

**MIT 2.853/2.854**  
**Manufacturing Systems**  
**Introduction to Simulation**

Lecturer: Stanley B. Gershwin  
*... with many slides by Jeremie Gallien*

# Optional References

- **Law, A. and W. Kelton, *Simulation Modeling and Analysis*, 3<sup>rd</sup> ed., McGraw-Hill (2000).**
- **Harrell et al., *Simulation Using ProModel*, McGraw-Hill (2004)**
- **Ross, S., *Simulation*, 3<sup>rd</sup> ed., Academic Press (2002).**

# What is Simulation?

A *computer simulation* is a computer program ...

- that calculates a hard-to-calculate quantity using statistical techniques; *OR*
- that models the behavior of a system by imitating individual components of the system and their interactions.

*I am not entirely satisfied with this definition — it does not seem restrictive enough. If all goes well, you will know what a computer simulation is after this lecture.*

# What is Simulation?

By contrast, a *mathematical model* is ...

- a mathematical representation of a phenomenon of interest.

Computer programs are often used to obtain quantitative information about the thing that is modeled.

# What is Simulation?

Simulation is used ...

- for calculating hard-to-calculate quantities ...
  - ★ from mathematics and science,
  - ★ from engineering, especially system design.
- for developing insight into how a system operates,
- for demonstrating something to bosses or clients.

# What is Simulation?

- Static, for the evaluation of a quantity that is difficult to evaluate by other means. The evolution of a system over time is not the major issue.
- Dynamic, for the evaluation of quantities that arise from the evolution of a system over time.

# What is Simulation?

- Static — Monte Carlo
- Dynamic
  - ★ Discrete time
  - ★ Discrete event
  - ★ Solution of differential equations — *I don't consider this simulation, but others do.*

# Types of Simulation

**Static / Monte-Carlo**

**(Crystal Ball)**

Examples:

- Options Contracts
- Insurance
- Demand Models

**Discrete Events**

**(ProModel)**

Examples:

