

# **CRITICAL DESIGN REVIEW**

---

## **Aircraft Repair Model Simulation**

October 25, 2000

### **TEAM 9: GOLDRUSH**

Nick Capron

Dave Stone	Carina Lansing
Russell Swannack	Kyle Klicker
Miguel Vilchez	Jonathan McPherson
Kelly Yearout	Andy Reynolds

## **Agenda**

---

1. Requirements Changes
2. Task Update
3. Assumptions Update
4. Context Diagram
5. High Level DFD
6. Lower Level DFDs
7. Impacted Modules Update
8. Functional Design (P-Specs, Data Dict.)
9. Structural Design (Structure Chart)

## **Agenda Continued**

---

10. Transactions
11. Traceability Approach
12. Schedule
13. Open Issues
14. Current Status
15. Action Items

## **Requirements Changes**

---

### **Extensibility Support:**

Code shall *facilitate* the user being able to “create” his own queuing scenarios. Ultimately (not for this release) the user will be able to:

- Set number of service stations
- Add different aircraft
- Designate which aircraft are served by which service stations
- Set number of queues
- Designate which aircraft can enter a given queue
- Designate queue priority
- Designate queuing style (e.g., FIFO)

## **Requirements Changes Cont'd**

---

### **Insert/Delete Clarification:**

The following functions shall be available:

- Insert a single event (arrival, departure, or endsim) to event list
- Delete a single event (arrival, departure or endsim) from the event list
- Delete an aircraft from any queue
- Delete an aircraft from any service station
- Downtime costs for deleted aircraft are included in the final report

## **Requirements Changes Cont'd**

---

### **Progress Screen:**

- The ARMS progress screen shall start in "paused" mode.

### **Initialization Procedure:**

- User shall be able to set event execution speed (delay time).
- User shall be able to change default pseudo-random number seed.

## **New Tasks**

---

- **Extensibility requirement increases effort on design and coding tasks**

## **Assumptions Update**

---

- **User is knowledgeable enough to use the simulation software properly.**
- **User understands that if he deletes a departure event (for example) and never reschedules a departure for that aircraft, then that service station will be “frozen” for the rest of the simulation.**
- **User understands that if he deletes the “endsim” event, the simulation will not stop until he inserts a new one.**

