MOSS - Micah os animated-simulator  
  
Proposed logic  
  
   
MOSS Architecture  
 MOSS is written flexibly, to symbolize simulated os software/partial hardware mechanisms, comprising   
of a SIMULATED PROCESS ENVIRONMENT. (SPE)   
  
The simulated process environment consists of a number of SIMULATED PROCESS ENVIRONMENT REGIONS (SPR’s), SIMULATED PROCESS ENVIRONMENT REGION CONNECTORS (SPRC’s), and SIMULATED TASK SCHEDULER (STS), which consists of an object collection of SIMULATED TASKS (ST’s).  
  
  
  
  
  
  
  
  
  
  
The SPE represents the total picture, and consists of all the mentioned segments to symbolize the simulation in totality.  
The SPR’s represent physical graphical regions that a task may be admitted, dispatched, ioeventcompleted, ioeventwaited, interrupted, or terminated into. An example of SPR is any one of the regions in the process flow; (new, ready, running, waiting, terminated)  
  
The SPRC’s represents the PHYSICAL connections or VIRTUAL task manipulation routes which connect the SPR’s together.( admitting, dispatching, ioeventcompleting, ioeventwaiting, interrupting, and terminating)  
  
The STS apart of the SPE uses its access to the SPE’s, uses SPR’s and SPRC’s to direct the ST’s along the SPRC’s, and into SPR’s, since SPR’s and SPRC’s are declared publicly accessible.   
  
Modules  
MOSS’ has a microkernel based simulated **os architecture**. Each SPR or simulated process region, acts as modulated servers, while each SPRC acts as a route along which messages pass, to accomplish task executions per simulated cycle.  
  
**Threading** is represented as exploration indices. In MOSS, ST’s, move from region to region (SPR to SPR or server to server), and along the various task routes (SPRC’s or message passage routes) through the use of different sets of these exploration indices. Once one set of exploration indices are consumed, another set is called upon per simulated task per new process step, to allow continued traversal throughout the process.  
  
**Kernel process handling** is insurgent in the STS. **Processing** wise, each process is seen in SPR’s, connected by SPRC’s whereby both SPR’s and SPRC’s are packed with utilities to reveal the different contexts which a ST might incur throughout the cycle.  
  
**Virtual memory and IO** is simulated throughout the different regions.  
  
  
  
  
Nice features:  
  
1. user may add simulated tasks at runtime.  
  
2. MOSS represents multi-core cpu software control (MCSC), where the user may toggle between 4 simulated cpu step sizes. (1, 2, 4, and 8 core simulation)   
  
3. Status reporting and deadlock symbolization. The status report bars show the following:  
a. MOSS is realistic, possessing deadlock symbolization. Through the use of MCSC, when the user toggles single core simulation, and loads multiple tasks, not only is execution delayed by some time, per task, BUT also AND more importantly, SOMETIMES deadlocks (process execution schedule collisions) may occur.  
  
b. Each simulated tasks possesses a gui label, which reports what is happening to it per process step. As the simulated task scheduler comes packed with a gui panel that collects each simulated task, the user gets a status report atop the MOSS ui layout, displaying these.   
  
c. Number of active simulated cores display.