An Electrical Power Planning Simulation and Graphical User Interface for an Advanced Space Habitat Life Support System

Gary Huband and Michael Dowell
Math/Computer Science Department, Georgia Southern University

ABSTRACT

BIO-Sim is a software package developed at Georgia Southern University that simulates the electrical power consumption of devices in a planetary habitat life support system. BIO-Sim consists of four parts: a graphical user interface (GUI), a power usage simulation, a data model, and a server/database. The GUI allows planners to easily enter/adjust device schedules; view the resulting power consumption on a bar chart, as well as the average power and total energy requirements. The simulation uses device schedules as input and creates the power consumption data. The database stores device properties and schedules. This paper concentrates on a detailed description of the software package.

INTRODUCTION

There are currently a number of Lunar/Mars test bed habitats [1],[2]. Each seeks to answer major issues associated with long-term human presence in a hostile environment. One of these issues is power management. The habitat must carry its own power generation equipment that is limited in the amount of power it can generate. A habitat will contain hundreds of devices that require power and scheduling [3]. The habitat will need high-level planning and scheduling software to reduce crew workload and maximize the use of limited power.

This paper describes the software package BIO-Sim, a tool to schedule and simulate power utilization for a habitat. This version of BIO-Sim will only model device power consumption. The user will enter a schedule of start/end times for each device, then view the resulting power vs. time chart. The model does not currently include the complex interactions with the life support system such as automatic regulation of temperature and oxygen/carbon dioxide levels. This paper describes the first step in providing a high-level scheduling tool – the software framework for the simulation and graphical user interface.

BIO-Sim maintains a list of power consuming devices where each device is modeled by a name, amount of power consumed, and a schedule of activities. Each activity models when the device is in use and is represented by a start and end time. BIO-Sim allows the user to view and manipulate the schedule graphically and view power statistics on a bar chart (see Figure 1).

The software has four major parts: the graphical user interface (GUI), the power usage simulation, the data model, and the server/database.

The GUI allows the user to enter and manipulate a list of device schedules. This data is saved in the database. The simulation then takes the device schedules as input and produces power usage vs. time data as well as statistics on power use. The power usage data is displayed as a bar chart in the GUI.

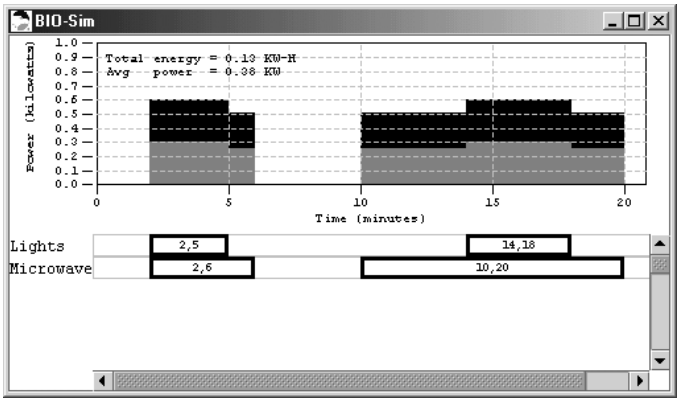

Figure 1: Graphical User Interface

BIO-Sim is written in the Java programming language. It can be used as a stand-alone application, or shared across an intranet or the internet.

The following sections describe BIO-Sim in detail. The Data Structure section covers the software data organization using set notation. This notation is used throughout the paper to describe software algorithms. The Architecture section describes the overall structure of each part of the software package and includes detailed descriptions of specific classes.

DATA STRUCTURE

The schedule consists of a list of devices. Each device has a list of activities, and each activity has a start time and an end time. This is represented using set notation.

Let D be the set of all devices:

$$D = \{d_1, d_2, d_3, d_i \ldots, d_n\},$$

where d_i is the i^{th} device in set D and n is the number of devices. For each device,

$$d_i = \{\text{name, power, } A_i\},$$

where A_i is the set of activities for d_i. The power is in Watts. The activities for a device are represented by

$$A_i = \{a_{i,1}, a_{i2}, a_{i,3}, \ldots, a_{i,m}\},$$

where $a_{i,j}$ is the j^{th} activity for device i, and m is the number of activities for device i. Each activity is an ordered pair

$$a_{i,j} = (s_{i,j}, e_{i,j}),$$

representing a start and an end time in minutes.