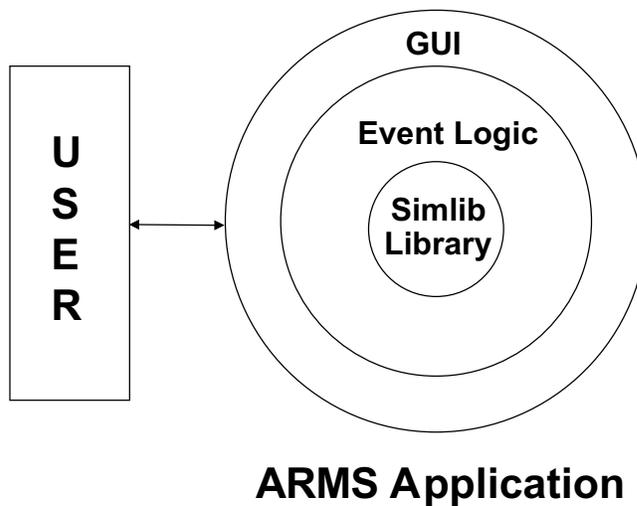
## Assumptions Update

---

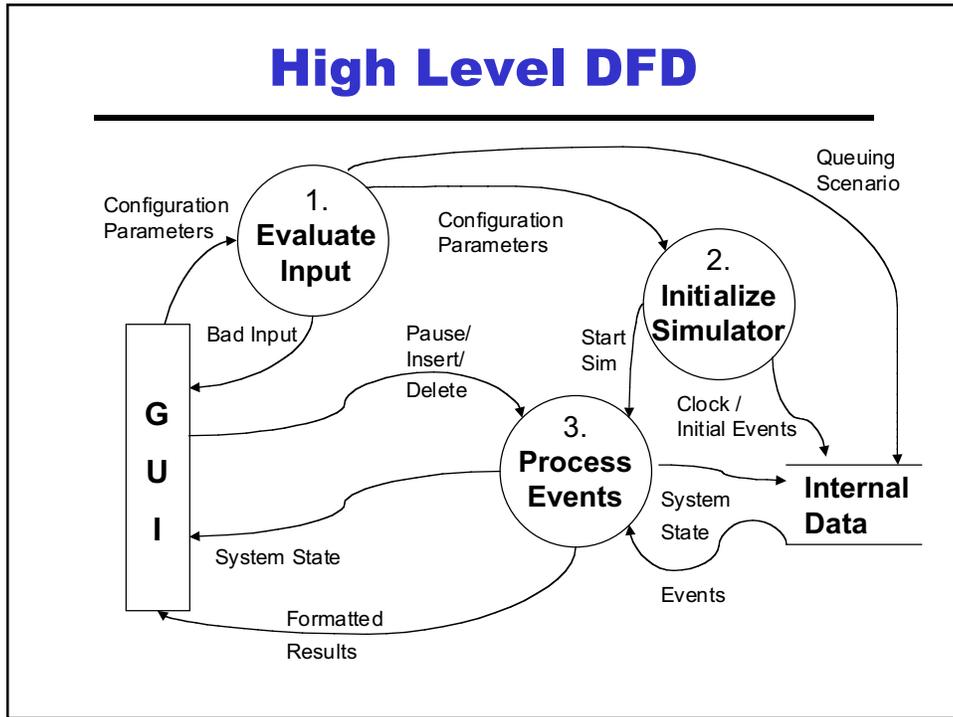
- Simlib lists will be sufficient to handle future extensions in the number of service stations and queues