- Business Processes
- Education
- Military

**Continuous Time**

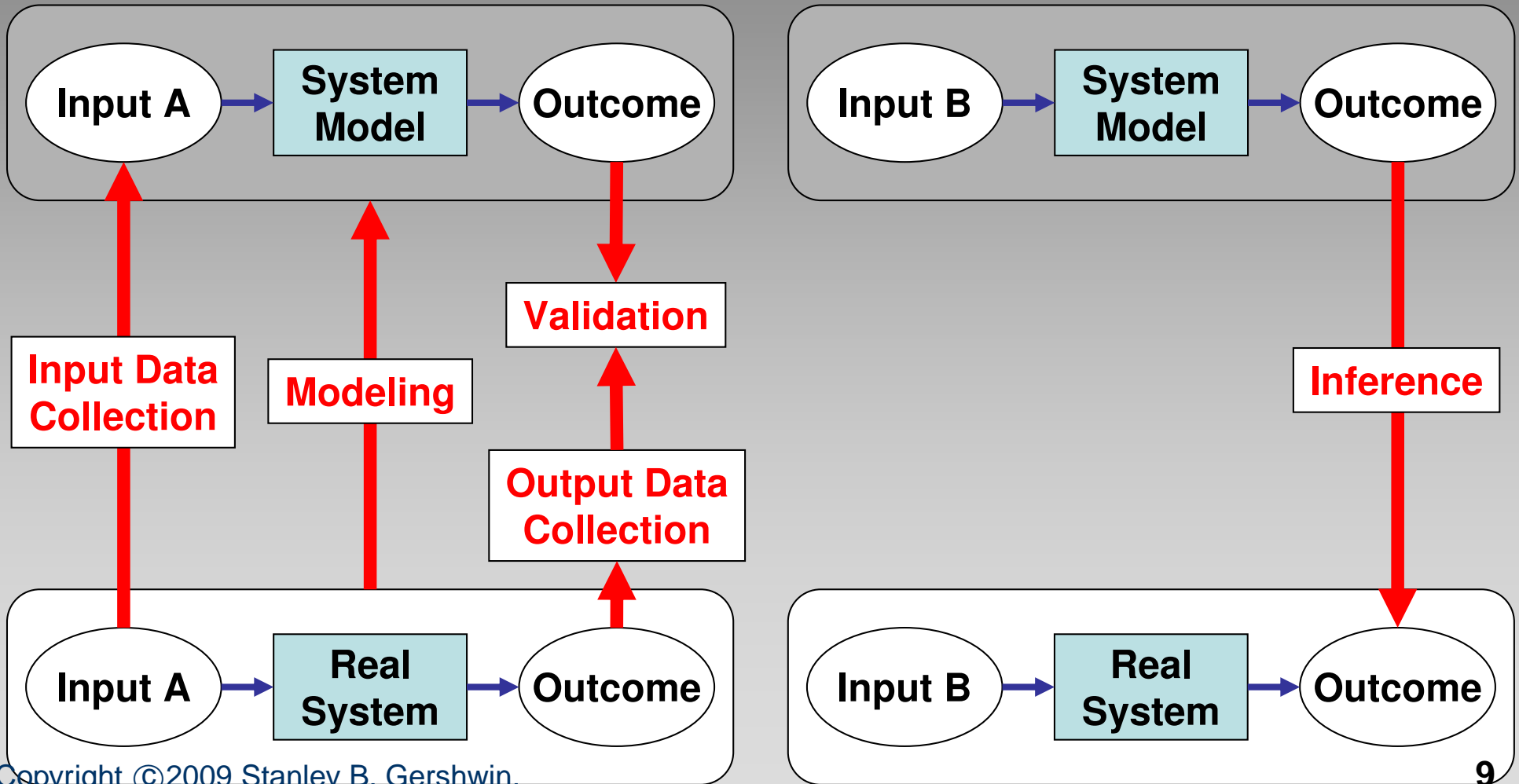
**(application-specific)**

Examples:

- Weather models
- Engineering design
- Scientific research
- Entertainment
- Education



# The Simulation Methodology



Copyright © 2009 Stanley B. Gershwin.

Copyright 2003 © Jérémie Gallien

# Dynamic Simulation

## Discrete Time Simulation

Appropriate for systems in which:

- Time is discrete.
  - ★ If time in the real system is continuous, it is discretized.
- There is a set of events which have a finite number of possible outcomes. For each event, the outcome that occurs is independent of the other simultaneous events and of all past events.
- There is a system state which evolves according to the events that occur.
  - ★ That is, the system is a discrete time Markov chain.
  - ★ Often, other systems can be transformed into or approximated by discrete time Markov chains.

To model a random event that occurs with probability  $p$ , let  $u$  be a pseudo-random number which is distributed uniformly between 0 and 1.

- Generate  $u$ .
- If  $u \leq p$ , the event has occurred. Otherwise, it has not.

### PERL code excerpt, Machine 1:

```
for ($step=0; $step <= $T; $step++){
```

- `if($alpha1[$step]==1)`

- ★ `{if (rand(1)<$p1){$alpha1[$step+1]=0}`  
`else{$alpha1[$step+1]=1}}`

- `else`

- ★ `{if (rand(1)<$r1){$alpha1[$step+1]=1}`  
`else{$alpha1[$step+1]=0}}`

- `if(($alpha1[$step+1]==1)&&($n[$step]<$N)) {$IU=1}`  
`else{$IU=0}`

Machine 2 is similar, except:

```
if (($alpha2[$step+1]==1)&&($n[$step]>0)){$ID=1;$out++;}  
    else{$ID=0}
```

where \$out is a counter for the number of parts produced.

Buffer:

```
$n[$step+1]=$n[$step]+$IU-$ID;
```

- **Advantage:** easy to program
- **Limitations:**
  - ★ geometric distributions, or others that can be constructed from geometric distributions;
  - ★ discrete time, or highly discretized continuous time
- **Disadvantage:** slow. Calculation takes place at every time step.

- Appropriate for systems in which:
  - ★ Events occur at random times and have a finite set of possible outcomes.
  - ★ There is a system state that evolves according to the events that occur.
- Time can be discrete or continuous. Discretization is unnecessary.
- These conditions are much less restrictive than for discrete time simulations.

## Method:

1. Starting at time 0, determine the next time of occurrence for all possible events. Some times will be obtained deterministically; others from a random number generator.
2. Create an *event list*, in which all those events are sorted by time, earliest first.
3. Advance the clock to the time of the first event on the list.
4. Update the state of the system according to that event.



