SAN095-1122C

Integrated Facilities Modeling Using QUEST and IGRIP

Kenneth R. Davis

Sandia National Laboratories
Intelligent Systems and Robotics Center
Albuquerque, NM 87185-1007[1]

Eric R. Haan

Production Modeling Corporation
3 Parklane Blvd., Suite 910 West
Dearborn, MI 48126

Biographies

Kenneth R. Davis earned a bachelor of science degree in electrical/computer engineering from Oklahoma State University (Stillwater) in 1987 and a master of science degree in electrical engineering from the University of Arizona (Tucson) in 1989. He has been at Sandia National Laboratories for the past six years and has been using Deneb's IGRIP® and QUEST® products for the last two years. Prior to Sandia, Mr. Davis worked for the Dow Chemical Co., General Dynamics, and Burr-Brown. He has been involved in a variety of projects ranging from instrumentation in chemical plants to simulating breakdown effects in semiconductor devices. His research interests include the application of numerical analysis in control and robotics, and the modeling and control of robot mechanisms.

•

Eric R. Haan holds a bachelor of science degree in industrial and operations engineering from the University of Michigan (Ann Arbor). He worked at General Motors - Lansing Automotive Division as a co-op engineering student from 1986 to 1992 and was hired as a Quality Engineer in May 1992. Since November 1992, he has been an Applications Engineer with Production Modeling Corporation in Dearborn MI working primarily with discrete-event simulation chiefly involving automotive-industry manufacturing.

Abstract

A QUEST model and associated detailed IGRIP models were developed and used to simulate several workcells in a proposed Plutonium Storage Facility (PSF). The models are being used by team members assigned to the program to improve communication and to assist in evaluating concepts and in performing trade-off studies which will result in recommendations and a final design.

The model was designed so that it could be changed easily. The added flexibility techniques used to make changes easily are described in this paper in addition to techniques for integrating the QUEST and IGRIP products. Many of these techniques are generic in nature and can be applied to any modeling endeavor.

[1] This work was supported by the United States Department of Energy (DOE) under Contract DE-AC04-94AL85000.

