ELSEVIER

Computer Integrated Manufacturing Systems Vol. 10, No. 1, pp. 35–48, 1997

Printed in Great Britain
0951-5240/97 $17.00+0.00

PII: S0951-5240(96)00017-1

A production order-driven AGV control model with object-oriented implementation

Manish Shah, Li Lin and Rakesh Nagi

Department of Industrial Engineering, 342 Bell Hall, SUNY at Buffalo, Buffalo, NY 14260, USA

To effectively manage the material handling in Flexible Manufacturing Systems (FMS), where a large amount of data is required in the dynamic decision-making, integrated control is needed to consider the overall production schedule. The focus of this research is on the development of an integrated Automated Guided Vehicle System (AGVS) control model that includes essential features like dynamic vehicle path determination and conflict-free routing. An object-oriented implementation of the AGVS model is proposed that forms the basis of systems integration with a production planning module such as MRP. Static and dynamic informational and functional models of the AGVS are developed. The system incorporates: (i) conflict-free shortest path routing procedures, and (ii) vehicle assignment rules or scheduling strategies. A prototype version of each of these has been developed for demonstrative purposes. This object-oriented modelling methodology provides the capability of rapid development and change. The approach has been demonstrated for a real manufactured product through simulation studies which confirm the superior performance of anticipatory AGVS control rules, even in a production order-driven environment. © 1997 Elsevier Science Ltd.

Keywords: Flexible Manufacturing System (FMS), Automated Guided Vehicle (AGV), Material Requirements Planning/Manufacturing Resources Planning Systems (MRP, MRP II), Systems Integration, Scheduling

Introduction

With advances in computer and electronic technologies making automation an integral part of manufacturing systems, the recent technological trend is towards developing Flexible Manufacturing Systems (FMS) due to their obvious advantages. The primary benefits of an FMS are reduced setup times, increased equipment utilization, reduced work-in-progress inventories, better throughput and reduced manual intervention[1,2]. An FMS typically consists of several numerically controlled machines for production processes, an automated storage/retrieval system (AS/RS) for raw material and finished goods, and an automated material handling system for material transfer between the work centres. An Automated Guided Vehicle System (AGVS) is a material handling system for transferring material between work centres (or flexible manufacturing cells) by means of one or more Automated Guided Vehicles (AGVs). AGVs controlled by a centralized computer system typically move along a guide path in the shop.

Since FMS involve high capital costs, significant attention has been paid to improving system efficiency via production scheduling[3–5]. Similar efforts have independently been undertaken on various aspects of AGVS design and operation (see the next section); most of the AGVS research has revolved around AGV allocation and dispatching strategies. The central idea of any of these strategies is to improve the AGVS performance, i.e. to improve the AGV utiization, to reduce total distance travelled, etc. For example, the nearest work centre allocation rule assumes that since the nearest vehicle will take the shortest time to reach the work station, the allocation will improve the performance for the job; and hence, such a strategy will improve the overall system efficiency.

However, even though the scheduling of work centres and AGVs are intimately related, and which collectively define the productivity of the overall system, little attention has been paid to considering them in a unified manner. Most research in AGVS lacks interest in the manufacturing system requirement-based allocation of AGVs. For this reason, the effect of a 'good' vehicle allocation strategy for a particular environment may cause only localized improvement; and the performance of the entire shop in terms of throughput does not necessarily improve. It has been realized that the very important link

between the shop requirements and AGV allocation is largely missing so far.

Recently some literature has focused on developing hierarchical approaches to the FMS control model[6–8]. The advantage of hierarchical control is that it allows the control problem to be partitioned to limit the complexity of any module in the hierarchy, regardless of the complexity of the entire structure[9,10].

In this paper, we develop an AGV control model that is driven by shop production orders. An object-oriented implementation of the AGVS model is proposed that forms the basis of system integration. Static and dynamic informational and functional models of the AGVS are developed. The system incorporates: (i) conflict-free shortest path routing procedures, and (ii) vehicle assignment rules or scheduling strategies. A prototype version of both of these has been developed to demonstrate the system's effectiveness. This object-oriented modelling methodology provides the capability of rapid software development and adaptability to changes of system configuration.

For production planning or generating shop requirements, a Material Requirements Planning (MRP) system is assumed. To integrate the MRP module with the AGVS and thus make it a shop requirement-driven AGVS, an order file acts as a common link. The MRP module, based on the BOM data file and final order due date, computes due dates for the component parts and accordingly places part orders in the order file. The AGVS retrieves orders from the order file and correspondingly allocates the AGV. Further, once an operation is performed, the part order updates the order file according to the next operation due date.

We also study the effects of vehicle selection strategy and an anticipatory dispatching on the overall system performance for an adopted order dispatching policy. Three models that adopt different strategies are implemented: Dedicated vehicle & Non-anticipatory; Nearest vehicle & Non-anticipatory; and Nearest vehicle & Anticipatory dispatching. These models are evaluated in a simulation framework. The comparison of their system performance indicates the superiority of anticipatory dispatching.

The paper is structured as follows. A literature review of the current topics of interest in AGVS design is presented in the next section. The proposed modelling methodology is then presented, and the system development issues, simulation framework and performance are discussed. Several important issues identified in the research are addressed, and finally, the conclusions are presented.

Literature review

The following discussion of the literature focuses on the basic elements of AGVS design: (i) guide path network design; (ii) optimal number of AGVs; (iii) vehicle dispatching; (iv) vehicle routing; and (v) traffic control. While these elements have been well studied, it must be noted that production order-driven AGV control has received little attention.

Guide path network design

A guide path usually defines the possible vehicle movement path. From the modelling viewpoint, the guide path is represented by a set of links and nodes. The nodes represent the action points (e.g. pick-up/drop-off points, maintenance areas) and intersections, whereas the links represent the aisle along which AGV movement takes place. The guide path design can be of an active or passive nature. For active guide paths, the central controller is connected to the entire guide path, and hence the AGVs receive command directives and also provide feedback to the controller via the guide path. However, the high implementation cost and inflexibility of such a design makes it impractical in many situations. A relatively inexpensive alternative, the passive type of guide path, has a primary function of keeping the vehicle on the track. In this case, the AGV communicates with the controller by means of RF/IR (radio frequency/infra red) waves.

The guide path network can be divided into four types, according to the allowed vehicle movement: uni-directional single lane; bi-directional single lane; multiple lanes; and mixed[11]. A uni-directional network, even though simple from the control and routing perspectives, is inferior in performance to a bi-directional network[11]. The multi-lane approach is practically cost prohibitive due to the extra space requirements.

The concept of single loop guide paths was introduced by Tanchoco and Sinriech[12]. In this approach, the entire path is divided into valid single loops which have at least one out of several arcs which make up each department. For single loop guide paths, the control system becomes simple due to the absence of possible collisions, etc. However, the system needs to have additional transfer points, as products may need to traverse through several loops before reaching their destinations.

Optimal number of vehicles

The cost and complexity of introducing additional vehicles against the marginal improvement in the system performance dictates the optimal number of vehicles in the system. Maxwell and Muckstadt[13] used an analytical model for determining the number of vehicles required under static and dynamic conditions. A simulation approach was used by Tanchoco[14] for determining the number of vehicles, and a non-simulation approach by Egbelu[15].