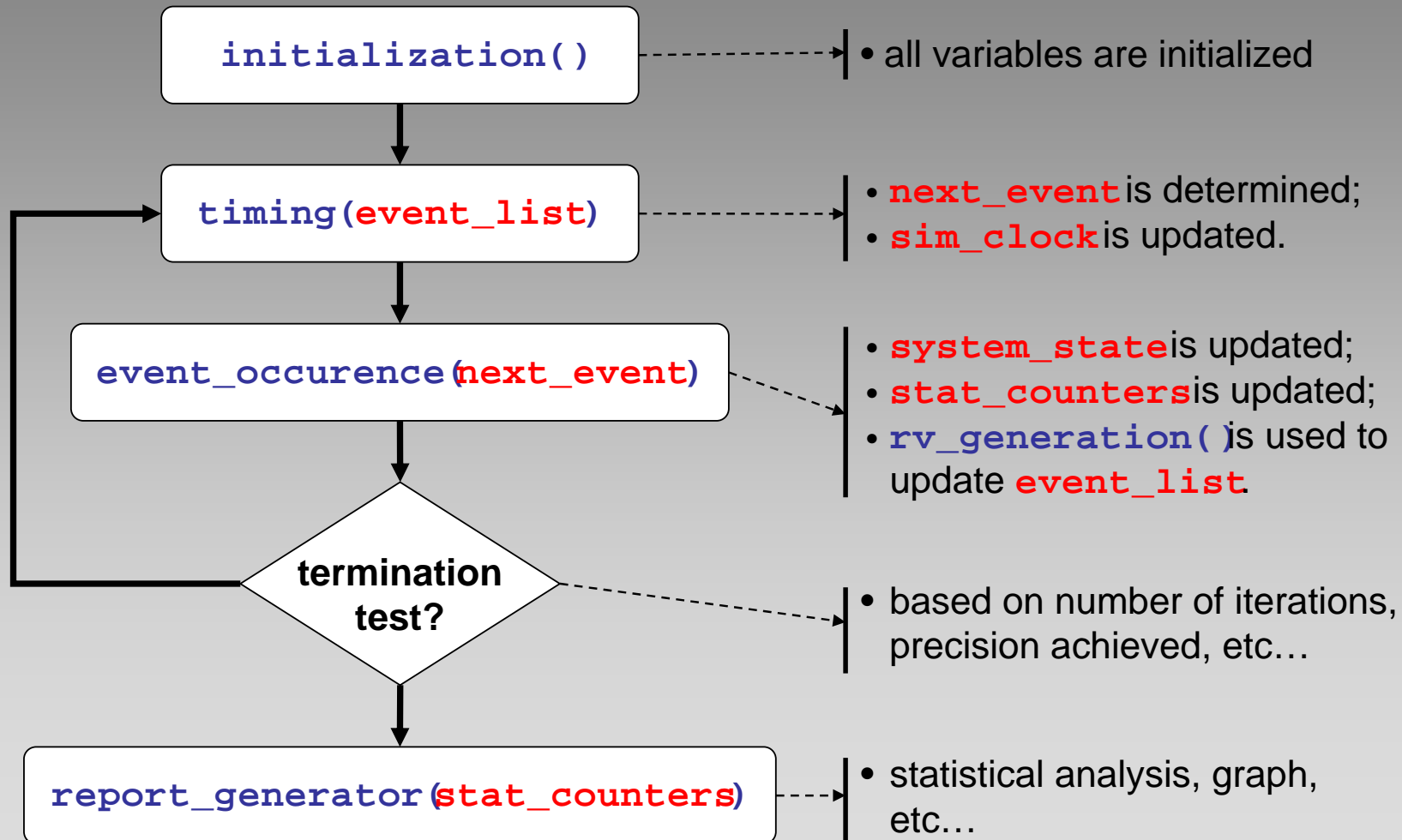
5. Determine if any new events are made possible by the change in state, calculate their times (in the same way as in Step 1), and insert them into the event list chronologically.
6. Determine if any events on the list are made impossible by the current event. Remove them from the event list. Also, remove the current event from the list.
7. Go to Step 3.

Assume the system starts with both machines up, the buffer with  $n > 1$  parts in it. The clock is at  $t = 0$ .

- Possible next events:  $M_1$  failing,  $M_2$  failing. Pick times for both events from the up-time distributions.
- Advance clock to the first event. Suppose it is  $M_1$  failing.
  - ★ Possible next events:  $M_2$  failing (already on the list), buffer emptying,  $M_1$  getting repaired.
  - ★ Pick the time for  $M_1$  getting repaired from  $M_1$ 's down-time distribution. Calculate the time for the buffer emptying *deterministically* from the current buffer level.