Suppose the schedule contains two devices: room lighting and a microwave oven. The set representation is

$D = \{d_1, d_2\}$
$d_1 = \{ \text{lights}, 100, A_1 \}$
$A_1 = \{ a_{1,1}, a_{1,2} \}$
$a_{1,1} = (2, 5) \quad a_{1,2} = (14,18)$
$d_2 = \{ \text{microwave}, 500, A_2 \}$
$A_2 = \{ a_{2,1}, a_{2,2} \}$
$a_{2,1} = (2, 6) \quad a_{2,2} = (10, 20)$

The power is measured in watts. In this example, the lights use 100 watts and the microwave uses 500 watts. The microwave starts two minutes into the simulation and operates for four minutes. It runs again starting at ten minutes and operates for ten additional minutes. This example is used later to illustrate how the simulation builds the power chart from the schedule.

ARCHITECTURE

BIO-Sim uses a client/server/database (Figure 2) architecture. This architecture is commonly used in internet applications where the client is a web browser that connects to the server across the internet to provide access to the database. This architecture can also easily be used for a stand-alone application or an application shared on an intranet.

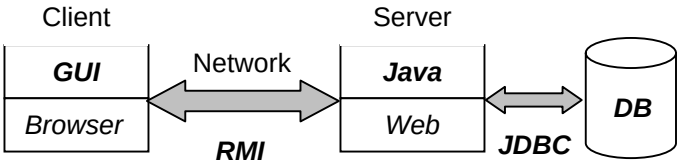

Figure 2: Client/Server/Database Architecture

BIO-Sim uses a Java application for the client, a Java application for the server, and a Microsoft Access database. The client includes the GUI for changing the power schedule and viewing results and the simulation to produce the results. When the user changes the schedule, the new data is sent to the server via Java Remote Method Invocation (RMI), the server then updates the database using the Java Database Connectivity (JDBC) programming interface. Finally, the GUI runs the simulation and displays the new results.

Graphical User Interface

The GUI is separated into two parts – the device schedule at the bottom, and the results at the top (usage statistics and bar chart). The schedule portion (see Figure 3) consists of a list of devices. Each device has its name on the left side and a schedule of start/end times (activities) on the right side represented as bars arranged on a time line. The beginning of the bar represents the start time and the end of the bar represents the stop time for the activity. To modify an activity, the bar can be repositioned to change both start and end times or, the bar can be resized to change one of the times.

Figure 3: Device Schedule

The user adds a device to the schedule simply by right-clicking in the device area and choosing a device from a list of available devices. To add an activity to this new device or an existing device, the user right-clicks in the device activity area and enters a start and end time. To delete a device the user right-clicks on the device and chooses delete. Right-click on a bar and choose delete to delete the activity. As the user makes each modification, the simulation runs and automatically updates the bar chart and statistics in the top part of the GUI.

The bar chart shows the total power used versus time, the total energy used, and the average power during the simulation time (see Figure 4). The bars show both the power used by all devices (black portion of the bar) and the total power needed to cool the waste heat from the devices (gray portion of the bar). The chart is horizontally scrollable and the scrolling is synchronized with the device schedule.

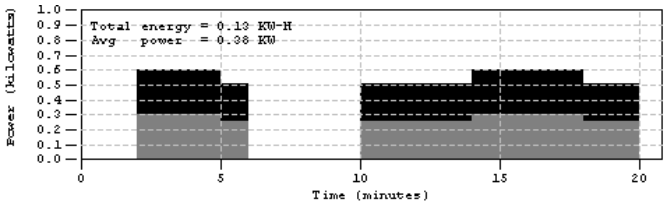

Figure 4: Bar Chart

The interface was constructed using software classes from the Java Swing application programming interface (API). Swing is a very powerful set of graphics components designed to make GUI construction relatively easy.