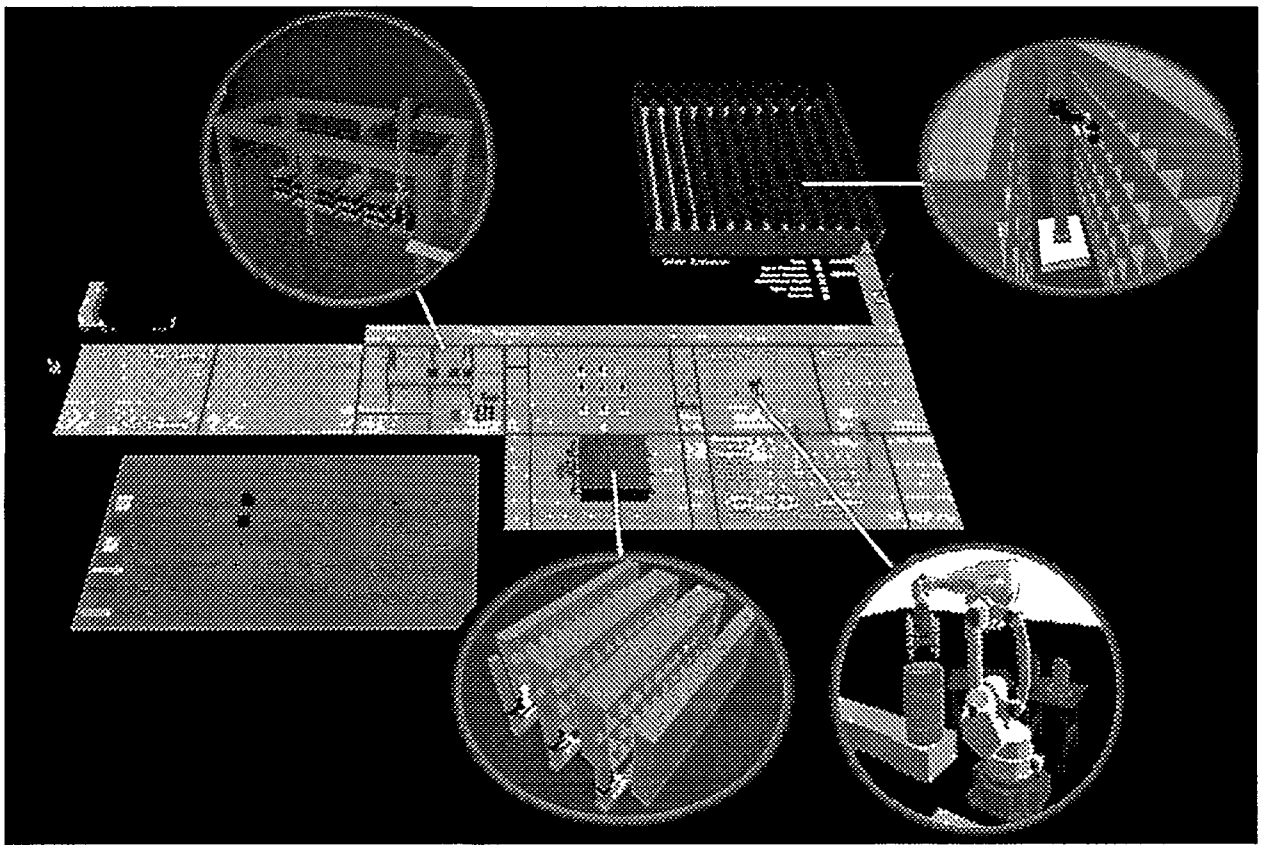

Figure 1: QUEST model of the Plutonium Storage Facility including "blow ups" of the detailed IGRIP workcell automation concepts.

1.0 Introduction

A multi-discipline team comprising the Fissile Material Disposition Program (FMDP) has been formed and has been tasked by the United States Department of Energy (DOE) to define and evaluate beginning-to-end options for the disposal of surplus fissile materials. Members of Sandia National Laboratories' Intelligent Systems and Robotics Center (ISRC) represent this team in the Automation and Robotics (A&R) area. Other disciplines represented in this program include Safeguards & Security, Transportation, Systems Analysis, Technical Integration and the specific disposition technologies. Members of these groups are from other DOE laboratories, industry, and academia. The end product of this program will be a recommendation to and supporting documentation for the disposition of the U.S.'s surplus fissile materials.

A&R activities at this stage of the project consist mainly of concept evaluations. Therefore, a capability was needed to visualize an entire facility including the discrete-event occurrences as well as detailed robotic workcells depicting the automation concepts being evaluated. In addition, an easy method of replacing one concept for another to experiment with alternatives (to do "what if" analyses) was desired. Deneb's QUEST product was chosen to perform the discrete-event simulations coupled with Deneb's IGRIP product to perform the detailed robotic workcell simulations. The inter-cell transport of material between workcells was modeled using QUEST as well.

It was the intent to build the model with a great deal of flexibility. Arrival rates and characteristics, numbers of parallel operations, times to complete operations, machine failure

rates and repair times, and buffer capacities can be changed easily in the model. It is anticipated that several iterations of analyses will be performed before deciding on a final concept recommendation. This model will assist greatly in the analyses being performed.

Sandia ISRC contracted with Deneb and Production Modeling Corporation to produce the base model in the flexible fashion as described above. Accordingly, the two authors of this paper will present this project from two different aspects; the developer of the base model and the user of the model. In this paper/presentation, we show and describe the QUEST model and the associated detailed IGRIP models. The hierarchy of the QUEST model allows communication between QUEST and IGRIP through the use of TCP/IP sockets (dynamically) and the link and jump functions (non-dynamically) found in QUEST. Also, a third method of recording an IGRIP model and playing it in QUEST will be discussed. In addition, the use of the Deneb products have greatly influenced the analysis and design processes of this program and have improved communications between a wide variety of team members.

2.0 Facility Description

One of the concepts being evaluated in this program is the long-term storage of the surplus fissile material. Figure 1 shows the QUEST model of this storage facility. Normal inbound flow is from left to right in this figure and, after processing, outbound flow is in the opposite direction. Material is transported through the model by conveyors or automated guided vehicles (AGVs). A block flow diagram of the material flow is shown below in Fig. 2.

