AGV 시스템에 대한 객체지향 시뮬레이션

Object-Oriented Simulation Approach for AGV Systems

김경섭*
Kyung-Sup Kim

Abstract

This paper presents an object-oriented simulation approach, AgvTalk, to design and analyze AGV system configuration and control. Smalltalk-80 is used as an implementation language in AgvTalk. AgvTalk includes 25 object classes and more than 300 object methods in its library. Compared to general purpose simulation languages, AgvTalk provides several important benefits. First, the hierarchical features and modularity create possibilities for the extension and reuse of simulation object components. This extensibility and reusability provide more flexible modeling capabilities for simulation of many alternative AGV systems. Second, detailed behavior of each object in the AGV system can be modeled easily and exactly in AgvTalk because there are no limiting modeling constructs. Third, AgvTalk provides a user-friendly simulation modeling environment through the MVC-triad of Smalltalk-80.

1. INTRODUCTION

1.1 Background

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.[7].

When designing and planning an automated material handling system, a large number of factors must be considered. In particular, with an automated guided vehicles (AGV) system, things such as the number of vehicles, a

*삼성데이타시스템

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 [14] or SLAM [15]. 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 [10] 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.

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 [3], and SIMAN-based SCG [5]. However, these still do not overcome the modeling limitations of their implementation languages.

Systems and processes in most manufacturing environments are constantly changing to keep pace with new technologies. 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 [13]. 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 hanguage, was developed in the 1960s for simulation purposes [1], 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 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 [12]. 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 [8]. Every representation and characteristic of an object defined in a superclass is *inherited* by objects of its subclasses; therefore, differences from the superclass are definced in the subclass. From the above characteristics, an object-oriened approach provides such advantages as reusability, extensibility, modularity, and portability.

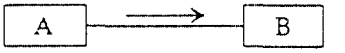
2. OBJECT-ORIENTED DESIGN FOR AGV SYSTEMS

An AGV system consists of objects like vehicles, workstations, machines, and jobs (parts). A natural one-to-one correspondence can be applied between physical objects in a real AGV system and instances of simulated objects. Message passing in an object-oriented approach (instead of

function calls in a conventional approach) is a more natural and convenient way to represent an AGV system. For example, when a message tasks is sent to an object representing a workstation, the first object (part) in the input buffer is transferred to the workstation and processed, and the status of the input buffer is updated. The same message can be sent to an AGV to define its travel characteristics from one station to another. The inheritance of methods and representation by a hierarchical class structure allows the construction of new objects from existing objects by adding, reducing, or modifying their functionality. For example, the modeling of a general AGV system may be extended to more specific AGV systems by adding their own specific operational characteristics in the subclass.

In this section, the logical design of the AGV system is discussed from an object-oriented view point. This discussion includes the object-oriented design of a model, and an experiment for AGV systems. The design is represented by an object diagram [2]. Since it is generally infeasible to capture all the subtle details of a complex AGV system, the key mechanism will be represented in an object diagram.

An object diagram is used to show the existence of objects and their interrelationships in the logical design of a system. The purpose of the diagram is to illustrate the dynamic semantics of the key mechanism in the logical design. A relationship between two objects simply means that the objects can send messages to one another, and a line is used to designate these relationships. To specify the visibility between the objects, the arrowhead line is added next to the line. The symbols used for the object diagram are shown in Figure 1.



There is a relationship between objects A and B.
An object B is visible to an object A.

Figure 1. Icons for Object Relationship

2.1 Object-Oriented Design of a Model

2.1.1 AGV System

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 [2]. 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 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 matches the

physical architecture of the AGV system naturally. Thinking about operations that the interface can meaningfully 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.

Looking at this same problem from the opposite 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

This design protocol leaves us with clear separation and modularity 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 showen in the object diagram in Figure 2.

2.1.2 Material Handling System

Figure 3 shows the semantics and relationships between objects in the material handling system. The material handling system controls how AGVs move from one location to the other location. The material handling system only communicates with the Interface in two ways. First, the communication is performed between the AGV and the Interface when the AGV arrives at the station for a pick up or delivery task. Second, when a work station finishes processing a part and the part needs to be moved to the next station according to the part routing, the Interface where the finished part stays directly sends message activateIdleVehicleIfAvailable to the AGVSController in the material handling system.

The AGVSController is the information collection and distribution center in the material handling system. Every object in the material handling system is visible to the AGVSController; however, the AGVSController is visible only to an instance of an AGV. With the exception of the AGVSController and the AGV, all other classes are encapsulated by themselves. Each time an AGV stops in the network, the AGV passes the message update to the AGVSController. In response to this message, the AGVSController passes appropriate messages to the simulation support objects (e.g., theRoute, theZone, etc.) to receive the information necessary for the AGV to restart. This information is distributed to the AGV with an appropriate message for the next travel. In fact, AGV movement consists of travel through a sequence of the divided zones. Each time an AGV complets a zone travel, the AGV passes the message update to the controller with information of the current zone. Then, the controller collects relevant information about the next zone such as zone identification, zone availability, zone distance, zone travel time, etc. If the next zone is available, the controller passes the message travelZone to the AGV. If not, the message waitForAWhile is sent to the AGV from the controller.

2.1.3 Production System

As can be seen in the object diagram (Figure 4), there is one class, **WorkStation**, in the production system. The WorkStation communicates only with the Interface. When the WorkStation needs to process a part, the message releaseLoad-

