Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

A JAVA CLASS LIBRARY FOR SIMULATING PERI-OPERATIVE PROCESSES

Philip M. Troy

Les Entreprises TROYWARE
5755 Eldridge
Cote Saint-Luc, QC H4W 2E3 CANADA

Lawrence Rosenberg

Department Of Surgical Services
The Sir Mortimer B. Davis Jewish General Hospital
Montreal, QC H3T 1E2 CANADA

ABSTRACT

To address the mismatch between supply and demand for peri-operative capacity, senior management of the Department of Surgical Services at the Sir Mortimer B. Davis Jewish General Hospital in Montreal designed and developed a discrete event simulation platform for modeling its peri-operative processes. Design goals included ensuring that the platform could be used for both long-term capacity planning and short-term scheduling, that it could be readily modified and extended, that it be comprehensive and fast, that it have a multi-level 2D animation capability, that it reuse software components, and that it could be embedded into other software. The primary outcomes achieved were the development of a Java class library to support the development of peri-operative process simulation models, and a preliminary model, built with the class library, that is currently being used to help understand the need for surgical beds.

1 INTRODUCTION

The Jewish General Hospital Center provides a broad range of inpatient and outpatient services to a large and diverse population in and around Montreal. It has 637 beds, including 154 surgical beds and a combined Medical-Surgical ICU (Intensive Care Unit) of 20 beds, of which 16 to 18 are usually staffed. The department of surgical services performs approximately 15,000 operative procedures per year, of which approximately 40% lead to overnight patient stays in the hospital. There has been a steady increase of about 2% per year in the volume of these procedures, and this increase is expected to continue until at least 2015. The institution is a full service medical center having a large cancer center, and tertiary and quaternary programs that include cardiac surgery and neurosurgery, but not major trauma or transplantation.

A consequence of the fixed capacity of the hospital, and the random nature of the demands placed on that capacity, is that it has not always been possible to find beds in the PACU (Post Anesthesia Care Unit), ICU (Intensive Care Unit) and surgical wards when needed. This in turn has lead to cancellations of operative procedures and delayed care for patients. In particular, in addition to occasional days in which operative procedures needed to be canceled because of a lack of appropriate beds, in the last year and a half there was an extended period in which all operative procedures requiring ICU stays were canceled because of the unavailability of beds in the ICU. There was also a period of time in which an unusually large number of operative procedures were canceled because of a lack of available beds on the surgical wards.

To gain insight into the mismatch between supply and demand for operative capacity, to provide bed utilization forecasting tools for schedulers, and to make it possible to investigate alternative approaches to address these problems, a decision was taken to design, develop and deploy a detailed and comprehensive simulation model of the hospital's peri-operative processes. This decision was made based on the understanding that there are many interactions, most of them involving randomness, that affect peri-operative throughput, and that it was important to have an analytic capability that reflected this reality. It was expected that this model would be used to develop a better understanding of these interactions, to predict short term bed insufficiencies, to improve short-term scheduling, and to support long-term capacity decision- making. It was also expected that in addition to using this model for specific individual analyses, it would be embedded into decision support tools to be used by hospital staff.

There has been a considerable amount of published research on the application of simulation to analyze and improve surgical processes. This research appears to have been championed first by Fetter and Thompson (1965), who performed a simulation study intended to evaluate the effect of surgery scheduling policy changes. A more recent study Carter (2006) reported the application of simulation to surgical processes to determine the number of beds required in the ICU and in the surgical wards in context of expanding operating room activity. In order to determine management guidelines for evaluating the utilization of an operating room suite, Tyler, Pasquariello and Chen (2003) used simulation to investigate the optimal utilization for those suites when the first procedure of each room starts within fifteen minutes of the start of the day, and the last procedure ends at most fifteen minutes after the end of the day. While trying to determine the best approach to sequencing operative procedures, e.g. to determine whether short procedures should be performed before longer procedures, Marcon and Dexter (2006) examine the impact of surgical sequencing on PACU staffing. Faced with numerous cancellations of operative procedures requiring ICU stays because of a lack of available ICU beds, Troy and Rosenberg (2009) used simulation to determine the number of ICU beds needed for surgery patients. Cahil and Render (1999) used simulation to determine the need for ICU bed needs for a Veterans Administration hospital in the context of a rapidly expanding growth of medical and neurological patients. Of particular note is their finding that "increased ICU bed availability resulted in increased telemetry and medical floor bed utilization downstream and increased length of stay on the medical service as the proportion of post-ICU patients increased on the floors." This unexpected finding illustrates the interconnectedness of peri-operative processes, and the need for analytical capabilities that include all peri-operative processes. In an effort to "describe the entire trajectory that patients have to fulfill," Cardoen and Demeulemeester (2007) investigated the use of "clinical pathways to structure" simulation models.