Vehicle dispatching

Once a demand for an AGV is initiated by a work centre, a choice needs to be made regarding which vehicle is to be dispatched. Conversely, when several

work stations need service and a vehicle becomes available, a choice has to be made as to which work station is to be serviced. On the basis of the above two scenarios and several optimizing criteria, Egbelu and Tanchoco[16] studied two types of selection rules: work centre Initiated Task Assignment (WITA) and Vehicle Initiated Task Assignment (VITA). A WITA applies in the case of excess vehicles, and the selection criteria can be based on random vehicle, nearest vehicle, farthest vehicle, longest idle vehicle or least utilized vehicle. A VITA is used in the case of one vehicle serving several stations. The determining rules are random work station, nearest work station, farthest work station, maximum queue size, minimum remaining queue size, modified FCFS, unit load arrival time, unit load due date, priority of unit load, etc.

The authors established that, in the long run, WITA rules have little application because the system is expected to have an optimum number of vehicles, so the chances of having several vehicles available at one point of time are quite slim. Thus, VITA rules are more important and practical.

Vehicle routing

To dispatch the AGV to a station at any point in time, a shortest feasible path from the existing position is desired. For a guide path network representation of links and nodes, the algorithm developed by Dijkstra[17] provides the shortest path. One approach is to determine a static path assuming that all the guide paths are available. However, in the presence of other vehicles, this assumption is not valid due to possible collisions. Collisions can be of two types: head-on and catching-up. To resolve these conflicts, a dynamic routing is required. It takes into account the current and future positions of all the vehicles in generating a feasible conflict-free path, which may not be the shortest one.

Several approaches have been proposed to create a conflict-free route. One is to generate the shortest path using Dijkstra's algorithm, and then define the node occupation times for each vehicle on the shortest path. These occupation times then can be used to detect partial conflicts[18]. The catching up conflict can be resolved by slowing down the trailing vehicle, and head-on conflict can be avoided by removing the conflicting segment from the network and using Dijkstra's algorithm to find the shortest path from the remaining network links. Another approach of using 'time-windows' was introduced by Huang[19], which used the physical arcs of the original network as time windows. However, it allows only one vehicle at a time in a zone. This approach was improved by Kim and Tanchoco[20] by maintaining time windows for reserved and free time slots on each node of the network.

Traffic control

In a system having conventional (intersecting) guide paths, the issues of AGV routing and collision avoidance are very important. The entire working of the system depends upon the control logic applied, which is divided into three main categories: (i) forward sensing control; (ii) zone sensing control; and (iii) combinatorial control.

Forward sensing control can be termed as a localized control in which each AGV is equipped with obstruction detecting sensors that can identify another AGV in front of it, and slow down or stop. The benefit of such a control is improved vehicle utilization due to closer allowable distances between the vehicles. However, this approach is unreliable for detecting obstacles around a corner and at intersections.

The zone control approach is more global. The control computer keeps track of the entire guide path, which is divided into zones. Once an AGV enters a zone, that zone (or a number of adjacent zones, depending on the location) is blocked so that no other AGV can enter that path[21]. In a simplistic implementation, zone control introduces some inefficiencies. However, sophisticated logic and rules can be used to improve the performance of such a control.

In combinatorial control, both the above strategies are selectively used to obtain the benefits of both strategies. For long straight paths, sensor control is appropriate, while for intersections, zone control is more suitable.

In summary, FMSs are capital-intensive systems that comprise expensive manufacturing centres and AGVS for material handling, therefore the efficient use of these resources is essential for their cost justification. Alongside, with ever-intensifying competition in the global market, there is an increasing need to further reduce product manufacturing cycle-time. Consequently, for material handling systems, it is necessary to minimize: (i) undesirable waiting time at machines due to the unavailability of AGVs; and (ii) excessive distances travelled by the AGVs in idle or loaded moves. Historical control approaches in FMSs have put much emphasis on the scheduling of work centres. Without a sufficient integration of the AGVS, only simple dispatching rules are adopted for AGV dispatching. The need for a unified approach to scheduling the material processing system and the material handling system has been well recognized, and effective control systems are expected to be adopted soon. This is exactly the focus of this research.

Modelling methodology

The following sections discuss the stand-alone AGVS modelling with related issues of conflict-free routing, the shop requirements planning by the MRP module, and finally, the AGVS integration with the shop

requirements. A two-step approach is used to present the modelling and integration of the system. First, the design of an independent AGVS is described using the object-oriented modelling technique. Both the static and dynamic informational and functional models, and an algorithm for generating the conflict-free routing, are discussed. In the second step, based on a typical product structure and bill of materials, the AGVS control and an MRP module are unified in one framework. The relevance of the MRP-based methods to generating part orders and back-calculating due dates is established, and finally, the integration of AGVS with the MRP module and related issues are explored.

AGVS design

Given the complexity of manufacturing systems and frequent product changes, system changes are often introduced. It gives the need for a stand-alone AGVS that can be integrated with a shop-order or requirement module. Traditional modelling and development methods fall short of these adaptability expectations. Hence, a powerful modelling methodology is essential. As a result of recent software engineering development, object-oriented modelling is gaining increasingly wider acceptance. Several methodologies have been proposed to model particular systems[22,23]. The object-oriented AGVS design will be employed later to demonstrate system performance for alternative global strategies.

The concept of 'objects' is at the heart of object-oriented modelling. An object in a system can be defined as a physical or conceptual entity with a well defined boundary, components and responsibilities[23]. The first step of the modelling is to identify the basic objects in the system and analyse the interaction between them. *Table 1* summarizes the objects identified in the AGVS along with their key attributes.

As for the modelling approach, Information Processing Object Hierarchy (IPOH), proposed by Changchien[24], a system has a dual dichotomy: a functional and an informational model, each with a static and dynamic view. These models are developed and implemented for the current system.

Static functional model. This is the basic model of the system, depicting the attributes (data members) and responsibilities (functions) of each object without considering the dynamic interactions between them. The information is displayed in a structured template called CARD (Class, Attributes, Responsibilities and Directive) classes.

Table 1 Objects in the AGVS

Objects	Major attributes
Traffic controller	No_of_AGV, No_of_Workstation, No_of_Sensor
AGV	AGV_id, current_position, destination, moving_status, loading_status, route
Sensor	Sensor_id, connectivity, importance
Order	Order_id, order_route, order_processing_time, due_date
Work station	Workstation_id, processing_status, workstation_position

Class stands for the name of the object class, while the attributes refer to the key features of the object. Responsibility is divided into three sub-sections, Input, Output and Processing. Directive classes is the list of all the classes which are connected to this particular class. *Figures 1* and *2* represent the CARD template for AGV and Traffic_controller. Appendix A contains the remaining CARD templates.

In the CARD templates for AGV class and Traffic_Controller, the attributes included some inherent characteristics of the object class that do not change during operation, such as AGV_Id_#, List_AGVs,

Class : Automated Guided Vehicle (AGV)

Attributes

- AGV_Id_#
- Load _Time
- Unload_Time
- Cur_Position
- Pre_Position
- Cur_Status (Waiting,Moving)
- AGV_avail(Assigned,Unassigned)
- Load_Status (Loaded, Unloaded)
- Dest_Position
- Initial_Position
- Assigned_Wait_Time
- Unassigned_Wait_Time
- Path_List

Responsibilities

Input
- Move_Position
- Load_Order
- Unload_Order
- Assigned_Wait
- Unassigned_Wait

Processing
- Update_Position
- Update_Status
- Update_Avail
- Update_%_Util
- Update_Path

Output
- Report_Cur_Pos
- Report_Cur_Status
- Report_Task_Comp
- Report_Statistics

Directive Classes

- Traffic_Controller
- Sensor
- Work_Station
- Part_Order
- MRP

Figure 1 CARD template for AGV class

List_Sensors and Layout_Of_Sensors. Since these are static information often related to a given system configuration, they are called static attributes. Other attributes in CARD that change values during operation are system states, termed dynamic attributes. These include Load_Time, Unload_Time, Cur_Position, etc. Responsibilities of the object classes are divided into input and output handling functions, and processing functions that update system states, i.e. dynamic attributes of the objects. Some examples are Update_Position, Update_Status, etc. Finally, the directive classes are objects that may be affected by the actions of the object. For AGV, for instance, Traffic_Controller, Sensor, Work_Station, are likely to change due to changes in the AGV's states.

