Design of Conventional and Tandem AGV Systems in Object-Oriented Simulation Modeling Environment

Kyung Sup Kim* · Russell E. King**

〈Abstract〉

An object-oriented simulation modeling environment, AgvTalk, is presented to provide flexible modeling capabilities for simulation of many alternative AGV systems. The hierarchical features and modularity of AgvTalk create possibilities for the extension and reuse of simulation object components. Also, detailed behavior of each object in the AGV system can be modeled easily and exactly in AgvTalk because there are no limiting modeling constructs. The modeling capabilities of AgvTalk is demonstrated by designing and simulating a conceptually different configuration of AGV systems, known as, the tandem configuration. Between the tandem and conventional AGV systems, the characteristics and design methodology in AgvTalk are described. Also, simulations between two systems are compared with AgvTalk in the job shop environment.

1. INTRODUCTION

1.1 Simulation of AGV Systems

Automated guided vehicle (AGV) systems have been regarded as one of the most exciting and growing areas of material handling and automation today. Although the early uses of AGVs were warehousing and distribution applications, development of hardware and software technology makes the AGV systems the most visible system in the application of manufacturing systems. In particular, because microprocessor technology has developed over the past decade, AGV systems are considered as highly flexible material handling systems for the effective operation of such systems as flexible manufacturing systems (FMS) and flexible assembly systems. Several efforts to make AGV systems more flexible are underway by eliminating the constraints which are imposed by the floor-imbedded guidance wire [9].

When designing and planning an automated material handling system, a large number of factors must be considered. In particular, with an automated guided vehicle (AGV) system, things such as the number of vehicles, a guidepath network configuration, control logic for dispatching vehicles, routing of vehicles from origin to destination, and interface with other material handling systems must be considered. Because of such complexities, simulation has been used as the primary method in designing, planning, and analyzing AGV systems.

Most of the modeling and simulation in the literature of material handling systems have been done by general-purpose simulation languages such as SIMAN [16] or

* Dept. of Industrial System Engineering, Yonsei University

** Dept. of Industrial Engineering, North Carolina State University

SLAM [17]. Recently, material handling features have been added to SIMAN and SLAM extensions. Although these extensions make the size of the simulation code much smaller, they are not easy to implement due to the inherent functional complexity of each feature. Moreover, these features are not flexible enough to serve as a basis for simulating the great diversity of many alternative material handling systems.

Some manufacturing simulation systems such as FACTOR, XCELL+, MAST, PROMOD, and SIMFACTORY have modeling constructs for manufacturing systems including an AGV system. However, these systems are even more inflexible in that they are limited to modeling only those manufacturing configurations allowed by their standard features [12].

To provide flexibility and reduce the complexity of the task of simulating an AGV system, simulation program generators specific for an AGV system have been developed. These include the C-based AGVSim [5], and SIMAN-based SCG [7]. However, these still do not overcome the modeling limitations of their implementation languages.

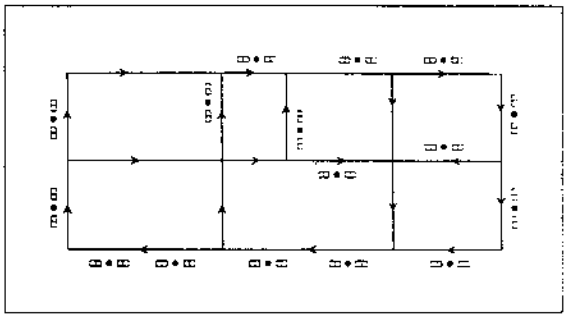
In most dynamic manufacturing environments today, systems and processes are constantly changing. Simulation tools are required that can accurately model a system in detail, yet still be easy to use, and allow rapid model redevelopment to react quickly to system changes [15]. Inherently, conventional approaches do not provide such capabilities. Object-oriented simulation is one such technique that allows an easy adaptation to system changes due to its inherent characteristics of extensibility and reusability. Although object-oriented simulation has a long history since SIMULA, the first object-oriented language, was developed in the 1960s for simulation purposes [2], the extensive use of object-oriented simulation using object-oriented languages such as Smalltalk and C++ is relatively new. In particular, simulation of material handling systems such as AGV systems using object-oriented simulation has not been examined in the literature.

1.2 Tandem AGV System vs Conventional AGV System

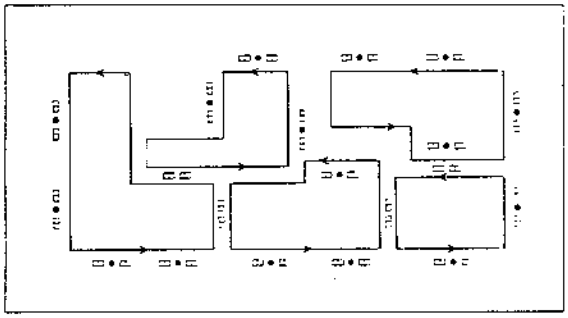
In conventional AGV systems in Figure 1, each AGV can move around the system along the system guidepath. If a station is reachable through the guidepath, the station is accessible to any AGV in the system. Since all stations are accessible to each AGV in conventional AGV systems, sophisticated control systems are required to manage vehicle dispatching, vehicle routing, and flow control

In the tandem configuration for AGV systems as shown in Figure 2, which was suggested as an alternative to the conventional AGV system by Bozer and Srinivasan [4], the movement of AGVs is defined differently in the system guidepath which is different from conventional AGV systems. The system paths are divided into non-overlapping, single vehicle closed loops. The locations of the stations remain the same as those of the corresponding conventional AGV system; however, each station belongs to only one loop. Only one AGV is assigned to each loop, completely eliminating traffic problems such as vehicle collision and vehicle congestion due to zone blocking. The movement of the AGV in the loop is determined by the First-Encountered-First-Served (FEFS) dispatching policy presented by Bartholdi and Platzman [1]. Under this policy, an unloaded AGV keeps moving to the next (adjacent) station in its loop until it finds a part waiting in the output buffer. The first encountered part is loaded to the AGV and delivered to the next station in its routing sequence. The communication between adjacent loops is performed by the interface. The main reasoning behind the tandem configuration is to reduce control problems inherent to conventional AGV systems. Bozer and Srinivasan also pointed out that the efficient operation of tandem AGV systems would require

balanced workloads among the loops to avoid creating bottleneck loops.



