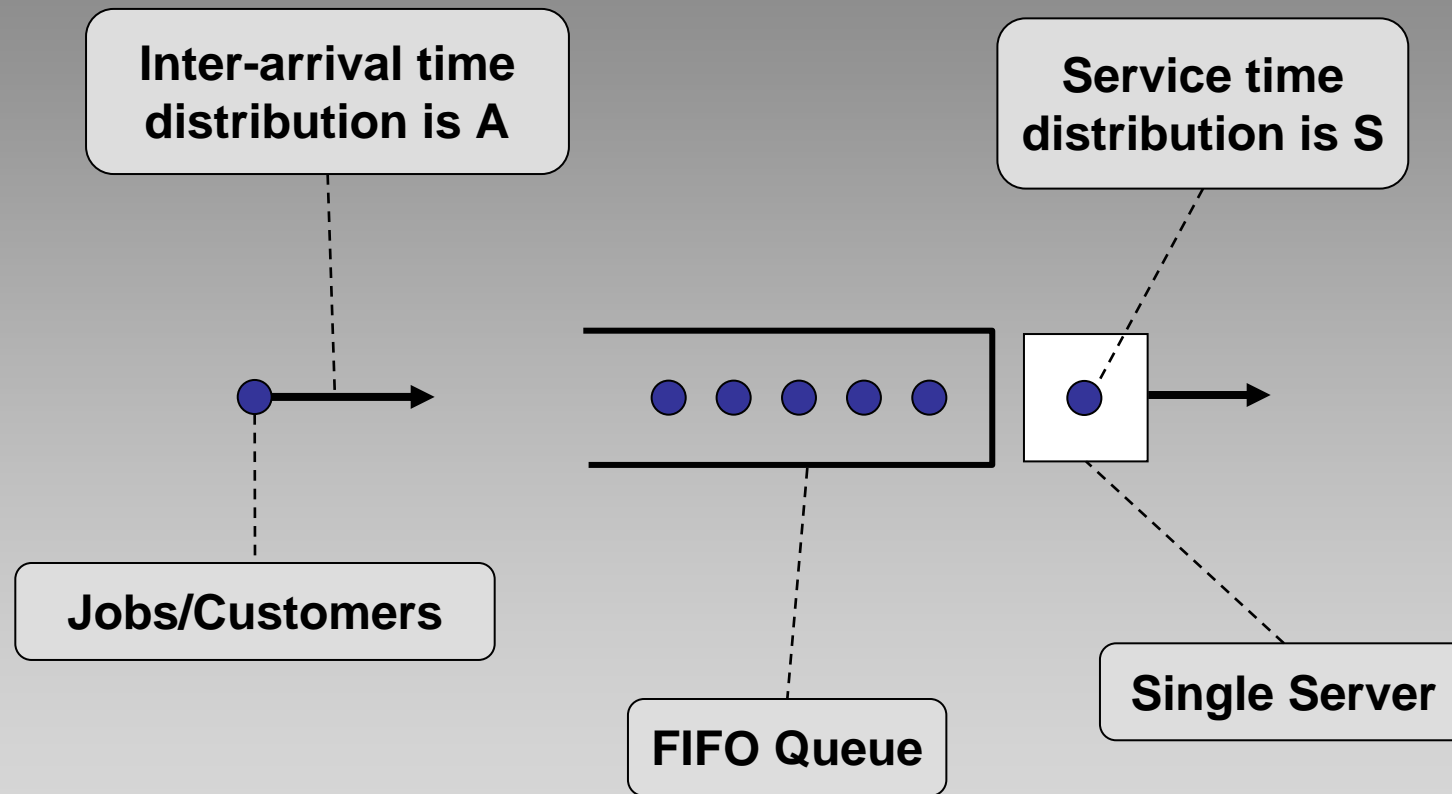


Class 3 Outline

Dynamic Discrete-Event Simulation:

- 1. Discrete-Event Framework**
- 2. Software Demo (Simul8)**
- 3. Modeling Examples**

A G/G/1 Queueing System



- How to simulate this system?

