A Programmed Solution to ARGESIM Comparison C 6 'Emergency Department' with DSOL, a Java- based Suite

Roland Lezuo, Felix Breitenecker, Vienna Univ. of Technology, Austria; *fbreiten@osiris.tuwien.ac.at*

Simulator: *DSOL* is an open source, Java based suite for continuous and discrete event simulation. *DSOL* is written with distributed computing in mind and supports distributed models. Models are implemented in Java code. *DSOL* offers facilities for graphical data evaluation like charts but also supports 2D and 3D animation. There are predefined classes for standard entities like generators, stations and resources. There is also a big library of statistic classes, including distributions and tallies which compute mean and standard deviation automatically.

Model / Implementation of Strategies for tasks. Model and experiments (tasks) of C6 are implemented using one class for each station (*registration*, *casualty ward*, *x-ray* and *plaster*). There is just one casualty ward object hiding the details of two wards with two doctors each. Additionally a patient source and a patient sink are used, primary for evaluation purposes and implementation details. The patients themselves are a class also and know the details about their path through the hospital and their priority for task c. The glue between all the classes is DSOL's event mechanism.

The patient source schedules a `goto next station` event for newly created patients and re-schedules the `create patient event` as long as needed.
The patient schedules an arrival event on his next station and gets queued there. As soon as the station treated the patient it will create an `goto next station` event for the patient and reschedule itself as long as there are patients queueing. The patient sink terminates this mechanism and stops the simulation when all patients left the hospital.

The method `getNextStation` returns the next target of the patient. In that way only the patient has to know about his way through the hospital, so new types of patient could be added easily.

Each station implements a `queuePatient` method, basically re-scheduling itself: in pseudo code:

```
void queuePatient(Patient p)      {
  if (queue.isEmpty())  {
   queue.add(p);
    scheduleProcessing(drawStationDelay());
 } else{
    queue.add(p);    } }
```

The last missing puzzle piece is the implementation of `scheduleProcessing` which draws a random delay and after that delay calls a method named `processing`. This method in pseudo code:

```
void processing()                 {
  p = queue.pop();
  p.notify();
  if (!queue.isEmpty())       {
   scheduleProcesing(drawStationDelay());}}
```

It basically takes the first patient from the queue and puts him to the next station. If there are still patients left it is re-schedules itself. The patient sink finally stops the mechanism by implementing a method, which writes statistically data to the experiment database (to be examined by the *DSOL GUI*).

Tasks a -c: Different strategies for operation. As special actions (change of doctors) and priorities are programmed directly in the method definitions, no further effort is necessary for complete the more complex tasks b and c.

Results for 50 simulation runs are given in Table 1 below. In general, results from task b are almost identically to task a, so swapping doctors seems to have no effect. Task c shows improvements of the priority strategy: although the overall treatment time for all patients did not change, the overall treatment time for each patient is reduced.

Type	sm[a]	sd[a]	sm[b]	sd[b]	sm[c]	sd[c]
1	205	12.47	215	13.16	169	14.97
2	98	14.13	99	12.63	109	15.68
3	286	22.22	287	17.52	165	21.63
4	98	4.81	106	5.97	122	6.18
all	171	13.41	176	12.32	141	14.62

Table 1: Mean and standard deviation for treatment times.

Classification: Programmed DEVS library-based approach.

Corresponding Author: Felix Breitenecker
Roland Lezuo, Felix Breitenecker, Vienna Univ. of Technology, Inst. f. Analysis and Scientific Computation, Wiedner Hauptstrasse 8-10, 1040 Vienna, Austria;
fbreiten@osiris.tuwien.ac.at

Received: October 24, 2005
Revised: January 23, 2006
Accepted: March 30, 2006