Static information model. This model depicts the static relationship between the system objects. The model bears a very close resemblance to the Entity Relationship Diagram (ERD) in relational database design, where each entity (object) is placed in a box and is connected to one or more objects via one-to-one, one-to-many or many-to-one relationships. The model does not contain any dynamic information on how data are processed, which is achieved by the dynamic information model. *Figure 3* illustrates the static information model for the AGVS.

This model shows the general interaction among object classes in the AGVS. For instance, the relationship between Work Station and (production) Order is a many-to-many process, i.e. one Work Station can process more than one order, and one order can be processed by more than one Work Station.

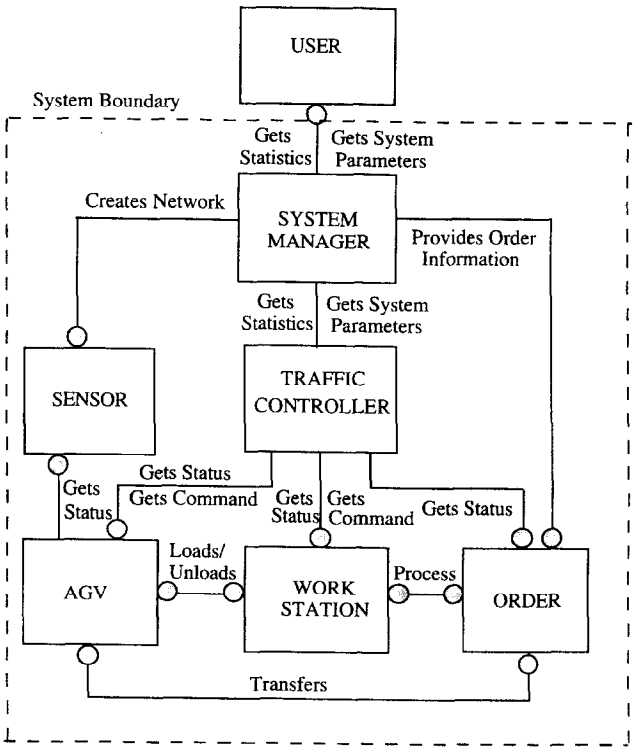

Figure 3 Static information model of AGVS

Class : **Traffic_Controller**
Attributes List_AGVs List_Sensors List_workstations List_Routing Layout_Of_Sensors Order_Queue
Responsibilities Input WS_Status AGV_Status Order_Routing_file Processing Find_Shortest_Path (AGV_Id_#) Select_AGV(WS_Id_#,AGV_Id_#) Assigned_Wait(AGV_Id_#) Unassigned_Wait(AGV_Id_#) Output Move_AGV Begin_Loading Begin_Unloading
Directive Classes AGV MRP Order Sensor Work_Station

Figure 2 CARD template for Traffic_Controller class

Dynamic functional model. This model depicts the temporal behaviour of the objects in the system. Typically, the dynamic functional model is implemented with the help of State Transition Diagrams (STD). Each object has several states during the system execution which consume a finite amount of time. A state transition is caused by an event which occurs at a discrete point in time. The events can be of the external or internal type. *Figures 4* and *5* show the STDs for the Traffic controller and the AGV. For instance, an AGV will change its state from 'AGV Loading' to 'AGV Loaded' by the event 'Loading done'; and it will proceed to an 'AGV Moving' state by the event of 'Moving to drop-off (load)'.

Dynamic informational model. This model depicts the dynamic data flow between the objects. Data flow diagrams are used to implement this model. The data flow diagrams have the following main components

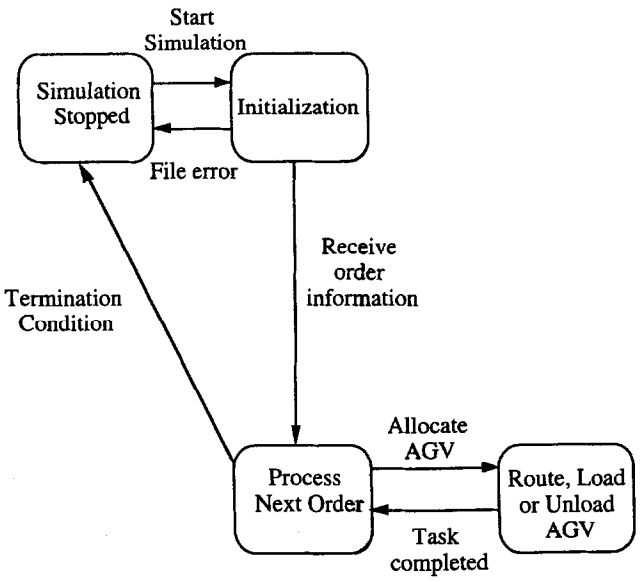

Figure 4 State transition diagram for Traffic_Controller class

which define the origin, conversion and destination of data[23]:

(a) Processes: to transform the input values to output values.
(b) Data flows: to connect the output of one object to the input of another.
(c) Data stores: the permanent store from which the input values are read, i.e. databases.
(d) Objects: which trigger different processes to start converting the input values.

Figure 6 illustrates the data flow diagram for the AGVS and the symbols used to represent the components of the data flow. Note that all data flow between processing function are attributes, such as 'Order Number', 'Sensor Position', etc. All attributes are contained in the attributes of the object class defined by the static function model.