To build a simulation platform that could be used to address our hospital's needs, design constraints and guidelines were specified, design decisions were made, and a Java class library was developed and used to build a preliminary model. The purpose of this paper is to discuss these efforts and their results.

2 DESIGN CONSTRAINTS & GUIDELINES

2.1 Discrete Event Simulation

To meet these analytical needs, it is necessary to use a tool that could address the randomness inherent to peri-operative processes. Queuing theory, the approach preferred for such analysis by many Operations Management analysts, was not, in the authors' opinions, appropriate for the analyses described above. This is because the queuing models that are most tractable when dealing with sequences of processes, i.e. operative intervention, followed by a stay in the ICU or PACU, followed by a stay in a surgical ward, possibly followed by another stay in the ICU, require that the underlying stochastic processes exhibit characteristics not typically found in these peri-operative processes. Thus it was decided to build a discrete event simulation of the processes, as this approach can be used in much more general situations, though it tends to require considerably more processing capability than does queuing theory.

2.2 Performance

Because of this need for more processing capability, and because the platform was planned to be used frequently for scheduling peri-operative resources, it needed to run quickly, and/or be able to be run in parallel on multiple computers. This is because evaluation of an individual test schedule generally requires simulation experiments consisting of many simulation trials, and because simulation based scheduling generally involves experimenting with a large number of individual schedules. Thus if the scheduling process were to experimentally test 10,000 schedules, and each experiment were to consist of 100 trials, and each trial were to take 36 seconds, and only one processor was used, it would take 10,000 hours to find the best schedule. If however each trial were to take .36 seconds and one processor was used, it would take 100 hours to find the best schedule, and if each trial were to take .36 seconds and 100 processors could be used in parallel, it would take 1 hour to find the best schedule.

2.3 2D Multi-Level Animation Engine

While animation is not needed to obtain useful simulation results and tends to significantly slow down the computation of those results, animation is generally needed to make it possible for hospital administrators and staff to understand complex simulation models, to ensure that the models are valid, and to get buy in for results obtained from the models. Thus any simulation development capability that was used needed to include an animation engine.

A decision was made by management that the animation capability be geographically based, i.e. that it present physical movements around the hospital, rather than animate the steps of a flow chart. The reason for this was that management felt that geographic animation would make it easier for staff to understand the model.

In addition, while the animation only needed to be 2-dimensional, it was necessary that users of the simulation be able to limit the animation to specific areas of the hospital, and to be able to interactively vary the areas being displayed while the simulation was running. Applying this requirement to the hospital's multiple levels required that users also be able to interactively change the floor of the hospital being animated at a particular time.

2.4 Component Reuse

An important goal of modern software development is to maximize the reuse of software components; more recently this has also become a goal of simulation model development. Reuse can be particularly helpful when building simulation models involving hierarchies of staff objects such as doctors, nurses, respiratory therapists, orderlies, unit agents; reuse can also be particularly helpful when building hierarchies of physical objects such as operating rooms, surgical wards, and beds. Thus an important design guideline for the project was to facilitate component reuse.

2.5 Design Methodology

While it may seem unnecessary to use a design methodology for building discrete event simulation models, the authors' had previously found that when using simulation software in which models were built by drawing them, as simulation model complexity increased, it often became harder to maintain and expand the models. One reason for this was that when building simulations using drawing tools, when editing the simulation, it often becomes necessary to move around other visual objects in the simulation model to make room for new objects; it can also become necessary and increasingly more difficult to move around the lines connecting simulation objects so as to make it possible to follow them and differentiate between them. Even when not using drawing tools to build simulation models, as models built without a methodology grow, the likelihood of difficulty in modifying or expanding them seems likely to increase.

Thus to facilitate the development of detailed and comprehensive models, and hence complex models, it was decided to select a design methodology before beginning platform design and development. The particular goals for the design methodology were that it be simple, efficient, straightforward to translate designs into implementation, expandable and easily validated, and that it lend itself to component reuse.