- Delete current event from the list, and advance the clock to the next event. Assume it is the buffer emptying.
  - ★ Now we have to adjust the list because  $M_2$  cannot fail while the buffer is empty.
  - ★ The only possible next event is the repair of  $M_1$ . Pick the time from the random number generator.
- etc.

# Discrete-Event Algorithm



Copyright 2002 © Jérémie Gallien

Slide courtesy of Jérémie Gallien. Used with permission.

- Advantages:
  - ★ Fast. Calculation only occurs when an event occurs.
  - ★ General. The time until an event occurs can have any distribution.
  - ★ Commercial software widely available.
- Limitations: ?????
- Disadvantage: Hard to program. (*But see “Advantages.”*)

Random numbers are needed. This is not trivial because a computer is inherently deterministic.

- They must be
  - ★ distributed according to a specified distribution; and
  - ★ independent.
- It is also very desirable to be able to reproduce the sequence if needed — and not, if not needed.

# Random Number Generation

- The key in a Monte-Carlo simulation is to generate sample values drawn from a known probability distribution. How is this done?

Generate Sample Value from Uniform[0,1]

```
graph TD; A[Generate Sample Value from Uniform[0,1]] --> B[Discrete Distribution]; A --> C[Continuous Distribution];
```

## Discrete Distribution

- Inverse transform method
- Rejection method

## Continuous Distribution

- Inverse transform method
- Rejection method
- Composition method
- Polar method

A *pseudo-random number generator* is a function  $F$  such that for any number  $\omega$ , there is a set of numbers  $Z_1(\omega), Z_2(\omega), \dots$  that satisfy

$$Z_{i+1}(\omega) = F(Z_i(\omega))$$

$$Z_0(\omega) = \omega \text{ is the } \textit{seed}$$

and the sequence of  $Z_i(\omega)$  satisfies certain conditions.




# U[0,1] Sample Generation

1. Calculate  $Z_1, Z_2, \dots$  given by the Linear Congruential Generator:

$$Z_i = ( aZ_{i-1} + c ) \text{ modulo } m$$

where  $m, a, c$  and  $Z_0$  are non-neg. integers

  
modulus

  
seed

2. Take  $U_i = Z_i / m$  as a proxy to an independent sequence of U[0,1] random variates (note:  $0 \leq U_i \leq 1$  for all  $i$ )

- The sequence  $Z_i(\omega)$  is determined — *deterministically* — by  $Z_0(\omega) = \omega$ .
- A small change in  $\omega$  causes a large change in  $Z_i(\omega)$ .
- Since the user can specify the seed, it is possible to re-run a simulation with the same sequence of random numbers. This is sometimes useful for debugging, sensitivity analysis, etc.
- Often, the computer can choose the seed (for example, using the system clock). This guarantees that the same sequence is *not* chosen.

# Discrete Distributions

- We want to simulate a random variable  $X$  defined by  $P(X = x_j) = p_j, \quad j=0,1,\dots,m$

1. Generate a sample  $U$  from  $U[0,1]$

2. if  $U < p_0$  then  $X = x_0$

if  $p_0 < U < p_0 + p_1$  then  $X = x_1$

.....

if  $p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j$  then  $X = x_j$

- **Why does this work?**

# Solution

1. Generate a sample  $U$  from  $U[0,1]$
2. if  $U < p_0$  then  $X = x_0$   
if  $p_0 < U < p_0 + p_1$  then  $X = x_1$   
.....  
if  $p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j$  then  $X = x_j$

- $$P(X = x_j) = P(p_0 + \dots + p_{j-1} < U < p_0 + \dots + p_j)$$
$$= p_j$$

# Continuous Distributions

- We want to simulate a random variable  $X$  defined by  $P( X < t ) = F(t)$  ( $F$  inversible)