〈Figure 1〉 Conventional AGV System



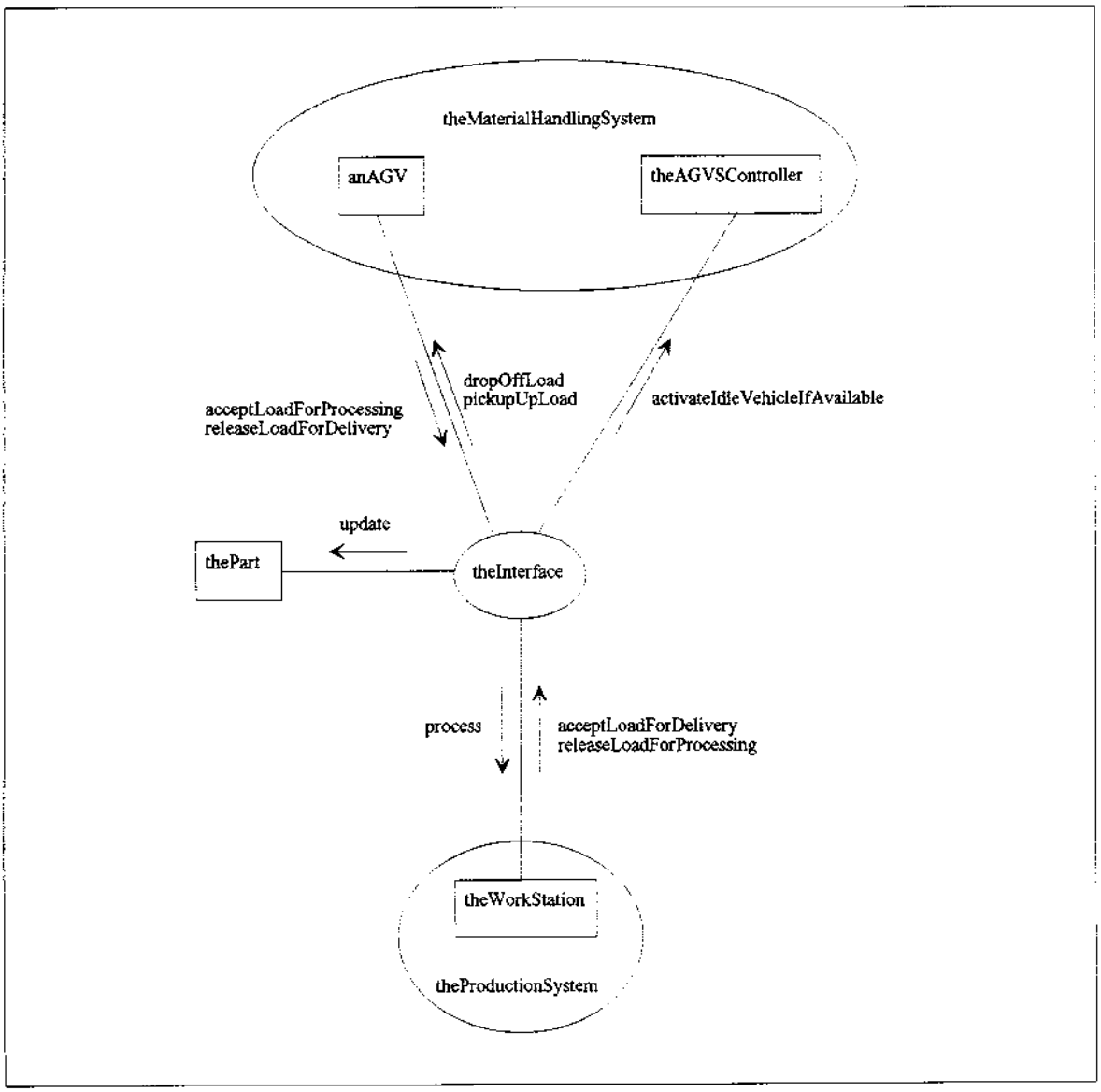
〈Figure 2〉 Tandem AGV System

1.3 Object-Oriented Approach

In an object-oriented approach, a system is modularly decomposed based on classes of objects the system manipulates, not on the function the system performs [14]. Objects are characterized and *encapsulated* by their state and the operations that can be performed on that state. Objects communicate with each other by *sending messages* in order to perform a task. When an object receives a message, the appropriate method is invoked. In the method, the appropriate reaction to the message is defined. Unlike in conventional programming languages, the message invokes the method specific to the object, as determined at an execution time rather than at a compile time. This is known as *dynamic binding*. Thus, the single message can produce different interpretation by different receivers. This condition is termed *polymorphism*. In an object-oriented approach, classes are hierarchically organized in subclass relationships [10]. Every representation and characteristic of an object defined in a superclass is *inherited* by objects of its subclasses; therefore, differences from the superclass are defined in the subclass. From the above characteristics, an object-oriented approach provides such advantages as reusability, extensibility, modularity, and portability.

2. OBJECT-ORIENTED DESIGN FOR CONVENTIONAL AGV SYSTEMS

An AGV system, at the highest level of abstraction, consists of a cooperating collection of other objects, including vehicles, zones, machines, and other objects. Direct visibility among instances of all the key abstractions would be a bad design since many of these objects are at quite different levels of abstractions [3]. Instead, key abstractions that represent the largest conceptual groups are selected at the highest level of abstraction for their visibility.In each group, key abstractions that are at the same level of abstraction are selected and grouped for their visibility. This process continues for the next highest level of abstraction until all the key abstractions are selected. For an AGV system, the material handling system, the production system, the interface, and the part are selected as the largest conceptual groups. The part represents the passive medium of communication among the other three objects. The material handling system describes the movement characteristics of vehicles from one location to another along the fixed guidepath, and includes objects such as vehicles, controllers, zones, a request queue, etc. The production system describes the part processing operations at workstations, and includes objects such as workstations and machines. The interface



〈Figure 3〉 AGV System Object Diagram

represents the interface between the material handling system and the production system, and includes input and output buffers. These four objects are included as direct components of the AGV system. However, many different designs are possible according to the definition of the relationships between these objects. In particular, designs at the higher level of abstraction are more critical and sensitive to the design of a whole system because each design feature directly affects the design at the lower levels of abstraction.