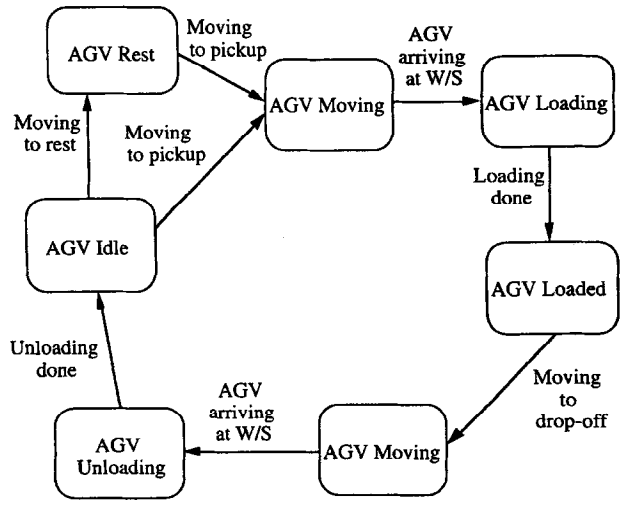

Figure 5 State transition diagram for AGV class

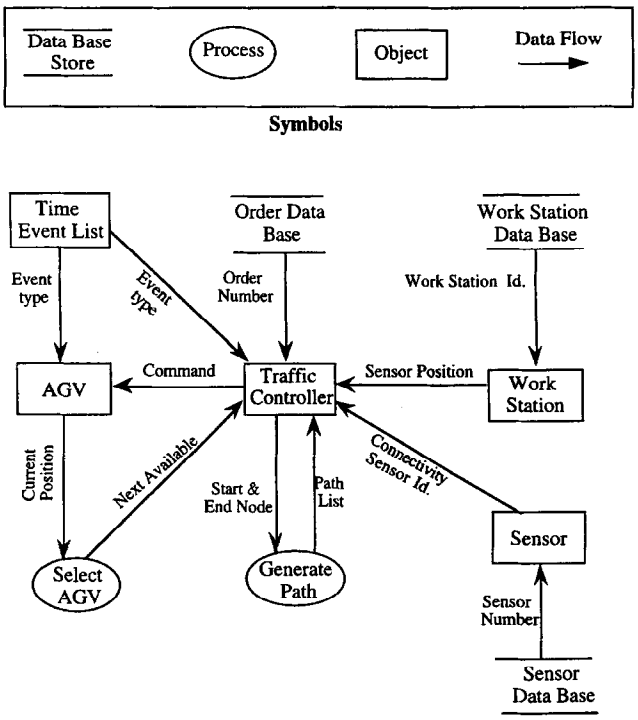

Figure 6 Dynamic information model for AGVS

Conflict-free routing of AGVs. As discussed earlier, a number of research efforts have focused on conflict-free routing of AGVs. The purpose of this section is to present a simple algorithm for conflict-free routing to be incorporated into the AGVS for demonstrative reasons. This simple algorithm can always be replaced by more appropriate techniques with relative ease due to its object-oriented implementation.

This algorithm considers n AGVs in the system. The concept of time window is applied to predict the position of each AGV with reference to the AGV under consideration. Consider the case where it is required to generate a conflict-free path for the nth vehicle. In the first step, depending on the origin and destination positions, a static shortest path is generated using Dijkstra's algorithm. Based on this static path, a time window is generated which has the time unit and corresponding sensor position as its attributes.

Next, similar lists having lengths equal to the time window list are generated for all the other $n-1$ vehicles. If a vehicle is at a stop (loading, unloading or waiting) or comes to a stop during the time window span, it is assumed to remain at the same location, at least for the remaining time period. Based on this assumption, after creating the time window lists, each time window is compared with the original one, and if a conflict is found, it is stored in a conflict-set.

If the conflict-set is non-null, the respective sensor associations are removed for Dijkstra's algorithm and a path is recalculated. This process is repeated until a feasible path is found. It is realized that the algorithm will require separate queues, and a more sophisti-

cated logic in case of a very large number of vehicles in a congested network. However, for this study, the algorithm is found to generate conflict-free routines for a moderate number of AGVs and medium-size window list. A flow chart of the algorithm is as shown in *Figure 7*.

MRP module

For planning the production tasks of multi-level assemblies, although contemporary planning techniques such as Just-In-Time (JIT) and Optimized Production Technology (OPT) have gained some popularity, the Materials Requirements Planning (MRP) (and Manufacturing Resources Planning, MRP II) philosophy is still employed by the majority of manufacturing enterprises. MRP has been found as an effective way to translate the requirements of a Master Production Schedule (MPS) into a detailed schedule of manufacturing parts and sub-assemblies.

MRP uses an explosion calculus procedure to result in the time phased plan for all component parts and raw materials required to produce all the products in the MPS. This materials plan can thereafter be utilized for detailed capacity planning, and finally shop-floor scheduling. Explosion calculus relies heavily on Bills-Of-Materials (BOM, or product structures), and refers to the set of rules by which gross requirements at one level of the BOM is translated into a production plan at that level and the requirements at lower levels. A BOM details the assembly structure or the parent–child relationship of components and end-items, the number of components per assembly, and the lead-time required for the production of each component. An accurate estimate of a component lead time considers (or should consider) the operation sequence (routing, or process sheets) and for each operation accumulates process time, setup time, queue time and material handling duration. The material handling duration estimates can be derived from the same process sheet by estimating the time required to move the material from one station to another.

The estimation of material handling duration and the process duration can be done with a fair amount of accuracy. However, the estimation of queue time due to either unavailability of a machine or unavailability of a vehicle introduces a degree of complexity to this particular problem. Various approaches have been proposed in the literature[25]. As the main objective of this paper is to develop a control model that integrates AGVSs and shop production orders, sophisticated methods of estimating queue times are beyond our research scope. Thus we did not include such an estimation, and only assumed static travel time.

The current work is based on the scheduling of production orders by an MRP module and its integration with an AGVS to dispatch the vehicles accordingly. To illustrate the approach, consider an example of a product having a 3-level structure as shown in *Figure 8*. Product A (representing the final assembly) is made up of three sub-assemblies (S1, S2 and S3) which are in turn made of several parts (P1 to P5) with different associations. Thus, product A is at level 0, sub-assemblies are at level 1 and the parts are at level 2. Each part and sub-assembly has a set of operations to undergo at several work stations.

To schedule the operations of the parts and sub-assemblies, MRP's backward scheduling logic is applied. The procedure for backward calculation, illustrated in *Figure 8* is summarized below by taking an example of sub-assembly S1 and its component parts P1 and P2:

Symbols	
Sub-assembly:	S1
Parts:	P1, P2
Work stations:	i, j, k, l, m
Process times: (assuming, one operation per process sheet)	P1→PP1 P2→PP2 S1→SP1
Travel time:	t (start_work_station, end_work_station)
Process route:	P1→i–j–l P2→i–k–l S1→l–m
Due date for vehicle dispatch:	P1→a1, b1 P2→a2, b2 S1→d1
Due date for S1:	T2

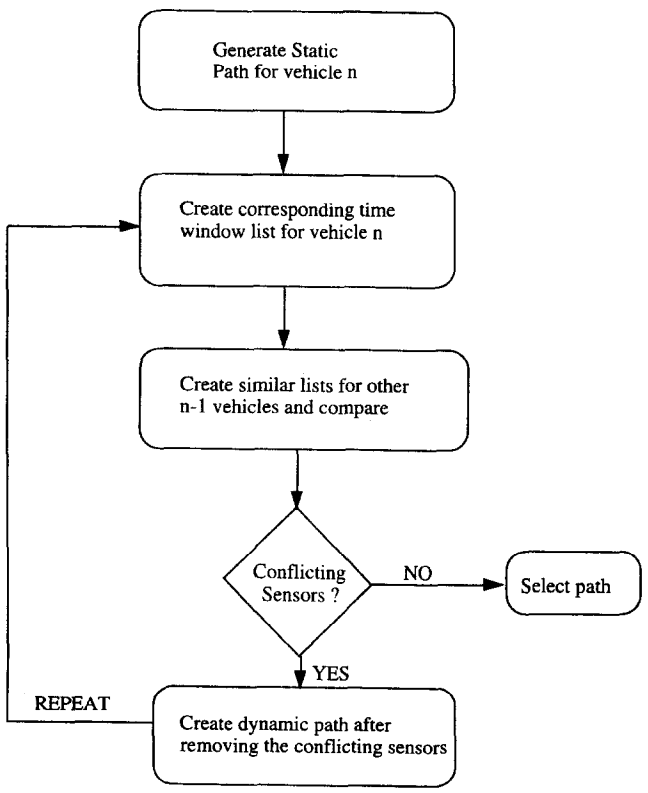

Figure 7 Conflict-free routing algorithm

- T2 represents the time at which the sub-assembly S1 is required for further assembly. Hence this is the due date for S1 from which all other due dates are back calculated using the above process route, process time and travel time estimates.
- Since it takes time t(l, m) to transfer S1 from station 'l' to station 'm', the due date for vehicle dispatch becomes dl, and is the time by which the operation for S1 must complete. This also means that parts P1 and P2 must be ready for assembly at station 'l' by time [dl − SP1]. This further leads to the due dates b1 and b2 by similar logic.
- Extending the same logic further and using the process times and travel times for parts P1 and P2, it is found that the first due date of part P2 falls before that of part P1 (a2<a1). Hence, the dispatch of part P2 is scheduled before that of part P1. In other words, P1 has a slack time compared to P2. This scenario also leads to the latest start time T1 by which the dispatch must start.