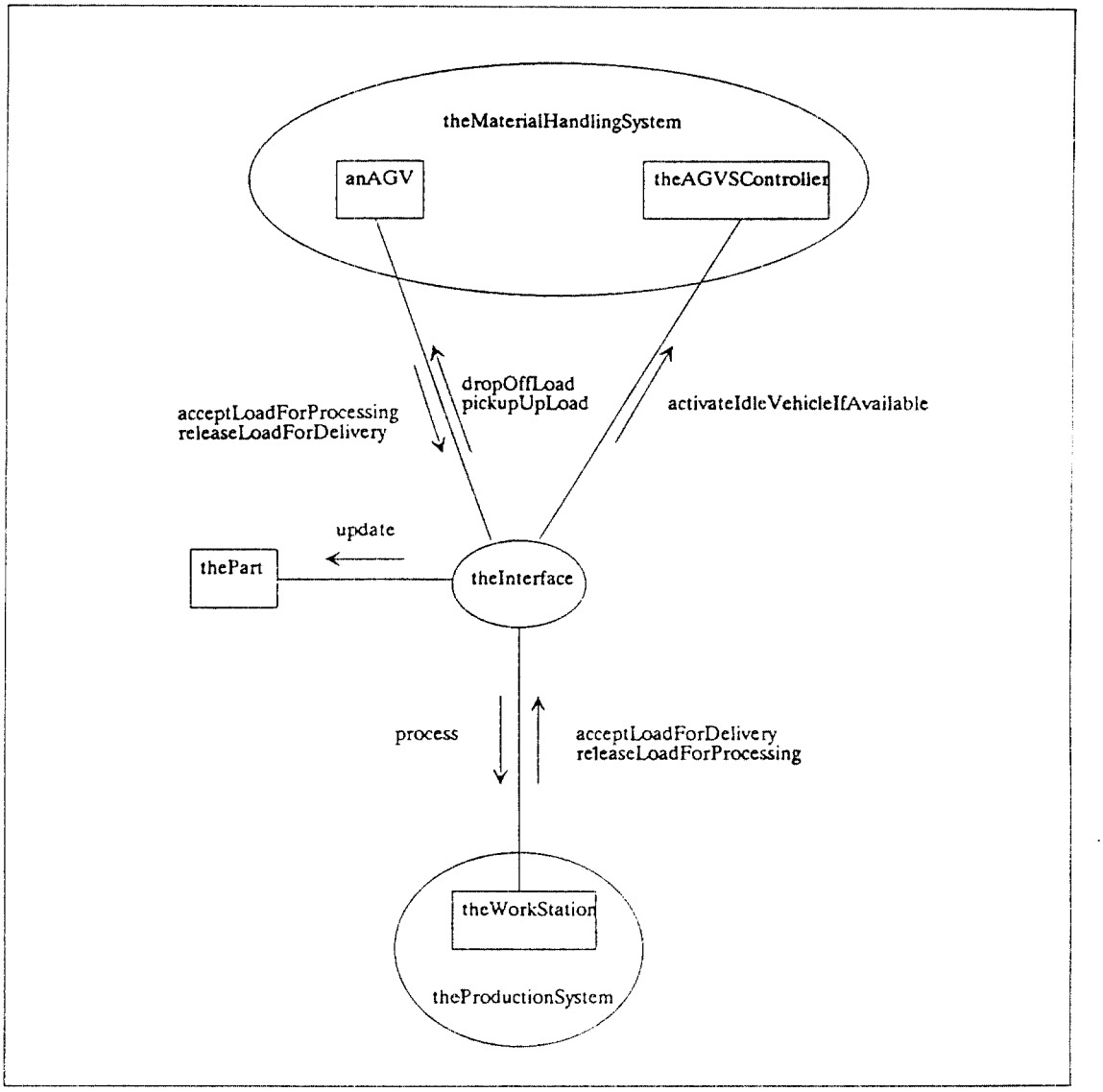


Figure 2. AGV System Object Diagram

ForProcessing is sent to the Interface. Then, the Interface passes the message process. If the resource is not available, the processing operation is blocked temporarily. When the WorkStation finishes the processing operation and passes the message acceptLoadForDelivery, the Interface activates the work center initiated dispatching rule in the material handling system if an idle vehicle is available. This is done by sending activateIdleVehicleIfAvailable to the AGVS Controller.

2.1.4 Interface

The object diagram for the Interface is shown in Figure 5. The material handling system and the production system