For the design of an AGV system at the highest level of abstraction, if we let three objects (the material handling system, the production system, and the part) only be visible to the interface and vice versa, this approach represents the real structure of the AGV system naturally. Considering operations that the interface can

perform upon each of three other visible objects, the following operations are derived from the point of each object.

- On the material handling system
  - dropOffLoad
  - pickUpLoad
  - activateIdleVehicleIfAvailable
- On the part
  - update
- On the production system
  - process

The above operations are the only communication between the interface with the material handling system, production system, and part. If we look at this problem from the other direction, we can determine what operations the material handling system and the production system perform on the interface. Since the part is a passive object, it does not perform any operations on other objects.

- By the material handling system
  - acceptLoadForProcessing
  - releaseLoadForDelivery
- By the production system
  - acceptLoadForDelivery
  - releaseLoadForProcessing

In this design protocol, the AGV system has been clearly separated and modularized among objects at the highest level of abstraction. The material handling system encapsulates by dropping off and picking up loads, and by activating an idle vehicle if available. The part encapsulates by updating the status of itself. Similarly, the production system encapsulates by processing parts. The interface encapsulates by serving as a communication center between the material handling system and the production system, and receiving or sending a part for processing or delivery. These design decisions are shown in the object diagram [3] in Figure 3.

The design concepts discussed have been implemented to the lowest level of abstraction using Smalltalk-80 [6]. The object-oriented simulation environment with the library of classes developed for an AGV system in Smalltalk-80 will be referred to as AgvTalk. AgvTalk includes 25 object classes and more than 300 object methods in its library which allows modeling of many detailed features of AGV systems.

3. HYBRID MODELING APPROACH IN AGVTALK

In an AGV system, the system logic can be usually determined by characteristics of two active physical objects which are independent of each other: how a vehicle moves in the system (vehicle move process) and how a part is processed in the workstation (part process). By combining these two processes, the complete AGV system logic is constructed.

In AgvTalk, the generic behavior of both processes are defined in the methods tasks in class AGV for the vehicle move process and process: in class WorkStation for the part process. That is, two different AGV systems represent different system logic, which correspond to two different combinations of methods tasks in class AGV and process: in class WorkStation.

Considering class AGV, all methods defined represent the generic behavior of an AGV such as load, unload, travel, wait, etc. However, one of generic senarios for an AGV system among many possible alternatives is performed by the method tasks. In a method tasks, the major characteristics for an AGV system such as vehicle flow pattern, existence of the staging area, shop operating condition (job shop, flow shop, etc.) are determined by the sequences of other methods defined in the AgvTalk library, and the method tasks in class AGV is referred to as a "Base" for the given AGV system. Each different sequence represents the specific alternative of the AGV system behavior. All possible

alternatives can be stored into the method tasks in the subclasses of class AGV, and the methods tasks in subclasses of class AGV are referred to as "Alternatives". Subclasses are different from class AGV only in AGV system behavior defined in the methods tasks, and all other characteristics are inherited from the class AGV.

"Alternatives" can be retrieved and handled by the user and allow a very fast model completion when the considered real system is close to one of the available "Alternatives". When the real system is identical to one of the "Alternatives", the user has only to initialize the right AGV class which has the identical "Alternative" in its method tasks. This initialization process can be easily implemented by the menu and window, which explains the patterns of the "Base" and all "Alternatives". The implementation of the method tasks shows inheritance and polymorphism mechanism of AgvTalk, which are shown in Figure 4. This approach is referred to as a *hybrid approach* since an AGV system model is constructed by providing system logic through procedure-oriented methods tasks within an object-oriented environment of AgvTalk and Smalltalk.

4. EXTENTION TO TANDEM AGV SYSTEM IN AGVTALK

For conventional AGV systems, many different configurations can be modeled very easily in AgvTalk because all elements in AGV systems are modularly designed as objects. Different configurations can be immediately implemented by assigning new or changed characteristics to the related objects. These processes may involve creating subclasses, modifying tasks, and assigning new instance variables.

However, from the design and operational points of view, the tandem AGV system is radically different from the conventional AGV system. The major differences fall into two categories, 1) system network definition and 2) travel of AGVs. The design of these differences in AgvTalk is discussed in the following sections.

4.1 System network definition

In AgvTalk, the system network is defined by the following three processes:

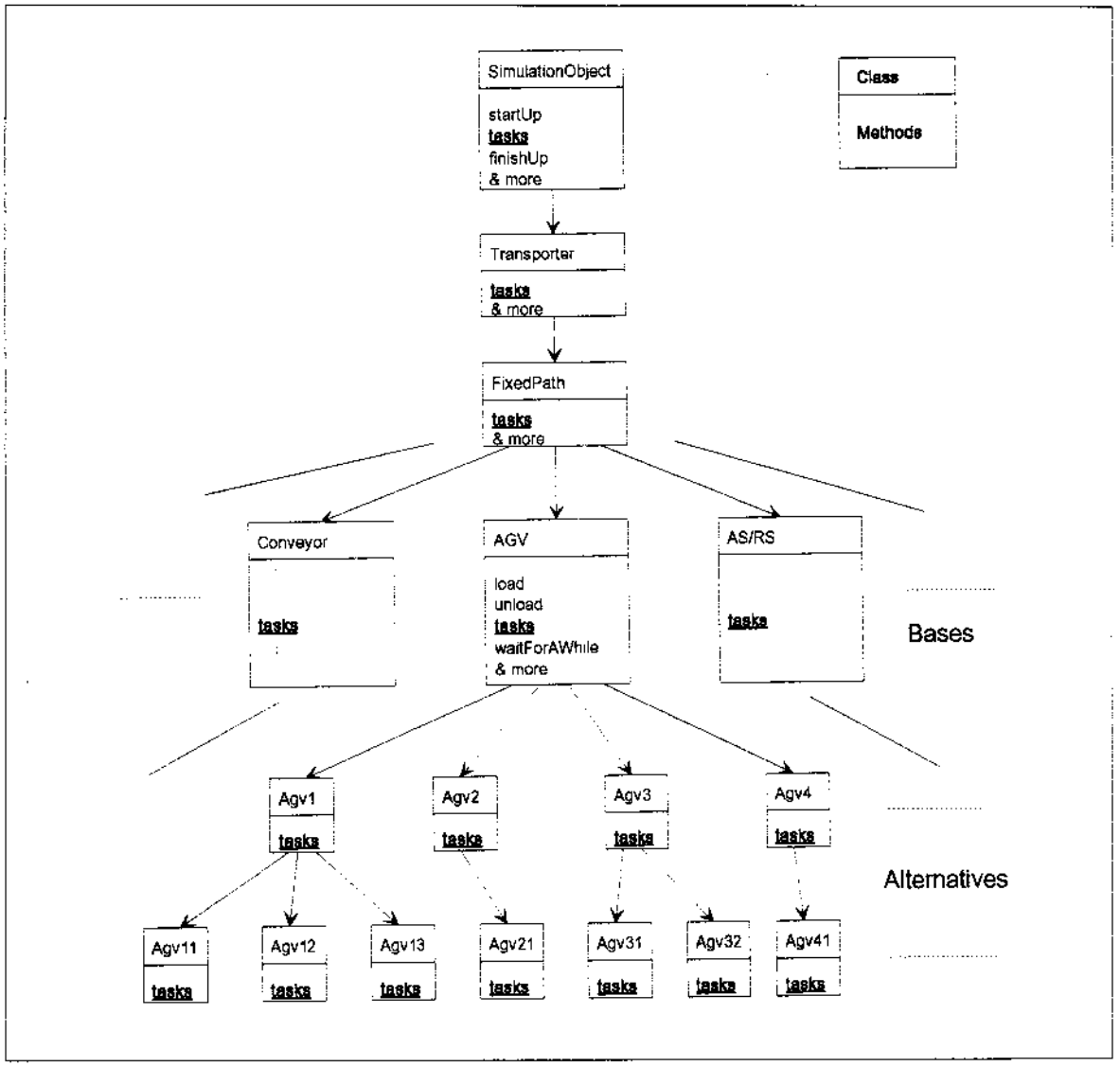
1. Definition of the control points (stations, intersections, etc) as instances of the class Node,
2. Connection of all possible combinations of two adjacent control points, and
3. Generation of the shortest routes between any combination of two control points.

From the operational point of view, the system network of the tandem AGV system is radically different from that of the conventional AGV system. However, the differences between two systems in defining the system network are minimal. The additional process required for the tandem AGV system is the assignment of an instance variable *loop* to the instances of the Node, which are defined as control points in process 1. For example, if station 1 belongs to loop 1, the process in AgvTalk is defined as follows;

```
Node new name: 'station1' inLoop: 1
```

In this process, the number of the loop to which the control point belongs is assigned to the *loop*. The interface among loops is defined just like another control point; however, the string of 'interface' is assigned to the variable, *loop*. The instance variable *loop* is used to prevent each AGV from moving into other loops.

In process 2, for each system path segment between two adjacent control points, the number of zones is defined for zone blocking. In the tandem system, only one AGV is allowed in each loop and each system path segment must belong to one loop. Thus, each system path segment is accessible to only one AGV, which completely eliminates traffic problems. Therefore, the number of zones between control points does not affect



〈Figure 4〉 Inheritence and Polymorphism in Hybrid Approach

system performance, and each path segment is defined to have only one zone for simplicity purposes.

There is no difference in defining process 3 between the conventional and the tandem AGV systems.

4.2 Travel of AGVs

From the material handling point of view, the AGV system can be summarized as a group of repeating processes of AGVs which travel from one station to the other station in the given system network. The travel between two stations can be characterized in many different ways. It may be travel for pick up or deposit tasks. It may be travel to the next station without a prior task assignment. Also, it may be travel to the staging area, or the recharging station.

In AgvTalk, the repeating process of each AGV, that is, the behavior from the arrival at a station to the arrival

```
tasks
        "Conventional system : This shows how a vehicle behaves from
        the arrival at the station to the arrival at the next station. The AGV
        system includes a staging area."

        status = 'arrival'
        ifTrue:
                [currentStation := 'staging']
        ifFalse:
                [currentStation := self controller nextStation].

        [true]
        whileTrue:
        [
        (currentStation = 'recharging')
        ifTrue:
                [self beingRecharged
                ].

        (currentStation = 'staging')
        ifTrue:
                [self waitForAWhileAt: 'staging'
                ].
        (((currentStation = 'staging') not) and: [(currentStation = 'recharging') not])
        ifTrue:
                [task = 'pickUp'
                ifTrue:
                        [
                        Buffer releaseLoadForDeliveryAt: currentStation to: self
                        ]
                ifFalse:
                        [
                        Buffer acceptLoadForProcessingAt: currentStation from: self
                        ]
                ].

        self controller updateInformationFor2: self.
        self controller transport: self.

        currentStation := self controller nextStation
        ]!
```