The orders are scheduled by their operation due dates, and hence the vehicles are assigned accordingly. A logical first choice in case of multiple vehicles being available is to allocate the nearest vehicle. A significant advantage of this strategy is that the parts are dispatched in the order of their required

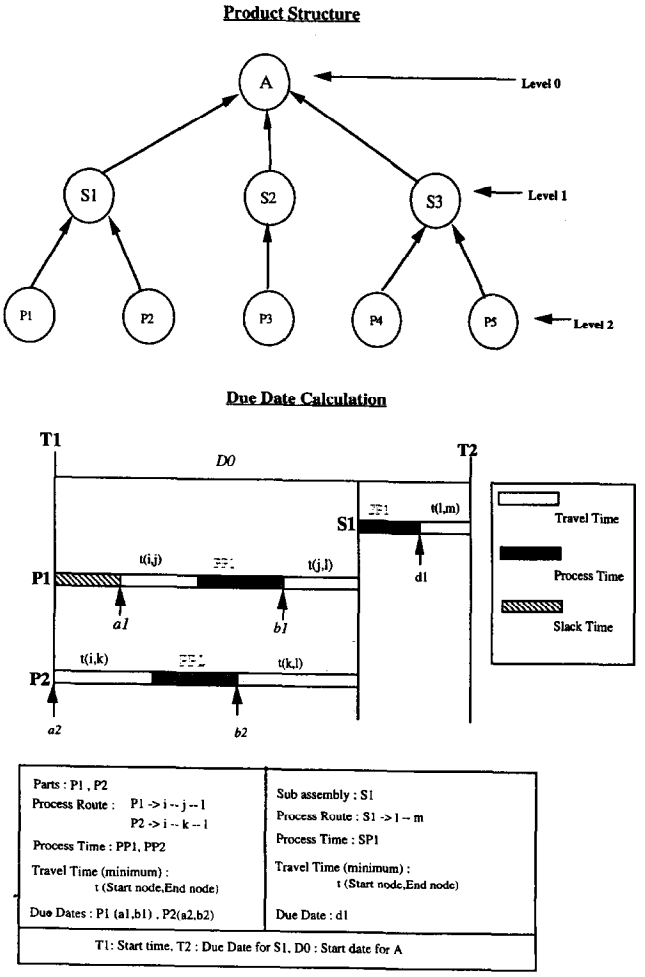

Figure 8 Product structure and due-date calculation

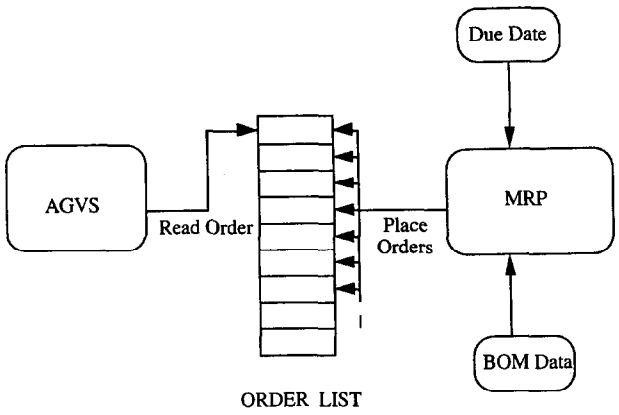

Figure 9 AGVS–MRP integration

completion time, and hence the overall final due date is the ultimate driving force for scheduling and AGV dispatching in the integrated control system.

Integration of MRP module and AGVS control

To integrate the MRP module with the AGVS and thus make it a shop requirement driven system, the order file acts as a common link. As shown in *Figure 9*, the MRP module, based on the BOM data file and final due date, calculates and assigns the due dates for the parts and accordingly places the part orders in the order file.

The AGVS retrieves the first order from the order file and correspondingly allocates the AGV. Further, once an operation is performed, the part order is updated in the order file according to the next operation due date.

It is realized that the material handling time estimates should be arrived at by considering not only the static distances between the work centres, but expected distances for non-conflicting paths and, more importantly, queue times due to waiting for an AGV. This problem is similar to the one of determining accurate estimates for part lead-times, where parts spend time in queues waiting for machines to become available[25]. In the lack of a systematic material handling time estimation procedure, we have adopted the static travel time between work-centres for demonstrative purposes. Possible improvements of this simple strategy are offered later in the discussion.

An industrial example

Another objective of the AGVS design is to demonstrate the overall system performance based on alternative parameters and strategies. Since AGVS is a shop support system, and its utility and performance are related to improvements in the performance of the system being served, operational parameters like AGV selection, order selection and routing vary according to the strategy chosen. The proposed strategy for a typical manufacturing scenario is

exemplified in this section by considering an example of a real life manufactured product and the coresponding shop-floor layout.

The integrated AGVS–MRP model was applied to a situation observed in a manufacturing company with some simplifications. *Figure 10* represents the layout of the manufacturing shop, and *Table 2* identifies the significance/utility of each of the workstations. *Table 3* represents the BOM data for the product, a viscous torsional vibration damper, which has a 2-level product structure. The final product is made of three parts, a Housing, a Cover and a Flywheel, with corresponding process routes and times. *Table 4* represents the static (minimum) distance between the various work centres which is used in the back calculation of due dates.

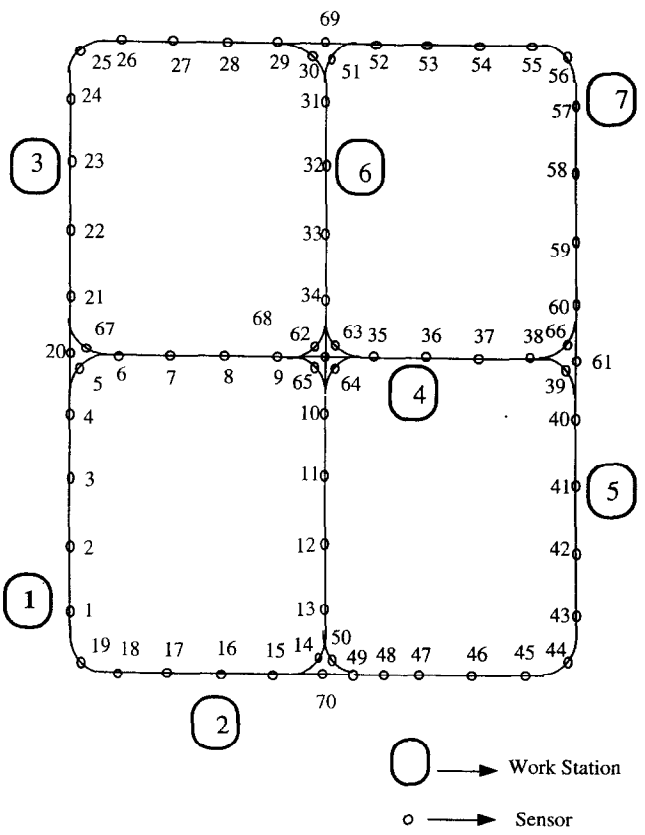

Figure 10 The facility layout

Table 2 Workstation data

Work station ID	Sensor ID	Function
1	1	Entry/storage
2	16	Cover manufacturing
3	23	Housing manufacturing
4	36	Turning
5	41	Inspection
6	32	Assembly
7	57	Exit/finished goods storage

Table 3 Bill-of-material information

Component name	Component ID	No. of operations	Process route	Process duration (at work station)
Housing	1	2	1→3→5→6	30(3) 10(5)
Cover	2	2	1→2→5→6	20(2) 10(5)
Flywheel	3	2	1→4→5→6	40(4) 10(5)
Damper	4	1	6→7	20(6)

Table 4 Static distance data

	WS1	WS2	WS3	WS4	WS5	WS6	WS7
WS1	***	4	7	11	15	12	18
WS2	4	***	11	9	11	10	16
WS3	7	11	***	10	15	9	13
WS4	11	9	10	***	5	5	7
WS5	15	11	15	5	***	10	6
WS6	12	10	9	5	10	***	8
WS7	18	16	13	7	6	8	***

AGVS control strategies

As mentioned earlier, the orders are processed in the sequence of their completion requirements, and hence the driving force for AGV dispatching is the order completion time. In the short run, however, this strategy may lead to the increase in tardiness of a particular job. However, as local optimality is not the objective, an overall reduction in the tardiness of all the jobs is expected in the long run. This is a significant potential benefit.

