Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

AN OBJECT-ORIENTED FRAMEWORK FOR SIMULATING AUTOMATIC DATA COLLECTION SYSTEMS

Manuel D. Rossetti
Bradley J. Hobbs

Industrial Engineering Department
University of Arkansas
Fayetteville, AR 72701, U.S.A.

Paul D. Faas

Logistics Readiness Branch
Warfighter Readiness Research Division
Human Effectiveness Directorate
Air Force Research Laboratory
Wright-Patterson Air Force Base, OH 45433, U. S. A.

ABSTRACT

In this paper, we discuss the design and use of a proto-type object-oriented framework for simulating automatic data collection systems within their operational contexts. We motivate the purpose of the framework and how the framework can be used through the use of a simple scenario on an airbase. In addition, we overview the design by examining the major conceptual artifacts within the object-oriented model. The framework is built on a Java Simulation Library (JSL) and permits easy modeling and execution of simulation models. The results and discussion indicate the flexibility and power of modeling with the framework. Finally, we summarize our future research efforts to model more complicated automatic collection systems which include health monitoring systems within the Air Force's new sense and respond logistics paradigm.

1 INTRODUCTION

Automatic data collection systems are information systems which utilize computerized data collectors to capture and record data associated with events at or near the spatial occurrence of the event and at or near the actual time of the event. Cohen(1994) defines a computerized data collector as an "independent entity that captures, stores, processes and forwards data to a host computer". The increased use of automatic identification systems has enabled the widespread implementation of automatic data collection systems. Automatic identification systems use sensors to automatically identify physical objects within a system given some pre-established coding or classification scheme. Examples of computerized data collectors include programmable logic controllers, portable data terminals, and radio frequency decoders. Examples of automatic identification devices include bar code readers and radio frequency tagging devices.

This research is focused on creating an object-oriented framework for the simulation of automatic data collection (ADC) systems. To provide a context for this research, we present and discuss our prototype designs for examining the impacts of ADC technology within an Air Force aircraft maintenance scenario. Much of this technology is based on sensors and automatic data collection devices, either on the aircraft or on the flight line. There exists a need for flexible models of the maintenance environment with enough detail and accuracy to serve as a mechanism for impact analysis of various types of maintenance innovations. In addition, these models should take into account realistic field measurement plans and procedures that allow the validation and benchmarking of unit-level impacts of innovations at the worker level.

Automatic data collection is appropriate for these systems because they are inherently event-based or Discrete Event Dynamic Systems (DEDS). Discrete event dynamic systems consist of sets of interacting objects, which compete for services from a variety of resources dynamically at discrete points in time. Automatic data collection (ADC) systems support system monitoring by capturing data on the interactions of the objects as they compete for resources within the DEDS. Despite their growing use, automatic data collection systems and management information systems have not produced all of the desired benefits in terms of their ability to improve the socio-technical productivity of the system. The introduction of an automatic data collection system can significantly impact the operational performance of the system in both positive and negative ways. The purpose of our simulation framework is to provide engineers the tools they need to analyze the impacts of ADC systems on a *simulation of the actual operational situation*. Thus, we need to design a framework that allows both the simulation of the operational setting and the simulation of the ADC system.

In this paper, we present our initial designs and preliminary testing of the framework. To provide context for our research and to contrast it with other work, we present a brief review of background literature in Section 2. In Section 3, we overview the scenarios that will be used in

testing the framework. In Section 4, we discuss the simulation classes within the framework. In Section 5, we discuss the means of comparing the models when an ADC system is being used versus when it is not. Finally, in Section 6, we will discuss the key conclusions of this research and discuss future areas of research that have been identified through the work done.

2 BACKGROUND

Automatic data collection systems are often (but not necessarily) implemented on top of wireless sensor networks. Wireless sensor networks are becoming more widespread in industrial and military situations. The tasks that these networks perform can include surveillance and monitoring. The most well known example of these sensor networks is Radio Frequency Identification (RFID) applications. Because wireless sensor networks are becoming more ubiquitous there has been recent work examining their computer science performance aspects via simulation.

Park et al (2001) describes the simulation of wireless sensor networks. In their models, three types of nodes are needed: sensor nodes that monitor the environment, target nodes that serve as stimuli for the sensors, and user nodes that serve as clients and administrators. Park et al (2001) states that the framework of such a simulation is subject to three main concerns. These concerns are the node placement and traffic generation, sensor stack, and power issues. Node placement can vary depending on the application and desired results. Traffic generation can be broken down into three types. First, user-to-sensor traffic involves commands given to the sensors. Second, sensor-to-user entails reports sent back to the users. Third, sensor-to-sensor traffic involves the processing of sensor events among sensors before they are sent to the user. The sensor stack is a signal sink that triggers the application whenever a sensor event occurs. Node placement and traffic generation determine what kind of application is being used and the sensor stack simulates how a sensor node processes sensor signals.