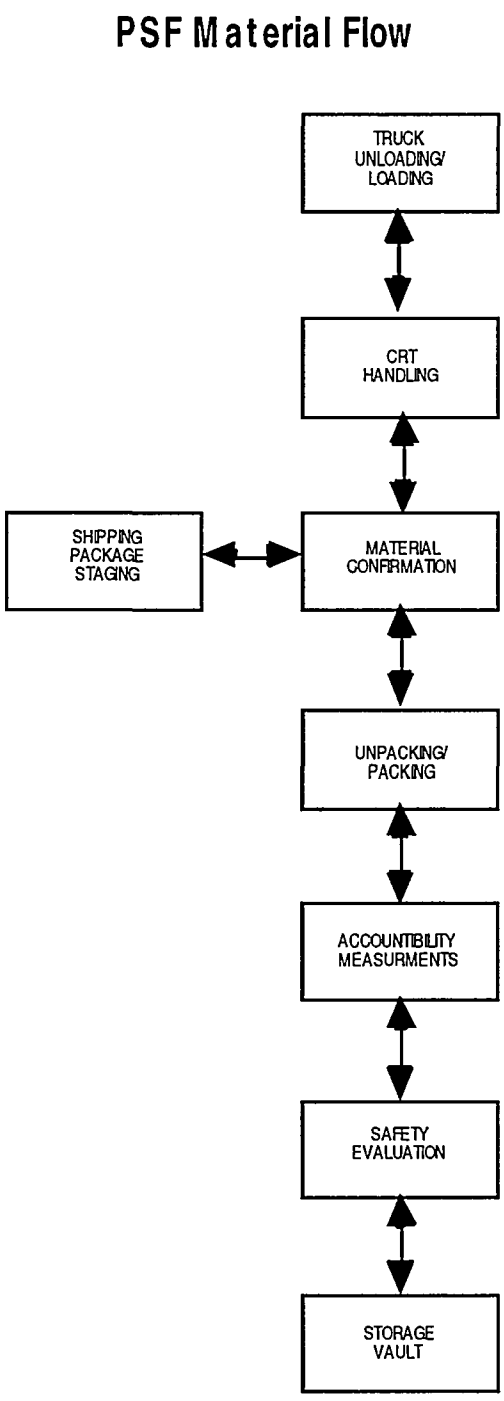

Figure 2: Block flow diagram showing material flow.

2.1 Material Flow in the Facility

Trucks arrive at the facility in convoys. These convoys are made up of a user-defined random number of trucks and arrive at user-defined random intervals. Each truck contains pallets known as Cargo Restraint Transports (CRTs) containing a number of shipping packages. Each shipping package contains a storage container which is described later in this section. After being unloaded from the trucks, the CRTs are taken to the CRT

Handling area where each individual shipping package is removed using a gantry robot. This overhead gantry was modeled in detail using IGRIP.

If available stations exist in the material confirmation (MC) area, then a shipping package is taken to this area where a total of four operations are performed including bar-coding and weighing. If all the stations in the MC area are occupied, the shipping package is taken to the shipping package staging (SPS) surge area. The concept shown for the SPS area is a three-lane automated stacker/retriever system (AS/RS) and has been modeled in detail using IGRIP. As stations in the MC area become available, the containers in the SPS area are moved to the MC area.

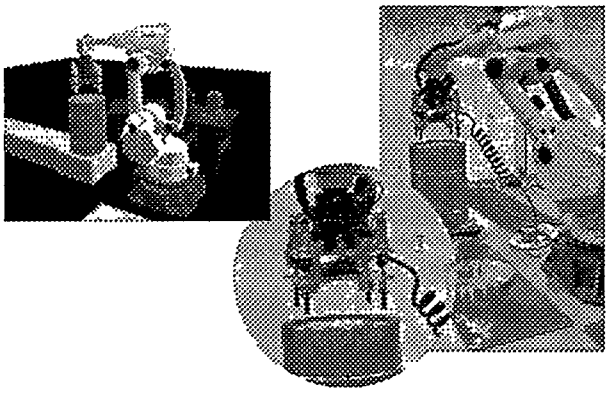

Figure 3: IGRIP model and real image of the S-800 robot doing the unpacking operation.