Power Simulation

A simulation is a model of a real-world system; in this case a habitat. The habitat as a system includes the habitat structure, humans, plants, an atmosphere, and electronic devices. Since the objective is to model power consumption, the simulation only includes the devices. Each device has two discrete behaviors – start of power consumption and end of power consumption. These events must be aggregated into partitions to produce the power versus time bar chart; allow calculation of the average power and total energy consumed. Using set notation, the algorithm for the simulation is as follows:

Let $S = \{s_{i,j} \mid s_{i,j} \in A_i\}$ be the set of start times and $E = \{e_{i,j} \mid e_{i,j} \in A_i\}$ be the set of end times. Then create Q, an ordered set of discrete times (q) that form partitions of the activities. Thus,

$$Q = \{ q_k \mid q_k \in E \cup S \text{ and } q_k < q_{k+1} \}$$

where k is a discrete event and k+1 is the next discrete event. Each partition is a single bar on the chart and represents a period of time when one or more devices are on. The total power used during the time period is the sum of the powers of all devices in the partition

$$Power_{k+1} = Power_k + \sum_{t_k \le q_k \le t_{k+1}} power_i * \delta$$

where Power is the total power for a partition, $power_i$ is the power for the i^{th} device and δ is defined as

$$\delta = \begin{cases} 1, q \in S \\ -1, q \in E \end{cases}$$

The δ operator adds the power for a start event and subtracts the power for an end event. Using the example data,

$$Q = \{ 2_{1s}, 2_{2s}, 5_{1e}, 6_{2e}, 10_{2s}, 14_{1s}, 18_{1e}, 20_{2e} \}$$

where the s subscript represents a start event and the e subscript an end event. Iterating through Q produces the total power during a partition:

@t=2, Power = $(power_1*1)+(power_2*1)$ = 100+500 = 600
@t=5, Power = 600+$(power_1*-1)$ = 600-100 = 500
@t=6, Power = 500+$(power_2*-1)$ = 500-500 = 0
@t=10, Power = 0+$(power_2*1)$ = 500
@t=14, Power = 500+$(power_1*1)$ = 500+100 = 600
@t=18, Power = 600+$(power_1*-1)$ = 600-100 = 500
@t=20, Power = 500+$(power_2*-1)$ = 500-500 = 0

Pairs of times form the partitions (bar chart divisions). The first partition is from 2 to 5 minutes with a total power of 600 Watts and the second partition is from 5 to 6 minutes with a total power of 500 Watts (see Figure 4).

The bar chart also shows the total energy used and the time averaged power for the simulation. The total energy is calculated from

$$Energy = \sum power_k * \Delta t_k$$

where Δt = e-s is the time interval for a partition. The time averaged power is then

$$AveragePower = \frac{1}{T} \sum power_k * \Delta t_k$$

where T is the total simulation time.

The simulation software implements the algorithm using the classes shown in Figure 5 and is based on a generalized discrete event framework [4]. This figure uses the Uniform Modeling Language (UML) conventions [5], [6] to show the relationship between classes. A line from one class to another ending in an arrow indicates that the class is a child class (inheritance). A line ending in a diamond indicates the class is used as an attribute of the class (composition). A line without an arrow indicates the class is simply used within the other class.

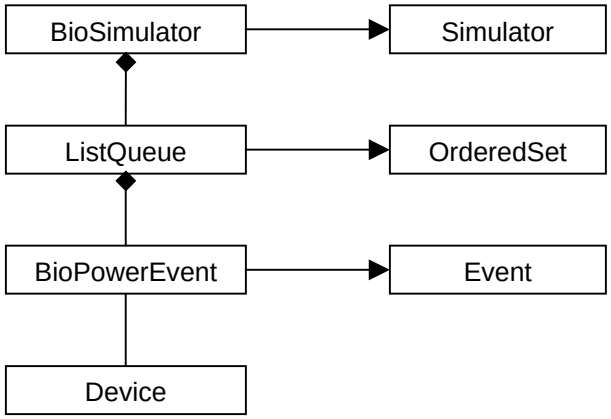

Figure 5: Simulation classes

The simulation receives the schedule data and places each activity for a device in a priority queue. The queue orders the devices by start time with the lowest start time at the beginning of the queue. Each activity is then removed from the queue and a bar for the chart data is

created. The activity is then placed back into the queue but is ordered by the stop time. The next start or stop time taken off the queue ends the data for one bar on the chart and starts the next bar. The simulation runs until all activity stop times are processed.

The BioSimulator class contains the ordered queue represented by the ListQueue class. The queue contains many events. Each event is an start or end time for a device.

The device class contains the data for one device - the device name, power, and the activities. Table 1 shows all of the attributes and methods for the Device class. Besides the usual get and set methods, the class has methods to add an activity, remove all activities, check if the device has any activities, and test for equality with another device.