Our framework will need to provide some constructs to simulate wireless sensor networks, but our initial focus will not be on the computer performance issues of the sensor network, but rather on what the network enables within the operational setting. Angeles (2005) gives examples of different areas in which RFID implementation can help the efficiency of a facility. These benefits can include using RFID tags to store information such as pallet weight as a way to speed up the receiving process at distribution centers. RFID has also been used to indicate which items belong in certain areas of a warehouse, decreasing the amount of misplaced inventory. Another use of RFID in a warehouse setting is to set up a network of sensors throughout the building in order to keep track of the location of every item.

When simulating the collection of data on an object, the most reasonable method would be to use discrete event data. These events can be triggered by several methods. These methods can include time until event, probability of event happening at a given point, and scheduling events to occur. Edwards (1991) gives an example of an automatic data collection system that simulates mission data for an aircraft. The data gathered in this simulation included engagement opportunity data, location, and weapons data. This system used a combination of discrete event and continuous data and implemented a system that included visual displays of events, visual displays of variables being studied, and a tape to allow for studying the data. The simulation and recording of operational events needs to be a key element in any ADC simulation.

Once sensors are placed within a facility, it may become desirable to place them at all levels of the supply chain. This can be beneficial when more than one facility is communicating with each other. When implementing RFID in a supply chain, Lee et al (2004) describes three factors that will be impacted. The first is inventory accuracy, which through shrinkage, incorrect product identification and transaction error costs companies for incorrect replenishment decisions. The next factor is shelf replenishment policy, which through RFID could reduce the number of lost sales by managing shelf replenishment. The third factor is inventory visibility. This deals with the ability of everyone in the supply chain to see inventory at any level at any time. Ultimately, it is our goal to be able to measure the impact of ADC systems within the facility and across a larger scale system, such as the supply chain or a sense and respond military network.

The simulation of sensor networks has been limited in past efforts to just considering the theory of the sensors and the communication network and not the operational setting. Also, in such papers as Lee et al. (2004) the operational setting is simulated, but not the sensor network. This research is focused on creating simulations of ADC systems in situations with the actual process or operational setting. This will give a more practical side to the simulation of ADC systems and will provide design tools for understanding their operational impacts.

An ADC system which has had some simulation research on the effect of the data collection system on the operational setting has been health monitoring systems (HMS). Larder (2003) discusses the benefits of using these systems in helicopters. In addition to the safety benefits that come from these systems recognizing when to perform preventative maintenance, there are many other benefits. These benefits include the reduction of component damage, enhanced maintenance planning, and the accurate recording of when aircraft limits are exceeded. Health monitoring systems can have several functions. Tanner and Crawford (2003) describe some of these functions as they relate to aircraft. These functions include engine vibration

monitoring and reporting, engine event monitoring, and engine mechanical fault detection. These systems gather data to determine if certain features appear and diagnose the problem from those features. These features can appear through outputs from specific detectors in the system. Once a group of detectors sends output, the system is compared to the detector readings of a properly functioning system and also compared to the detector readings for every known problem to determine the state of the system. Once a problem has been determined, it is sent to the end user in order to make the appropriate preparations.

Other work has been done concerning the effects of using HMS and other related technologies. Faas (2003) describes the simulation of an autonomous logistics system (ALS). The purpose of using ALS is to allow for aircraft to return to working order sooner after a fault is discovered. This paper describes the advantages that can be gained by using HMS and other systems to alert the system that a fault has occurred and that a service and/or a replacement part is needed, as illustrated in Figure 1. These advantages can be found in the areas of preventative maintenance, parking and recovery, and unscheduled maintenance.

The maintenance flight line context of Faas (2003) serves as the main application area for our current research efforts.