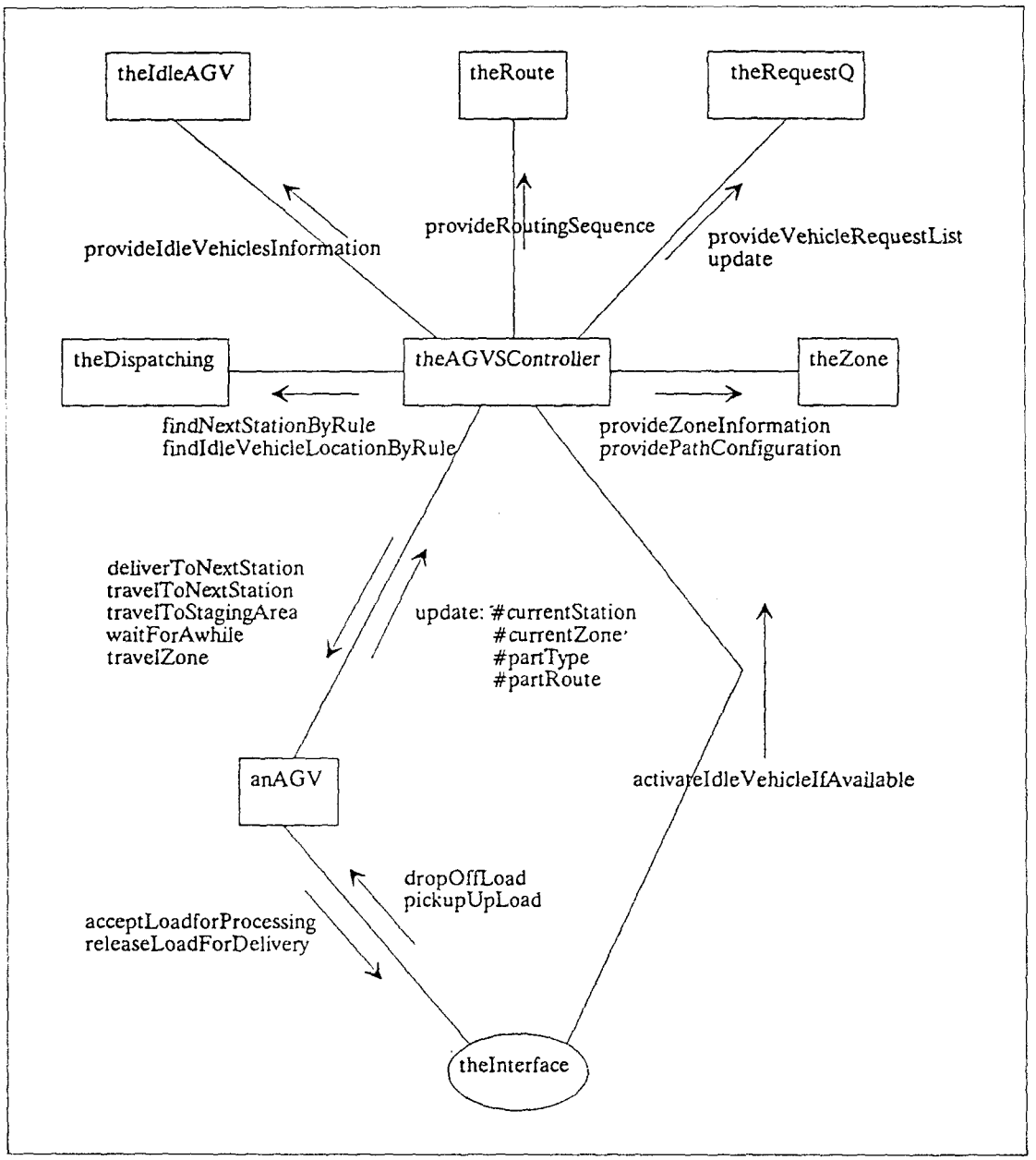


Figure 3. Material Handling System Object Diagram

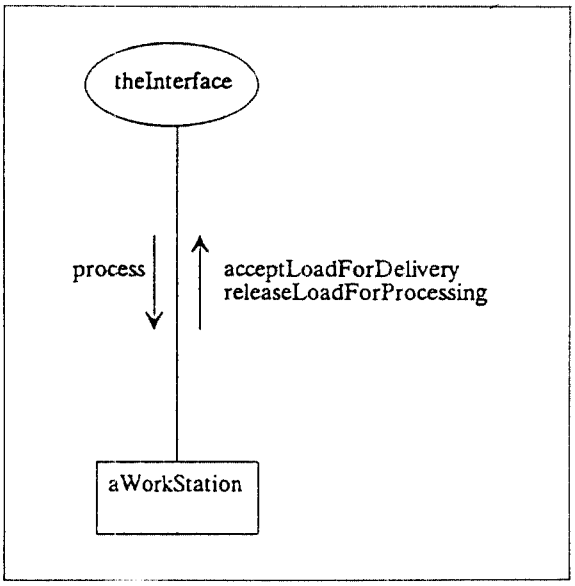


Figure 4. Production System Object Diagram

communicate with each other through the Interface. Messages sent to the Interface from the material handling or the production system, (such as acceptLoadForProcessing, releaseLoadForDelivery, acceptLoadForDelivery, and releaseLoadForProcessing) are processed as follows. First, the Interface sends the message update to itself. Then, the status of the appropriate subclass (InputBuffer of OutputBuffer) is updated. In response to the message update, the buffer sends the appropriate message (enter: or exit:) to the AGVSStatisticsControl with the instance of the Part as a parameter. The AGVSStatisticsControl calculates the flow time and time averaged statistics internally. Whenever the Interface updates its status, the status of the Part (Job) is also updated by sending message update to itself. This status includes the current location (current station, input or output buffer), the entry time of the current buffer, etc.

2.2 Object-Oriented Design for an Experiment

Figure 6 represents the top level of an experiment. Three objects are defined: theDiscreteEventSimulation, theAGVProblem, and theAGVSystem. TheDiscreteEventSimulation provides the discrete-event simulation framework in which theAGVProblem is defined. As discussed in the design of the model, theAGVSystem includes the material handling system, the production system, the interface, the part, and the network.

The experimental portion is composed of two initialization processes at the beginning of the simulation. These are initializations of theAGVProblem and theAGVSystem. The initialization of theAGVProblem includes the definition of any stochastic arrival processes into the AGV system and resources. For example, when an AGV receives the message defineArrivalSchedule from theAGVProblem, the instance of the AGV is initialized and the arrival pocess of that instance is defined, and it sends itself the message defineInitialLocation, defineVelocity, and defineAcceleration. Also, the AGV sends the message defineArrivalSchedule to the objects Breakdown and Recharging to define their stochastic distributions. The objects which receive the messages defineArrivalSchedule (theAGV, theBreakdown, theRecharging) send a message to an appropriate distribution object which represents their arrival processes. This is illustrated in Figure 7.

The implementation of thePart in an experiment is similar to that of theAGV. The Part receives defineArrivalSchedule from theAGVProblem and sends messages to the distribution object to define its arrival process and the processing time distribution. In addition, the instance of thePart defines its routing sequence and the entry location by sending itself the messages defineRoutingSequence and defineEntryLocation.

The initialization of theAGVSystem consists of the initialization of its lower level classes, that is, the material handling system, the production system, the interface, and the network system.

The initialization of the material handling system represents clearing the system statistics and redefining them. In particular, by sending the message selectRules to the Dispatching, two rules in each category of the vehicle initiated rule and the work center initiated rule are selected. The initialization of the Production System and the Interface defines the capacities of the work sations and buffers,