After completing the operations in the MC area, the shipping package is taken to the unpacking/packing area where the inner storage container is removed from the shipping package. A detailed workcell of this operation has been modeled using IGRIP. The concept for this area uses a Fanuc S-800 robot with three different Sandia-designed tools. The first tool, a quad air socket wrench, removes the bolts securing the lid of the container. The second tool, a vacuum gripper, removes the lid and upper packing insert. The third tool removes the inner storage container and places it on an outbound conveyor. The robot then reassembles the empty shipping package. A picture of the real and modeled S-800 workcell is shown in Fig. 3.

The storage containers arrive at the accountability measurements (AM) area where five different operations are performed. Two of these operations (bar-coding and weighing) take minimal time and are performed first. The order for the next three operations (gamma count, neutron count, and calorimetry) is unimportant and one of these operations can take up to twelve hours to perform. It was important to be able to modify the numbers of parallel machines in this area because of the long times it took to do the operations. A small surge area exists in the AM area as well for staging.

After completing the operations in the AM area, the storage containers are taken to the safety evaluation (SE) area prior to entering the vault. Seven different operations exist in the SE area and the origin of the container determines which operations to perform. The SE area receives inbound containers heading to the vault, outbound containers from the vault, and containers from the vault which undergo a random inventory check.

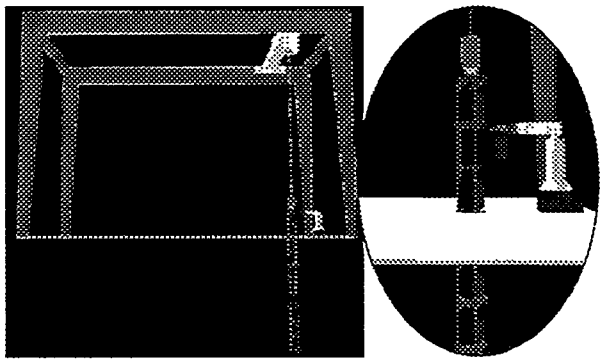

Figure 4: IGRIP model of the well storage vault automation concept.

The final area is the vault which provides an area for storage. The concept in the model shown in Fig. 1 is a high-stack AGV which places the containers into shelf locations. This

workcell has been modeled in detail using IGRIP. One of the alternative concepts for the vault is a well storage option and the IGRIP model for this concept is shown in Fig. 4.

Outbound containers move through the facility in reverse order of the inbound containers. The one exception being that when containers leave the vault, they are taken to a glove box area for inspection prior to arriving at the AM area.

2.2 Facility Requirements and Assumptions

The previous section gave a brief overview of each area in the facility. Since this project is still in the analysis phase, these different areas of the facility could change. It did seem natural to break up the facility into these areas based on intuition, and the modeling being performed confirms these intuitions.

There exists overall throughput requirements of material making it through to the MC area in a specified time period and all the way to the vault in another specified time period. Therefore, the entire facility as a whole is being evaluated.

It is assumed that the SPS area will be used during every convoy arrival. Therefore, increasing the number of truck unloading bays or gantry cranes in the CRT handling area will only decrease the amount of time that it takes for the shipping packages to reach the SPS area. The first bottleneck occurs at the MC area. Increasing the number of stations in the MC area does relieve this bottleneck only to have another bottleneck occur at the unpacking area. Because of cost considerations, it is very desirable to have only one S-800 robot doing the unpacking and packing in this facility. The unpacking operation takes approximately two minutes to perform so having a second S-800 as a backup if one of the robots were to break down would be the reason for two and not to increase throughput.

There is a fair amount of routing logic needed in the AM area. A set order of the operations is not followed in this area. Different routing schemes were modeled including filling the stations that took the most amount of time first. Quantities of identical machines were varied to analyze the impacts of these variations as well.

The operations that occur in each of the areas are not always successful. A certain percentage will fail and these percentages are set through the data file that is read in at the beginning of each simulation run as described in the data input section (3.1). Process logic is written for the various workcells to handle these failed operations according to the requirements for the facility. QUEST's user interface allows for the failure of certain machines with random distributions and user-specified repair times.

