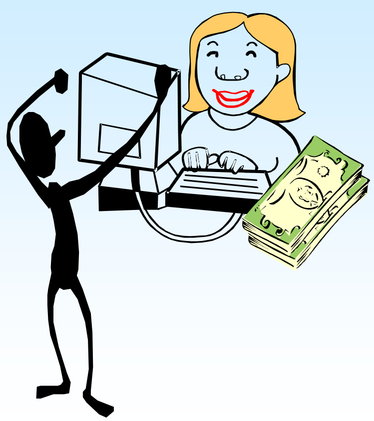
An Introduction to Simulation

Simulation Modelling

- Constructing a dynamic model of a given system is called simulation modelling.
- The function of the model, called a *simulator*, is to mimic the behaviour of the system within the limitations of the system description.
  - Give some examples of simulations we see around us

A Bank Simulation

The current *state* of the *system*: A teller lady with access to lots of money; a man who wants to convince the teller that he should get some money from his mate's account; a lady patiently waiting in the queue; and a girl who's walking in to join the queue.



A Bank Simulation

- How much time does a customer spend in the bank on average?
- What percentage of customers wait for service?
- What is the average waiting time per customer?
- What is the average waiting time of those customers that wait?

A Bank Simulation

- What is percentage of time the teller idles?
- What performance difference would we see if we have two tellers?

A Bank Simulation

Customer	Arrival time	Service time	Time service begins	Time service ends	Time in system	Idle time	Time in queue
1	0	3	0	3	3	-	0
2	8	2	8	10	2	5	0
3	11	2	11	13	2	1	0
4	15	6	15	21	6	2	0
5	17	4	21	25	8	0	4
6	23	9	25	34	11	0	2
7	30	1	34	35	5	0	4
8	38	9	38	47	9	3	0
9	39	5	47	52	13	0	8
10	42	1	52	53	11	0	10

Input data

Why Simulate?

- A real system may not be there
  - E.g., A new processor design at its conception
- A real system may be too difficult or too expensive to access
  - E.g., a nuclear reactor, cockpit, etc.
- A real system may not be perturbed
  - E.g., an airline reservation system, because of potential loss of revenue if system goes down due to perturbations

Simulation Modelling

- How do we get the input data?
  - Measurement
  - Random numbers
- How do we generate the output, given the input?
  - By hand
  - By a computer programme (the *simulator*)
- What insight do we get from the output data?
  - What are the performance figures we are looking for?

Simulation Modelling

- A system consists of several physical entities, or components.
- At any given time, each of these entities has state information associated with it.
  - For instance, a server might have two states: *busy* and *idle*.
- Ideally, the state of the simulator at a given simulation time should correspond to the state of the system at the corresponding real time.

Simulation Modelling

- The change of state is called an *event*.
- An event triggers an *activity* - a unit of work - in the simulator.
  - An activity will typically cause the creation of further events.
- A logically-related set of activities constitutes a *process*.
- As the simulation proceeds, the simulation time advances in steps, depicting the changes in states and mimicking the corresponding activities.

Endogenous and Exogenous Events

- Events internal to a system are called *endogenous* events; events external to the system are *exogenous* events.
  - A customer arrival event in the bank simulation is an exogenous event.
  - A teller acquisition event is an endogenous event.

Time-based Simulators

- In a *time-based* or *time-driven* simulator, the time steps are regular, that is, the interval between any two successive time steps stays constant.
  - If the time interval is too large, the simulator might miss some state changes.
  - On the other hand, if the time interval is too small, the simulator would waste time advancing through time steps during which there are no state changes.
  - Thus, in general, a time-based simulator lacks either accuracy or efficiency, or both.

Time-based Simulators

```
int gclock = 0;

for ( ; ; ) {     // repeat forever
        if ( eventsExistAt(gclock) ) {
                // do what's required for the time step
                processEventsAt(gclock);
        }
        ++gclock;
}
```

Using global variables to represent global entities is perfectly OK. Time, for example, is a global entity.

Event-based Simulators

- *Event-based* simulators advance the simulation time only to those points where there are state changes.
  - Consequently, the time steps here are irregular.
- These simulators maintain an event list that is a diary of all unprocessed events.
- The simulation proceeds by removing from the list the event with the earliest time and modelling the corresponding activities.