Example: G/G/1 Queue

- **system_state**: number of jobs in system (in service + waiting)
- **next_event** is either
 - **new_arrival (n_a)**: a new job arrives in the system; or
 - **service_completion (s_c)**: the job that was in service is completed

simulated time

other variables

system

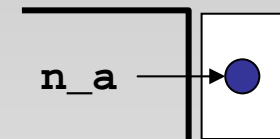
sim_clock = 0

system_state = 0
event_list = {(n_a, 1.2)}
next_event = (n_a, 1.2)

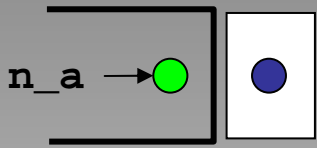
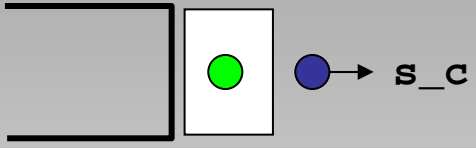
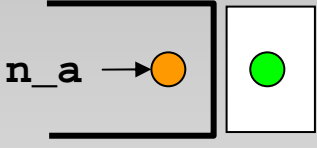
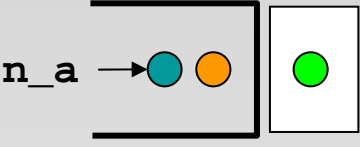


sim_clock = 1.2

system_state = 1
event_list = {(n_a, 1.8);
 (s_c, 2.2)}
next_event = (n_a, 1.8)



Example: G/G/1 Queue

simulated time	other variables	system
<code>sim_clock = 1.8</code>	<code>system_state = 2</code> <code>event_list = {(n_a, 2.6); (s_c, 2.2)}</code> <code>next_event = (s_c, 2.2)</code>	
<code>sim_clock = 2.2</code>	<code>system_state = 1</code> <code>event_list = {(n_a, 2.6); (s_c, 3.7)}</code> <code>next_event = (n_a, 2.6)</code>	
<code>sim_clock = 2.6</code>	<code>system_state = 2</code> <code>event_list = {(n_a, 3.5); (s_c, 3.7)}</code> <code>next_event = (n_a, 3.5)</code>	
<code>sim_clock = 3.5</code>	<code>system_state = 3</code> <code>event_list = {(n_a, 4.4); (s_c, 3.7)}</code> <code>next_event = (s_c, 3.7)</code>	

Example: G/G/1 Queue

- At the simulated time point `sim_clock` = 3.7 = t_6 , the performance collection variable `stat_counters` might look like:

event number	0	1	2	3	4	5	6
<code>sim_clock</code> (t_i)	0	1.2	1.8	2.2	2.6	3.5	3.7
<code>system_state</code> (S_i)	0	1	2	1	2	3	2

- If called at that point t_6 , the routine `report_generator()` might perform an operation like:

$$Y = \frac{1}{t_6 - t_0} \sum_{i=0}^5 (t_{i+1} - t_i) S_i$$

Discrete-Event Algorithm

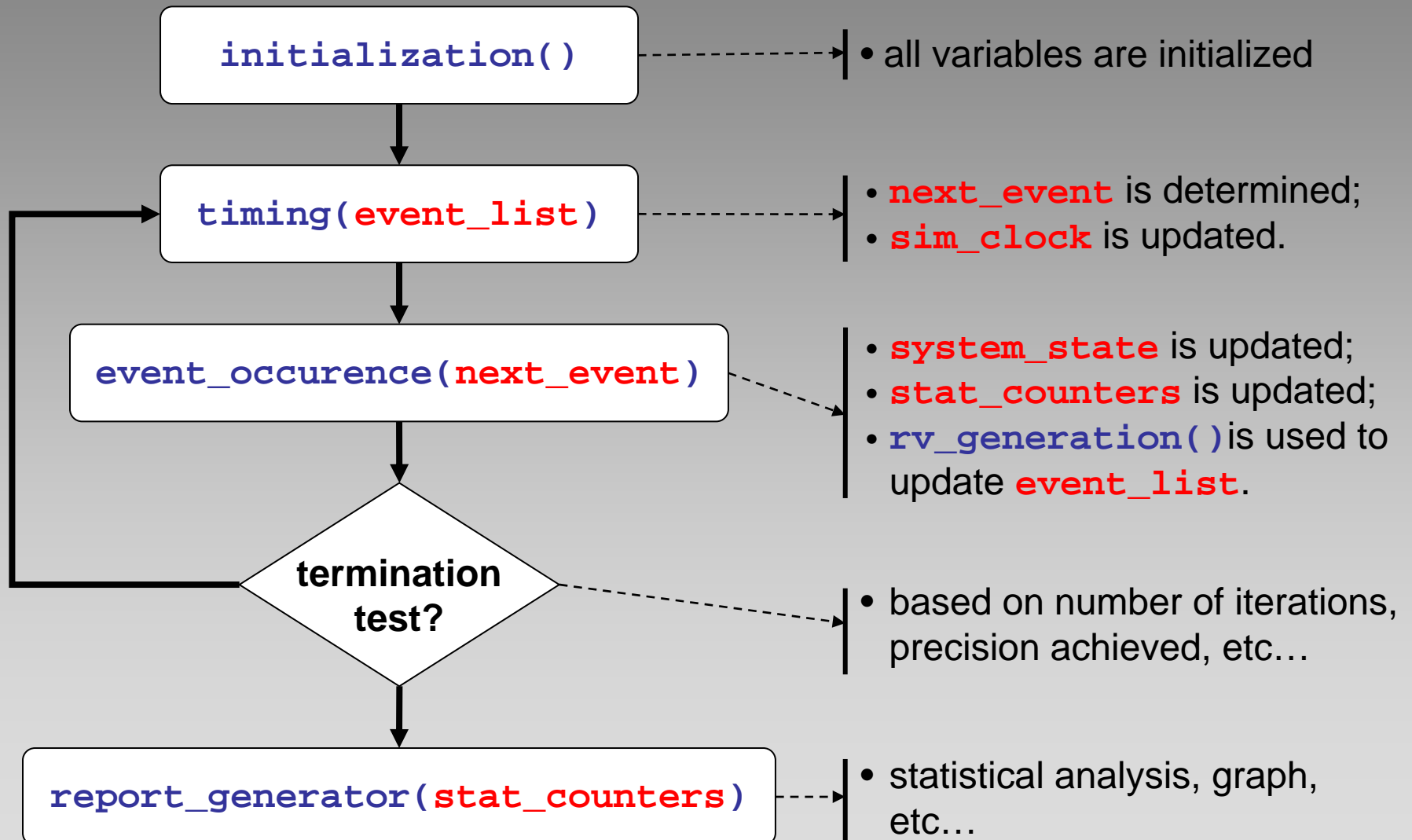
Variables:

- **system_state**
- **sim_clock** current value of simulated time.
- **event_list** contains the next time when each type of event will occur.
- **next_event** is the type of the next event that will occur.
- **stat_counters** contains historical performance/behavior information.