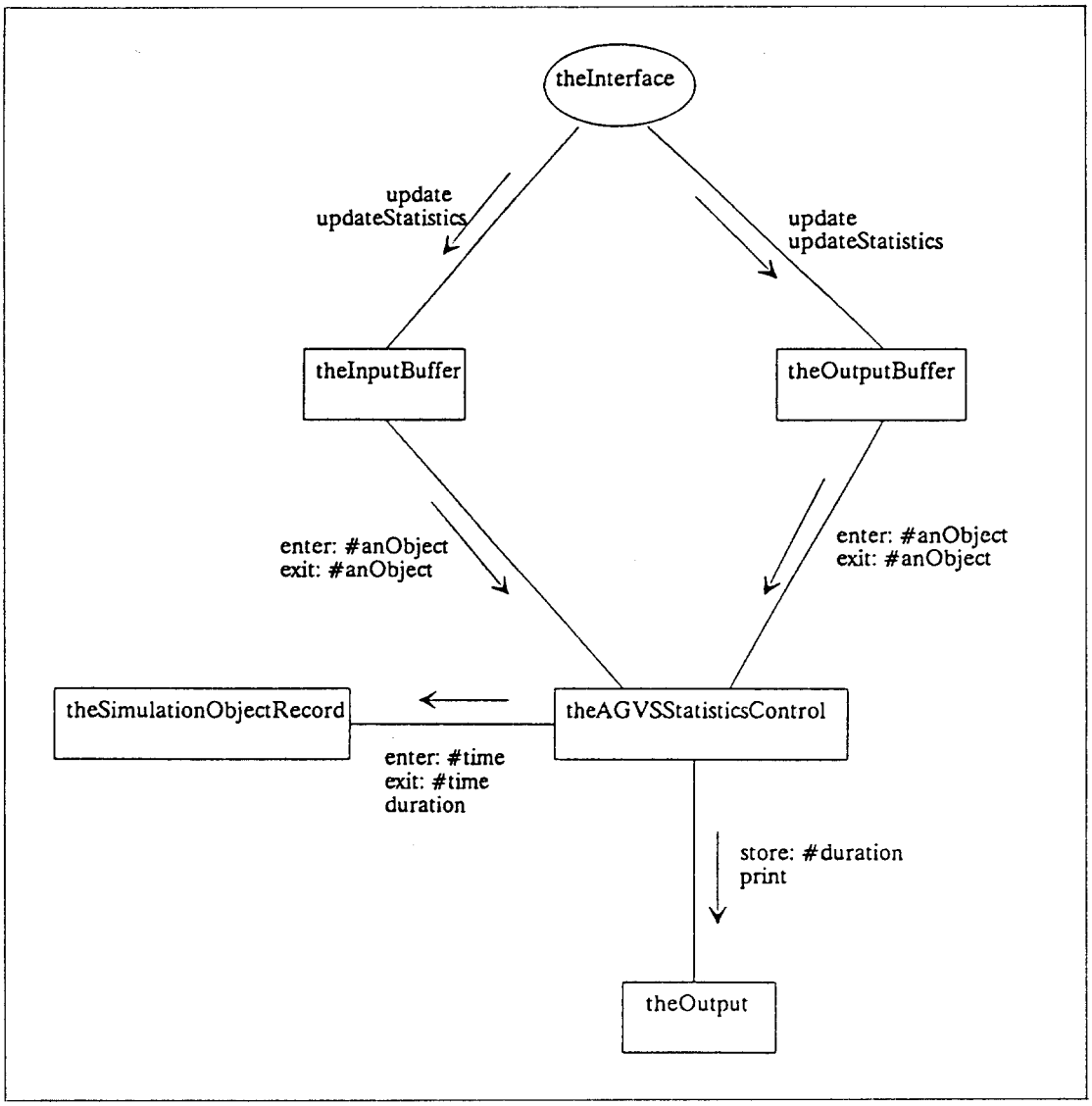


Figure 5. Interface Object Diagram

respectively by sending the message defineCapacity. In the initialization process of the NewworkSystem, the messages defineControlPoints and constructNetworkandShortestRoute are sent to the objects Node and Net, respectively. Then, theNet constructs the system network by connecting the instancs of theNode defined as control points. The object flow (link) of the connection can be uni- or bi-directional. In a special case, spur-link can be defined with a dead end in the link. Once the system network is constructed, the shortest routes between any two combination of the control points are generated in the class Net. These generated routes are stored into the object Route by passing the message

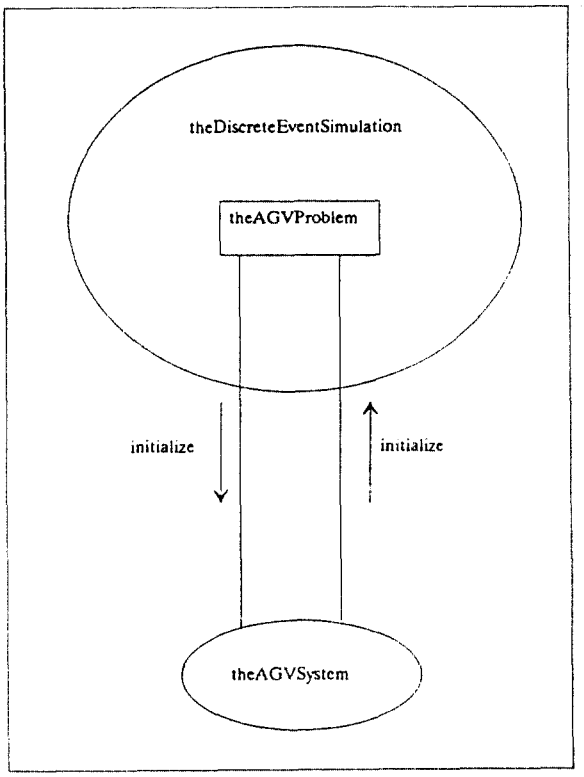


Figure 6. Top Level Object Diagram (Experiment)

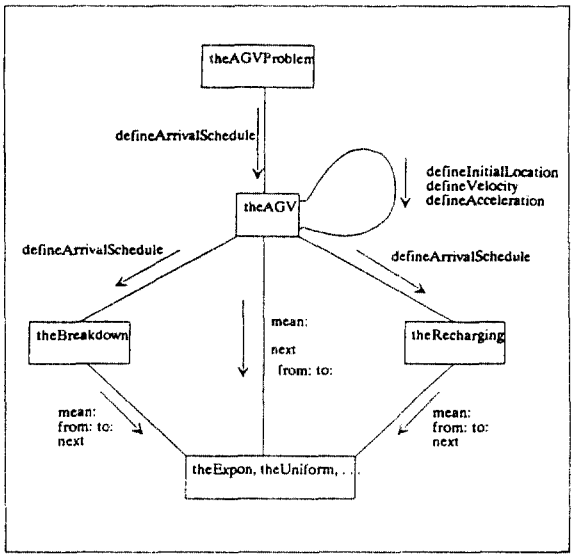


Figure 7. AGV Object Diagram (Experiment)

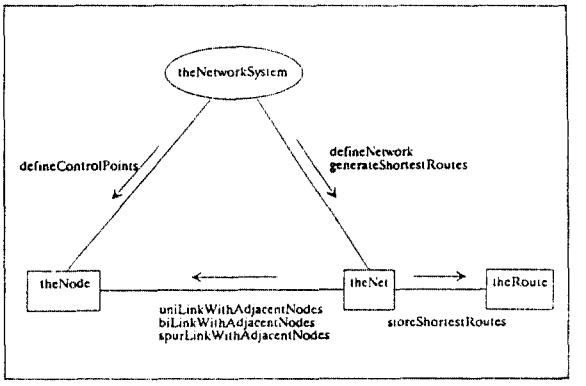


Figure 8. Network System Object Diagram (Experiment)

storeShortestRoute. This is shown in Figure 8.