3 DESIGN DECISIONS

To address these constraints and guidelines, several design decisions, discussed in the sub-sections below, were made.

3.1 Design Methodology

As a starting point for specifying a design methodology, Leung and Lai's (1997) methodology, based on Yourdon's Structured Methodology (YSM), was considered for analyzing simulation modeling requirements. Their methodology consists of three stages. The system definition stage consists of building a diagram that specifies the boundaries of the system by specifying the entities in the system, and building a list of external and internal events that trigger activities in the system. Their modeling formulation stage consists of building a state transition diagram for each entity type that indicates the effect of events on the state of individual entities, building a data flow diagram to specify the activities to occur in response to specific events, and building an entity relationship diagram to specify the attributes of each entity, and the relationships between entities, including relationships because of data, activities and events. Their validation stage consists of using the above diagrams to validate the analysis

A limitation of the Leung and Lai methodology, from the perspective of making it as simple, direct, and fast as possible to build simulation models, is their emphasis on the difference between determining what needs to go into the model, and how the model is built, i.e. "A key principle for YSM is to model the functional requirements and the implementation requirements separately." In particular, by not using the same approach for both analyzing and building the simulation model, their methodology requires that one map the first set of requirements into a second, possibly very different set of requirements. It also becomes necessary to perform two steps, design and building.

To bypass these limitations, the discrete event simulation model for the hospital was conceptualized in terms of programming language based objects used in several Java based discrete event simulation tools such as SSJ (L'Ecuyer 2009). In particular, it was decided that entities would be built using Java objects hereafter referred to as entity objects, events would be built as Java objects hereafter referred to as event objects that are scheduled to occur in the future, and the activities to occur in response to those events objects would be programmed as procedures, or methods, owned by the entity objects. This object orientation would make it possible to adapt most of Leung and Lai's steps so that they could be simultaneously used to analyze, build, and document the simulation model. The initial set of steps that resulted from this adaptation were:

- Specify the the system's entities and build them as entity objects.
- Specify the attributes of those entities and build them as fields of the entity objects.
- Specify the system's events and build them as event objects.
- Specify the activities to be performed by entities in response to those events and build them as methods of the entity objects.

To facilitate the validation of models built with this methodology, the following steps were added:

- Specify the states each entity object can take on.
- Initialize the state of each entity object upon its creation.
- Modify the state of each entity object whenever that state is modified in response to an event.
- Before performing an activity, have entity objects check whether they are in an acceptable state for performing that activity, and if not, log the simulation time, the event, the entity object, and the entity objects' current state, and then stop the simulation.
- Add a logging capability that could be turned on or off as needed to log all state changes, the activities that performed them, and the events that triggered those activities.

It is interesting to note that the derived set of steps are similar to those suggested in Gilbert (2004) as a methodology for building agent based social simulations, which tend to have a lot in common with discrete event simulations.

3.2 Simulation Environment

To address the need for performance, it was deemed necessary to use an enterprise capable development environment that would run simulation trials quickly, that could be used to distribute the simulation so that individual trials could be run in parallel on different computers, and that would not require expensive licensing for that distribution. Likewise, to address the need for component reuse, it was deemed necessary to use an object oriented development environment. These needs suggested the use of Java, whose performance now comes close to that of compiled object oriented programming languages; Java applications can also be easily distributed due to the existence of Java engines for a large number of modern operating systems and processors, and also because Java applets can be distributed via web pages in a manner that can securely take advantage of unused personal computer processing power.

To preclude the need for developing core simulation capabilities such as the pseudo-random number generator needed for repeated random sampling, or the event list needed to track future simulation events, it was originally planned to use a publicly available Java simulation class library such as SSJ (L'Ecuyer et al 2009), a simulation class library developed by Pierre L'Ecuyer and his team. However, since L'Ecuyer's random number generator had been translated into Fortran by Miller (2001) and since Sun had publicly published the code for its TreeMap object which could be readily adapted for use to track future events, it was decided to take advantage of them so as to preclude possible licensing problems. This was also done to optimize the implementation of the TreeMap object, which (L'Ecuyer et al 2009) "was not efficient." It was relatively straightforward to add code for the main simulation event loop.

