable to change the synchronous slot headway time or relax operating constraints such as use of discrete, as

opposed to continuous, merges. All these items strongly pointed to a high degree of modularization along functional lines. Since functional changes or additions must, in general, solve the problem of the proper sequencing of the corresponding modules, the conclusion was that a single, highly automated module sequencing scheme is desirable. Adapting to a change in density of a sector will be facilitated if the program and data structure are designed to handle a generalized sector. The attributes of a particular sector should be defined in the data base for that sector. Geographical expansion will be facilitated if the system treats sectors as generalized building blocks that can be configured in any number. Incremental functional installation of a system is facilitated if the system is partitioned along functional lines. Mixed vehicle operation is facilitated if the nature of the vehicle is transparent to the management system.

Finally, system maintenance was considered. The importance of system maintainability has already been established. System flexibility, by definition, facilitates system maintenance because it provides for system expansion and operation in a changing environment. Two other aspects of system maintenance are the ease, or difficulty, with which system flexibility can be exploited and the reliability of the maintenance work. Both aspects are intimately related to system comprehension and, since maintenance is often performed within a fixed time frame, to the rate at which system comprehension can be attained. System comprehension, and its rate, may be constrained by system complexity, by the effect of time on the designer's memory, or by the use of new personnel to perform the system maintenance.

Modularization, by its nature, reduces the apparent complexity of a system by partitioning the system into units that can be easily perceived. System designers separated from a system by time or new personnel must study a system to comprehend it. The multiple perception levels provided by modularization and a logical modularization structure facilitate system study. In particular, maintenance personnel can view different parts of the system at a level appropriate for the maintenance task at hand. For example, maintenance functions are often localized to a particular module at a sublevel in the structural hierarchy. The multiple perception levels allow maintenance personnel to view loosely related modules as functional black boxes while providing, by virtue of the boundary conditions, a detailed view of the modules affected by the maintenance function. Furthermore, the better comprehension provided by modularization enhances reliability of maintenance functions. Therefore, the conclusion was that a highly modularized system and the use of modularization to provide intermediate perception levels are highly desirable.

Another aspect of maintenance is the sequencing of new functional modules. Different users may wish to add functional modules that are unique to their particular applications. A problem with the addition of a functional module is establishment of the proper processing sequence. Therefore, an automated task sequencing scheme and the capability of a user to insert new modules in the task sequence without intimate familiarity with existing modules are desirable.

At this point, the logic of the parameter and desirable attributes was reviewed. Had there been major or numerous conflicts in the attribute dictates, the development process would have considered relating the attributes to costs, but this was not the case. The review also indicated that no significant conclusions about system structure would be achieved by further parameter study.

Taken alone, many of the dictates of individual parameters were trivial. However, taken as a group, the individual dictates consistently pointed toward a highly modularized system. This, however, did not contribute to a structural definition. Several parameters dictated modularization along management system functional and general technological lines. Since the technological lines related to computer speeds and storage costs and since the functional lines often were related to large data base needs, these two dictates were consistent. Although this provided a framework for modularization, it still did not contribute to a structural definition. Several parameters dictated a need for an automated task-sequencing scheme, which led directly to the management system executive scheme.

A point that recurred in different forms throughout the development was the desirability of using modularization to facilitate program understanding. In the consideration of the maintenance parameter, it specifically addressed using modularization to provide an intermediate level of perception of the system. This point connected with an earlier conclusion, drawn in the prenetwork management development, that the management system had its own unique and high level of perception of the real world. This led directly to the development of a generalized primitive model set to generate the image that modeled the real world for the management system. The level of perception offered by the primitive models was carefully defined to be appropriate for the function the management system was designed to perform. The following structural philosophy evolved from several iterations of a trial structure.

NETWORK MANAGEMENT SYSTEMS STRUCTURE

The network management systems structure is characterized by three primary aspects: executive job control, management image perception set, and dynamic image reconfiguration.