Table 1: Device attributes and methods

Device
String _name double _power Vector _activities
Device(String name,double power) String get_name() double get_power() Vector get_activities() void set_name(String name) void set_power(double power) void add_activity(Schedule_Db) void removeAllActivities() boolean isScheduleEmpty() boolean equals(Device device)

Data Model

The data model enables other class objects to change the schedule data and retrieve updated schedule data. The model provides access to the schedule through the server by calling server methods. Figure 6 outlines the model classes.

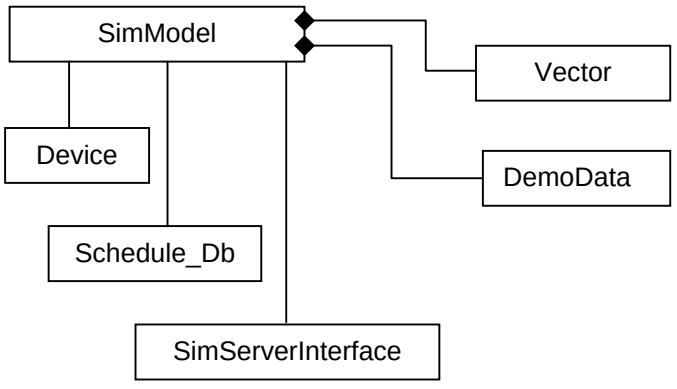

Figure 6: Model classes

Each class that changes schedule data must call a model method. The model has a method for each data change (see Table 2). For example, when the user deletes an activity using the GUI, the GUI must call the delete method of the model. The delete method then calls the delete method of the server, which deletes the activity from the database.

Each class that wants to view schedule data must register as a listener with the model. When the schedule changes the model will notify each listener and the listener calls the appropriate model method to retrieve data. For example, the chart data class first registers as a listener by calling the addListener method of the model. Whenever the schedule changes the model will execute the fireChange() method and the chart data class will be notified. The chart data class then calls the getScheduleData() method of the model which in turn calls the getScheduleData() method of the server.

Table 2: Model Methods

SimModel
String _serverURL SimServerInterface _server Vector _listeners boolean _db DemoData _demo
SimModel() addListener(SimListener) SimServerInterface get_server() String get_serverURL() removeListener(SimListener) fireChange() insertActivity(Device,Schedule_Db) deleteSchedule(Device) deleteActivity(Schedule_Db) updateActivity(Schedule_Db) getScheduleData() getDeviceList() SimServerInterface getServer() void callServer(String operation, Schedule_Db sDb)

Server/Database

The server provides access to the schedule data in the database through methods that correspond to methods in the model (i.e. getScheduleData(), getDeviceList(), etc.). The server could be located on the local machine or a remote machine connected by a network. The model and server communicate across the network using a Java technology called Remote Method Invocation (RMI).

RMI allows two objects to communicate transparently across a network. Transparent means that the method calls for a local object and a remote object are identical – whether the object is local or remote is transparent to the user. The local object only needs to know the location of the remote object (a machine address) and the available methods of the remote object. In BIO-Sim the server methods are described in an interface class called SimServerInterface. The model uses this class to connect to the server and call the methods to manipulate the schedule in the database.

The server is responsible for retrieving information from the database using queries written in the Structured Query Language (SQL). SQL is a standard query language for relational databases. The server uses Java Database Connectivity (JDBC) classes (see Figure 7) to connect to the database and submit queries.

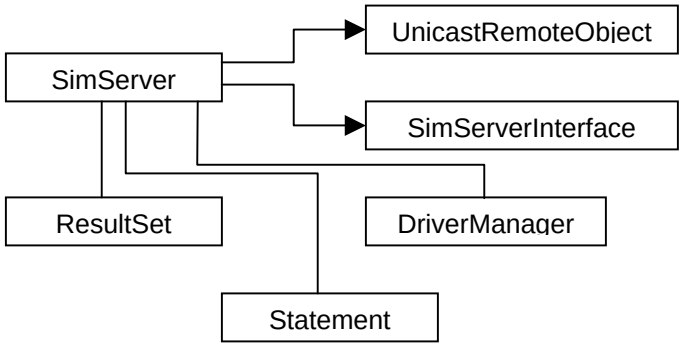

Figure 7: Server classes

UnicastRemoteObject provides the functionality for the server to communicate across a network. SimServerInterface defines the server methods that are invoked by remote objects such as SimModel. The DriverManager creates database connections, Statement is used to form SQL statements, and ResultSet holds the data returned from the successful execution of a SQL statement.