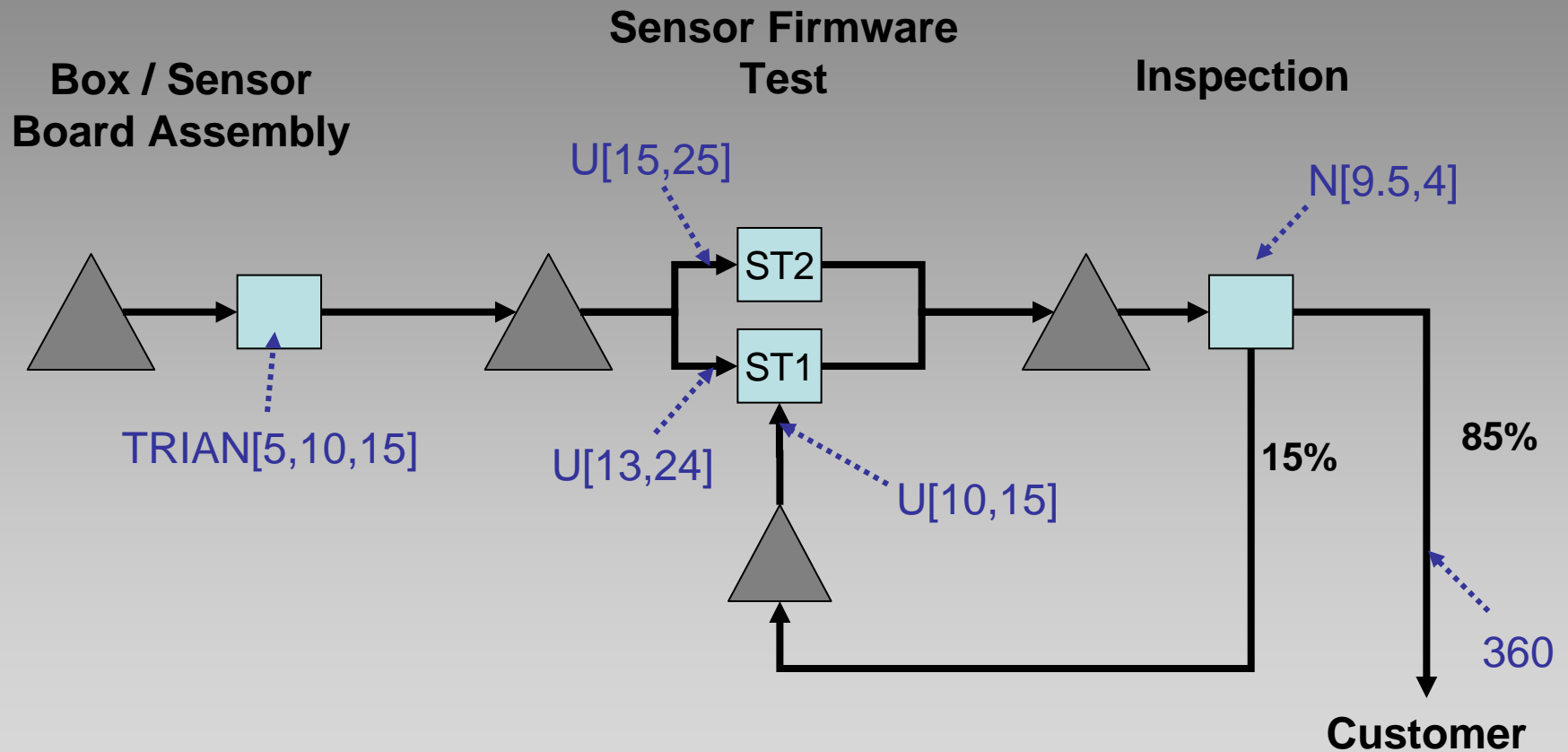
Subroutines:

- **initialization()**
- **timing(event_list)** determines **next_event** from **event_list** and advances **sim_clock** to the time this event occurs.
- **event_occurrence(next_event)** updates all variables according to the model logic when each particular type of event occurs.
- **rv_generation()** generates samples of random variables according to the probability distributions specified in the model.
- **report_generator(stat_counters)** computes estimates of the performance variables of interest from the **stat_counters** variable.