〈Figure 5〉 The Method tasks for Conventional Configuration

```
tasks
        "Tandem configuration : This shows how a vehicle behaves from
        the arrival at the station to the arrival at the next station when an
        AGV system is a tandom configuration of loops "
        | outputBuffer |

        [true]
        whileTrue:
        [
        task = 'deposit'
        ifTrue:
              [
              Buffer acceptLoadForProcessingAt: currentStation from: self
              ].

        outputBuffer := (OutputBuffer at: currentStation).
        (outputBuffer isEmpty) not
      ifTrue:
            [
            (((Node interface includes: currentStation) not)
            or:
            [(Node interface includes: currentStation) and:
                  [(outputBuffer first loop = self loopNo) not]])
            ifTrue:
                  [
                  task := 'pickUp'.
                  Buffer releaseLoadForDeliveryAt: currentStation to: self
                  ]
            ifFalse:
                  [task := 'moveInLoop']
      ]
      ifFalse:
            [task := 'moveInLoop'].

      self controller updateInformationInLoopFor: self.
      self controller transport: self.

      currentStation := self controller nextStation
      ]!
```

〈Figure 6〉 The Method Tasks for Tandem Configuration

at the next station for a given AGV is defined in the method tasks of the class AGV. That is, when an AGV arrives at the station, the appropriate interactions with the buffer of the station are performed according to the AGV's travel purpose to this station. For conventional AGV systems, the travel purposes are to pick up or deposit a part, to wait at the staging area, to be recharged at the recharging station, etc. Even though there are some changes in the system configuration, the processes defined in the method tasks remains the same as long as the system logic for the AGV movement behavior remains unchanged. However, different logic-based AGV systems such as a tandem AGV system should have different behavior in the method tasks, since the travel purposes of AGVs are different. In the tandem AGV system, the travel purpose of AGVs should be one of the following two: to pick up a part or just to travel to the next adjacent station in its loop until it encounters the pick up task.

Unlike conventional AGV systems, the tandem configuration includes interfaces. When an AGV arrives at the interface and there are parts waiting for an AGV to be delivered, the first part in the interface is examined if it is an incoming part from another loop. If it is an incoming part, it is picked up by the AGV and delivered to the next station in its routing sequence. For this process in AgvTalk, the instance of the Part has an instance variable *loop*, and the loop number of the instance AGV is assigned to the variable *loop* whenever a part is loaded onto the AGV. As long as the part instance remains in the loop, the value of *loop* is unchanged. When the AGV instance arrives at the interface, and if there are parts waiting and the loop number of the first part is the same as the loop number of the AGV, the part must be an outgoing part to another loop, and must not be picked up by the AGV. If the loop numbers of the part and the AGV are different, the part must be an incoming part from an adjacent loop. Thus, the part instance is picked up by the AGV instance, and the loop number of the part is updated to that of the AGV. The method tasks for the conventional and tandem AGV systems are presented in Figures 5 and 6.

5. SIMULATION EXPERIMENT

In this section, AgvTalk is used to experiment with two different AGV systems. The two systems are a conventional system and a tandem system.

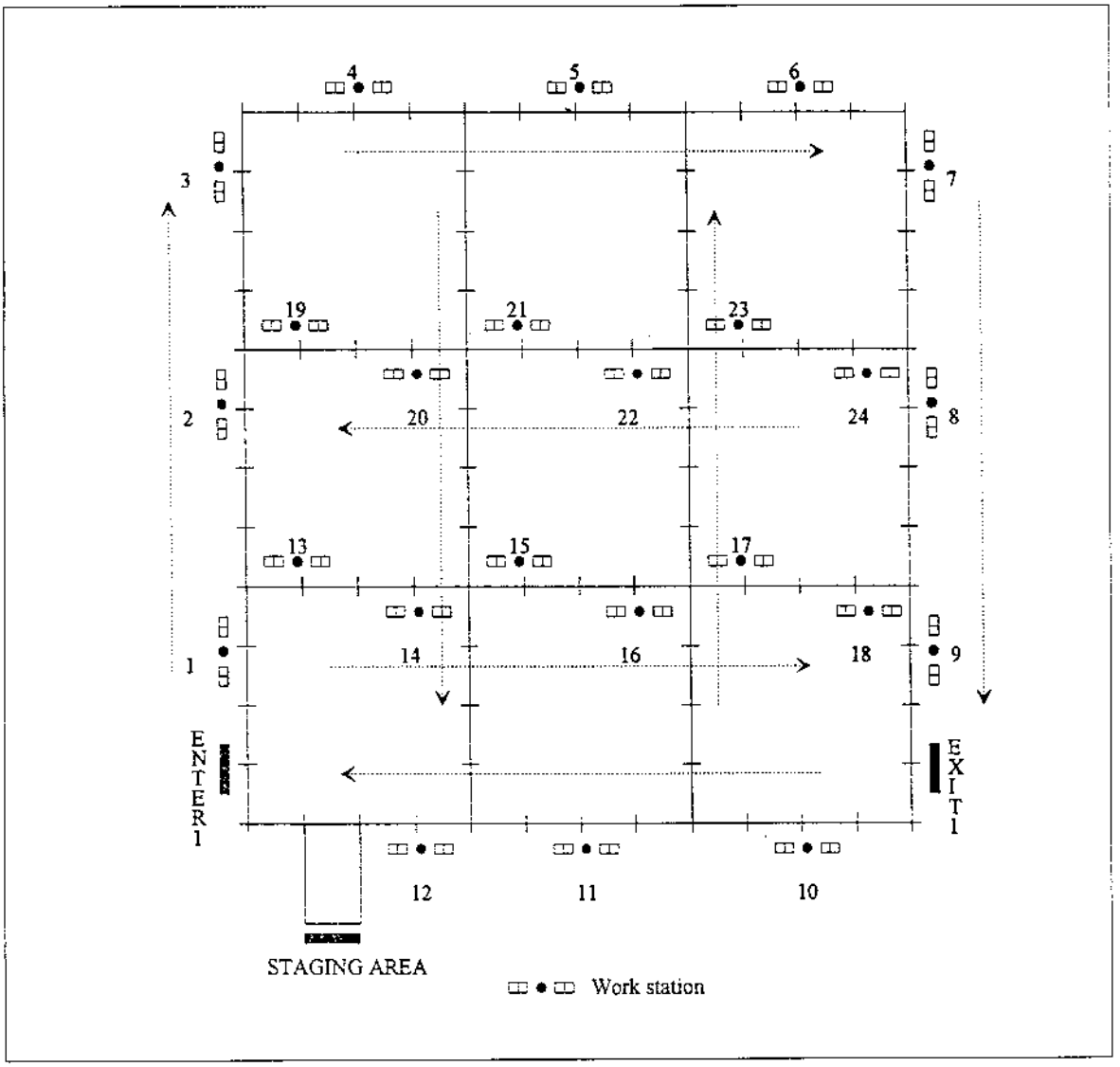
As pointed out by Bozer and Srinivasan [3], the development of an efficient tandem configuration from a given set of stations and part flow data is not a simple task and is another research area. Thus, tandem configurations are developed only with a given set of stations. Then, different sets of part flow data are provided for each configuration.