Executive Job Control

The automated network system appears to the management system as an event-oriented system. The management system must perform a particular function, or set of functions, in response to an event or in response to the absence of an anticipated event. Furthermore, the management system must, as a function of the current operating environment, perform the functions in the proper sequence. The management system uses an executive to manage event responses. The executive recognizes the availability of event data, recognizes events, recognizes the absence of events, and orders responses to events or missing events or both.

To perform these functions, the management system uses a timer-priority job scheme. The executive contains a set of timer jobs, priority jobs, and data cells. The jobs correspond to the functions, which are performed for a particular entity by applying the corresponding job to the data cell related to that entity. For example, slot assignment is a management function. Hence, the management system contains a slot assignment job. Each vehicle that requires a slot assignment is identified in a data cell. To generate a slot assignment for a particular vehicle, a slot assignment job is applied to the corresponding data cell. Hence, a job is executed by setting up a link to the data cell and executing the job software.

Priority jobs are sequenced as a function of priority; timer jobs are sequenced as a function of time. To manage the sequence, the executive maintains a priority

job list and timer job list. Each entry in the timer list contains the timer job name, the timer job time, and a link to the data cell on which the job is to operate. A timer job is initiated by, and executed in, a computer interrupt generated by a programmable timer (Figure 1). A priority job is executed when the job has the highest priority of all jobs currently in the priority job list and the computer is not in an interrupt state (Figure 2). A job, upon completion, returns to the executive. For a timer job completion, the executive sets the programmable timer to execute the next job in the timer job list. For a priority job completion, the executive initiates processing of the highest priority job remaining on the priority job list. The executive, upon request, will schedule a timer job on the timer job list (Figure 3). The job requester must provide the executive with the job name, the job time, and the data cell on which the job is to operate. Similarly, the executive, upon request, will schedule a priority job on the priority job list (Figure 4). The job requester must provide the executive with the job name, the job priority, and the data cell on which the job is to operate. A job may schedule any type of job. Hence, both a timer job and a priority job may schedule a priority job or a timer job.

The ordering of responses to events is inherent to the executive. To recognize and provide rapid response to events, the system uses external interrupts of the management system to process event data (Figure 5). The external interrupt, as a function of the event data, schedules a job to process the event data. The job is then executed in its designated time or priority sequence. The event data, which is used by the external interrupt to schedule a job, implicitly or explicitly contains the job time or priority and the size of the required data cells or the identification of a previously established data cell.

The timer job scheme provides a technique for detecting the absence of an event. A system user simply schedules a task to verify the occurrence of an anticipated event. Vehicle position verification provides an example of this process (Figure 6). At the time of the most recent position verification, a timer job was scheduled to occur following the anticipated time of the next vehicle position check-in; check-in is activated by a priority interrupt. At the scheduled time, the time-interrupt executive routine will remove the top job, i.e., verify vehicle check-in from the timer job list and link it to the data cell. Control will then be transferred to the verify routine, which will determine whether the anticipated event has occurred and either reschedule the verify routine or initiate the appropriate response strategy.

Some management functions must be executed as a series of short segments over an interval that is quite long compared to computer execution times. Similarly, some functions consist of subfunctions, each of which has a different priority or must be executed at a different time. To prevent wasting processor time, these functions use the job scheme to perform segmented processing. In this scheme, the initial function job establishes a data cell and executes the first functional segment. It provides for execution of subsequent functional segments by

1. Scheduling a job or set of jobs to execute the next segment or set of segments for the function or
2. Generating an output that will precipitate a subsequent input because of a closed loop with the real world.

In the latter case, processing of the input will generate a job to execute the next functional segment.

Management Image Perception Set

The management image perception set has two primary aspects. First, it is a set of models that can be interconnected in any number and configuration to form a model of the real world for the management system. Second, the models, to be consistent with the management system function, are characterized by a high level of perception of the corresponding real-world entities.

The automated network system contains two major and distinctly different subfunctions. The first is vehicle control, which is concerned not with the interaction of vehicles but with the control of a single vehicle with respect to a single control point. The second is network management, which is concerned with the interaction of all vehicles in the system and not with the microrelation of a single vehicle to a single control point. The level of perception required to perform these two functions is totally different.