Event-based Simulators

```
Event e = EventManager.NextEvent();

while ( e != null )
{
        switch ( e.Type )
        {
                // process each event, possibly generating more
                …
        }
         e = EventManager.NextEvent();
}
```

Bank Simulation: Event-based

```
const int MAX_CUSTOMERS = 10;
Random arrivalGenerator = new Random();
const int MAX_INTERARRIVAL = 20;

const int MAX_TELLERS = 1;
Random tellerConsumptionGenerator = new Random();
const int MAX_TELLER_CONSUMPTION = 10;
```

Bank Simulation: Event-based

```
Resource teller = new Resource(MAX_TELLERS); // Modelled as resource
Entity god = new Entity("God");

Event adamsArrival
        = new LocalEvent(EventType.CUSTOMER_ARRIVAL, god);
EventManager.Schedule(adamsArrival, 0);

Event evesArrival
        = new LocalEvent(EventType.CUSTOMER_ARRIVAL, god);
EventManager.Schedule(evesArrival,
        arrivalGenerator.Next(MAX_INTERARRIVAL));

int customersSoFar = 2;
```

Bank Simulation: Event-based

```
for ( LocalEvent e = (LocalEvent)EventManager.NextEvent();
                e != null; e = (LocalEvent)EventManager.NextEvent() ) {
        switch ( e.Type ) {
                case EventType.CUSTOMER_ARRIVAL :
                        // Process customer arrival event
                        break;
                case EventType.TELLER_ACQUISITION :
                        // Process teller acquisition event
                        break;
                case EventType.TELLER_RELEASE :
                        // Process teller release event
                        break;
                case EventType.CUSTOMER_DEPARTURE :
                        // Process customer departure event
                        break;
        } // end switch
} // end for
```

Bank Simulation: Arrival

```
Entity thisCustomer = new Entity("Customer");
Event onAcquire = new LocalEvent(EventType.TELLER_ACQUISITION,
        thisCustomer);
teller.Acquire(onAcquire);
if ( customersSoFar < MAX_CUSTOMERS )
{
        ++customersSoFar;
        Event newArrival
                = new LocalEvent(EventType.CUSTOMER_ARRIVAL, god);
        long arrivalDelta = arrivalGenerator.Next(MAX_INTERARRIVAL);
        EventManager.Schedule(newArrival, arrivalDelta);
}
```

Bank Simulation: Teller Acquisition

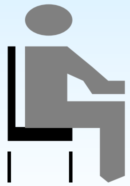
```
long howLong2keep =
        tellerConsumptionGenerator.Next(MAX_TELLER_CONSUMPTION);
Event releaseEvent =
        new LocalEvent(EventType.TELLER_RELEASE, e.Owner);
EventManager.Schedule(releaseEvent, howLong2keep);
```

Bank Simulation: Teller Release

```
teller.Release();
Event customerDepart = new
    LocalEvent(EventType.CUSTOMER_DEPARTURE, e.Owner);
EventManager.Schedule(customerDepart, 0);
```

Bank Simulation: Departure

```
// Nothing to do; sit pretty
```



How does the teller pick up the next customer when the current customer departs?

Exercise

What implications are there if we move the customer arrival generation code to the *DEPARTURE* event processing from the *ARRIVAL* event processing?

```
if ( customersSoFar < MAX_CUSTOMERS )
{
        ++customersSoFar;
        Event newArrival
                = new LocalEvent(EventType.CUSTOMER_ARRIVAL, god);
        long arrivalDelta = arrivalGenerator.Next(MAX_INTERARRIVAL);
        EventManager.Schedule(newArrival, arrivalDelta);
}
```

Bank Simulation: Event-based

- Go through the event-based bank simulation system supplied in the course resources.
- Modify the system to collect useful performance metrics (you define what's useful) and statistics.
  - E.g., Do female customers require less service time at the teller?
- Modify the system further to answer more "What-if" questions.
  - E.g., What effect giving a two-hourly 15 min break to each teller has on the performance of the system?

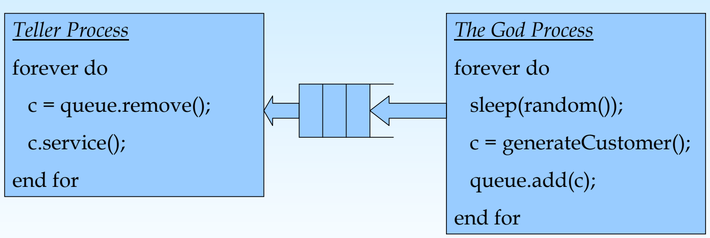
Event-based Simulators

- In an event-based simulator, the system is modelled as a collection of events.
- Coding an event-based simulator is tedious and it is hard to get the code correct.
- Maintaining and updating the simulator is also tedious and time consuming.