It is recognized that the performance of the system would strongly depend upon the system parameters like bill of material structure, product mix, order arrival times, order due dates, shop layout and the number of vehicles. Thus, the strategy to sort the part orders only on the basis of the most pessimistic time is too simplistic to produce consistently good performance for all shop conditions. Alternative strategies could (i) assign priorities by studying the effect of a new part arrival in the system on the existing parts being served, or (ii) sort orders by their most pessimistic dispatch times and by their arrival times so as to reduce the tardiness in the short term (at the expense of possibly higher tardiness in the long run).

The complexity involved in the determination of the possible logic for such a scheduling and dispatching renders it outside the scope of this work. Instead, we keep our focus on integrated control and study the effects of vehicle selection strategy and anticipatory dispatching on the overall system performance for the adopted order dispatching policy. The vehicle selection strategy involves determining the dynamic distance of each available vehicle (i.e. the shortest distance considering the obstacles in the path) and the subsequent selection of the nearest vehicle. The anticipatory dispatching strategy assumes that information related to the location of the immediate next order arrival is available, and therefore dispatches the assigned vehicle before the order arrives. This means that the AGV, once available,

receives information about the location of the next ariving order, proceeds to the destination and waits for the processing to complete. The combination of the above two strategies gave rise to the following three models:

The base model (dedicated vehicle & non-anticipatory). The base model consisted of a dedicated vehicle selection rule. In the case of multiple vehicle availability, a dedicated vehicle was always assigned irrespective of its distance from the order location. the model was non-anticipatory in nature, meaning that once the vehicle became available, it would wait at its current location for the arrival of the next order to start moving.

Model 1 (nearest vehicle & non-anticipatory). This model employed the distance criteria to dispatch the nearest vehicle from the order location. The distance was determined based on the existing layout configuration and location of other vehicles, hence the model was dynamic in nature. The vehicle dispatch strategy was non-anticipatory.

Model 2 (nearest vehicle & anticipatory dispatch). This model included both dynamic distance selection criteria as well as anticipatory dispatching on the basis of the availability of the next order information.

The three models gave rise to three different sets of computer codes. They correspond to three sets of order files that have 6, 12 and 18 orders, respectively. The following section provides a comparison of the system performance using these three strategies, and comments on the simulation results obtained.

System implementation and results

Implementation issues

The object-oriented models and order assignment methodology developed were implemented using Borland C++ as the programming language on the IBM/OS/2 platform. In the following, we discuss the software implementation approach and highlight the important features of the program structure and development.

The entire C++ code was divided into three main modules, namely a class declarations module, a member function declaration module and the main program. These three modules were developed in three different files under a common directory, so as to keep the complications at minimum and at the same time make changes and additions easier.

The header file contains the definition for all the objects defined in *Table 1*, as well as the objects of the MRP module. In addition to these system objects, two more objects for simulation (Calendar and TEvent) were declared. All the type casting for the container classes of Borland C++ was also defined in this file.

For simulation purposes, it is imperative that all the objects be stored or acted upon at different stages of the program by the same framework. Hence, the concept of a virtual class was used to refer to all the relevant objects together. A virtual base class ROOT is defined with common data member 'object_type' and a common member function 'event_handler'. Other objects like AGV, Traffic_Controller, Sensor, Work_St, Partorder, etc., are derived from this class. The main purpose of this declaration is to achieve dynamic binding and a commonality in event handling.

The container classes of Borland C++ were used while developing the program. The availability of such class libraries made it possible to directly use the data structures like Queue, Sorted Array, Array for objects and Sets without concerning the development of their underlying mechanisms. Some common functions available for such container classes also made it easy to perform operations on the entire structure. It should be noted that the availability and use of such classes is in accordance with the object-oriented philosophy of 'reusability'.

Simulation framework

To implement the order processing strategy and demonstrate system performance, a discrete event simulation framework was developed. This framework consisted of two main classes: TEvent: the event class, and Calendar: the manager of events. The class TEvent has three main data members:

(a) EventTime: the time of occurrence of the event.
(b) EventObject: the active object to be acted upon with the event.
(c) EventType: the type of event for the object.

As the EventObject can be of any one of the objects involved, it is type cast as the ROOT which is the virtual parent object. The events are organized into an array sorted by their EventTime, and are identified by the EventType. The array is a private data member of the Calendar class, and the primary responsibility of the calendar object is to schedule the event and retrieve the first event in the array.

The simulation is initiated by reading an order from a file, which is further decomposed into several part orders. Each part order has several types of processing to be done. After each process, it is updated in the part order array for further transportation. Each object which is acted upon during an event retrieval has an event_handler member function that performs the intended task and also schedules other events as required.

The simulation is run till an event termed as 'END SIMULATION' is encountered, which is the prescribed time to end the simulation. After the termination of the simulation, an output report showing the statistics of the system is printed to an output file. *Figure 11* shows the logic flow of this simulation framework.

Program description. The following describes the sequence of events in the program, its input data file structure and its output file specifications. The program was developed with industrious effort in enforcing the data and logic independency required of such systems. Hence, the first steps of the main program included retrieval of data from various data files. This modularity is necessary to facilitate future changes or additions. *Table 5* reviews the data files used for input and output.