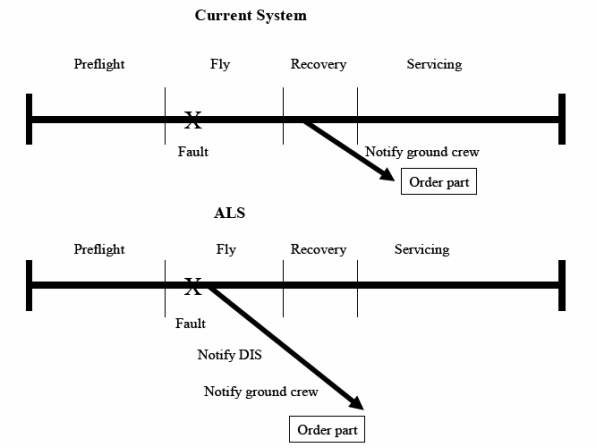

Figure 1: Timeline for Service and Part Replenishment

3 OVERVIEW OF SCENARIO

In this section, we provide a description of a basic scenario derived from the flight-line operational setting discussed in Faas (2003). This scenario follows the repair process of an aircraft on an airbase. The repair process contains a sequence of steps that requires several types of workers and equipment. This main scenario contains four different types of workers and five types of equipment that will work to complete a seven step repair process. Figure 2 depicts the repair process and the personnel and equipment that is necessary for each step. The diagram also depicts when resources are requested compared to when they are actually seized and released. The solid, curved lines depict when the resource is seized or released and the dashed, straight lines represent when the resources are first requested.

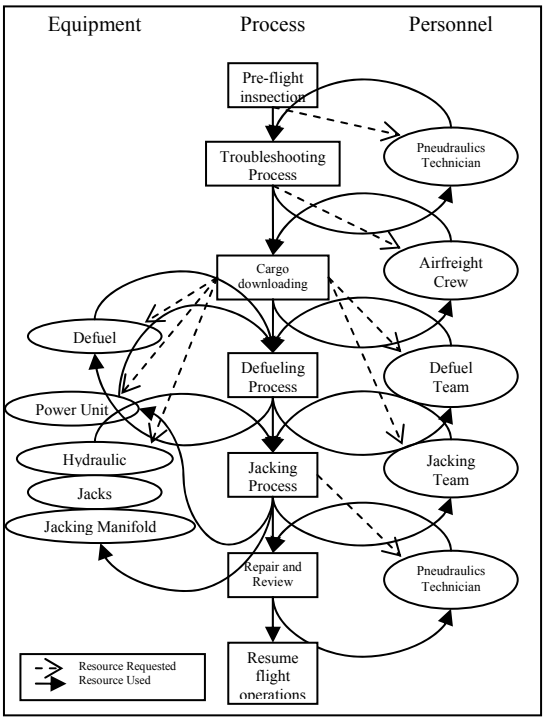

Figure 2: Example Repair Scenario

Since this scenario is such a complex one, it would be impractical to attempt to build the model without testing the classes that are built in order to complete it. For this purpose, a simpler scenario was designed to test the objects in the model for correctness and to provide a framework for creating the more complex test scenario. The basic scenario consists of a system of three aircraft, three repair stations, five workers, and four pieces of equipment. This basic scenario does not specify different kinds of workers or equipment, as any idle worker can be selected to perform the repair task when an aircraft enters the repair station. When an aircraft is sent to a repair station, a worker is requested. The worker travels to the repair station and the attempts to find a piece of equipment somewhere on the air base. Once the equipment has been found and seized, the worker and equipment travel back to the repair station to complete the repair process. When the repair process is completed, the worker takes the equipment back to its storage location and then travels back to the worker rest site. Workers travel between locations according to a Euclidean movement rationale. The workers walk in a series of steps in a direct Euclidian line from location to location. We simulate each step of the worker in order to make the tracking of the workers and equipment on the air base rele-

vant to sensor tracking. The U.S. Manual on Uniform Traffic Control Devices (MUTCD) refers to 4.0 feet (1.2 meters)/second as the "assumed" normal walking speed of a human being while the Canadian Uniform Traffic Control Devices for Canada (UTCD) says that normal walking speeds are from 1.1 to 1.4 meters/second. Based on an analysis of human walking velocity, we model the velocity of movement as a triangular random variable with (min = 53.64, mode = 80.47, max = 53.64) all in meters/minute.

This scenario will be examines under for two different situation. The first situation uses the system as is, using search patterns for locating equipment and no real method for tracking workers and equipment. The second situation utilizes an ADC system. This system includes RFID tags and readers. After these situations have been modeled the performance of each can be measured. One of these performance measures includes the time that a worker has to spend searching for an idle piece of equipment. Another measure is the utilization of the worker.

In order to model this scenario, several classes had to be designed into a simulation library. These classes include the airbase, worker, aircraft, a resource class which represents the repair stations and equipment, and the relevant ADC classes such as tags, readers, and the network. A more detailed look at these classes is given in the next section.