Process-based Simulators

- An easier and more natural approach to model a system is to describe the behaviour of its components and the way they interact.
- *Process-based* simulators take this approach in which every active component of the system is modelled by a process, so that the actions and interactions of the processes correspond to those of the system's active components.

Bank Simulation: Process-based

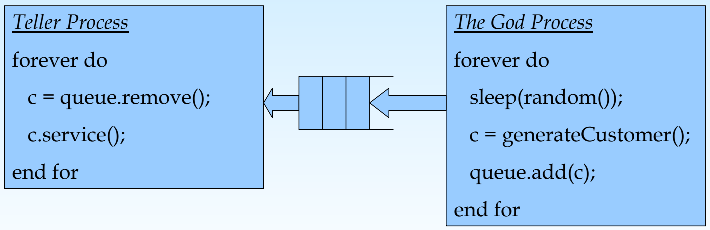


Two parallel processes, the *teller* and the *God*, communicating via a common queue structure.

Process-based Simulators

- A process could simply be a description of the system component's operation in the simulator's host language.
- Should the definition of a system component change, the simulator is updated by modifying the corresponding process that models the component.
- Process-based simulators are modular and thus make the construction and maintenance of large-scale models easy.

Bank Simulation: Process-based



1. Modify the system so that customers gets service only if there aren't any disabled customers waiting. Assume that there is no pre-empting.
2. Examine the effect of having a queue for each teller rather than having a single queue.

Static and Dynamic Structures

- In modelling the system components, it is necessary to specify their static and dynamic structures.
  - The *static* structure of a system component specifies its physical framework. The *dynamic* structure, on the other hand, specifies the way the component accomplishes its work.

Static and Dynamic Structures

- It is the dynamic structure that contributes towards the *active* nature of a component; thus, components that have no dynamic structure are said to be *passive*.
- In general, a system has both active and passive components.
  - E.g., A *resource* is a passive entity that can be *acquired* and *released* by active entities.
  - E.g., the queue in the process-based bank simulation is passive while the customers are active.

Random Variables

- Most simulation models use random variables to mimic the input data (e.g. customer arrival time).
- Given a phenomenon that we intend to model, we must choose an appropriate probability distribution.
  - This choice is critical to a successful model.
  - The data set of random observations from a distribution must be statistically indistinguishable from the empirical observations of the phenomenon we intend to model.

Random Variables

Phenomenon	Example	Distribution that often describes the phenomenon
Choice outcome	Tossing a coin; Sex of a customer	Bernoulli
Quantity	Weight of a shipment	Normal
Interval	Time between customer arrivals	Exponential
Frequency	Number of customer arrivals per hour	Poisson
Duration	Time to complete a bank transaction	Erlang

Validation

- The results from the simulations are only as good as the model
- Validation of the results is an important aspect of simulation. Where possible:
  - compare results from a real system to the results from the simulated system
  - perform sanity checks
  - check conformance with analytical models

Software Engineering Rules

- A simulator is a software, so the rules of software engineering hold for the simulator.
  - Modularity, extensibility, and re-usability
  - Design for ease of maintenance
- Performance matters!
  - Simulators typically run for hours. Profile and optimize.
  - Consider distributed or parallel simulation.

Summary

- A simulator is a dynamic model that mimics the behaviour of a system (within the limitations of the system description).
- The quality of the simulation depends on the quality of the model. There are no known GIGO systems.
  - Build a well-focussed model that will answer your questions about the system
  - Ensure, however, the model is extensible so that you can modify it to answer further questions

Further Reading

- Jerry Banks, “Introduction to simulation”, In the *Proceedings of the 2000 Winter Simulation Conference*, pages 9-16, 2000.
- Arne Thesen and Laurel Travis, “Introduction to simulation”, In the *Proceedings of the 1990 Winter Simulation Conference*, pages 14-21, 1990.
- Richard Fujimoto, “Parallel and Distributed Simulation Systems”, In the *Proceedings of the 2001 Winter Simulation Conference*, pages 147-157, 2001.