The job shop has been chosen for the experiment. The layout configuration of the job shop in the conventional AGV system is presented in Figure 7. The corresponding job shop configurations in the tandem AGV system is shown in Figure 8. The tandem configuration has 9 loops. There are 24 work stations in the job shop, and the locations of work stations are the same in two configurations. The conventional configuration has an additional staging area for idle AGVs, and the tandem configuration has 12 interfaces to allow part routing among adjacent loops.

Three different part types are produced in the job shop, and the arrival rates of each part type are as follows:

Part type 1 : Exponential (12)
Part type 2 : Normal (12, 2)
Part type 3 : Triangular (10, 13, 15)

There are two different sets of part routings. In the first set, the routing sequence of each part type is randomly selected. In the second set, the routing sequence of each part favorably matches the station

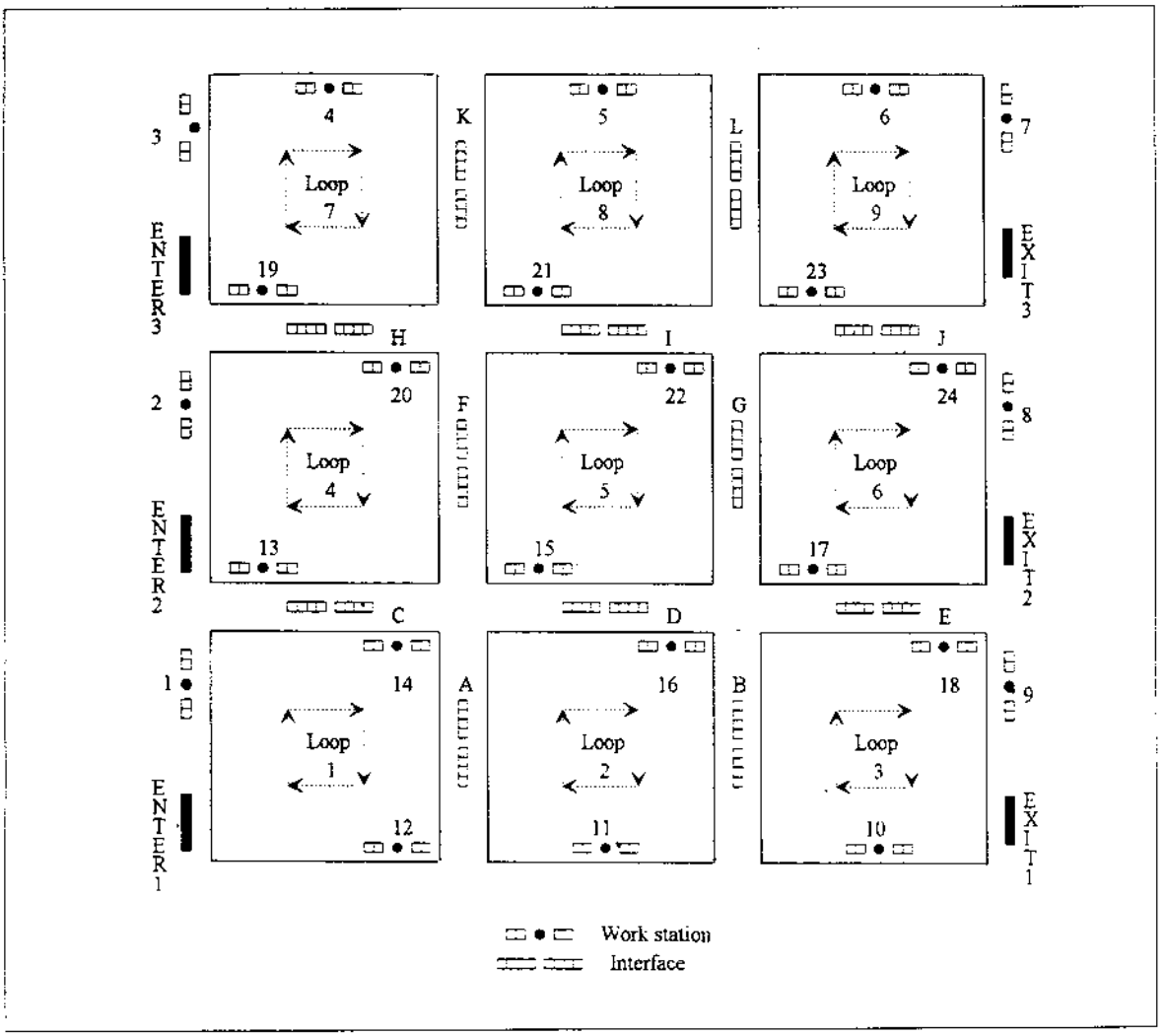


(Figure 7) Conventional Configuration for AGV System

sequences of loops in the tandem configuration. The routing sequences of three part types in each set are in Tables 1 and 2.

Therefore, the conventional system with its part flow data has its tandem counterpart. Likewise, the tandem system in which the part routing sequence matches the sequence of stations in its loops has its conventional counterpart Since each different configuration can be combined with each part flow set to complete one AGV system, there can be four different AGV systems.