To maximize performance of the core simulation capabilities, a few changes were made to the code described above. Perhaps the most important of these was that simulation time was defined to be an integer between 0 and 2^31 - 1 minutes, rather than the floating point simulation time used by SST, or the Object field used as an index in Sun's TreeMap class. In addition, the main simulation event loop was designed so that it runs in a thread, so that multiple simulation runs could potentially be run simultaneously on multi-core processors such as the eight core processors expected to be available within the next two years. The simulation engine was also designed to support, in future versions, the distribution of simulation processing to other personal computers in the hospital via web pages.

3.3 2D Multi-Level Animation Engine

As SSJ does not have an animation capability, and as no Java toolkits were found to have an animation ability that could be set interactively to change the floor being animated, it was decided to build the animation capability. To help obtain user acceptance of the simulation model, it was decided to provide users with an interactive ability to specify the ratio between simulation time and real time. To ensure that the animation engine would have minimal impact upon performance when animation was turned off, it was also decided to integrate calls to the animation engine into the simulation event loop via a single if statement that would check whether the simulation time of the next animation update was less than the simulation event time of the next event. When this was not the case the simulation would proceed, and when it was the case the simulation would be delayed until the appropriate real time for the next animation update.

To make it possible for users to view different areas of the hospital, the simulation frame displaying the animation was designed to allow users to click on buttons, painted on the frames, to scroll the animation forward, backward, left and right, and to move the animation up or down one floor at a time.

3.4 Simulation Class Library

While many of the base classes of the simulation class library were designed to be similar to that of other Java event oriented simulation libraries, the class library was also designed to include base classes that could be used for logging and animation.

To model entities, an Entity interface was designed to provide the basis for a hierarchy of general simulation entity objects that need to respond to simulation events such as the simulation is about to start, the simulation has ended, the simulation is being reset, and the simulation's warm-up period has ended. Numerous implementations of this interface were made to represent hospital entities such as beds, patients, doctors and operating rooms

To model those and other events, an Event interface was designed, and numerous event objects were built to represent events such as patients calling a surgeon's office for an appointment, patients arriving for the appointment, receptionists leaving for work and surgeons finishing operative procedures. Of particular note is the simplicity of the event classes, whose design was motivated by SSJ's (L'Ecuyer et al 2009) Event class. When events are created they are initialized with needed references and values; when events occur their actions method is invoked and in general a single method call to one of the reference objects is made using the other references and values. Thus the purpose of the event class is to trigger activities; these classes generally leave activity details to entity objects.

4 USAGE

This section discusses the efficacy of using the class library, after implementing it as described above, to build a preliminary peri-operative simulation model currently being used to evaluate surgical bed requirements at the hospital. This model entails surgical requests being generated randomly by all of the hospital's surgeons at historical rates, operative procedures taking random lengths of time based on actual procedures performed by the same surgeons, and patient post-operative bed stay durations randomly generated using bed stay data collected for those operative procedures from all surgeons at the hospital. The model was run for four 28 day periods to generate a backlog of surgical requests, for an additional thirteen periods to bring the simulation to a state closer to steady state, and for twenty two data collection periods. While the model currently only collects the maximum number of beds utilized, it will soon be modified to collect the amount of time 1, 2, 3, . . . 200 beds are needed per year, both overall, and by individual surgical services. Subsequent to that it will also be used to investigate whether scheduling changes can be used to increase peri-operative throughput.

4.1 Design Methodology

As anticipated, the design methodology simplified and speeded up the building of the preliminary model. In particular, the first step of the methodology resulted in classes for each of the entities being modeled (see Table 1); the major effort entailed by this step was just to list the set of entities. The second step resulted in the addition of a set of attributes for each of those entities (see Table 2 for a list of the bed manager entity's attributes); the major effort entailed by this step was just to list the attributes. The third step resulted in a set of event classes (see Table 1); the major effort entailed in this step was just to list the events being modeled, or more precisely the list of happenings corresponding to the end of each activity in the model. The fourth step resulted in the specification of the activities for each entity class (see Table 2 for a list of the bed manager's activities); this step required both the straight forward identification of the activities, as well as the identification, by a simulation analyst, of how these activities were to be translated into Java. Of particular note is that it the design and development steps were and could be done simultaneously.