2.3 AgvTalk: AGV System Simulation Environment in Smalltalk

The design concepts discussed in previous sections have been implemented in Smalltalk-80 [4]. 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. MODELING PROCESS IN AGVTALK

To provide the insight on how a simulation model is developed in AgvTalk, the modeling process is presented in this section according to the AgvTalk structure shown in Figure 9.

3.1 Model Specification

The main modeling advantage of AgvTalk is the simplicity in defining and changing system requirements and specifica-

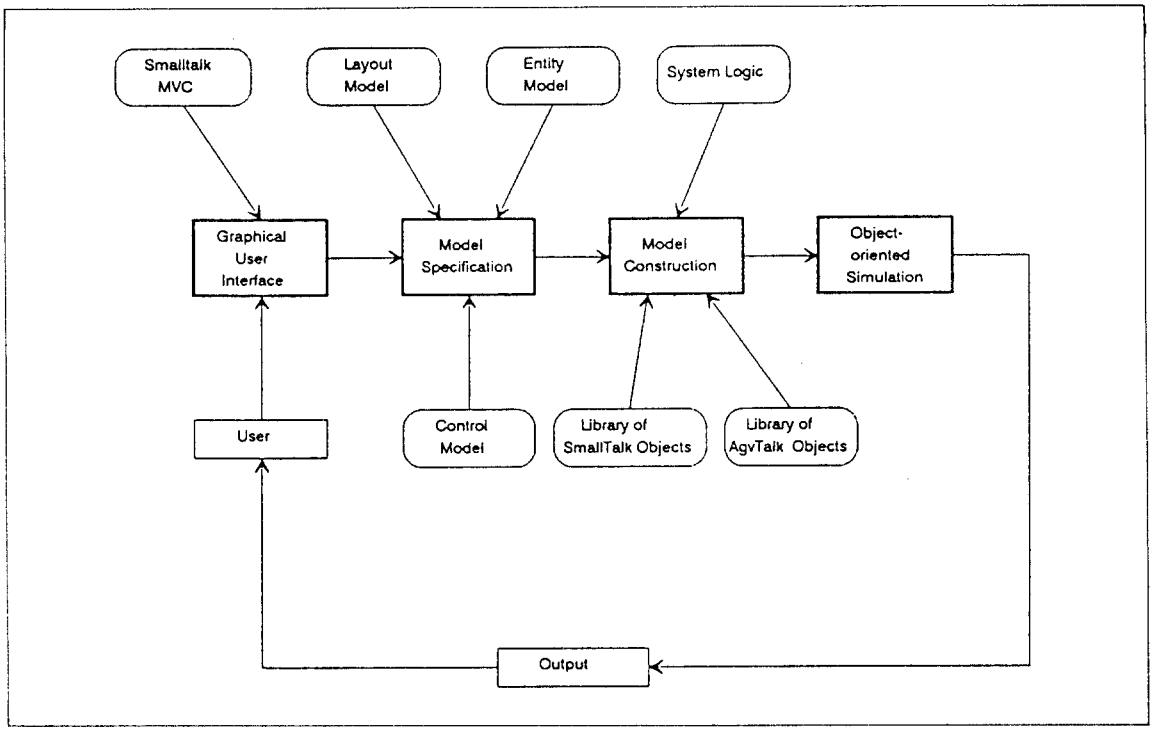


Figure 9. The AgvTalk Structure

tions. By invoking the method initialize in class **AgvProblem**, the system is initialized through the MVC (Model-View-Controller) triad in Smalltalk-80 [9]. The system initialization includes definition of the layout model, entity flow model and control model.

3.1.1 Layout Model Specification

Layout model specification involves description of the system guide path configuration and resources, including:

- identification of control points (workstations, intersections, staging area)
- definition of distances and number of zones between any two adjacent control points
- definition of uni- or bi-directional paths between any adjacent control points

The required information for the abvoe description is provided by invoking the method constructNetworkAndFindShortestRoutes in class **AgvProblem**. Once information for all combinations of adjacent control points is given, shortest distance routes between control points are produced using Dijkstra's shortest route algorithm.

3.1.2 Entity Flow Model Specification

Entity flow model specification includes a description of the physical objects such as number of vehicles, workstation characteristics, part routing, etc, including:

- —Vehicle information
  - —vehicle number
  - —speed, acceleration/deceleration
  - —loading/unloading time
  - —loading capacity
  - —breakdown/recharging rate distribution
- —Workstation information
  - —location of pickup/deposit station

- number of resources (e.g. machines)
- inter-breaking time distribution
- repair time distribution

- Part information
  - number of part types
  - arrival point into the system
  - interarrival time distribution
  - processing route
  - processing time distribution at each station
- Buffer information
  - capacity of the input buffer and output buffer of each workstation.

The above information is initialized by invoking the method initialze in **AGVProblem**, which in turn invokes the initialization methods of the physical object classes, that is, **AGV, WorkStation, Part, Buffer, InputBuffer** and **OutputBuffer**. For example, Figure 10 represents the initialization process of the **Part** instance through the menu-driven interface.