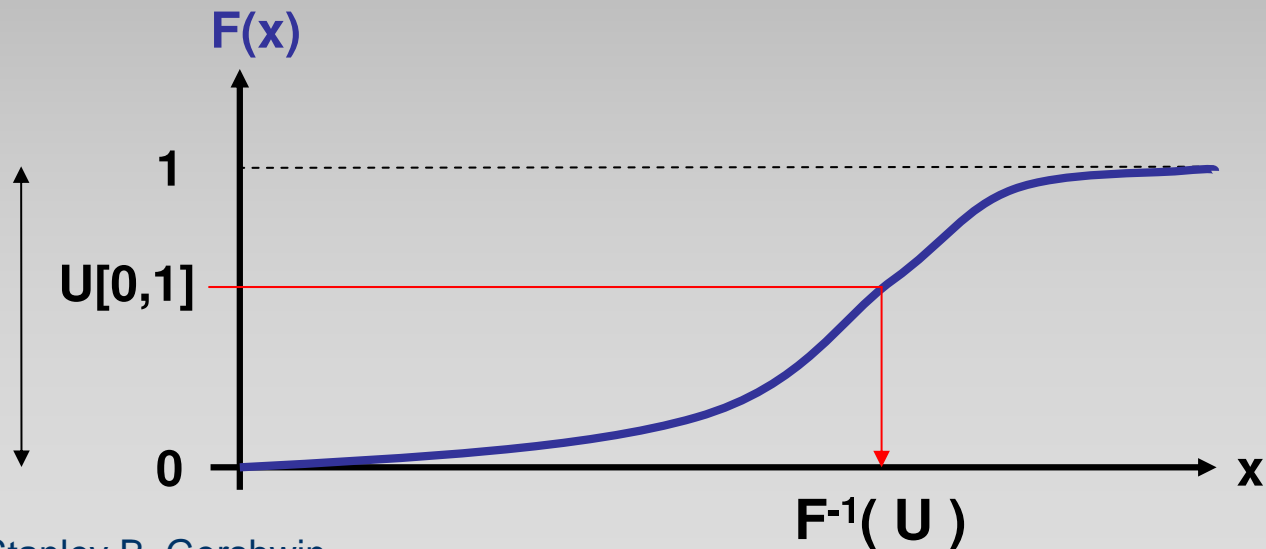
1. Generate a sample  $U$  from  $U[0,1]$
2. Set  $X = F^{-1}( U )$

- Why does this work?

# Solution

1. Generate a sample  $U$  from  $U[0,1]$
2. Set  $X = F^{-1}( U )$

- $P ( X < t ) = P( F^{-1}(U) < t ) = P( U < F(t) ) = F(t)$



# Exponential Distribution

- Variable  $\exp(\lambda)$  has cdf.  $F(t) = 1 - \exp(-\lambda t)$ ;  
 $F^{-1}(y) = -\log(1-y) / \lambda$

1. Generate  $U$  from  $U[0,1]$
2. Compute  $X = -\log(1-U) / \lambda$
3. The distribution of  $X$  is  $\exp(\lambda)$  !

# Dynamic Simulation

## Technical Issues

### Statistics, Repetitions, Run Length, and Warmup

- The purpose of simulation is to get quantitative performance measures such as production rate, average inventory, etc.
- The data that comes from a simulation is statistical. Therefore, to get useful numerical results from a simulation,
  - ★ there must be enough data; and
  - ★ the data must be obtained under stationary conditions ... although there are sometimes exceptions.



#### Enough data:

- The data is treated like measurement data from the real world. Sample means, sample variances, confidence intervals, etc. must be calculated.
- The *more times* the simulation is run (ie, the greater the number of repetitions) the smaller the confidence intervals.
- The *longer* the simulation is run, the closer the results from each run are to one another, and consequently, the smaller the confidence intervals.
- However, ...

#### *Warmup:*

- To evaluate average production rate, inventory, etc., the system must be in steady state.
- But the simulation will start with the system in a known state, so it will not be in steady state immediately.
- Some time is required to get the system in steady state. This is the *transient period*. It depends on the characteristics of the system.

- Therefore, we should not collect data starting at time 0; we should start only after the system reaches steady state.
- The initial period before data is collected is the *warmup period*.
- Required: warmup period  $\geq$  transient period.
- Problem: it is very hard to determine the transient period.