## Context Diagram

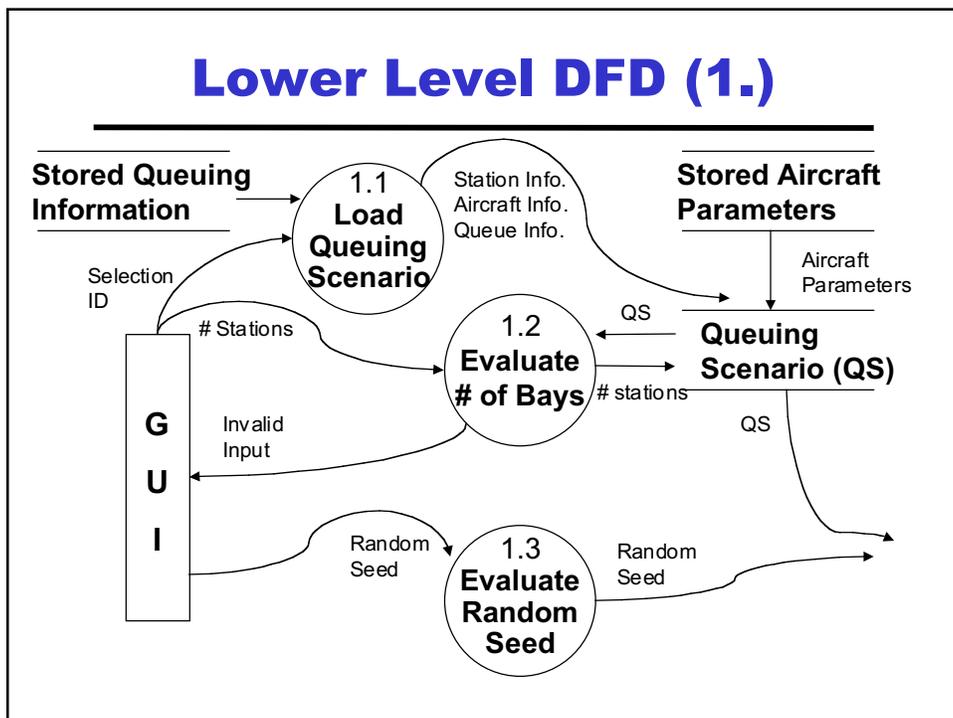
---



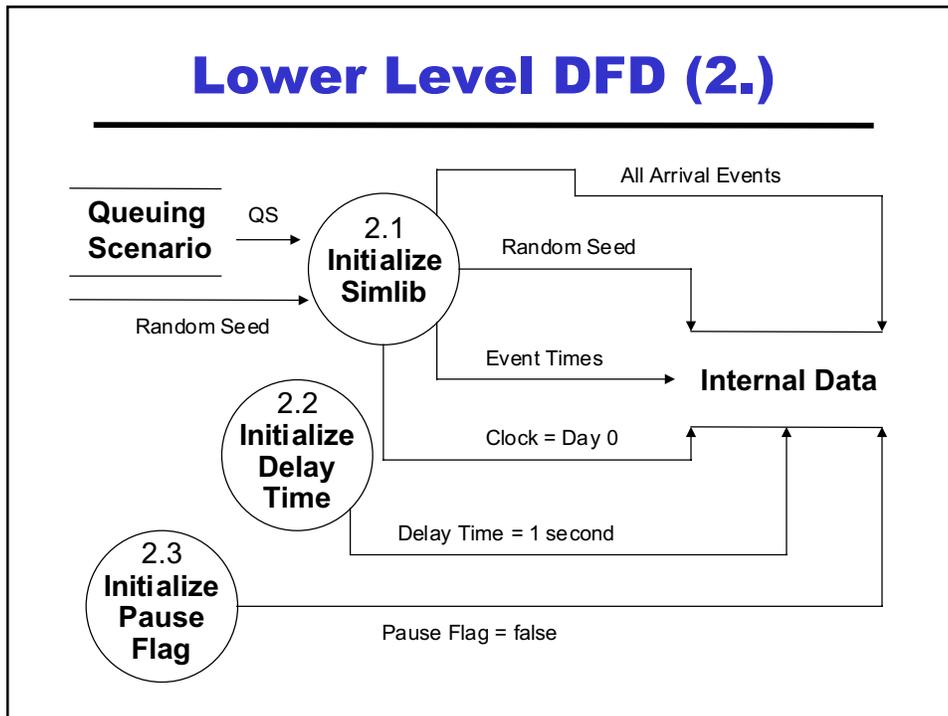
## High Level DFD



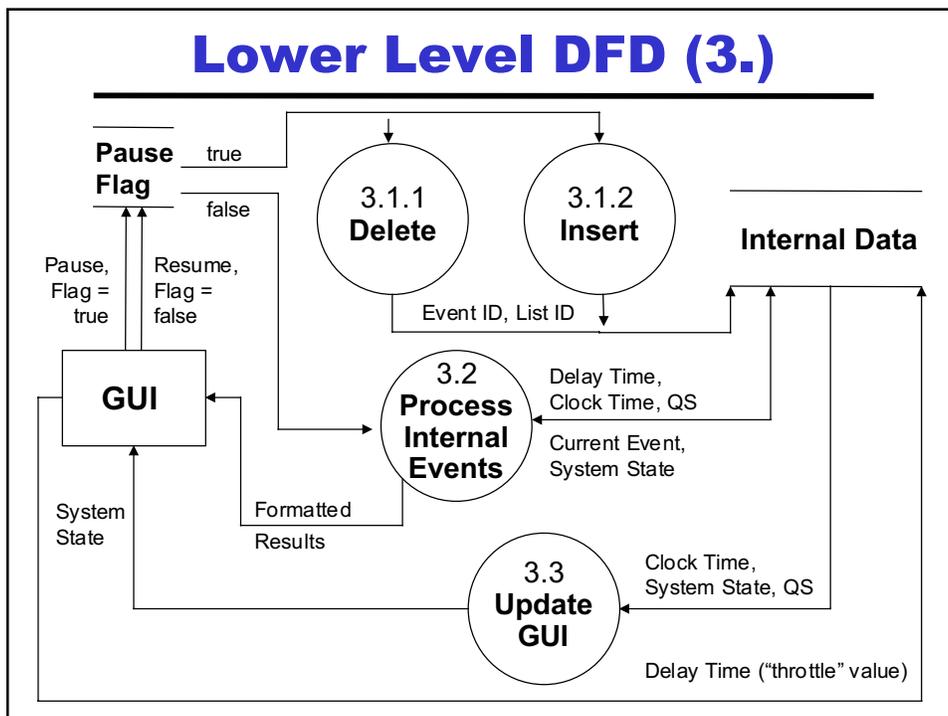
## Lower Level DFD (1.)