3.0 Modeling Techniques

This section covers various aspects of the QUEST modeling that was developed for this project. Methods for integrating QUEST and IGRIP are discussed in Section 4.0. The following six modeling issues are discussed in the following subsections:

* data input
* setting and changing the quantities of parallel machines
* periods of inactivity
* preventing two-way flow
* output reporting
* model debugging

3.1 Data Input

A data file is read by the model at the start of each run. The model was designed to be very generic and flexible in nature and consequently model users are provided with considerable control over model parameters via the data file. Container arrival and departure characteristics, process cycle times, transportation times, buffer capacities, machine quantities, and various other parameters are all controlled through the data

file. A portion of this data file (which we called "system_data.dat") is shown below:

```
          :
          :
Process Cycle Times (all times given
in seconds)

900                        -- truck
      unloading time per CRT
900                        -- truck
      loading time per CRT
Normal(1/12*3600,0,21)     -- CRT
      unpacking
Normal(0.75*3600,0,22)     -- CRT
      Assembly
Normal(0.50*3600,0,23)     -- Material
      Confirmation
Normal(0.25*3600,0,24)     -- Shipping
      Package Unpacking
Normal(0.25*3600,0,25)     -- Shipping
      Package Packing
          :
          :
```

(Note: In the actual data file, each item is contained on one line. It was necessary for each item to take two lines in this paper because of less space in the two column format.)

The input data file can be generated with any editor. In fact, a script was written to automatically change certain parameters in this file prior to running the model each of several times. This method allowed the running of the model with a multitude of scenarios without user interaction.

The general strategy that was used to read in the data and set the model's parameters was to read the data into a variable using QUEST's READ_LINE and SCAN_STR commands. Then a BCL (Deneb's Batch Control Language) command is executed from within SCL (Deneb's Simulation Control Language) to manipulate the model accordingly. To illustrate, the time for CRT assembly (packing), which is given in the data file segment above, is read into the model as follows:

```
READ_LINE( #1, temp_str_1 )
SCAN_STR( temp_str_1, "--",
      T_CRT_packing )
```

The first line of the above segment reads the distribution (one line) from the data file into a temporary string variable. The second line removes the associated description (comment) from the temporary string variable via the SCAN_STR statement. Finally, the following code is used to set the parameter in the model to incorporate the data that was read:

```
bcl1 = "SET WORKCELL 'CRT_packing'
      CYCLE TIME OF 'process_1' TO "
bclsend = bcl1 + T_CRT_packing
err = BCL(bclsend)
```

This code combines a BCL statement with the data that has been read into the model and then executes the BCL statement that has been created. The data input portion of the code consists mostly of numerous READ_LINE and SCAN_STR statements combined with code for BCL statement generation and execution. A total of 72 parameters are read into the model through the use of this file.

It should be pointed out that the flexibility provided to the user in terms of reading in data in this manner, including the ability to specify a distribution type in addition to it's parameters, is a capability in QUEST which is usually not found in other simulation packages. Also, it should be pointed out that the model is designed to allow the user the option of viewing the data after it has been read into the model, which is useful in verifying that the model is actually using the parameters which the user believes it is using.

3.2 Setting and Changing the Quantities of Parallel Machines

It was desired for the model user to be able to specify the number of each type of machine to use in the model for each run. The strategy that was employed to accomplish this was to

have a large number of workcells in the model for each machine type and then to "bring down" those machines which are extraneous to the amount requested by the model user. This is done by use of the FAIL statement. For example, machine type X may be represented in the model by 20 different workcells, but the user may wish to run the model using only 6 of these machines. In this case, 14 of the 20 workcells would be effectively removed from the model via the use of FAIL statements. The process logic code which is utilized to accomplish this strategy is illustrated below:

```
if (leftstr(cres->name,4) == 'AM_C')
  then
    if (AM_C_if_op[int(val(rightstr(
      cres->name, int(val(leftstr(
      rightstr(cres->name,2),1))
      + 1.5))) +0.5)] == 1 ) then
        do_process( ANY )
    else
        do_process( ANY )
        fail 0, 10^10000
    endif
endif
```

The code above checks the variable value to see whether a particular machine is to be operating (based upon the quantity of machines of the particular type specified by the user to be in use) and then, if the workcell is not needed, effectively removes the workcell from the model by use of a FAIL statement with an approximately infinite failure time. The portion of the code shown above is used for the calorimeters in the accountability measurements area of the facility. The workcells in this area are named AM_Calorimeter_01, AM_Calorimeter_02, AM_Calorimeter_03, etc. (Note: an extremely valuable enhancement to QUEST would be to allow the definition of arrays of machines, thus eliminating the need to define numerous machines all of which are the same, as was necessary here.)