# Dynamic Simulation

## Technical Issues

Statistics, Repetitions, Run Length, and Warmup

*Example:*

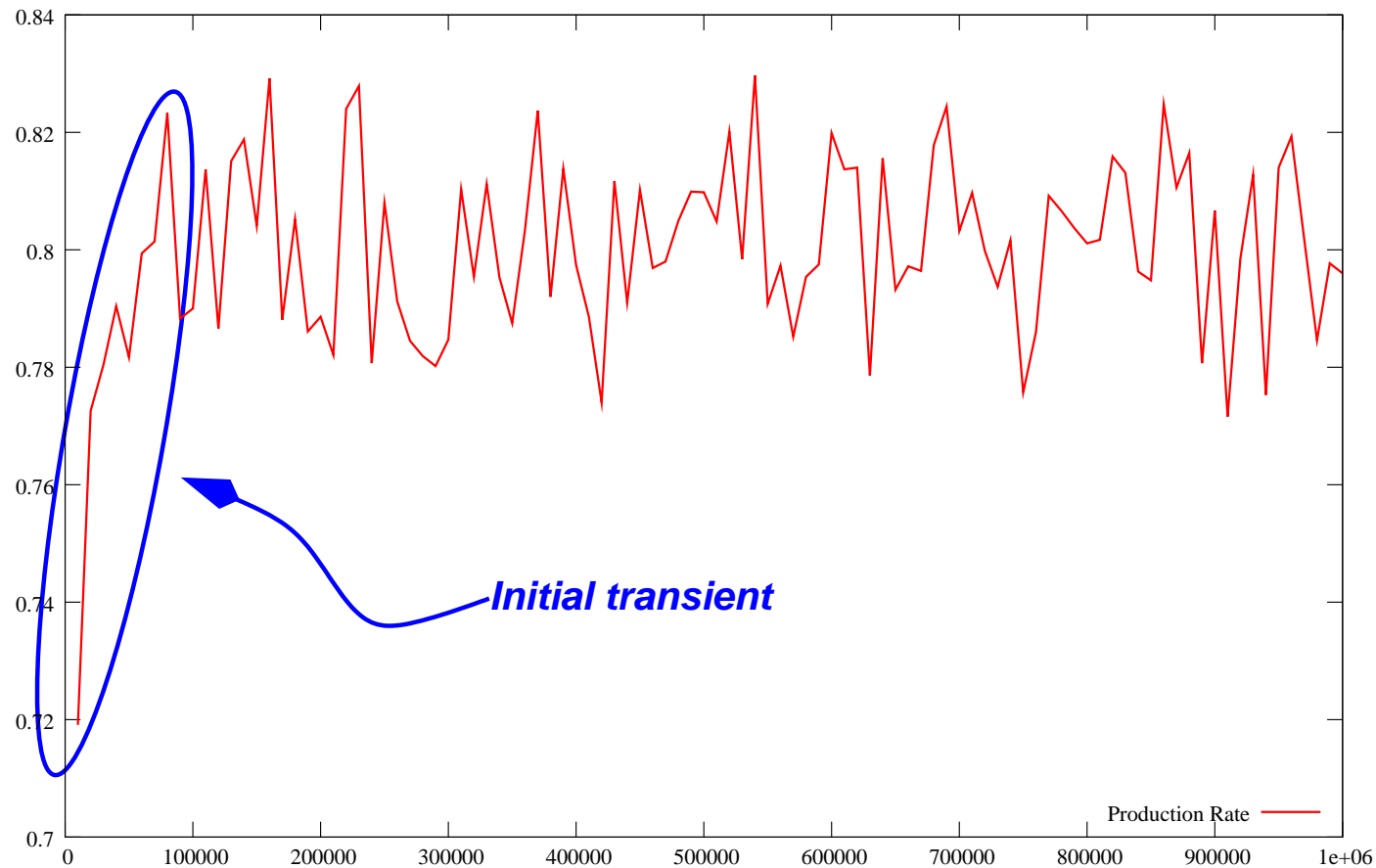
- 20-machine line.
- $r_i = .1, p_i = .02, i = 1, \dots, 19; r_{20} = .1, p_{20} = .025.$
- $N_i = 1000, i = 1, \dots, 19.$
- The simulation is run for 1,000,000 steps; data is averaged over 10,000 steps.