The database contains a table for the devices and a table for the schedule as shown in Figure 8.

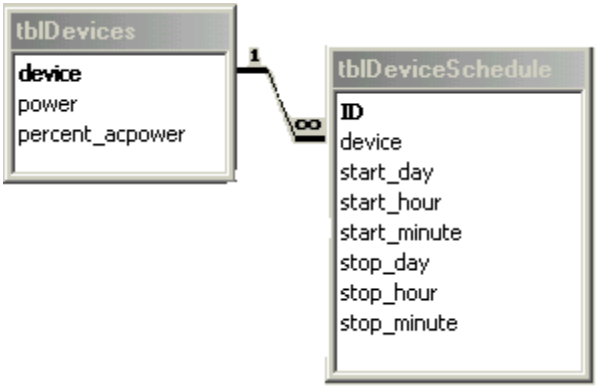

Figure 8: Database Tables

Each entry in the device table has the data for one device: device name, power used, and cooling power. The schedule table contains the start and end times for each activity. The tables are related by the device name. To retrieve all of the activities for one device the SQL statement selects the data for the given device from the devices table and all records from the device schedule table with the device name.

CONCLUSION

Bio-Sim is an easy-to-use power management tool for design and testing of a planetary habitat. A user can create a power schedule and view the resulting power use, average power, and total energy used. The user can then explore different scenarios by graphically manipulating the schedule and viewing the results.

In the future the Bio-Sim simulation will include more advanced features such as

- Modeling of cooling system.
- Modeling of oxygen/carbon dioxide levels.
- Automatic generation of schedules for statistical analyses.
- Integration with control software to provide plan evaluation.

These upgrades will increase the habitat model fidelity and increase usefulness in planning and design.

REFERENCES

1. "Bioregenerative Planetary Life Support Systems Test Complex", http://advlifesupport.jsc.nasa.gov/Bio~Plex/Bio~Plex~Enter.htm.
2. "Mars Society Flashline Arctic Research Station", http://arctic.marssociety.org/
3. Chen, G., and Dowell, M., Simulation of Power Utilization for Planning and Control of BIO-Plex Activiities.
4. Altmann, Michael, "Writing a Discrete Event Simulation: Ten Easy Lessons", http://www.nmsr.labmed.umn.edu/~michael/des/.
5. Rossetti, Manuel, D., et al, "SIMFONE: An Object-Oriented Simulation Framework", Proceeding of the 2000 Winter Simulation Conference, Dec, 2000.
6. Booch, G., Rumbaugh, J., and Jacobson, I. *The Unified Modeling Language User Guide*, Addison – Wesley, 1999.

CONTACT

Gary Huband
(Instructor, Computer Science)
ghuband@gsu.cs.gasou.edu
http://www.gsu.cs.GaSoU.edu/faculty/ghuband
(912) 681-0364

Dr. Michael Dowell
(Assistant Professor, Computer Science)
dowell@gsu.cs.gasou.edu
http://www.gsu.cs.GaSoU.edu/faculty/dowell
(912) 681-0251

Georgia Southern University
Department of Math and Computer Science
Landrum Box 8093
Statesboro, GA 30460
FAX (912) 681-0654.

ADDITIONAL SOURCES

A BIO-Sim demo is available on the web at http://www.gsu.cs.gasou.edu/faculty/ghuband/research/habitat_power_planning.htm. Your browser will automatically download the latest version of Java needed to run the demo.

For more information on Java you can go to the Sun java site at http://www.java.sun.com.

DEFINITIONS, ACRONYMS

Definitions

D: set of all devices.
d: a single device.
a: a single device activity.
A: set of all activities for a device.
s: device activity start time (minutes).
e: device activity end time (minutes).
S: set of all device activity start times.
E: set of all device activity end times.
q: a discrete time event (s or e).
Q: an ordered set of discrete times.
t: time (minutes).
T: total simulation time (minutes).

Acronyms

API: Application Programming Interface.
GUI: Graphical User Interface.
JDBC: Java Database Connectivity.
RMI: Remote Method Invocation **.**
SQL: Structured Query Language **.**
UML: Uniform Modeling Language.
.