Discrete-Event Algorithm



ClearPictures Production Model



Label-Based Distribution for ST1

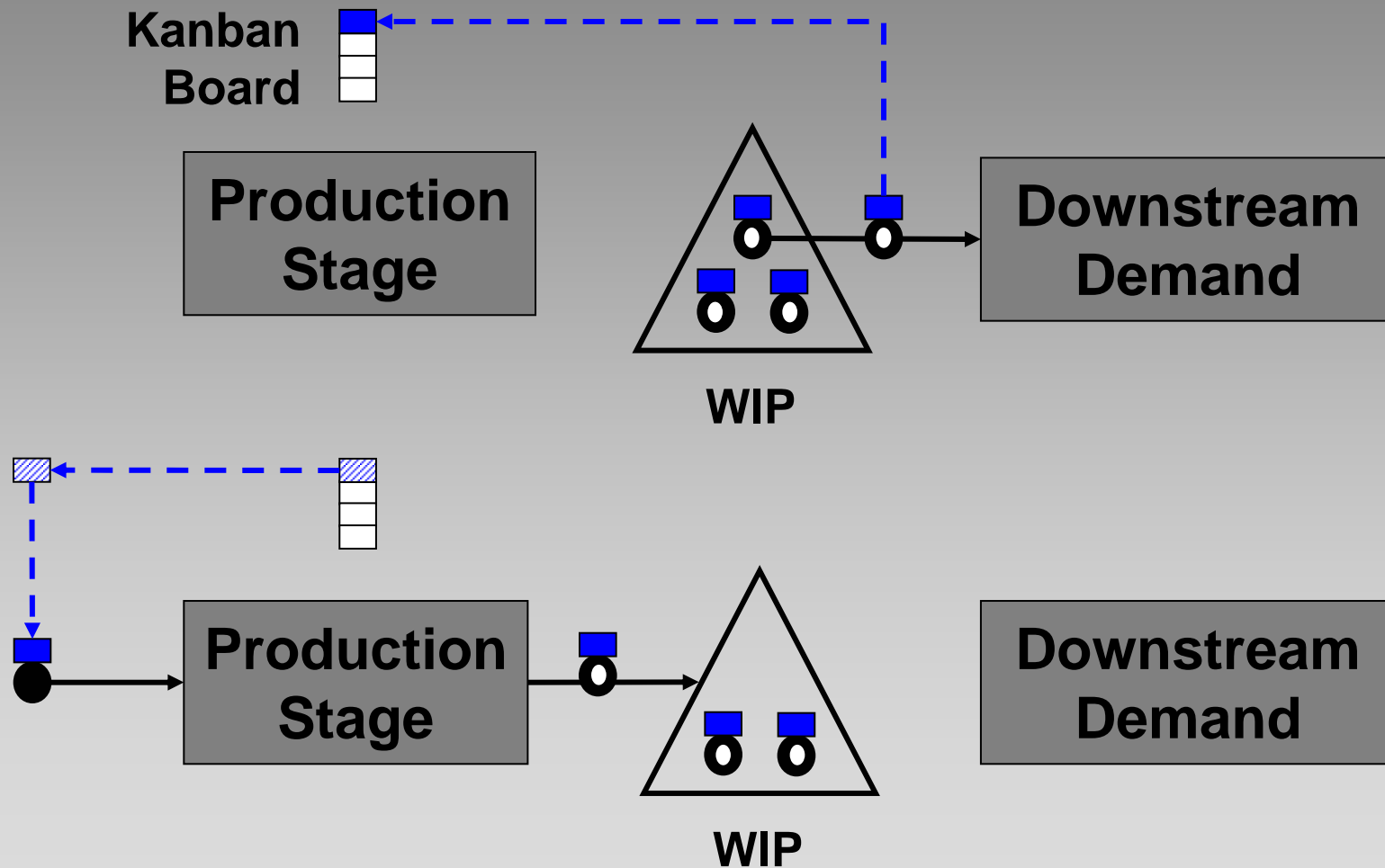
Set-up:

- A work item label ***Distribution ST1*** is created
- A probability distribution named ***Sensor Test ST1*** and equal to U[13,24] is created
- A probability distribution named ***Firmware Adjustment after Inspection*** and equal to U[10,15] is created