The data base that the management system uses to represent the real world is called the system image. The image contains data on the present and projected state of the real world. The management system uses these data to make decisions on what functions shall be executed to effect efficient and safe vehicle management. The management system issues commands to other systems to effect its management decisions and updates the image to reflect its desired projected real-world states. Conceptually speaking, the image has three main parts: the abstract image, the relational image, and the projection image. The abstract image is the representation of the guideway network, the imposition of synchronous slots on the network, guideway stations, and sector boundaries. The relational image relates real-world entities to the abstraction and to one another. It defines events currently in progress, the current system state, and future system states that will be the result of events currently in progress. An event is, in the context of the management system, considered to be in progress if it will run to completion unless an alternative action is taken by the management system. Hence, an event may be in progress due to a previous management system command or due to an anomalous real-world condition that negated a previous management system command. The projection image defines events that the management system is scheduled to execute, but that are not yet in progress. These events are scheduled for execution by the management system because either the priority job list or the timer job list contains a job that will issue commands that will initiate the event.

The management system contains a set of models that present a high level of perception of the real world and are the fundamental building blocks of the system image. The management system interconnects these primitive, high-level models in any configuration and any number to generate the system image, makes management decisions based on the system image, and generates and issues commands to the appropriate subsystem to effect the decisions. These commands relate to the system image and, hence, are high-level commands. The receiving subsystem has a level of perception of the real world lower than that of the management system.

Thus, these subsystems contain models that provide for translation of the system management command to a set of commands appropriate to the level of perception of the subsystem. For example, a slot slip is a common maneuver in a synchronous slot scheme. The management system issues a command to the vehicle control system to perform a slot slip on a certain vehicle on a certain link. The vehicle control system is responsible

Figure 1. Timer job initiation.

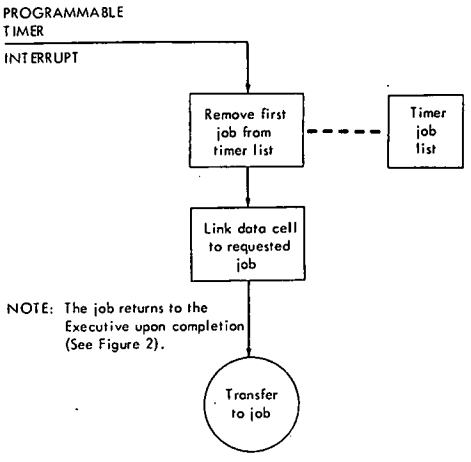


Figure 2. Job completion and priority job initiation process.

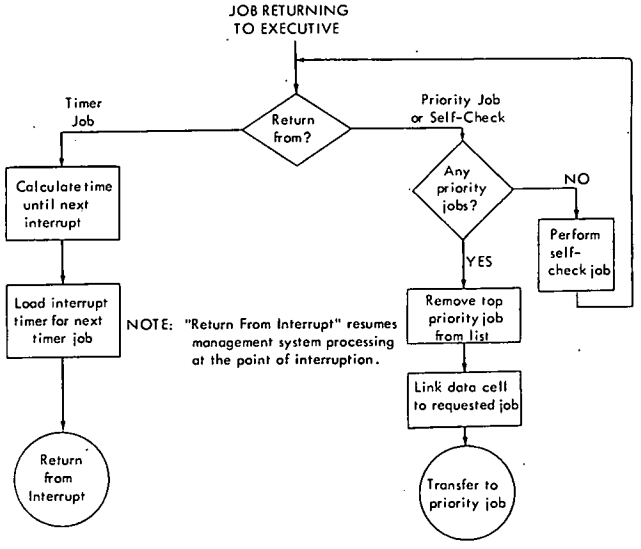


Figure 3. Timer job scheduling.

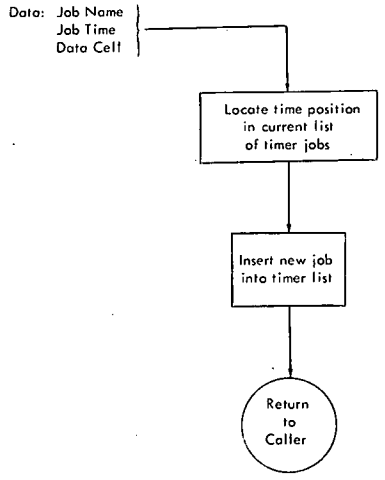


Figure 4. Priority job scheduling.