Each system was simulated for 480 minutes with a warm up time of 50 minutes, and this 480-minute simulation is replicated 12 times. Thus, the actual simulation time for each system is 5160 (430*12) minutes. There are nine AGVs in each AGV system. For the tandem configuration, one AGV is assigned to each loop. The processing time at each work station is given as 5 minutes; the loading and unloading times of each AGV are given as 0.75 minutes. The velocity of



〈Figure 8〉 Tandem (9-Loop) Configuration for AGV System

〈Table 1〉 Part Rounting Sequence in Part Flow Set 1

	Part routing sequence(Station number)									
Part type 1	IN1	2	15	23	11	14	6	9	13	OUT1
Part type2	IN1	1	22	19	10	3	8	7	24	OUT1
Part type3	IN1	4	16	5	17	18	21	20	12	OUT1

each AGV are given as 0.75 minutes. The velocity of each AGV is 160 ft/min. Simulation results are in Tables 3 and 4.

〈Table 2〉 Part Routing Sequence in Part Flow Set 2

	Part routing sequence(Station number)									
Part type 1	IN1	1	14	12	16	11	18	9	10	OUT1
Part type 2	IN2	2	20	13	22	15	24	8	17	OUT2
Part type 3	IN3	3	4	9	5	21	6	7	23	OUT3

〈Table 3〉 Summary of Simulation Outputs for Conventional AGV System

	No. of parts arrived	No.of parts inducted	Throughput	Average flow time(min)	% of loaded vehicle travel
Part flow set 1	1217	1184	987	126.49	57%
Part flow set 2	1195	1188	1064	104.15	43%

〈Table 4〉 Summary of simulation Outputs for Tandem(9-Loop)AGV System

	No. of parts arrived	No. of parts inducted	Throughput	Average flow time(min)	% of loaded vehicle travel
Part flow set 1	1212	1002	171	296.95	38%
Part flow set 2	1199	1196	1105	93.45	31%

6. OBSERVATIONS

6.1 Conventional AGV System

As shown in Tables 3 and 4, and as expected, the system performance of the conventional system is not as sensitive as that of the tandem configuration to the part routing. In the conventional AGV system, the travel of AGVs to complete part routing is more flexible since each station is accessible to all AGVs, and there are many possible different ways to reach the destination station for the given locations of the AGVs. Thus, different part routing sequences do not affect the system performances very critically.

6.2 Tandem (9-Loop) AGV System

In a tandem AGV system, the loops which include the part-entering station have a high probability of being a bottleneck loop. Since only one AGV serves each part-entering loop, there is a limit to the rate of induction of parts to the system. In each part-entering loop, if all stations except the part-entering station do not require any delivery tasks and there always exist parts waiting to be delivered in the part-entering station, each time the AGV passes through the part-entering station, the part will be picked up and delivered to the interface of the entering loop. This maximizes the number of parts inducted into the system. Throughput will never exceed this limit. Even though the part arrival rate increases, the number of parts inducted will not increase.

Also, as shown in Table 4, the number of parts inducted is sensitive to the part routing sequence in the tandem system. With part flow set 1, 1002 units were inducted into the system. This is a low number of units compared to 1196 units for part flow set 2. Comparing

the number of parts inducted with that of the other configuration (Table 3), the tandem configuration with part flow set 1 seems to create a bottleneck loop. In the tandem system, in order to induct as many parts as possible, the AGV which belongs to the entering loop should not spend much time performing delivery tasks. The more time the AGV spends in delivery tasks, the more frequently the AGV passes through the part-entering station without picking up a part. In part flow set 1 (Table 1), the stations of the part-entering loop 1, that is, stations 1, 14, and 12, are in the routing sequences of different part types. Similarly, the stations of the part-entering loops 4 and 7 are in the routing sequence of different part types or are not sequentially located. Thus, a part arriving at one of these stations will require a delivery task to a station in another loop. This results in additional delivery tasks at the interfaces of the entering loop. In part flow set 2, stations of the part-entering loops 1, 4, and 7 are in the routing sequence of one part type and are sequentially located. This minimizes the number of delivery tasks at the interfaces. Therefore, in the tandem system, the part routing sequence is very critical to system performance. It is suggested for the entering loop to have only a part-entering station and interfaces.

The part routing sequences in part flow set 2 perfectly matches the station sequences in loops. However, since AGVs only move in one direction in each loop, the travel time from one loop to another loop may require a longer trip than the actual distance. For example, when a part (type 1) of part flow set 2 needs to be moved from station 12 (loop 1) to station 16 (loop 2) in the routing sequence, the routing path in Figure 8 is

12 - ENTER - 1 - 14 - A - D - 16.

If all paths are bi-directional, the path will be

12 - A - D - 16.

In summary, it is observed that there are some major factors that should be considered in developing the tandem configuration, and they are as follows:

- part routing sequences
- travel path between loops in part routing sequence
- design of the entering loop according to the part arrival processes
- vehicle speed

6.3 General Observations

In this experiment, the tandem AGV system with its matching part types in part routing sequences produce better system performances than the conventional AGV system. This is because the tandem AGV system has almost ideal configurations such as perfectly balanced load among loops, perfectly matching part types in its routing sequences, minimum number of parts' travel among loops, etc. In the real world, a conveyor system looks more suitable for these ideal environments than the tandem AGV system due to cost considerations. Also, this ideal tandem configuration is not the corresponding counterpart for a conventional AGV system. The corresponding counterpart should be produced from the given conventional AGV system which has already determined each part routing sequence. In this experiment, except the locations of workstations, the tandem configuration was constructed with its matching part routing sequences regardless of the configuration of the conventional AGV system. The methodology to produce the tandem counterpart has not been reported in the literature.

7. SUMMARY AND FURTHER RESEARCH

In this paper, the existence of objects and their relationships in AGV systems has been conceptualized. This conceptual organization was represented in the logical design of AGV systems by the object diagram.

Also, this logical design has been applied to the implementation of object-oriented classes providing the ability to create a model for AGV systems. The resulting simulation modeling environment, AgvTalk, includes 25 object classes and more than 300 object methods in its library for many detailed feature of AGV systems.