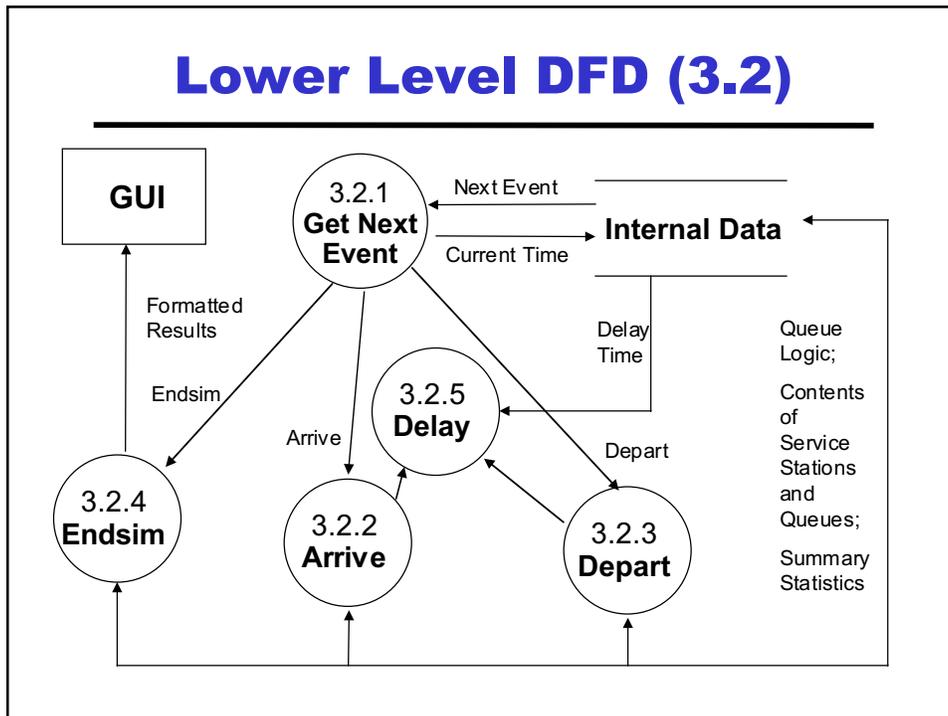
## Lower Level DFD (2.)



## Lower Level DFD (3.)



## Lower Level DFD (3.2)



## Impacted Modules

### Modifications to Simlib API:

- No additional modifications required.

## Functional Design (P-Specs)

---

### 1. Evaluate Input

*/\* Customer presses "start" button \*/*

#### 1.1 Load selected queuing scenario

1.1.1 Set information about service stations

1.1.2 Set information about aircraft

1.1.2 Set information about queues

#### 1.2 Evaluate number of service stations

1.2.1 If number of service stations entered is invalid for scenario, reprompt for input

#### 1.3 Evaluate random seed

1.3.1 If random seed not a number or blank,  
Use default random seed

## Functional Design (P-Specs)

---

### 2. Initialize Simulator

*/\* Customer presses "start" button and input data is valid \*/*

#### 2.1 Initialize Simlib

2.1.1 Initialize random number streams

2.1.2 Initialize event List

2.1.2 Set system clock to 0

#### 2.2 Initialize delay time to default value of 1 second.

#### 2.3 Initialize pause flag to default value of true.

## **Functional Design (P-Specs)**

---

### **3. Process Events (infinite loop)**

*/\* Always occurs after system is initialized \*/*

#### **3.1 If “pause” mode is activated, process external events**

3.1.1 If delete selected, process delete

3.1.2 If insert selected, process insert

#### **3.2 Else, process internal events**

3.2.1 Get next event from event list (via Simlib transfer array)

3.2.2 If arrive, process arrival

3.2.3 If depart, process departure

3.2.4 If endsim, process endsim

3.2.5 Wait the delay time

## **Functional Design (P-Specs)**

---

### **3.3 Update progress GUI**

3.3.1 Print event list

3.3.2 Print queue lists

3.3.3 Print service station lists

3.3.4 Print current simulation time

## Functional Design (P-Specs)

---

### 3.1.1 Process Delete

*/\* Occurs if user selects pause, then delete option \*/*

- 3.1.1.1 Find selected event from list using middle pointer algorithm.
- 3.1.1.2 Delete event from list
- 3.1.1.3 If deleting from bay, force aircraft departure.
- 3.1.1.4 If deleting from queue, update statistics.