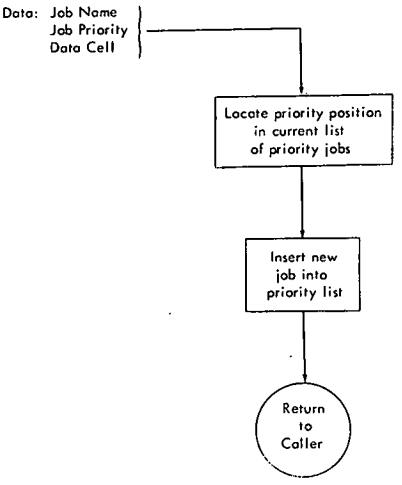


Figure 5. Priority interrupt processing.

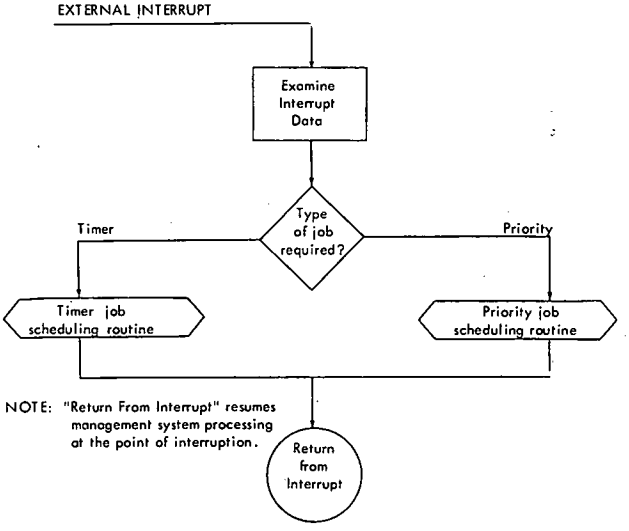
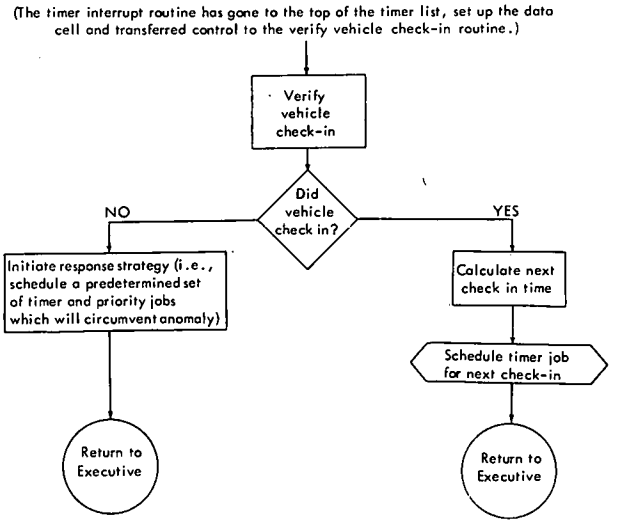


Figure 6. Detecting absence of an event.



for translating the system level command into a vehicle command. This requires that the vehicle control system translate the link number and current vehicle position into a time at which the vehicle will be on the designated link. The vehicle control system also translates the slot slip maneuver command into a velocity profile. To effect the slot slip, in the three-level computer control hierarchy, there are two levels of perception within the vehicle control system. The first level is at the sector computer, which translates the system management command into a velocity command. The second level is in the on-board vehicle computer, which translates the velocity command into braking and throttle commands to effect the velocity profile.

The primitive models are interconnected according to predefined rules. Each model has one or two connection points. Models are interconnected at a node, which is an image component that is the junction of two or three models. The node has at least one input and one output model. An input model receives a vehicle from a node, and an output model sends a vehicle to a node. The input and output are local attributes to a model at a node; a model with two connection points will be an input model at one point and an output model at the other. A node, as a junction point, is an abstract point and, hence, can never actually contain a vehicle.