At the start of each simulation run, dummy widgets are sent to each workcell to "take down" unnecessary workcells. To prevent widgets from being sent to workcells which have been disabled (using the code above), the CHECK_ACCEPT statement is used. An example of it's usage is shown below:

```
for ii = 1 to 46 do
   if ( Check_Accept(cres->out[ii],
     cwgt) ) then
      transfer cwgt to cres->out[ii]
      return
   endif
endfor
```

This coding goes through all of the output connections for a workcell and only sends widgets to workcells that are capable of processing.

3.3 Periods of Inactivity

This model is different from most models in that for much of the time, material flow is not taking place in the model. To make the model more "run friendly" and exciting, a pop-up message appears on the screen reporting the time at which the next convoy arrival of trucks are going to occur as well as the time at which convoy departures will begin to be taken from the vault. (The times between arrivals and departures are randomly sampled.) Once a random event occurs, QUEST schedules the time of the next occurrence of the same event. The reporting of these events is simple because the user has access to these global variables (times of next occurrence) in QUEST. This reporting allows users to skip through lulls in the system and focus on periods of activity, without having to stare at the screen waiting for something to happen.

3.4 Preventing Two-Way Flow

One of the characteristics of the facility is that containers will not flow both in and out of the system at the same time. Containers can flow through the system from the truck unloading/loading area to the vault or they can

flow from the vault out to the truck unloading/loading area, but they cannot be flowing in each direction at once. To handle this situation in the model, two variables are maintained which control system flow. When an incoming convoy is cleared to be unloaded, a variable is set to "1" and the coding for outbound flow is such that outbound flow does not occur when this variable is set to "1". When the convoy's containers have all been safely put into the vault, the variable is then set back to "0". A second variable is modified in a similar manner to "lock out" inbound flow while outbound flow is occurring.

3.5 Output Reporting

QUEST provides users with an extremely flexible and portable technique for accessing the list of resources in a model. This technique is based upon QUEST's maintenance of a global variable called first_resource and the fact that each resource has a type associated with it as well as a pointer called next_res. The general strategy that was employed is shown below:

```
temp_res = first_resource
while (temp_res->next_res <> NULL) do
   if (temp_res->resource_type ==
    WORKCELL) then
      -- Perform reporting for
         workcells
   endif
   if (temp_res->resource_type ==
    BUFFER) then
      -- Perform reporting for
         buffers
   endif
   temp_res = temp_res->next_res
endwhile
```

In this code, temp_res is a temporary variable of type "resource". It is first set to the first resource in QUEST's global list and performs output reporting for that resource, depending upon the resource's type. The loop then goes through all resources in the model and produces output reporting for each resource depending upon each resource's type.

An extremely beneficial aspect of the code segment above is its reusability. It's very easy to take the same code from one model to another. Thus, a QUEST modeler could develop his/her own customized reporting technique and use this technique in each model he/she develops with little or no added work.

3.6 Model Debugging

To facilitate model debugging, a logical variable (0 = false, 1 = true) was utilized. It is called debug_mode and was used to provide additional reporting (usually to the screen) whenever the model was being debugged. The general strategy for employing this debugging capability is as follows:

```
if (debug_mode == 1) then
   -- Perform add'l debug reporting
endif
```

Utilizing this technique proved to be very beneficial in expediting the verification and debugging of the model.

4.0 QUEST/IGRIP Integration

It was desired to model the discrete events of the facility to track material flow and assess utilization of workcells. Equally important was the ability to have detailed robotic workcells showing realistic dynamics and cycle times. We learned of the three methods of integrating QUEST and IGRIP from Deneb and employed them in our model. The following three subsections describe these integration methods.