# Dynamic Simulation

## Technical Issues

Statistics, Repetitions, Run Length, and Warmup

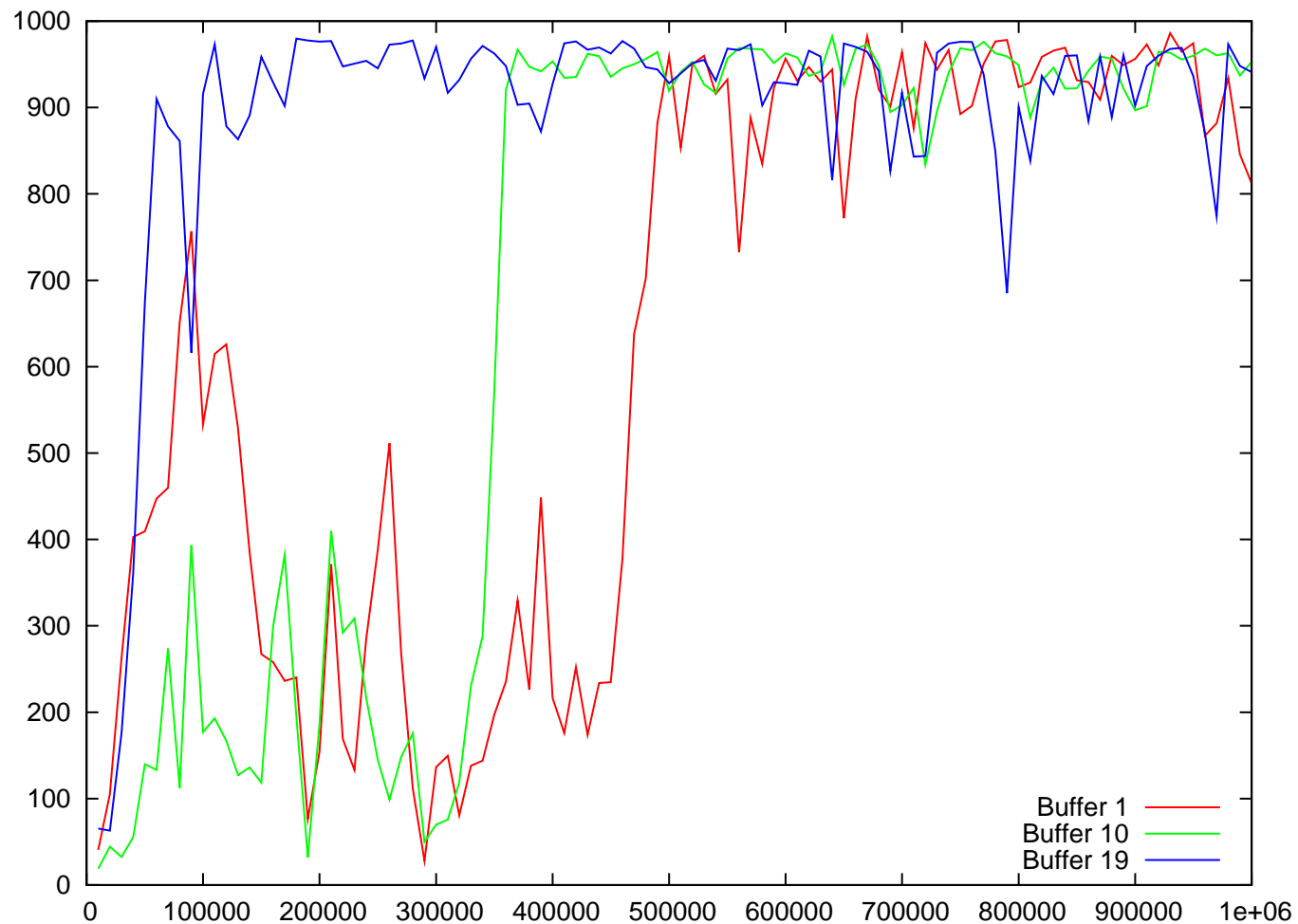
### *Production rate*



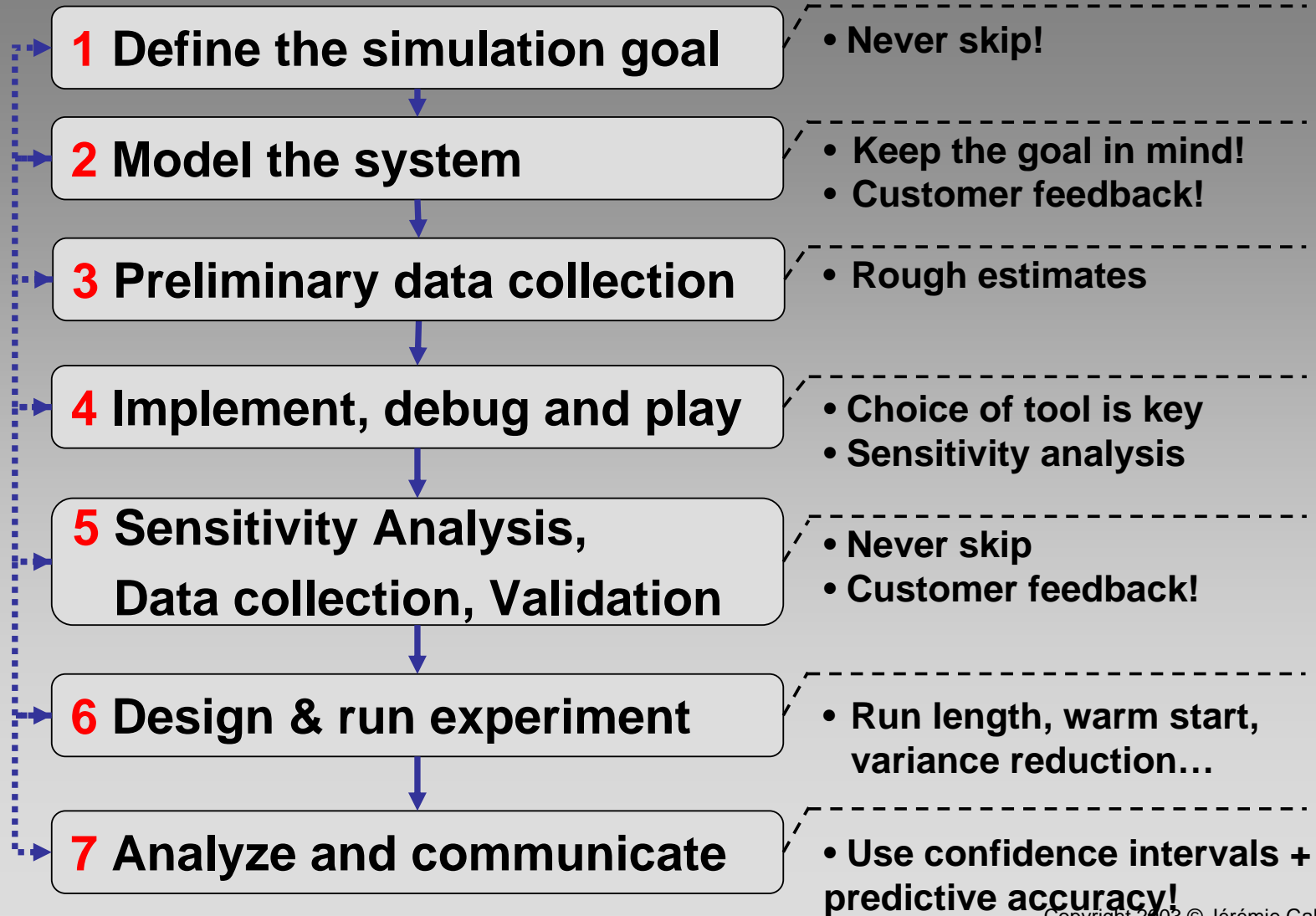
## Dynamic Simulation

Statistics, Repetitions, Run Length, and Warmup

### *Buffer levels*



# The Simulation Process



Slide courtesy of Jérémie Gallien. Used with permission.

# Typical Errors

- **Start modeling before knowing what question you want to answer**
- **Constructing a model of the universe**
- **Not enough feedback from person knowledgeable with system**
- **Collect data before knowing how your model will use it**
- **Report results without understanding where they come from/their qualitative interpretation**

Copyright 2003 © Jérémie Gallien

Slide courtesy of Jérémie Gallien. Used with permission.



# System Modeling

- **“Everything should be made as simple as possible, but not simpler.” *Albert Einstein.***
- **The simulation goal should be the guiding light when deciding what to model**
- **Start to build your model ON PAPER!**
- **Get client/user feedback early, and maintain model + assumption sheet for communication purposes**
- **Collect data and fit distributions... *after* modeling the system, with sensitivity analysis in mind!**

Copyright 2003 © Jérémie Gallien

Slide courtesy of Jérémie Gallien. Used with permission.

MIT OpenCourseWare  
<https://ocw.mit.edu>

2.854 / 2.853 Introduction To Manufacturing Systems  
Fall 2016

For information about citing these materials or our Terms of Use, visit: <https://ocw.mit.edu/terms>.