4 REQUIRED CLASSES

In order to model this scenario, several classes were constructed. These classes can be separated into three groups: spatial classes, model classes, and ADC classes.. The spatial classes facilitate moving within a spatial frame. The model classes include any classes that represent a model element in the scenario, such as the aircraft, worker, and resource classes. The last group of classes that were created are the ADC classes. These classes represent the ADC system, such as the network, RFID tag, and RFID reader.

4.1 Spatial Classes

The spatial classes were built into the Java Simulation Library (JSL). The JSL is an object-oriented framework for discrete-event simulation. The JSL consists of a number of Java packages that facilitate simulation modeling within the Java language. The primary packages of the JSL include a calendar package for managing the event list, a modeling package that contains the classes for modeling standard discrete event situations, the utilities package which contains classes for generating random numbers and random variables and for collecting statistics, a spatial modeling package for modeling spatial relationships, and an observer package for collecting and reporting simulation results via the Java observer/observable pattern. Information concerning the use of the JSL for research or educational purposes can be obtained by contacting <rossetti@uark.edu>.

The purpose of the JSL is to facilitate object-oriented simulation research and education. Within the JSL, the modeler can sub-class from a base class called ModelElement. A ModelElement represents an abstraction for elements or things that can be within a simulation model. ModelElements can react to specific default actions such as the beginning of an experiment, the beginning of a replication, the initialization of the model, the warm up of the model, the ending of a replication, and the ending of an experiment. The most important sub-class of ModelElement is Model, which represents the parent element in an object hierarchy of all model elements. Modelers create instances of their ModelElements and add them to an instance of Model. The model controls the default actions of all the model elements in the object hierarchy. The second most important sub-class of ModelElement is the SchedulingElement class, which allows easy access to the event calendar for scheduling/canceling events, etc.

Within the modeling package are additional packages for modeling common simulation situations such as queues, resources, processes, etc. Because the modeling of spatial relationships is a very common simulation concern, the JSL also contains a package for spatial modeling (jsl.spatial). For the purposes of this paper and because of space limitations, we simply overview the basic concepts involved in spatial modeling within the JSL in order to give the reader a conceptual understanding of what is available. The jsl.spatial.spatial2D package allows for 2-dimensional modeling through the SpatialModel2D abstract base class. This class represents a 2-D world and allows distances to be defined between (x,y) positions within the world. Instances of the class SpatialElement2D can be added to a SpatialModel2D to represent locations/elements within the spatial model. Figure 3 indicates that a SpatialModel2D may contain many instances of the class SpatialElement2D. In addition, a ModelElement may or may not be related to a SpatialElement2D. The relationship between model elements and spatial elements, allows the simulation model elements to reference an underlying spatial representation, and to delegate moving within the spatial model to the corresponding spatial element.

The key elements of the Spatial Model Package include the SpatialModel2D, SpatialElement2D, AbstractMovementController2D, and RectangularGridSpatialModel2D classes. Whenever an instance of a SpatialElement2D has its position changed, its corresponding spatial model is notified. The methods used in the class are used to set the current position, final destination, set velocity, set coordinates, update positions and handle collisions. This includes setting the initial position of the spatial element (setInitialPosition()), its current position (getX() and getY()), etc. The AbstracMover2D class is a ModelElement and serves as a base class for tying model ele-

ments that can move (e.g. trucks, people, etc) to a spatial representation. For each move a spatial element makes within the spatial model, the spatial model will be registered as an event. This event generation is facilitated by subclassing AbstractMover2D from SchedulingElement. The method moveTo() is called to move an element from its current position to another position. The private class EndMovementListener which inherits from ActionListenerIfc facilitates the scheduling of the move events. Another feature which is included in this class is the collision handling of the spatial elements. The user needs to build class which implements the CollisionDetector2DIfc in order to include logic for the collisions happening between spatial elements.