### 3.1.2 Process Insert

*/\* Occurs if user selects pause, then insert option \*/*

- 3.1.2.1 Find location to insert in event list using middle pointer algorithm
- 3.1.2.2 Insert event into event list

## Functional Design (P-Specs)

---

### 3.2.2 Process Arrival

*/\* Occurs if first event in event list is an arrival \*/*

- 3.2.2.1 If any service stations that can take this aircraft are empty, move aircraft into station and schedule next departure time.
- 3.2.2.2 Else, move aircraft into least filled queue that will accept it and update queue statistics.

### 3.2.3 Process Departure

*/\* Occurs if first event in event list is a departure \*/*

- 3.2.3.1 Remove aircraft from appropriate station list and update service station statistics. (Ignore departure if no such aircraft in that station.)
- 3.2.3.2 Choose next aircraft from appropriate queue, move aircraft into station, and schedule next departure time.

## Functional Design (P-Specs)

---

### 3.2.4 Process Endsim

*/\* Occurs if first event in event list is endsim\*/*

3.2.4.1 For each bay, update statistics for that service station (assuming aircraft have just departed).

3.2.4.2 Generate Simlib summary data (average time spent in queue, average downtime, etc.).

3.2.4.3 Calculate average downtime costs for all aircraft types.

3.2.4.4 Display results on output GUI.

3.2.4.5 Exit application.

## Functional Design (Data Dict.)

---

### Stored Queuing Info =

Saved information about ALL queuing scenarios (see below). This list will eventually be updateable by the user.

### Stored Aircraft Parameters =

Saved information about ALL aircraft types (see below) See SRS for the complete table. This list will eventually be updateable by the user.

### Queuing Scenario =

List of data for each service station + list of data for each aircraft + list of data for each queue defined in the user-specified scenario.

### Service Station data =

List of aircraft IDs that it services + label + ID # (note that the ID# identifies the corresponding Simlib list that hold the internal data for that station).

## Functional Design (Data Dict.)

---

### **Aircraft data =**

Mean time between failure + distribution types + repair data + label + ID #.

### **Queue data =**

List of aircraft IDs that it accepts + queuing style (e.g., FIFO) + priority level + label + ID #.

### **Streams =**

Lists of random numbers used to calculate event times (put what streams go to what statistics).

### **Event List =**

Simlib list of events in ascending time order.

### **Event =**

Time of event + event type (arrival | departure | endsim) + aircraft ID# + service station ID# (for departure events).

## Functional Design (Data Dict.)

---

### **Delay Time =**

The amount of time to keep each change in system state up on the progress GUI. Can be set by "throttle" on the progress GUI.

### **Transfer Array =**

Simlib data structure used to transfer events between lists.

### **Pause Flag =**

Global variable indicating if user pressed the pause button on the progress GUI.

### **Service Station Lists =**

Simlib lists showing the contents of each service station (if full, what aircraft is currently there).

### **Queue Lists =**

One Simlib list for each queue in the queuing scenario showing which aircraft are in the queue in increasing time order.

## Functional Design (Data Dict.)

### Simulation Clock =

Simlib global variable that indicates the current day of the simulation.