4.1 Recording and Playback

Several of the workcells in the QUEST model are represented as generic squares or pads. Detail of these workcells was not necessary. It was essential to be able to show detail of certain workcells, though, like the S-800 robot (unpacking area). A recording of this workcell was made in IGRIP and imported to the QUEST model. Each

sim_update in IGRIP represents a pose being imported to QUEST so setting the step size prior to recording is important so as to have the appropriate number of poses. The steps necessary to do this recording and playback are as follows:

1) Retrieve the model in IGRIP.
2) Turn recording mechanism on (Motion→Rec→Recording ON).
3) Run workcell for desired time (start and end poses should be identical).
4) Turn recording mechanism off.
5) Create a QUEST export version of the workcell (Motion→Rec→QUEST Export).
6) Choose the newly created recording file as part of the export process.
7) Exit IGRIP.
8) Start QUEST - include the IGRIP config files to provide access to the recorded file.
9) Import the IGRIP workcell (Kin→Build→Cre).
10) Choose 'Use IGRIP Workcell' from the popup and use IGRIP cycle time.
11) Characteristics of the workcell can be modified just like any workcell created in QUEST.
12) To have the script executed when a widget arrives, associate the script with a process using the Kin→Pose→Associate button.

4.2 Non-Dynamic (Static) Linking

It was desired to be able to show detailed robotic workcells of certain automation concepts within the facility at a layer below this higher-level QUEST model. Non-dynamic linking of QUEST workcells with IGRIP workcells was chosen to be able to show this level of detail. The user is able to "jump" to the more detailed IGRIP workcell through the use of a few buttons. The steps necessary to make this type of link and jump are as follows:

1) Create the IGRIP and QUEST models.
2) From QUEST, choose the Model→Flow→Link button (choose the desired QUEST workcell in the model).
3) Select the corresponding IGRIP model.
4) Save the QUEST model.
5) Select the Model→Flow→Jump button.
6) Size the IGRIP window.
7) An IGRIP session has been launched and all IGRIP functions are valid.
8) Exit IGRIP to return to QUEST.

QUEST workcells can be linked to other Deneb products as well including other QUEST workcells making the links several layers deep. A modeler is then able to show the appropriate amount of information at each level of detail. The steps to link to other products would be similar to the above.

4.3 Dynamic Linking

A third method for integrating QUEST and IGRIP is dynamically through the use of TCP/IP sockets. The CRT Handling area and the SPS area were modeled using this technique. The QUEST model is brought up on one workstation and the two IGRIP models are each brought up on different workstations (or the same workstation if desired). Over the local area network, the QUEST model continually sends update commands to the two IGRIP processes so that the timing of all three models is in sync.

Within the QUEST SCL code, the use of the 'open server' command was employed. Similarly, the 'open client' command is used in the IGRIP GSL code. The entire coding for this is rather lengthy and is included in the next four sub-subsections.

1) For each workcell link, open two sockets with QUEST as the server and IGRIP as the client:

QUEST SCL code:

```
-- Open the server socket for
   communication from QUEST to IGRIP
read_kbd("Enter Port Number 1 for
  SPS AS/RS: ", port1)
OPEN server port1 for update as 17

-- Open the server socket for
   communication from IGRIP to QUEST
read_kbd("Enter Port Number 2 for
  SPS AS/RS: ", port1a)
OPEN server port1a for update as 19
```

IGRIP GSL code:

```
-- receive notification to start
retval = read_kbd("Enter Port
  Number 1: ", port1)
host_port = server1 + str(':%g',
  port1)
OPEN CLIENT host_port FOR UPDATE as
  unit_no

-- receive notification OK to
   continue next update
retval2 = read_kbd("Enter Port
  Number 2: ", port2)
host_port2 = server1 + str(':%g',
  port2)
OPEN CLIENT host_port2 FOR UPDATE as
  unit_no2
```

2) Use a WHEN condition in IGRIP to send a signal to QUEST for each sim_update:

```
-- Check for incoming data to parse
   when the first socket has
   incoming data
WHEN (CHECK_STREAM(1)) DO
  parse_data() as 1

-- Execute routine to send
   notification to QUEST for each
   sim_update
WHEN (TRUE) DO send quest() as 2
WHEN_MASK(2, chk_on_update)

-- Execute routine to notify QUEST
   when action is finished
WHEN (alldone == 1) DO proc_done()
  as 3

WHILE (TRUE) DO
   ENABLE (1)
   ENABLE (3)
   sim_update
ENDWHILE
```