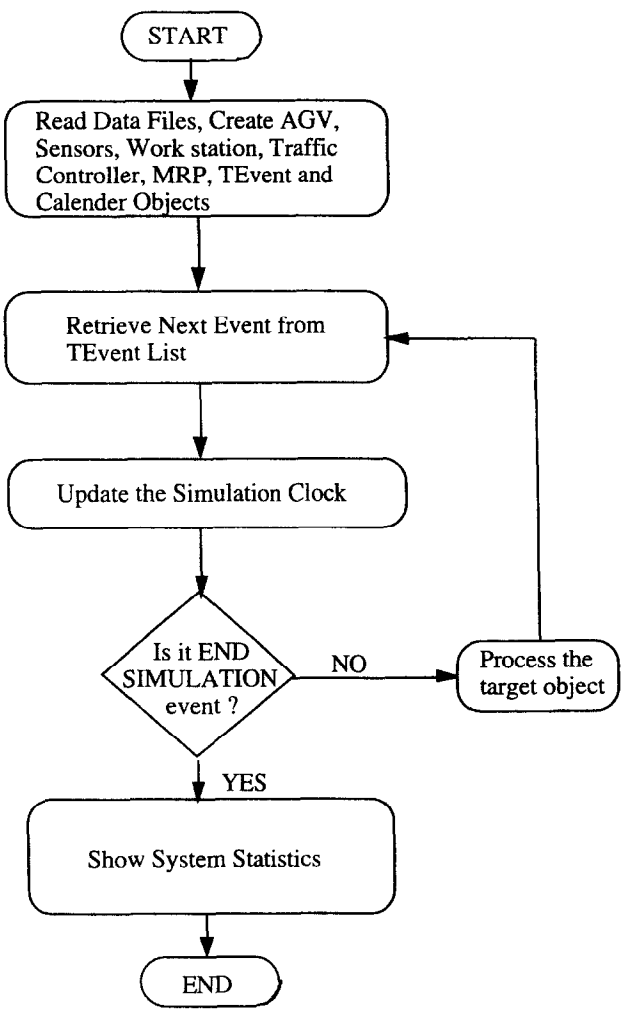

Figure 11 Flowchart for the simulation framework

In the main program, first the initialization of the system parameter takes place in which various data input files are read, and accordingly the objects of various classes are created. The pointers of various objects like Traffic_controller, Sensorlist, AGVlist, MRP and WSList are initialized for message passing and referencing. Next, orders from the order data file are read and the MRP module decomposes the orders into corresponding part orders in creating an order list. The system simulation then starts and proceeds to execute the various events in sequence. The simulation terminates when the END SIMULATION event is encountered and the system statistics are written to an output file.

Simulation results

As mentioned earlier, the code developed for each model was subjected to the three order sets containing 6, 12 and 18 orders. The simulation provided the statistics related to the overall system performance in the following format:

Overall system statistics

(a) Number of orders received
(b) Number of orders processed
(c) Number of tardy orders
(d) Maximum tardiness (Units)
(e) Minimum tardiness (Units)
(f) Average tardiness (Units)

Order statistics

(a) Order number
(b) Order arrival time
(c) Order due time
(d) Order completion time
(e) Tardiness

AGV statistics

(a) The travel statistics for the AGV number
(b) Unloaded travel time (Units)
(c) Loaded travel time (Units)
(d) Unassigned wait time (Units)
(e) Assigned wait time (Units)
(f) Loading time (Units)
(g) Unloading time (Units)
(h) Total number of part orders processed

Table 5 Data files used for input/output

Name of object	Container class	Used in class	Purpose
TEvent	BI_SArrayAsVector⟨TEvent⟩	Calendar	Used to store the events in an array sorted according to the event time
Integer	BI_QueueAsVector⟨int⟩	Traffic_controller	Used to store the node numbers of the route for AGV dispatch
partorder	BI_SArrayAsVector⟨partorder⟩	MRP	Used to store the part orders in a sorted array
partorder	BI_IArrayAsVector⟨partorder⟩	MRP	Used to store the permanent list of the components having BOM information
Sensor	BI_ArrayAsVector⟨Sensor⟩	Traffic_controller	Used to store the permanent list of sensors
Work_St	BI_ArrayAsVector⟨Work_St⟩	Traffic_controller	Used to store the permanent list of the work stations
AGV	BI_ArrayAsVector⟨AGV⟩	Traffic_controller	Used to store the permanent list of the AGVs

Table 6 The model statistics

	Dedicated (base model)		Nearest vehicle (model 1)		Anticipatory (model 2)	
No. of orders	Total tardiness	No. of tardy orders	Total tardiness	No. of tardy orders	Total tardiness	No. of tardy orders
6	293	6	227	6	42	2
12	1207	12	1257	12	274	7
18	2179	18	1903	18	658	10

Table 6 shows the relative performance, and *Figure 12* shows the comparative total tardiness of the three models. The following inference is drawn by analysing the graph.

The base model with dedicated vehicle selection and non-anticipatory dispatching shows the worst performance as expected. The vehicle dispatching logic of the other two models is designed to improve the baseline performance. From the figure, it can be seen that the situation marginally improves with the dynamically nearest vehicle selection strategy in model 1 when the number of orders increases. This slight improvement is attributed to the saving of vehicle travel time by selecting the nearest vehicles instead of dedicated ones. Given the small size of AGVS network, thus less difference in vehicle locations, the improvement was not significant, just as expected. However, model 2 with the anticipatory strategy (AGVs move to expected order locations in advance) shows remarkable improvements mainly due to the reduction in the wait time for the order pick up. It can also be seen that the total tardiness increases proportionally with the increase of orders. This is true for all three models.

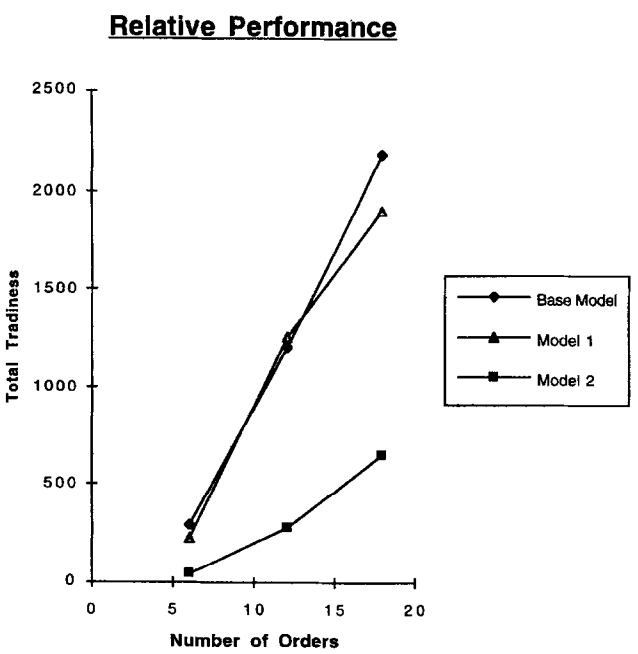

Figure 12 Relative performance of AGVS models

Discussions

The model developed in this research has touched upon some of the essential yet very important aspects of an AGVS like vehicle selection, dynamic path generation, vehicle dispatching, order scheduling and the integration of the MRP with the AGVS. However, the real challenges faced by a typical real world system are numerous and much too complicated to be addressed in a single framework such as this. Nonetheless, the advantage of using the object-oriented methodology to model a system effectively has been demonstrated in this work. Given the flexibility of the framework, the present model can be improved and embellished in numerous ways to handle the real world challenges, some of which are listed below:

- The current model uses a simulation framework to simulate the real world events and to capture the interaction between the various objects. To make the model applicable to a real system, it is necessary to replace the simulation framework with the relevant software/hardware interface to allow the transfer of data between the system and the physical objects. A desirable feature of the object-oriented implementation is the structured function definition in the object CARD templates. The real system signal, digital or analogue, may be included to replace the input/output handling functions in CARD. Of course, the memory location of the signals in a memory-mapped I/O interface must be specified.
- The present model rests on many underlying assumptions. The constant velocity of the vehicles, no external obstacles in the path, no machine break-downs, exact timings for travel and order processing, etc., are just some of these assumptions. However, the real system needs to address these issues with thoroughness to account for each normal and abnormal activity that could take place. One possible approach is to enhance the functions of sensors in the network so that exact vehicle movement can be monitored closely.
- The present model has a dynamic path generation algorithm which is capable of handling only a few normal conditions. For a real system, the algorithm needs to be improved and tested for each possible condition. In addition, an error-handling routine may also be necessary.

- The MRP module in the present model performs a very important function of incorporating the BOM structure and disaggregating an order into several part orders according to the due date considerations. It also utilizes a very basic dispatching strategy in terms of back-calculation. However, as noted earlier, this simplistic strategy may not be useful for many cases. The MRP module needs a significant improvement in terms of generality in storing and using the BOM structures for several parts, scheduling parts in an effective order and considering alternative strategies for relaxation of back-calculation in case of any abnormal events.
- The present model does not allow the vehicle path to be changed once it has been dispatched. However, the inclusion of such a strategy is a must in order to account for sudden obstacles or breakdowns. In this way, the control becomes truly dynamic.
- Eventually, the AGVS model needs to be run not by a primitive MRP module but by a more complex module based on the MRP II closed-loop framework, which incorporates the needs of shop capacity planning and other realistic constraints and performs the scheduling accordingly.