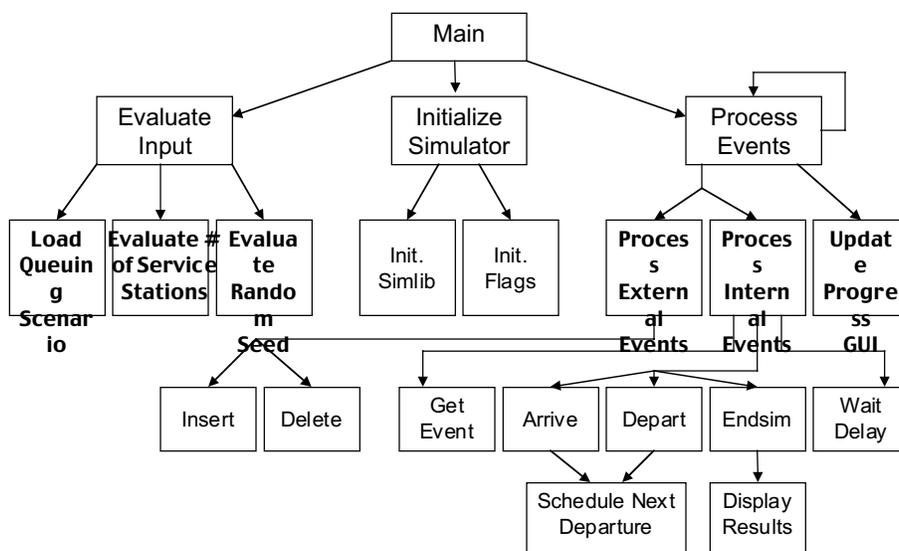
### Statistical Counters =

Simib's internal data storage for summary statistics about each list (such as the number of aircraft that have passed through the queue)

### Internal Data =

The data stores internal to Simlib and/or the ARMS java code. This includes the queuing scenario, a transfer array, an event list, a list for each queue, a list for each bay, the simulation clock, the delay time, the statistical counters, and the pause flag.

## Structural Design



## Transactions

---

### Case 1:

- Event: Running stored Queuing Scenario #1.
- Stimulus: User selects QS1, enters a valid number of service stations, and presses start.
- Activity: ARMS runs to completion using the logic defined by QS1.
- Response: Results are displayed to output GUI.
- Effect: Verifying the description and logic associated with QS1.

## Transactions

---

### Case 2:

- Event: Running stored Queuing Scenario #2.
- Stimulus: User selects QS2, enters a valid number of service stations, and presses start.
- Activity: ARMS runs to completion using the logic defined by QS2.
- Response: Results are displayed to output GUI.
- Effect: Verifying the description and logic associated with QS2.

## Transactions

---

### Case 3:

- Event: Running stored Queuing Scenario #3.
- Stimulus: User selects QS3, enters valid numbers of service stations, and presses start.
- Activity: ARMS runs to completion using the logic defined by QS3.
- Response: Results are displayed to output GUI.
- Effect: Verifying the description and logic associated with QS3.

## Transactions

---

### Case 4:

- Event: Running stored Queuing Scenario #3.
- Stimulus: User selects QS3, enters invalid numbers of service stations, and presses start.
- Activity: ARMS prompts user for re-entry until valid parameters are entered.
- Response: Error message is displayed to GUI.
- Effect: Verifying the error checking of input parameters.

## Transactions

---

### Case 5:

- Event: Running stored Queuing Scenario #2.
- Stimulus: User presses Pause button on progress GUI, presses button to Delete an aircraft from the queue, and presses Resume.
- Activity: ARMS deletes that aircraft from the selected queue and then runs to completion.
- Response: Progress GUI shows that the aircraft has been removed from the queue.
- Effect: Verifying the Delete function.

## Transactions

---

### Case 6:

- Event: Running stored Queuing Scenario #1.
- Stimulus: User presses Pause button on progress GUI, presses button to Insert an event to the event list, and presses Resume.
- Activity: ARMS prompts the user for a description of the event, inserts the event to the event list, and then runs to completion.
- Response: Progress GUI shows that the event has been inserted into the event list.
- Effect: Verifying the Insert function.