3) Have a resource in QUEST that does nothing but read the socket, write back a confirmation to IGRIP, and then delay for sim_update amount of time:

```
WHILE (TRUE) DO
   -- If IGRIP is active then sync
      to real time
   if (synctime == 1) then
      WHILE (TRUE) DO
         -- IGRIP will write back a
            "0" over stream 17
            whenever a second of time
            passes and a "1" whenever
            the current task is
            complete
         -- Read update notification
            from IGRIP and send back
            a signal to go ahead
         READ(#17, taskdone)
         WRITE(#19, taskdone, cr)
         -- If the task is NOT done
            then wait until the rest
            of the second has elapsed
         -- If the task is done then
            signal the workcell to
            continue
         if (taskdone == 0) then
            delay 2
            break
         else
            delay 2
            signal taskdone
            break
         endif
      ENDWHILE
   else
      delay 2
   endif
ENDWHILE
```

4) The socket can be used to pass information back and forth as well. The SPS AS/RS model sends the aisle, bay and tier from QUEST to IGRIP:

```
-- Signifies an example aisle, bay,
   tier combination
cres->aisle = 1
cres->bay = 5
cres->tier = 3
cres->action = 'p'
```

```
send_str = str('aisle %g',
 cres->aisle) + str('bay %g',
 cres->bay) + str('tier %g',
 cres->tier) + blank + cres->action
-- Write on socket to IGRIP
write (#17, send_str, cr)

-- wait for IGRIP to notify that
   task is finished
wait until signal 1
```

5.0 Results of the Base Model

The first objective of this project was to have a model that several people could view and use to understand the project and improve communications between a group of people. The QUEST model met this objective by showing a 3-D rendering of the facility including proper sizes of areas and equipment in addition to being able to show material (widgets) flowing through the facility with realistic cycle times. The concepts shown in Fig. 1 and the quantities of machines became the baseline model for this facility.

The next objective was to have a base model that could be changed easily and run so that comparisons could be made. It is rather simple to produce reports and graphs within QUEST for this purpose. These reports describe such things about a workcell as numbers of widgets consumed and produced, numbers of widgets to fail the operation, average widget residence time, time busy, down, and idle, and so forth. In addition to reports, graphs are easy to generate. A graph of the vault contents over long periods of time was of particular importance in making comparisons.

5.1 Trade-off Studies

The above-mentioned reports and graphs are being used to perform trade-off studies in comparing the many alternatives. Although Sandia ISRC is part of the team supplying the data, it is not our responsibility to perform these studies - we are making recommendations and supplying all of supporting documentation for the recommendations. An independent team from academia will be making an assessment of the data and alternatives.

Nevertheless, through the use of this model, data such as throughput and utilization can be produced easily for a wide variety of scenarios. Other information relating to cost of robotic workcells and the like is being supplied as well. The effects of altering parameters slightly are being investigated. This form of sensitivity analyses can be used to point out the requirements that have the largest effects on determining the best alternatives.

6.0 Conclusions/Summary

A project of this magnitude inherently required modeling and simulation to be able to produce the requested data. Deneb's IGRIP product had been used quite extensively by Sandia ISRC to perform detailed robotic simulations. An opportunity arose that required discrete-event data as well in a facility comprising many workcells. Deneb's QUEST product was chosen to model the discrete-event occurrences because of the promises Deneb made about marrying the two products. We were able to accomplish what we set out to do in developing the flexible model and using it to supply the results of many "what if" scenarios.

In addition to using this model for it's intended purpose of evaluating different concepts for the PSF, it is also a good framework for other modeling endeavors to follow. Besides having to draw and reconnect a new model together, many of the techniques described in section 3 can be applied to a new model with little or no re-work effort. This is especially true of the lessons learned from the integration of QUEST and IGRIP as was described in section 4.

Acknowledgments

The authors would like to thank Joseph Hugan of Deneb Robotics for a great deal of

help in setting up the model including the QUEST/IGRIP integration and for the updates to the QUEST product as a result of this project. In addition, we would like to thank Larry Shipers and his team at Sandia National Laboratories for gathering much of the information and requirements for the facility being modeled.

Many of the items called out in this paper in sections 3 and 4 referring to deficiancies in the QUEST product have been brought to Deneb's attention. We would like to thank the support team at Deneb for providing work arounds and suggestions to be able to accomplish certain items. This type of feedback from users has been encouraged by Deneb so that future versions of the product can be enhanced.

DISCLAIMER