Table 1: Entities and Events in Preliminary Peri-Operative Simulation Model

Entities	Events
Bed	Bed Cleaning Finished
Bed Manager	Doctor Arrives At Office
Bed Request	Doctor Arrives Home
Bed Unit	Doctor Time To Leave Home
Doctor	Doctor Time To Leave Office
Exam Room	Patient Admitted To Bed
Major OR Suite	Patient Arrival At Doctor Office
Medical Office	Patient Arrival At OR For Surgery
Medical Office Receptionist	Patient Arrives Home
Minor OR Suite	Patient Bed Stay Finished
One Day Surgery	Patient Discharged From Bed
Operating Room	Patient Generation
Post Anesthesia Care Unit	Patient Registration At Doctor Office Completion
Patient	Patient Time To Go To Doctor Office For Exam
Patient Generator	Patient Time To Go To OR For Surgery
Surgeon	Surgeon Arrives At OR
Surgery Request	Surgeon Surgery Completion
Surgical Ward	Surgeon Time To Leave Home For OR
Unit Agent	Surgeon Time To Leave OR
	Unit Agent Arrives At Home
	Unit Agent Arrives At Unit
	Unit Agent End Of Work Day
	Unit Agent Time To Leave Home

The design methodology also systematized and speeded up the process of modifying the model. In particular, to modify the model, it was only necessary to revise the results of the steps discussed above so that they were consistent in context of the modified model. This is in contrast to the modification of visually created models where it becomes necessary to move the constructs used by those models, and the inter-connections between those constructs, so as to conform with the modified model.

Table 2: Bed Manager Entity Attributes and Activities

Attributes	Activities
Bed Types	Assign Patients To Beds
Bed Requests	Be Notified That Bed Has Been Cleaned
Available Beds	Be Notified That Patient Stay Has Finished
All Beds	Be Notified That Patient Has Been Admitted To Bed
	Be Notified That Patient Has Been Discharged
	Order Bed To Be Cleaned
	Order Patient Discharge
	Order Patient To Be Moved And Admitted To Bed
	Receive Bed Request

4.2 Animation Engine

As illustrated in Figure 1, the animation capability made it possible to see the geographical flow of entities through the hospital. In that figure, orange balls surrounding a "P" are used to represent patients and blue balls are used to represent surgeons; in other parts of the animation other colors are used to represent unit agents, nurses, respiratory therapists and orderlies. In addition, the black triangles, plus sign, and minus sign icons in the bottom right hand corner of the display can be used to scroll the display left, right, forward, backward, up and down instantaneously, even when incorporating architectural plans as the background of the display.