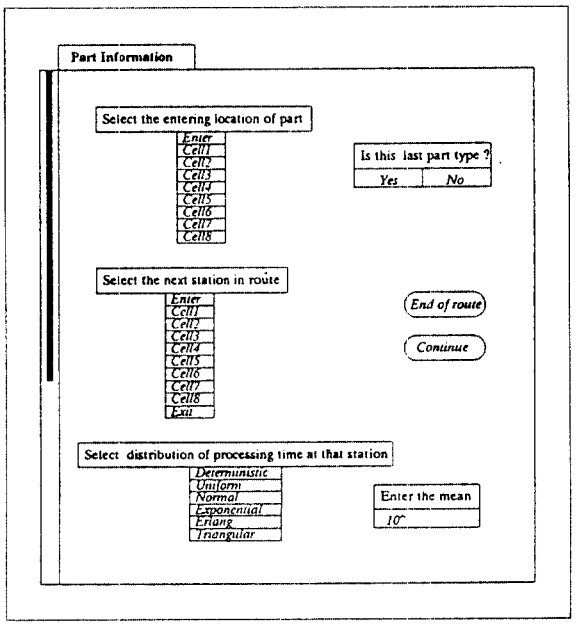


Figure 10. Initialization Process of Part Instance

3.1.3 Control Model Specification

Control model specification includes determination of the followings:

- Control policy
  - Vehicle initiated rule
    - shortest distance rule
    - maximum outgoing queue size rule
    - minimum remaining queue size rule
    - first-come first-serve rule
    - modified first-come first-serve rule
    - random workstation rule
    - extracted rule (ER, ERR) [6]
  - Workstation initiated rule
    - nearest vehicle rule
    - longest idle vehicle rule
    - random vehicle rule

The above information is also initialized by invoking the method initialize of class **AGVProblem**, and in turn, class **Dispatching**. The initialization process in the class **Dispatching** by the menu-driven interface is shown in Figure 11.

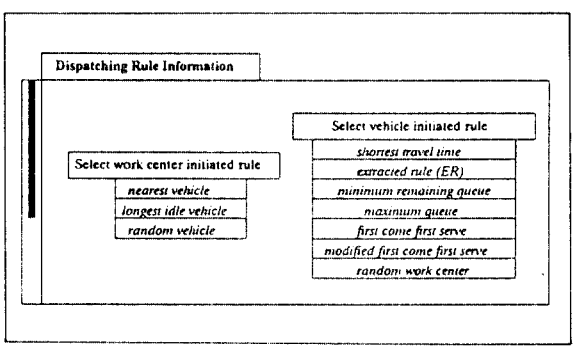


Figure 11. Initialization Process of Dispatching Class

3.2 Model Construction

In order to model a complex manufacturing system in an object-oriented environment, the object classes developed in section 2 should be assembled in some way. One way is for a user to state the system logic from the beginning to the end into the model file. The limitation of this approach is

that the user must be very familiar with the libraries and the specific system being analyzed for the fast model-building. To overcome these limitations, a hybrid approach is proposed.

3.2.1 Hybrid approach

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 **Work Station** 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 the generic scenarios 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 characterisics 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 12. 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. VALIDATION OF AGVTALK

For the validity of AgvTalk, the given AGV systems are simulated with AgvTalk and SIMAN, and the output data are compared. Assuming that a SIMAN model correctly models the AGV system, the percentage deviations of AgvTalk output are tested with SIMAN output. The measures of performance in these tests are throughput and flow time.

AgvTalk and SIMAN are capable of modeling many detailed features of AGV systems, however, how these features are modeled may be different between AgvTalk and SIMAN. Therefore, a simple AGV system is modeled to provide as similar system logic as is possible between an AgvTalk model and a SIMAN model. The assumptions for the AGV system in this validation test are as follows:

- Vehicle breakdown is considered.
- Battery recharging of vehicles is not considered.
- Vehicles travel with instantaneous acceleration and deceleration.
- Vehicles do not pass each other.
- Vehicles travel based on the shortest path.
- Vehicles transport one unit load at a time.
- First-in-first-out(FIFO) is used for the vehicle initiated rule.
- Shortest travel distance rule is used for the workcenter initiated rule.
- Only one vehicle can occupy a zone at one time.
- Idle vehicles are sent to the staging area.
- The routing sequence of each part type is deterministic.