The guideway network consists of merges, diverges, guideway links, and stations. The management system uses an image link to model a real-world link. A real-world link is a section of concrete guideway with no branches, and an image link is an image component that connects two nodes and contains an integral number of slots. A section of real-world guideway without any branches is normally defined as a single link. However, so that synchronous slot bookkeeping can be facilitated, a guideway section may be defined as several links. Image links, serving as the image representation of real-world links, have several attributes. Vehicles on different links in the image are not allowed under any circumstances to have mechanical contact in the real world. This means that any point on the real-world guideway can be on one and only one link in the image. However, points on different lanes on the real-world guideway network can be on the same link in the network image. Another attribute is that the management system views a link in terms of a set of available slots. Every slot time the slots on the link advance one slot. A new slot appears at the input end from the model at the input node, and the slot at the output end passes into the next model at the output node. The management system issues commands to the vehicle control system to move a vehicle down a link. The vehicle control system, as in the slot slip example, must actually move the vehicle.

A real-world merge or diverge is the junction of three links. A merge has two input links and one output link; a diverge has one input link and two output links. Merges and diverges do not have discrete models in the management system image. Rather, they are a natural consequence of the use of nodes to interconnect system models. Hence, a merge is represented by the junction of two input links and one output link at a node. Similarly, a diverge is represented by the junction of two output links and an input link at a node. Since slot length can be calculated, slot position could be drawn on a map of a guideway network. If this were done, nodes corresponding to merges would always occur upstream of the real-world concrete merge. Similarly, nodes corresponding to diverges would always occur downstream of the real-world concrete diverge. This is a consequence of the aforementioned link exclusiveness requirement that vehicles on different links in the image not be allowed to have mechanical contact in the real world.

Stations can be modeled several ways. One technique is to model a station, in the perception of the management system, as a vehicle sink and source. Another technique is to widen the perception of the management system by modeling a station as a network of synchronous links, wait queues, and launch queues. A vehicle entering a wait queue would be required to decelerate from the input synchronous speed to a stop in the last entry in the queue. The front vehicle in a launch queue would, upon command, be required to accelerate to the output synchronous speed of the queue. The queue output synchronous speed would be the synchronous speed of the model the queue connected to at its output connection point. Once again, the management system commands a vehicle into or out of a queue. The vehicle control system translates the command into vehicle control commands that will effect acceleration or deceleration of the vehicle. As in the slot slip example, the vehicle control system has its own corresponding model that it uses to translate the high perception management commands into vehicle commands.

A synchronous slot system can be modeled, for normal operation, with a very small set of models. Other model sets can be used, but the fundamental concept is the same. The model set provides a high level of perception to be consistent with the management task, and the models can be interconnected in any configuration to model any particular network.

Dynamic Image Reconfiguration

The use of the primitive model set provides the management system with a technique for generating an image of the real world. However, if there is a real-world anomaly, this image no longer accurately reflects the state of the real world. Also, the real-world anomaly generally introduces an illegal system state that the management system must respond to. Image correction is achieved by providing a dynamic image reconfiguration capability. Input event data indicate to the management system changes in the real-world state that negate the present image. The management system dynamically alters the image. It adds, deletes, or changes models in the image as appropriate. The response to the real-world anomalies and illegal real-world system states is achieved by the use of "transition" models. These transition models are members of the primitive model set. However, they have the special characteristic that they represent real-world entities that are illegal system states when confronted by events that are currently in process. Therefore, transition models have an associated set of functions that are performed to detect and circumvent any illegal system states that have arisen as a result of the real-world anomaly and concomitant dynamic image reconfiguration. The management system, by performing the associated functions, resolves all illegal system states. It then replaces the transition model with another model that reflects the real-world state, but that does not have the associated procedures for circumvention of an illegal state.

An example of a dynamic image reconfiguration is the management system response to a vehicle that suddenly slows down and stops on a link. The management system, upon notification of the vehicle stoppage, remodels the synchronous link as a transitional blocked link. The blocked link appears as a nonentity to any management system job attempting to route a vehicle over the link. A function is associated with the blocked link transition model that either parks or reroutes via an alternate path all vehicles headed for the blockage.