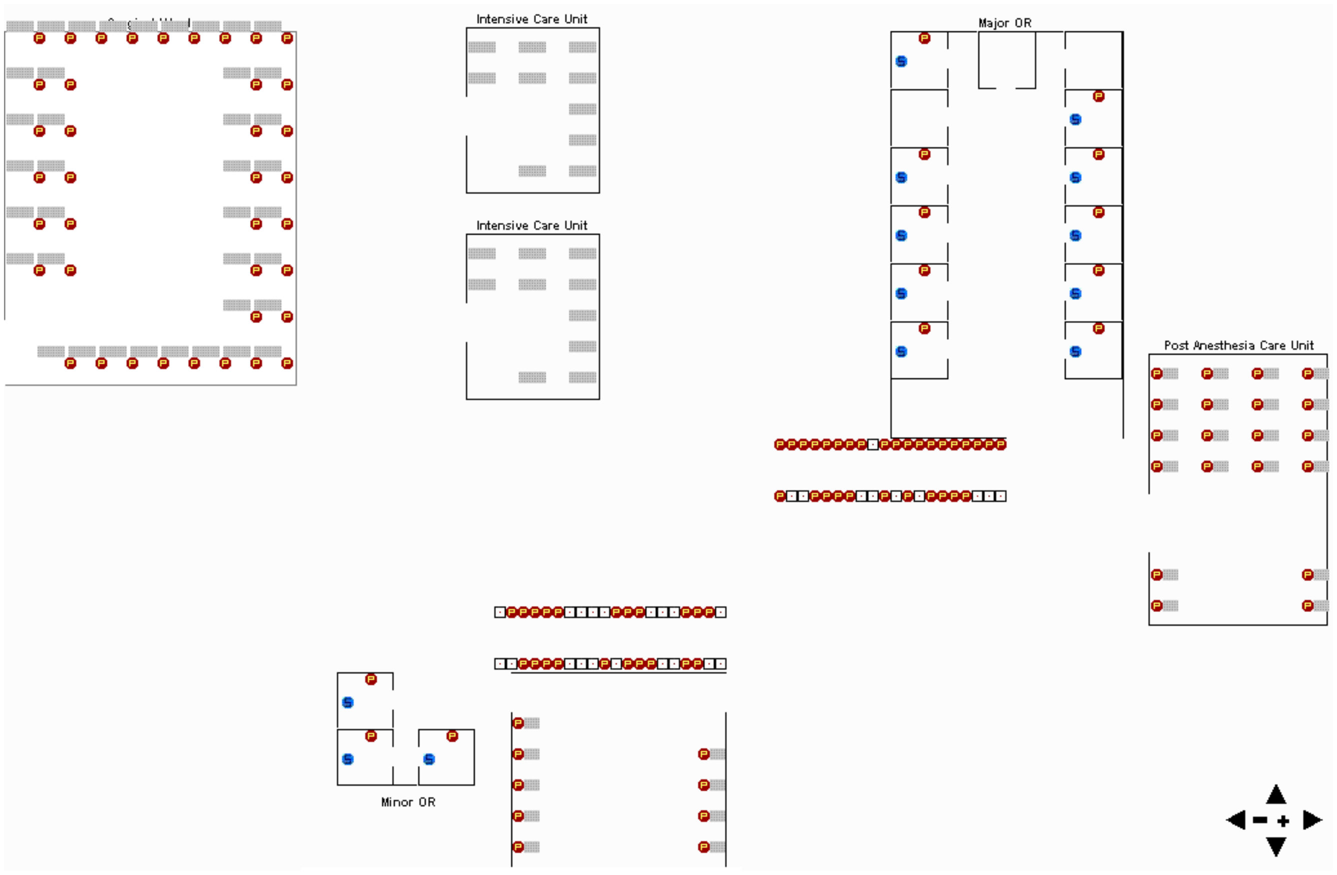

Figure 1 – Geographical animation of OR suites, PACU and a surgical ward

4.3 Performance

The class library is fast. In particular, for a preliminary peri-operative simulation model just involving surgeons and patients, when the animation speed was set to 50 frames per second and one day simulation time per frame, 1092 days (i.e. 39 four-week periods) of simulation were processed and animated in approximately 22 seconds, indicating that both the simulation and the animation were able to perform at the targeted speeds. These timings were performed on a ThinkPad T61 running Windows XP Professional with Service Pack 2, 2 GBytes of RAM, and a T9300 dual core processor. In addition, for the same model on the same computer, when animation was turned off, the total run time, including model creation and initialization was approximately two seconds; it is anticipated that when running experiments with multiple trials that the time for a single trial of 1092 days of simulations will be less than 1 second.

5 OTHER POTENTIAL USES

It is our firm belief that given the resource constraints that are pervasive in the provision of health care services, the mismatch, either short-term or long-term, between supply of services and the demand for services, is a fundamental issue that must be addressed. It is also our contention that the class library described in this paper can be readily used by other departments and hospitals to analyze alternative approaches to addressing this mismatch. As such, it is tentatively planned to release the class library as implemented to date at no charge for stand-alone usage.

6 FUTURE ENHANCEMENTS

While currently of considerable value to the hospital, to facilitate the further design, building and documentation of simulations, several enhancements are being considered.

6.1 Design Methodology

To facilitate future use of the class library described in this paper, the authors envision:

- The addition of a user interface to the Java IDE (Eclipse) to allow developers to directly enter the specifications for entity objects, i.e. their parent class, their additional attributes, changes and additions to their activities, their states, and the states allowed for each activity.
- The addition of a user interface to the Java IDE to allow developers to directly enter the specifications for event objects, i.e. their name, their parameters, and the actions to be performed when they occur.
- Extensions to the Java's built in documentation capability (JavaDoc) to make it possible to generate a document containing a hierarchical list of entity objects with their attributes, states, and activities.
- Extensions to the Java's built in documentation capability to make it possible to generate a document containing a list of event objects with their names, parameters and the actions to be performed when they occur.

6.2 Class Library

It is expected that as more detail is added to the peri-operative simulation model, and as the class library is used for other simulations within the hospital, that the class library will be expanded to facilitate those additions.