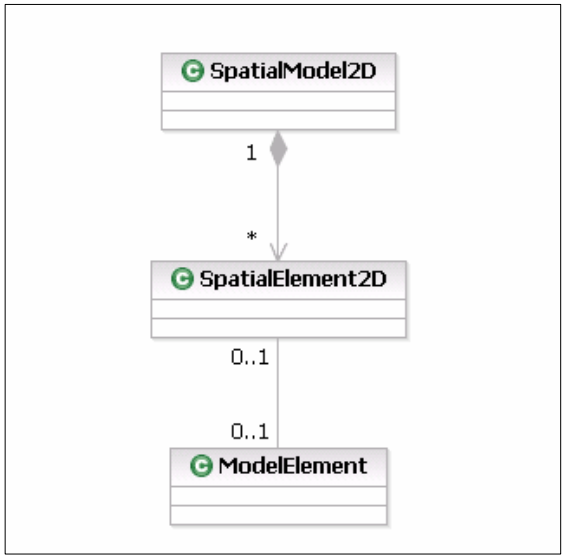

Figure 3: Relating Model Elements to Spatial Frame

The moveTo( ) method facilitates traveling from the current position to the desired destination which starts a trip. A trip is a series of movement to move from the current position to the specified coordinates. At the beginning of a trip, observers that monitor the start of trips are notified. If the trip is broken down into movements, then each movement can have its own velocity, distance, and direction (as long as the final movement ends at the destination). An instance of AbstractMovementController can be supplied to control each movement of the object. In this manner, walking can be easily modeled as well as more complicated path movements. Figure 4 illustrates how trips can be broken down into movements.

Based on the classes within the spatial package a number of additional classes were created that represent modeling elements within the scenario. These included the worker (modeled on top of an AbstractMover2D) and equipment (modeled as a Resource2D). These and the other classes used to model the scenario are described in the next section.

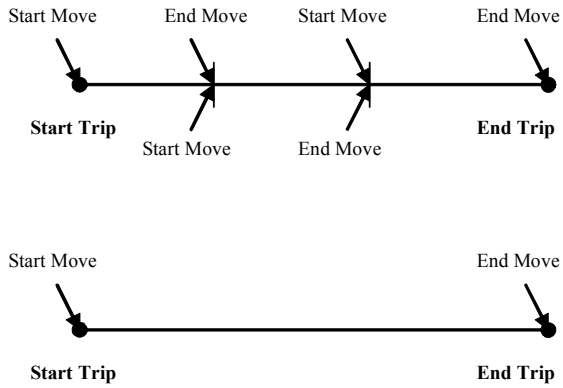

Figure 4: Modeling Trips as a Sequence of Movements

4.2 Model Classes

The model classes represent the operational model elements in this scenario. The aircraft was modeled, along with the worker, the airfield, the repair stations, and the pieces of equipment. These classes will be used in both the ADC and the non-ADC model situations.

The airfield is modeled as a rectangular grid which allows for other model elements to be spatially positioned within it. The aircraft is modeled with different attributes, such as sortie time and failure indicator. The Airbase class is considered to be a container class, placed on an airfield, which controls and keeps track of all other objects in the system. One of the responsibilities of the airbase is to hold the information concerning the equipment's home base locations. These locations are defined when a piece of equipment is added to the airbase. In addition, when modeling the ADC situation, the airbase knows where each piece of equipment is located via the use of the ADC network. In the ADC modeling situation, the worker can request the location of a piece of equipment rather than searching at each piece of equipment's home base. In order to do this, the airbase determines if there is an ADC system present. If so, the airbase asks the ADC network to provide the worker with the list of locations, and if not, the worker is given the list of equipment home base locations.

The resources (equipment and workers) are modeled as typical simulation resources with one key difference. Some resources have the ability to move on their own, while others do not. Workers, for instance, can move without assistance from any other object. Workers are created with attributes for velocity and step size used in traveling. A worker's move is also assigned a move type, which determines the next action for the worker.

Move types include :

- Moving to start repair.
- Searching for equipment.
- Moving to repair with equipment.
- Returning equipment.
- Returning to rest site.

As mentioned before, the worker moves according to a Euclidean movement rationale. The worker also controls seizing and releasing of equipment when necessary. Figure 5 depicts the Worker class.