Since the complexities and specific natures of AGV systems do not allow easy construction of a simulation model with only stand-alone nature of objects, the hybrid approach in AgvTalk was proposed. In the hybrid approach, the possible life cycle of active objects are modeled in methods, and the methods are stored in the AgvTalk library. The hybrid approach eliminates the modeling process of vehicles, work stations, parts, and repair stations behavior in AGV systems; instead, it provides the selection process among behavior already built in the library. This selection process and data input process for model construction can be performed through the window- or menu-based user interface in AgvTalk.

In comparing features such as modeling AGVs, processes, dispatching rules, breakdown, and system layout, the main difference between AgvTalk and general purpose simulation languages is that transporters (e.g. vehicles in an AGV system) are not modeled as active objects in general purpose simulation languages. This results in a limited modeling environment for detailed and exact behavior. AgvTalk also provides natural constructs for modeling AGV systems separately and distinctly among physical objects, objects' behavior, and control of objects resolving the inherent problems in general purpose simulation languages.

In order to demonstrate the potential of AgvTalk, that is, the modeling capabilities by extensibility and reusability, the tandem AGV system was designed and extended in the AgvTalk environment. The tandem AGV system has a radically different configuration from the conventional AGV systems in the system network layout and travel behaviors of vehicles. By redefining the system network layout and travel behaviors of vehicles, and reusing the AgvTalk library already developed for the conventional AGV systems, the tandem AGV system was easily extended.

The following topics are suggested for further investigation.

- Implementation of a more comprehensive AGV system in AgvTalk. The work in the current AgvTalk focuses on the material handling system. The various configurations of the production system may be extended.
- Implementation of other than a job shop environment in AgvTalk. Using a hybrid approach, other material handling shop environments such as flow shops, shops with pull system, etc. may be developed.
- Development of a class and methods for free-ranging AGV systems. The library in AgvTalk would be useful except the class FixedPath. The operations defined in the class FixedPath should be redefined for vehicles' travel behaviors. This class would be extended as a subclass of the class Transporter at the same level as FixedPath.
- Provision of graphics and animation on simulation in AgvTalk. The MVC-triad in Smalltalk-80 and the standalone nature of objects in the AgvTalk library would be helpful.
- Development of other material handling systems and integration with them. The design concepts implemented in AgvTalk would be useful and might be reused in developing AS/RSTalk, ConveyorTalk, CarouselTalk, etc.

REFERENCES

[1] Bartholdi, J.J., Platzman, L.K., "Decentralized Control of Automated Guided Vehicles on a Simple Loop", *IIE Transactions*, pp76-81, March 1989

[2] Birtwistle, G.M., Lomow, G., Unger, B.W. and Luker, P.A., "Process Style Packages for Discrete Event Modeling: Data Structures and Packages in SIMULA", *Trans. Soc. Comp. Simulation*, Vol. 1,

No 1, pp61-82, 1984.

[3] Booch, G., *Object-Oriented Design with Applications*, Benjamin/Cummings Publishing Company, Inc., 1991.

[4] Bozer, Y.A., Srinivasan, M.M., "Tandem Configurations for AGV Systems Offer Simplicity and Flexibility", *Industrial Engineering*, pp23-27, Feb. 1989.

[5] Gaskins, R.J. and Tanchoco, J.M.A., "AGVSim2: a Development Tool for AGVS Controller Design", *International Journal of Production Research*, Vol. 27, No.6, pp 915-926, 1989.

[6] Goldberg, A. and Robson, D., *Smalltalk-80: The Language*, Addison-Wesley, 1989.

[7] Gong, D. and McGinnis, L.F., "An AGVS Simulation Code Generator for Manufacturing Applications", *Proceedings of the 1990 Winter Simulation Conference*, pp 676-682, 1990.

[8] Hodgson, T.J., King, R.E. and Wilson, C.M., "Analysis and Control of Multiple Vehicle AGV Systems", Technical Paper, Dept. of Industrial Engineering, North Carolina State University, 1990.

[9] Kay, M.G., "Global Vision for Free-Ranging AGVS Control", Tech. Report CRIM-91-1, North Carolina State University: Center for Robotics and Intelligent Machines, 1991.

[10] LaLonde, W.R., Pugh, J.R., *Inside Smalltalk*, Volume 1, Prentice Hall, 1990.

[11] LaLonde, W.R., Pugh, J.R., *Inside Smalltalk*, Volume 2, Prentice Hall, 1990.

[12] Law, B.M. and Haider, S.W., "Selecting Simulation Software for Manufacturing Application: Practical Guidelines & Software Survey", *Industrial Engineering*, pp 33-46, May 1989.

[13] Law, B.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw Hill, pp311-312, 1982.

[14] Meyer, B., *Object-Oriented Software Construction*, Prentice Hall International (UK) Ltd., Hertfordshire, Great Britain, 1988.

[15] Najmi, A. and Stein, S.J., "Comparison of Conventional and Object-Oriented Approaches for Simulation of Manufacturing Systems", *1989 IIE Integrated Conference & Society for Integrated Manufacturing Conference Proceedings*, pp 471-476, 1989.

[16] Pegden, C.D., Shannon, R.E. and Sadowski, R.P., *Introduction to Simulation Using SIMAN*, McGraw-Hill, New York, 1990.

[17] Pritsker, A.A.B., *Introduction to Simulation and SLAM II*, Third Edition, Halsted Press, New York, NY, 1986.

KYUNG SUP KIM is an assistant professor of Industrial Systems Engineering Dept. at Yonsei University. He received a B.S. in mechanical engineering from Yonsei University, and he received M.S. and Ph.D. degrees in industrial engineering from University of Nebraska and North Carolina State University respectively. His research interests are in simulation modeling and analysis, object-oriented simulation, material handling system, and computer integrated manufacturing.

RUSSELL E. KING is an associate professor of industrial engineering at North Carolina State University. His research and teaching interests are in production scheduling and control, real-time control of automated systems, and stochastic processes. He is a member of IIE, ORSA and TIMS and is an associate editor of IIE Transaction. He holds a B. S. in systems engineering and M.S. and Ph.D. in industrial engineering from University of Florida.