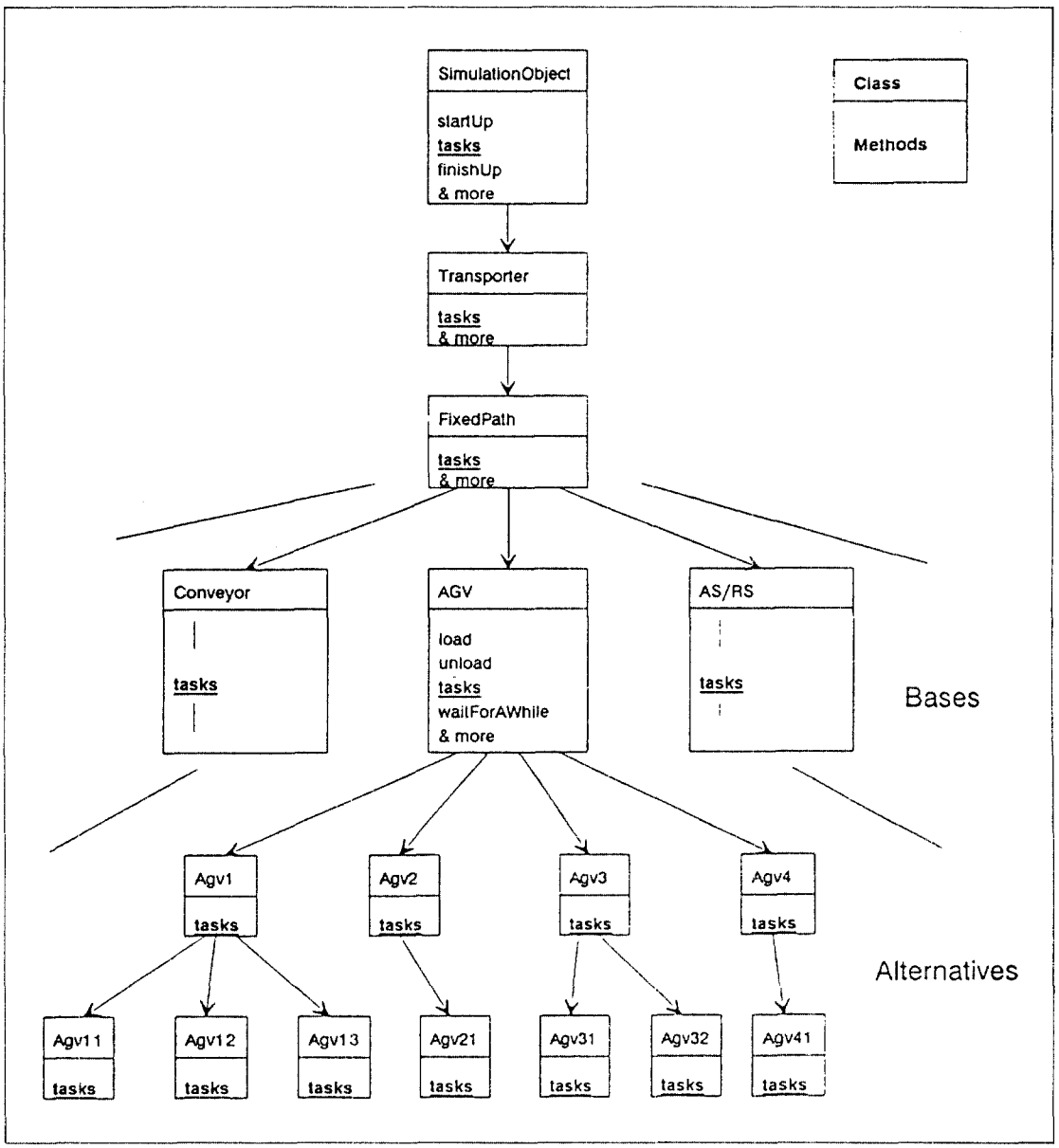


Figure 12. Inheritence and Polymorphism in Hybrid Approach

The layout configuration for the AGV system is presented in Figure 13, where there are 8 work stations, one staging area, and one recharging station. At each work station, the locations of pick up and deposit stations can be different. Three different part types are processed in a job shop environment, and their arrival processes, routing sequences (station ID), and the corresponding processing times (minutes) are as follows:

Part type 1						
Arrival process	:	Exponential (8)				
Routing sequence	:	ENTER	3	7	6	EXIT
Processing times	:	1	3	5	3	0
Part type 2						
Arrival process	:	Normal (9, 2)				
Routing Sequence	:	ENTER	4	5	EXIT	
Processing times	:	1	4	5	0	
Part type 3						
Arrival process	:	Triangular (10, 13, 15)				
Routing sequence	:	ENTER	1	2	8	EXIT
Processing times	:	1	5	5	6	0

There are 12 AGVs in the system. The loading and unloading times of the AGVs are 0.25 minutes. The velocity of each AGV is 200 fpm. Because of the limitations of SIMAN in modeling dispatching rules, only the shortest-distance rule and the first-come-first-serve rule are used for the work center initiated rule and vehicle initiated rule, respectively.

The AGV system is simulated with AgvTalk and SIMAN for 480 minutes with a warm up time of 50 minutes for each of 12 replications. From the replications, the IID measures of performance are collected. Since 12 runs are not enough for a classical approach, a jackknife approach [11] is used for an analysis of percentage deviation data of two outputs. In Table 1, the simulation output results are summarized for different performance measures.

For each performance measure, the summary for $\bar{Z}$ and its 95% confidence interval is provided in Table 1, where $\bar{Z}$ is the estimator for the percentage deviation data of AgvTalk from SIMAN. The results show that the estimators for the percentage deviation were less than 2%, and the confidence intervals included zero for all performance measures tested, indicating AgvTalk generates almost identical simulation outputs as SIMAN.

5. MODEL DEVELOPMENT AND EXECUTION TIMES COMPARISONS

5.1 Model Development Time Comparison

The model construction process in AgvTalk is simple due to the hybrid approach. In the hybrid approach, many models for conventional AGV systems are already developed. Thus, the model for the given AGV system can be developed by selecting the appropriate class which has the appropriate

Table 1. Summary for Percentage Deviation between AgvTalk and SIMAN Output Data

System Performance Measure	$\bar{Z}$	95% CI for $\bar{Z}$
Average flow time of all parts	-0.00062	(-0.02132, 0.02008)
Average flow time of part type 1	-0.00310	(-0.02713, 0.02093)
Average flow time of part type 2	-0.00548	(-0.02366, 0.01270)
Average flow time of part type 3	0.01766	(-0.00214, 0.03736)
Throughput	-0.00224	(-0.03621, 0.03173)