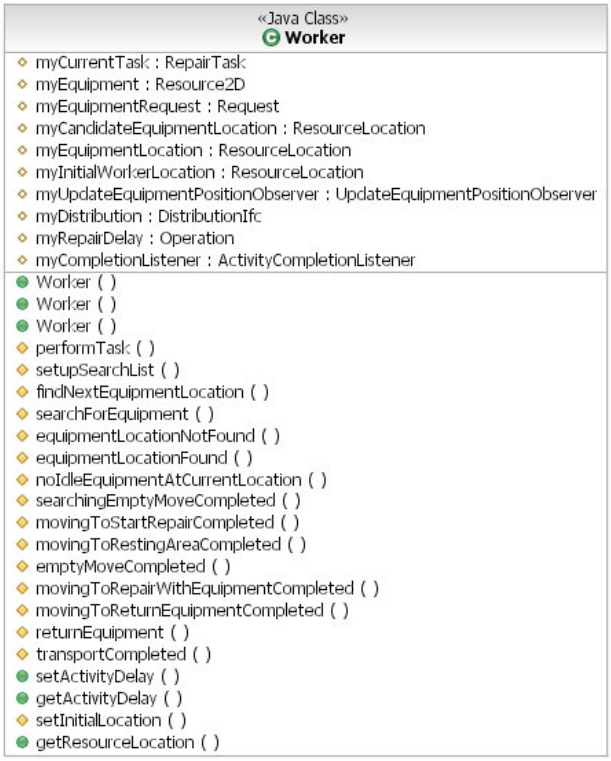

Figure 5: Worker Class Diagram

Repair Stations and Equipment fall into the other category, since they either cannot move or have to be moved by something else. For this reason, they have both been modeled with a Resource2D class. The Resource2D class has the attributes that a normal resource would be expected to have, but also has the ability to be placed on a spatial model and moved when required. It represents a resource that actually takes up space and has a location. They are attached to a resource location, which is used in searching and traveling procedures in the model.

4.3 ADC Classes

The classes that were constructed to represent the ADC system are the network, tags, and readers. The network functions as an overseer of the entire ADC system. The network holds attributes and methods that keep track of all tags and readers in the system. The network class also controls the addition of tags and readers to the model. This is because readers and tags that are created without a network serve no purpose and add no value to this model. The network also assigns the tags to objects and alerts those objects that a tag has been attached to them. The network also keeps running lists of tags, readers, and tags that are attached to pieces of equipment. The equipment tag list is used primarily when searching for idle pieces of equipment and giving workers specific locations to travel to. The method used to add a new RFID Tag is presented in Exhibit 1.

```
public    RFIDTag    addRFIDTag(SpatialElement2DIfc
spo){
        RFIDTag s = new RFIDTag(spo);
        myTagList.add(s);
        s.setNetwork(this);
        spo.addSpatialElementObserver(s);
        if (spo instanceof Resource2D){
            System.out.println("Tag Added!");
            myEquipmentTagList.add(s);
        }
        return(s);

    }
```

Exhibit 1: Adding an RFID Tag

As discussed earlier, when the worker requests the locations that equipment is stored at, the airbase checks to see if an ADC system is in place. If so, the airbase requests the list from the network. The network then calls a method for compiling a list of locations of the idle equipment in the system. Exhibit 2 shows the method used for obtaining that list.

The RFIDTag class represents passive RFID tags. The tags have been set up to where they can only be created by the network that they belong to. The key issue with this class was determining how the tags are tracked throughout the network. More specifically, should the network alert the tag that its position changes or should the tag alert the network when its position changes. It was determined that the most effective strategy would be to set the RFIDTag class as an observer to an object in the system and have it alert the network when its object changes position. The class diagram of the RFIDTag class can be seen in Figure 6.

The reader class can be attached to a location in the airfield. This can be achieved by attaching the reader to either an object in the model or to a resource location. The reasoning for attaching a reader to an object is to facilitate the possibility of attaching a reader to an object that may not be stationary, such as a vehicle or worker, if it was deemed desirable to do so. The reader, when asked by the network, can check to see if a specific tag is within its range and then return the location to the network. This is

performed by determining the distance between the tag and the reader in question. This method returns a true or false and is used in all search methods concerning the reader. Exhibit 3 shows the method that is used to determine whether a tag is within the reader's range.

```
public List<ResourceLocation> getEquipmentLocationList(){
        List<ResourceLocation> rList = new ArrayList<ResourceLocation>();
        for(RFIDReader rr: myReaderList){
            rr.checkForTags();
            List<RFIDTag> InRange = rr.getTagsInRange();
            for(RFIDTag rt: InRange){
                if(rt.getSpatialElement() instanceof Resource2D){
                    Resource2D equip = (Resource2D)rt.getSpatialElement();
                    ResourceLocation location = equip.getResourceLocation();
                    if(!rList.contains(location)){

    if(location.isResource2DAtLocation(equip))
                            rList.add(location);
                    }
                }
            }
        }
        return rList;
    }
```