This is done by picking diverge points before the blockage and generating the slot reservations from those points to the destination for each vehicle. If the attempt is not successful, the vehicle is commanded to park upstream from the blockage. The parking maneuver may be accomplished by making the transitional blocked link appear as a wait queue to vehicles en route. The management system then commands the vehicle control system to enter the appropriate vehicles into the wait queue. The vehicle control system executes the actual parking maneuver. When all vehicles that were destined for the link have either been parked or rerouted, the management system performs another reconfiguration that models the blocked link as an ordinary blocked link. The link still appears as a nonentity to any management system job attempting to route a vehicle over the link, but it no longer requires the associated functions for rerouting or parking the vehicles destined for the link.

The dynamic image reconfiguration technique inherently provides for system malfunction response. All dynamic image reconfigurations are precanned packages that are sequences of image reconfigurations. The management system uses a two-dimensional matrix to execute dynamic image reconfigurations. One dimension defines the current system state in terms of a system descriptor, and the other contains the event data that represent the real-world anomaly requiring reconfiguration. The matrix element refers to the precanned reconfiguration sequence that is the management system response to the real-world anomaly.

Dynamic image reconfiguration can also be used for normal system maintenance by using a similar maintenance matrix. For example, if a link requires any routine maintenance, the system, at the appropriate time, reconfigures the link as a closed link. A closed link, by definition, is a link that is still operable but will not accept any new vehicle routes. Hence, all vehicles destined for the link are still allowed to traverse the link, but no new routes across the link are generated. When all vehicles destined for the link have passed the link, the link is modeled as a blocked link. This is the same as the closed link, except it indicates that all vehicles destined for the link have passed it. As before, it appears as a nonentity to any job attempting to route a vehicle over the link. At this point, the normal system maintenance can proceed. When the maintenance is completed, the system again reconfigures the image to reflect the availability of the link. Hence, the image reconfiguration technique also provides a technique for management of non-anomaly-related system functions.

An inherent advantage of the dynamic image reconfiguration method is that malfunction response strategies or normal maintenance functions can be changed or added without changing the system. The management image perception set can model any real-world state. The dynamic reconfiguration logic provides a set of tools to execute any sequence of model changes, additions, or deletions under dynamic conditions, thereby allowing the management system to dynamically respond to any change in the real-world state or to dynamically create any desired new state. Since the reconfiguration logic uses matrices to define reconfiguration sequences, new or different response or operating strategies are incorporated by changing the reconfiguration sequences in the matrices. No program changes are involved.

Another advantage is that, in using a high level of perception for the management system, the system provides a system user with the same high level of perception tools for planning malfunction responses and normal operating strategies. Strategies can be planned solely in terms of the models in the management image perception set without ever getting involved in the intricacies of the management program. The well-defined, high level of perception also precludes redundant performance of functions at different levels in the three-level computer hierarchy. For example, if care were not taken in the design process, it would be easy to begin to explicitly deal with distances in the management system. The vehicle control system was designed to handle this level of detail. Furthermore, this level of detail violates the logic that established multiple sectors because a single computer was not fast enough to respond to the entire system at this level of detail.

SUMMARY

A prenetwork management development effort resulted (a) in the selection of a deterministic synchronous slot scheme as the fundamental control methodology for the automated network, (b) in the division of computer control of the system into a three-level hierarchy, and (c) in the recognition that the network management and vehicle control subfunctions were characterized by distinctly different levels of perception of the real world. The network management development effort dictated use of an executive that responded to task priorities and desired times of execution. The development also, by focusing attention on the particular nature of the system and its applications, resulted in the dynamically reconfigurable primitive model concept. This concept, recognizing the management and vehicle control dichotomy, provides a management analyst with a set of tools that perceive the system at the management level. In keeping with the management level perception aspect, the ability of the models to dynamically reconfigure allows malfunction and normal operating strategies to be couched in terms of reconfiguration sequences.