6.3 Animation Engine

As the use of the class library is extended, it is expected that there will be a need to display the animation of several parts of the hospital at one time. In turn it is expected that this may lead to:

- The development of a capability to simultaneously display animation of several parts of the hospital, one per computer monitor.
- The extension of the current capability to allow zooming in and out within a particular animation display.
- The extension of both of these capabilities so that they work over the internet.

To facilitate getting staff to accept models, it would also seem useful to provide an ability to display or verbally announce the events, activities and state changes as they occur for selected objects.

6.4 Performance

While a significant amount of effort has already been expended to ensure that the class library is fast, there are a few capabilities, including the future event list and surgeon schedules, whose speed could potentially be improved. However, it seems likely that most future performance efforts will be on the extension of the class library to enable it to distribute the processing of individual simulation trials, so that simulation experiments can be run in approximately the same time it takes to perform a single simulation trial.

REFERENCES

Bloch, J., D. Lea, Available via <http://www.docjar.com/html/api/java/util/TreeMap.java.html> [accessed August 11, 2009].

Cahill, W., M. Render. 1999. Dynamic Simulation Modeling Of ICU Bed Availability. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, 1573-1576. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Cardoen, B., E. Demeulemeester. 2007. Evaluating The Capacity Of Clinical Pathways Through Discrete Event Simulation. Working Paper KBI 0712, Department Of Decision Sciences and Information Management, Faculty Of Economics and Applied Economics, Katholieke Universiteit Leuven, Leuven, Belgium.

Carter, M. 2006. A Case Study of a Simulation-Based Decision Support Tool. Available via <http://www.ioz.pwr.wroc.pl/Pracownicy/Zabawa/ORAHS1/pliki/Carter_ORAHS.ppt> [accessed August 11, 2009].

DiLeo, J. J. 2005. FreeSML: Delivering On The Open Source Simulation Language Promise. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2513-2522. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fetter, R.B., J.D. Thomson. 1965. The Simulation of Hospital Systems. Operations Research 13(5):689-711.

Gilbert, N. 2004. Agent-based social simulation: dealing with complexity. Available via <http://cress.soc.surrey.ac.uk/web/resources/ABSS%20-%20dealing%20with%20complexity-1-1.pdf> [accessed August 12, 2009].

L'Ecuyer, P. 2009. Pseudorandom Number Generators. In *Encyclopedia of Quantitative Finance*, ed. E. Platen and P. Jaeckel. Hoboken, New Jersey: Wiley.

L'Ecuyer, P., et al. 2009. SSJ User's Guide, Package SimEvents, Simulation Clock and Event List Management. Available via <http://www.iro.umontreal.ca/~simardr/ssj/doc/pdf/guidesimevents.pdf> [accessed August 12, 2009].

Leung, J. W. K., K. K. Lai. 1997. A Structured Methodology To Build Discrete-Event Simulation Models. Asia-Pacific Journal of Operational Research 14:19-37.

Marcon, E., F. Dexter 2006. Impact Of Surgical Sequencing On Post Anesthesia Care Unit Staffing. Health Care Management Science 9:87-98.

Miller, A. 2001. MODULE Lin_Feedback_Shift_Reg. Available via <http://users.bigpond.net.au/amiller/random/lfsr258.f90> [accessed August 12, 2009].

Troy, P. M., L. Rosenberg 2009. Using simulation to determine the need for ICU beds for surgery patients. Surgery(in press).

Tyler, D. C., C. A. Pasquariello, C. H. Chen 2003. Determining Optimum Operating Room Utilization, Anesth Analg 96:1114-21.

AUTHOR BIOGRAPHIES

PHILIP TROY is the principal consultant for Les Entreprises TROYWARE, a consulting and software development corporation focusing on health care Operations Management. He received his Ph.D. In Operations Research from Yale University in 1992. He is also a part time faculty member of the John Molson School of Business at Concordia University.

LAWRENCE ROSENBERG is a Professor of Surgery and Medicine and Director of the Division of Surgical Research at McGill University, A.G. Thompson Chair of Surgical Research at the McGill University Health Center, and Chief of Surgical Services at the Sir Mortimer B. Davis Jewish General Hospital. He obtained his M.D.,C.M. From McGill University in 1979, trained in General Surgery at McGill, completed his surgical residency and Ph.D. in Experimental Surgery in the area of pancreatic cell differentiation in 1985, and undertook post-doctoral studies and fellowship training in transplantation surgery at the University of Michigan.