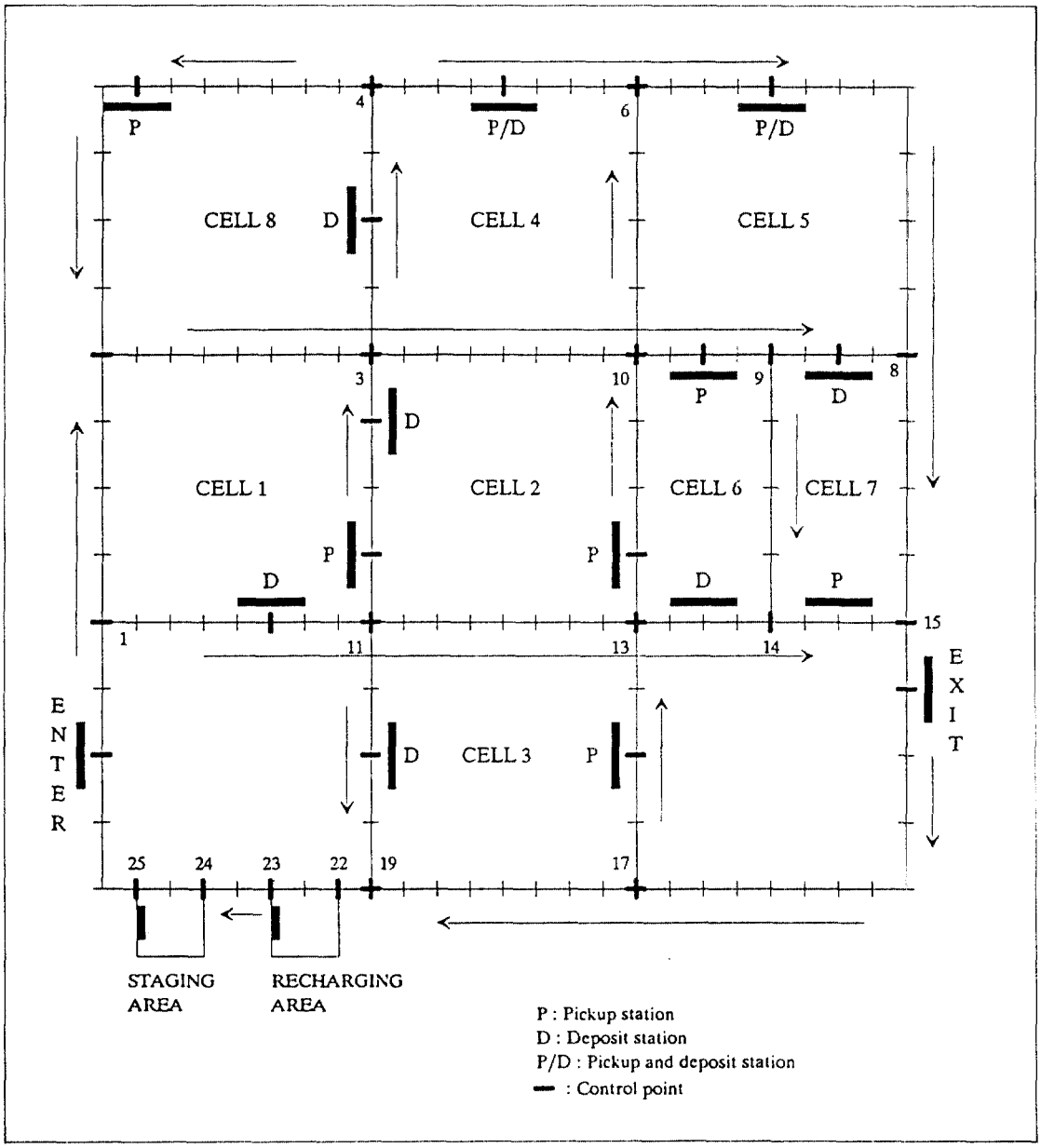


Figure 13. Layout Configuration of AGV System

system logic.

Some advantages in AgvTalk were discussed in modeling such features as breakdown, dispatching rules, different locations of pick up and deposit stations, bidirectional link, etc. In AgvTalk, the additional modeling process for these features is very minimal, and all these additional modeling

features are defined in the model specification process and are as simple as specifying the parameter values or selecting the appropriate menu by clicking the mouse.

However, in SIMAN, as the AGV system becomes more complex, the modeling effort and time increase rapidly. Unlike in AgvTalk, the modeling of AGV characteristics such as breakdowns and recharging requires creating entities in a model frame with a series of blocks which have sophisticated functions. Some high level constructs for controls such as different vehicle dispatching rules may not be implemented by the model developer because the functioning of these controls is hidden inside SIMAN. Station submodels are very useful for modeling workstations, however, they have limited capability in modeling workstations which have different locations of pick up and deposit stations. Moreover, because the functions of blocks defined in SIMAN material handling features do not always agree with the activities of objects in a real system, the more complex AGV system requires more modeling efforts and time.

In summary, even though the absolute model development time for AgvTalk and SIMAN cannot be determined, the relative model development time is summarized and compared in Table 2 where the model development time for a simple AGV system with AgvTalk is defined as 100 time units. This comparison is based on the assumption that the model developer has a through knowledge about an AgvTalk library and SIMAN material handling features. For a simple AGV system, the assumptions made in section 4 are applied.

5.2 Simulation Model Execution Time Comparison

From the point of simulation model execution time, AgvTalk is not as capable as SIMAN. The amount of time needed to execute an AgvTalk model is roughly seven or eight times longer. This statement is made from the test results in validation activities where an AgvTalk model and a SIMAN model of the same AGV system were executed repeatedly (12 replications) on a DEC station 3100. The execution times of an AgvTalk model and a SIMAN model for the AGV system summarized in Table 3.

However, the longer execution time of an AgvTalk model is not perceived as a major disadvantage because the model execution time (seconds, minutes) is much less than the model development time (hours, days) in which Agv Talk is much more favorable than SIMAN. Also, the continued increase in computer processing speed makes the long execution time less significant.

6. SUMMARY

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. This design includes the object-oriented design of a model, and an experiment for AGV systems. 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 features 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

Table 2. Model Development Time Comparison between AgvTalk and SIMAN

	Model development time with AgvTalk	Model development time with SIMAN
Base AGV system	100	300
Breakdown	10	150
Recharging	10	100
Pick up and deposit stations	10	200
Vehicle dispatching rule	10	300 or not possible
Bidirectional or spur links (without a deadlock)	10	300
Each additional feature	10 - 30	100 - not possible

Table 3. Execution Time Comparison between an AgvTalk Model and a SIMAN Model for AGV System

Replication	1	2	3	4	5	6	7	8	9	10	11	12
AgvTalk (seconds)	351	349	353	347	351	346	348	347	351	347	348	350
SIMAN (seconds)	47	46	47	47	48	46	47	46	47	48	47	47
Ratio	7.5	7.6	7.5	7.4	7.3	7.5	7.4	7.5	7.5	7.2	7.4	7.4

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.

REFERENCES

[1] 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, 1984, pp. 61-82.

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

[3] 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, 1989, pp. 915-926.

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

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

[6] 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.

[7] 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.

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

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

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

[11] Law, B.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw Hill, 1982, pp. 311-312.

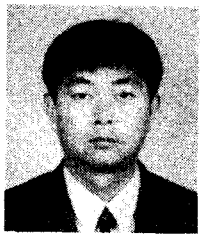
[12] Meyer, B., *Object-Oriented Software Construction*, Prentice Hall International (UK) Ltd., Hertfordshire, Great Britain, 1988.

[13] 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*, 1989, pp. 471-476.

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

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

저자소개



김경섭

1982 연세대학교 기계공학 학사
1986 미국 University of Nebraska-Lincoln 산업공학 석사
1993 미국 North Carolina State University 산업공학 박사
현재 삼성데이타시스템 CIM개발실/생산물류정보팀
관심분야: 시뮬레이션, 물류시스템, 제조시스템 등