## Traceability Approach

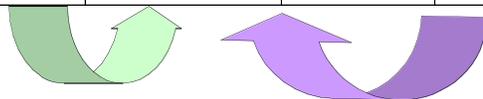
---

- Categorize/number requirements
- Identify testing method for each requirement
- Develop Requirements Traceability Matrix (RTM)
- Link DFD modules to RTM requirement categories
- Test modules against assigned requirements

## Traceability Approach Cont'd

---

Req. ID System Level	Req. ID Sub-system Level	DFD Identifier(s)	Module Name(s)	Verification Method	Tested
A001.00				I,D	
	A001.01			I,D	
	A001.02			I,D	
	A001.03			I,D	
	A001.04			I,D	
	A001.05			I,D	
	A001.06			I,D	



## Schedule

	September	October	November	December
T1	[Green bar spanning from start of September to end of October]			
T2	[Green bar spanning from start of September to end of November]			
T3	[Green bar spanning from start of October to end of December]			
T4	[Green bar spanning from start of November to end of December]			

**T1: Understand Problem/Requirements**

*(Client discussions, Simlib study, Java review, SRS development)*

**T2: Design ARMS Application**

*(GUI design, PDR, CDR, Design Notebook)*

**T3: Code ARMS Application**

*(GUI, Simlib extensions, event-handlers, User Manual)*

**T4: Test/Validate ARMS Application**

*(Test cases, requirements validation, code modifications)*

## Open Issues

- How to display extensibility options on input GUI.

## Current Status

---

### Accomplishments:

- Simlib/Java interface
- Project Plan, SRS, PDR
- Critical design
- RTM

### Plans:

- GUI finalization
- Design Notebook
- Coding
- Testing Plans

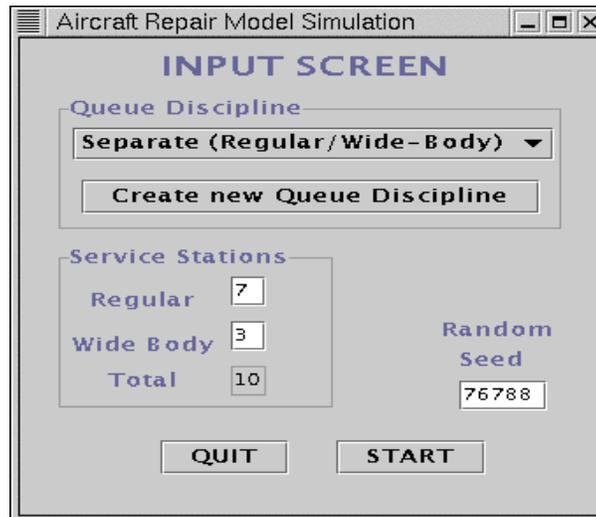
### Problems:

- Network outages
- Limited experience
- Communications

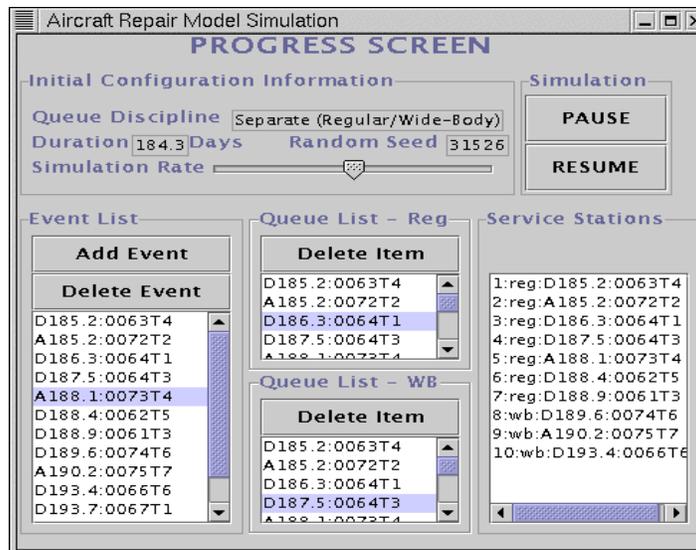
## Action Items

---

# Input GUI Prototype



# Progress GUI Prototype



# Output GUI Prototype

**REPORT SCREEN**

Queue Discipline: Separate (Regular/Wide-Body)

Duration: 365 Days Random Seed: 31526

Service Stations:  
Regular: 7  
Wide-Body: 3

Aircraft Type	Average Queue Delay (days)	Time-Average Number Down
1		
2		
3		
4		
5		
6		
7		

Average Queue Delay Overall (days):

Time-Average Number in Queue Overall:

Total Average Daily Downtime Costs Overall: X \$10,000

**Restart Application**