In addition to the above system development issues, vehicle dispatch strategy can also be improved. The simplistic strategy of using shortest static travel times as the estimates of transportation duration can be made more realistic by relaxing it using the following approaches:

- apply a uniform relaxation factor to the material handling duration estimates (e.g. a factor of 1.5);
- replace the shortest duration by an average of the shortest and the longest non-repetitive path;
- use different relaxation factors according to the shop loading conditions, etc.

Summary and conclusions

A software model for an AGVS was developed in this research. Salient features like dynamic path generation, AGV allocation and dispatch, order processing, etc. were incorporated in the model. In addition, the integration of the AGVS control to a production order system was established. The static and dynamic functional and informational models were adapted to receive orders from the MRP module and accordingly dispatch the AGVs. The aim of this strategy is to minimize the overall due dates for the orders. The MRP module was utilized as a demonstrative approach, and hence a very basic back-calculation strategy with shortest material handling duration was used.

To understand the system operation, a small scale discrete event simulation framework was developed, and the different objects in the model were conformed to pass the messages and retrieve the events as dictated by the simulation framework. The entire system was developed in C++ and was designed to be data independent. The system can be utilized for different scenarios by modifying the data files for system components like AGV, sensor connectivity, work station location, bill of material matrix and order sequencing. The MRP module was proposed for a general case, and was implemented for a specific case observed in a manufacturing company. The main task of the MRP module was to break down the final order into several part orders and sequence them according to the individual due dates. The code developed on the basis of the above model was then further modified to incorporate the various strategies of vehicle selection and vehicle dispatching. The three strategies were then subjected to the three order sets containing 6, 12 and 18 orders with varying due dates and arrival times, and the statistics were collected. The results obtained corroborated the inherent advantages and disadvantages of the individual strategies.

The concept of object-orientation was kept at the heart of the system development. It was realized during the stages of development that the model building was the most important phase which, once accomplished, led to a relatively easy coding. The inherently appealing concept of object-based development made the additions and modifications quite easy. The features of Borland C++ like virtual class and container class libraries helped to avoid the coding at a basic level, and thus made the tasks of dynamic binding and event handling quite simple. Thus, more effort could be applied for the actual system development.

In conclusion, it was found that the system design and development using the object-oriented technique was realistic in representing real system components and behaviour. It offers a flexible modelling and software implementation tool for the control of complex systems such as FMSs. The unique effort to integrate the AGVS; with the MRP module has sufficiently demonstrated the all important issue of integrated FMS control. The poptential application of this framework in the development of FMS control systems appears to be promising.

References

1 **Sethi, A K and Suresh, P S** 'Flexibility in manufacturing: a survey', *International Journal of Flexible Manufacturing Systems* Vol 2 (1990) pp 289–328

2 **Talavage, J and Hannam, R G** *Flexible Manufacturing Systems in Practice: Applications, Design and Simulation* Marcel-Dekker, New York (1988)

3 **Han, M-H, Yoon, K N and Hogg, G L** 'Real-time tool control and job dispatching in flexible manufacturing systems', *International Journal of Production Research*, Vol 27 No 8 (1989) pp 1257–1267

4 **O'Grady, P J and Menon, U** 'Loading a flexible manufacturing system', *International Journal of Production Research*, Vol 25 No 7 (1987) pp 1053–1068

5 **Stecke, K E and Kim, I** 'A study of FMS part type selection approaches for short-term production planning', *International Journal of Flexible Manufacturing Systems*, Vol 1 (1988) pp 7–29

6 **Davis, W J** 'Real-time optimization in the automated research facility', in J White and I W Pence (eds), *Progress in Material Handling Logistics* (1989)

7 **Jones, A and Mclean, C** 'A proposed hierarchical control model for automated manufacturing systems', *Journal of Manufacturing Systems*, Vol 5 No 1 (1986) pp 15–25

8 **Tirpak, T M, Deligiannis, S J and Davis, W J** 'Real-time scheduling in flexible manufacturing', *Manufacturing Review*, Vol 5 No 3 (1992) pp 193–212

9 **Simpson, J A, Hocken, R J and Albus, J S** 'The automated manufacturing research facility of the national bureau of standards', *Journal of Manufacturing Systems*, Vol 1 No 1 (1982) pp 17–31

10 **Jones, A T and Saleh, A** 'A multi-level/multi-layer architecture for intelligent shopfloor control', *International Journal of Computer Integrated Manufacturing*, Vol 5 No 1 (1990) pp 15–25

11 **Egbelu, P J and Tanchoco, M A** 'Potentials for bi-directional guide path for automated guided vehicle based systems', *International Journal of Production Research*, Vol 24 No 5 (1986) pp 1075–1097

12 **Tanchoco, J M A and Sinriech, D** 'OSL — Optimal Single-Loop guide paths for AGVS', *International Journal of Production Research*, Vol 30 No 3 (1992) pp 665–681

13 **Maxwell, W C and Muckstadt, J A** 'Design of AGVS', *IIE Transactions*, Vol 14 No. 2 (1982) pp 114–124

14 **Tanchoco, J M A, Egbelu, P J and Taghaboni, F** 'Determination of the total number of vehicles in an AGV-based material transport system', *Material Flow*, Vol 4 (1987) pp 33–51

15 **Egbelu, P J** 'The use of non-simulation approaches in estimating vehicle requirements in an automated guided vehicle based transportation system', *Material Flow*, Vol 4 (1987) pp 17–32

16 **Egbelu, P J and Tanchoco, M A** 'Characterization of automated guided vehicle dispatching rules', *International Journal of Production Research*, Vol 22 No 3 (1984) pp 359–374

17 **Dijkstra, E W** 'A note on two problems in connexion with graphs', *Numerische mathematik* Vol 1 (1959) pp 269–271

18 **Broadbent, A J, Besant, C B, Premi, S K and Walker, S P** 'Free ranging AGV systems: promises, problems and pathways', *Proc 2nd International Conference on Automated Materials Handling*, IFS Publishers, UK (1985) pp 221–237. (Reprinted in *Automated Guided Vehicle Systems*, R H Hollier (ed), IFS Ltd, UK, 1987.)

19 **Huang, J, Palekar, U S and Kapoor, S G** 'A labeling algorithm for the navigation of automated guided vehicles', *ASME Transactions, Journal of Engineering for Industry*, Vol 115 (1993) pp 315–321

20 **Kim, C W and Tanchoco, J M A** 'Conflict free shortest time bi-directional AGV routing', *International Journal of Production Research*, Vol 29 No 12 (1991) pp 2377–2391

21 **Koff, G A** 'Automatic guided vehicle systems: applications, controls and planning', *Material Flow*, Vol 4 (1987) pp 3–16

22 **Booch, G** *Object-Oriented Design*, Benjamin-Cummings (1991)

23 **Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F and Lorensen, W** *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ (1991)

24 **Changchien, S-W, Lin, L and Sun, D** 'A dynamic control model of flexible manufacturing cells using the information processing object hierarchy', *Journal of Factory Automation and Integrated Manufacturing* (forthcoming)

25 **Melnyk, S A and Piper, C J** 'Leadtime errors in MRP: the lot-sizing effect', *International Journal of Production Research*, Vol 23 No 2 (1985) pp 253–264