Exhibit 2: Retrieving the Equipment Location List

The reader can also be asked to create a list of tags that are within its range and return that to the network. For the purpose of testing for this scenario, an additional method is called when the network is updated and displays the tags that are in the range of each reader in the system. This method can also be used in searching for tags in the system.

```
public boolean isTagInRange(RFIDTag tag){
        double xDistance = Math.abs(getX() - tag.getX());
        double yDistance = Math.abs(getY() - tag.getY());
        if(Math.sqrt((xDistance * xDistance) + (yDistance * yDistance))<= range)
            return(true);
        else
            return(false);
    }
```

Exhibit 3: Determining When a Tag is in a Reader's Range

5 COMPARISON OF ADC VS NON-ADC

The key difference between the ADC scenario and the non-ADC scenario is the worker's method for searching for equipment. In the non-ADC scenario, the worker searches through the potential equipment locations until an idle piece of equipment is found. In order to do this, the worker requests a list of potential equipment locations from the airbase. The worker then works through that list until an idle piece of equipment is found. An important note is that the list and the order of locations never change throughout the experiment. In the ADC system, when the worker requests the list of potential equipment locations, the readers are activated. The network asks readers to go through the list of tags that are attached to equipment and determine whether any are within that particular reader's read range. Once an equipment tag has been discovered by a reader, a check is performed to determine whether the tag is located in its equipment's home base location. If so, the location of the reader is returned to the worker. This process repeats for each reader in the network.

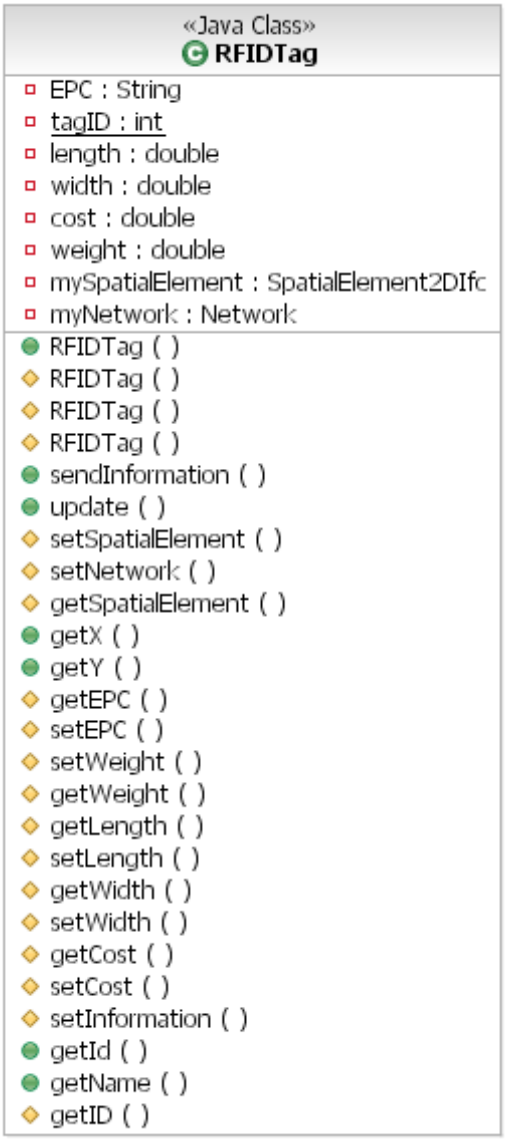

Figure 6: RFIDTag Class Diagram

By running the basic scenario, the search time was indeed decreased by using the ADC system. For this scenario, the average search time was defined as the time that

the worker left in search of the equipment until the time that the worker begins to travel back to the repair station with the piece of equipment. In order to test if there is statistically significant difference between the two scenarios, a paired t-test was performed. The two situations were run for 30 replications at a replication length of 480. The replication length was set to be equal to the number of minutes in an eight hour day, and the 30 replications is intended to allow for a month of observations. The average search time was gathered for each replication and compared to the search time for the other situation in the same replication. The difference between the search time of the non-ADC system is taken from the ADC system search time and the test is performed.

The statistical results are presented in Table 1. By using a confidence level of 95.0%, it is apparent that there is significant statistical difference between using the ADC system and using the non-ADC system.

Table 1: Statistical Results from Paired t-test

Mean	-0.079746128
Standard Error	0.002916966
Median	-0.080122471
Standard Deviation	0.015976881
Sample Variance	0.000255261
Kurtosis	0.303217768
Skewness	-0.332227584
Range	0.071251389
Minimum	-0.119131254
Maximum	-0.047879864
Sum	-2.392383848
Count	30
Confidence Level(95.0%)	0.005965865
LL	-0.085711993
UL	-0.073780263

This amounts to a savings of about 38.3 minutes per worker in terms of time spent searching for equipment.

6 CONCLUSION

This framework was built in order to create a more flexible method for simulating ADC systems. The aim of this paper is to demonstrate the application of this framework to a real scenario. In this scenario, the modeling elements were created to represent an airbase, along with the different objects that would be present on it, and tested them on a key performance measure. ADC constructs were then created and added to the model to test the potential improvements that these additions can bring to the system. After an experiment was conducted, the two models were compared and a statistically significant difference was found. The ADC system decreases the average equipment search time when compared to the searching methods of the model without the ADC system.

This paper details the main characteristics and considerations that were given to certain classes which were created for the model. Higher level classes, which some classes discussed in this paper inherit attributes and methods from, were not discussed unless specific detail was necessary to describe the actions of the classes discussed.

There are numerous possible research opportunities that can be pursued using this framework and its scenarios. For example, the ADC system can be expanded to include other technologies. These technologies can include using health monitoring sensors, active RFID tags, and RFID readers with more "real world" range functions. The model can be expanded further to make the system more applicable in an airbase context. This can be done by differentiating different types of workers and equipment, simulating a sequence of repair operations necessary to complete a task, and creating and testing new performance measures. Some of these performance measures can include worker utilization, total throughput time, and total aircraft downtime.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation and the Air Force Research Laboratories under grant EEC-0436687. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the Air Force.

REFERENCES

Angeles, R. 2005, "RFID technologies: supply-chain applications and implementation issues", *Information Systems Management*, v. 22 (1), p. 51-65.

Cohen, J. 1994 *Automatic Identification and Data Collection Systems*, McGraw-Hill Book Company

Edwards, J. 1991, "Full mission simulation data system", *Annual Forum Proceedings, American Helicopter Society*, v. 1, p. 565-577.

Faas, P. D. 2003, "Simulation of autonomous logistics system (ALS) sortie generation", *Master's Thesis*, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio

Larder, B. D. 2003, "Assessing the benefit of helicopter health and usage monitoring systems", *IEE Seminar on Aircraft Airborne Condition Monitoring*, p. 6/1-6/6.

Lee, Y. M., Cheng, F., Leung, Y. T. 2004, "Exploring the impact of RFID on supply chain dynamics", *Proceeding of the 2004 Winter Simulation Conference*, p. 1145-1152.

Park, S., Savvides, A., and Srivastava, M. B. 2001,"Simulating networks of wireless sensor net-

works", *Proceedings of the 2001 Winter Simulation Conference*, p. 1330-1338.

Tanner G. F. and Crawford, J. A. 2003, "An integrated engine health monitoring system for gas turbine aeroengines", *IEE Colloquium*, v. 3, p. 31-42.

AUTHOR BIOGRAPHIES

MANUEL D. ROSSETTI, Ph. D., P. E. is an Associate Professor in the Industrial Engineering Department at the University of Arkansas. He received his Ph.D. in Industrial and Systems Engineering from The Ohio State University. Dr. Rossetti has published over thirty-five journal and conference articles in the areas of transportation, manufacturing, health care and simulation and he has obtained over $1.5 million dollars in extra-mural research funding. His research interests include the design, analysis, and optimization of manufacturing, health care, and transportation systems using stochastic modeling, computer simulation, and artificial intelligence techniques. He was selected as a Lilly Teaching Fellow in 1997/98 and has been nominated three times for outstanding teaching awards. He serves as an Associate Editor for the International Journal of Modeling and Simulation and is active in IIE and INFORMS. He served as proceedings co-editor for the WSC 2004 conference. He can be contacted at are <rossetti@uark.edu> and <www.uark.edu/~rossetti>.

BRADLEY J. HOBBS is a graduate research assistant in the Industrial Engineering Department at the University of Arkansas. He earned a B.S. degree in Industrial Engineering at the University of Arkansas. He has worked on research projects in the area of computer simulation. His email address is <bjhobbs@uark.edu>.

PAUL D. FAAS is a Senior Logistics Research Engineer within the Logistics Readiness Branch at the Air Force Research Laboratories. He holds a M.S. degree in Operations Research from the Air Force Institute of Technology and a B.S. degree in Aeronautical & Astronautical Engineering from Purdue University. He has been the program manager for a number of research efforts including the Smart Systems for Logistics Command and Control (SSLC2) which is related to this paper. Mr. Faas is a member of the International Society of Logistics Engineers. Mr. Faas can be reached at <Paul.Faas@wpafb.af.mil>.