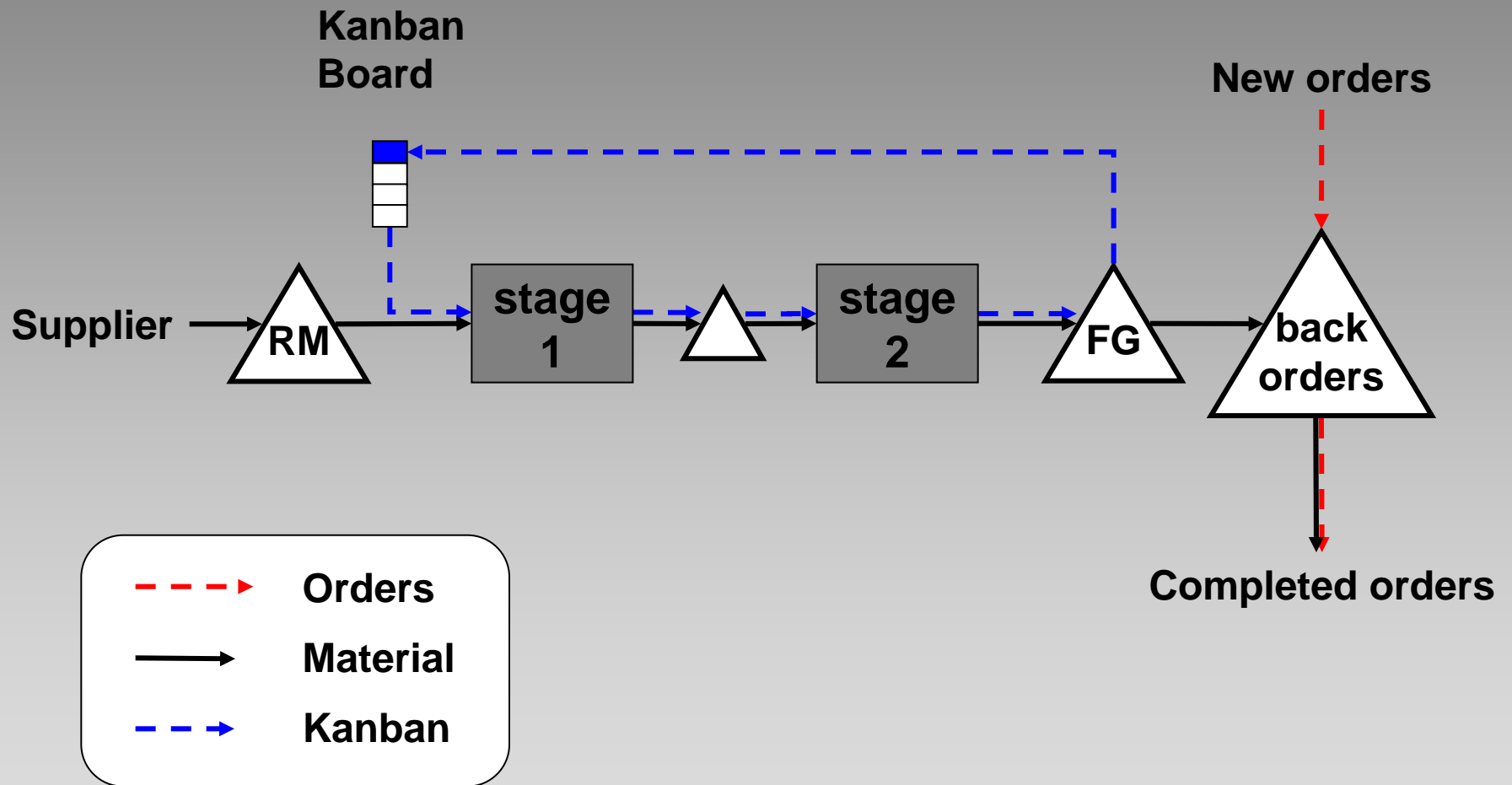
Logic:

- The work entry point initially sets the label ***Distribution ST1*** to ***Sensor Test ST1*** for all items
- The service time on work center Sensor Testing ST1 is defined as a *label-based distribution* associated with the label ***Distribution ST1***
- The work center “label-changing dummy” has 0 service time but sets the label ***Distribution ST1*** to the value ***Firmware Adjustment after Inspection***

What is Kanban?

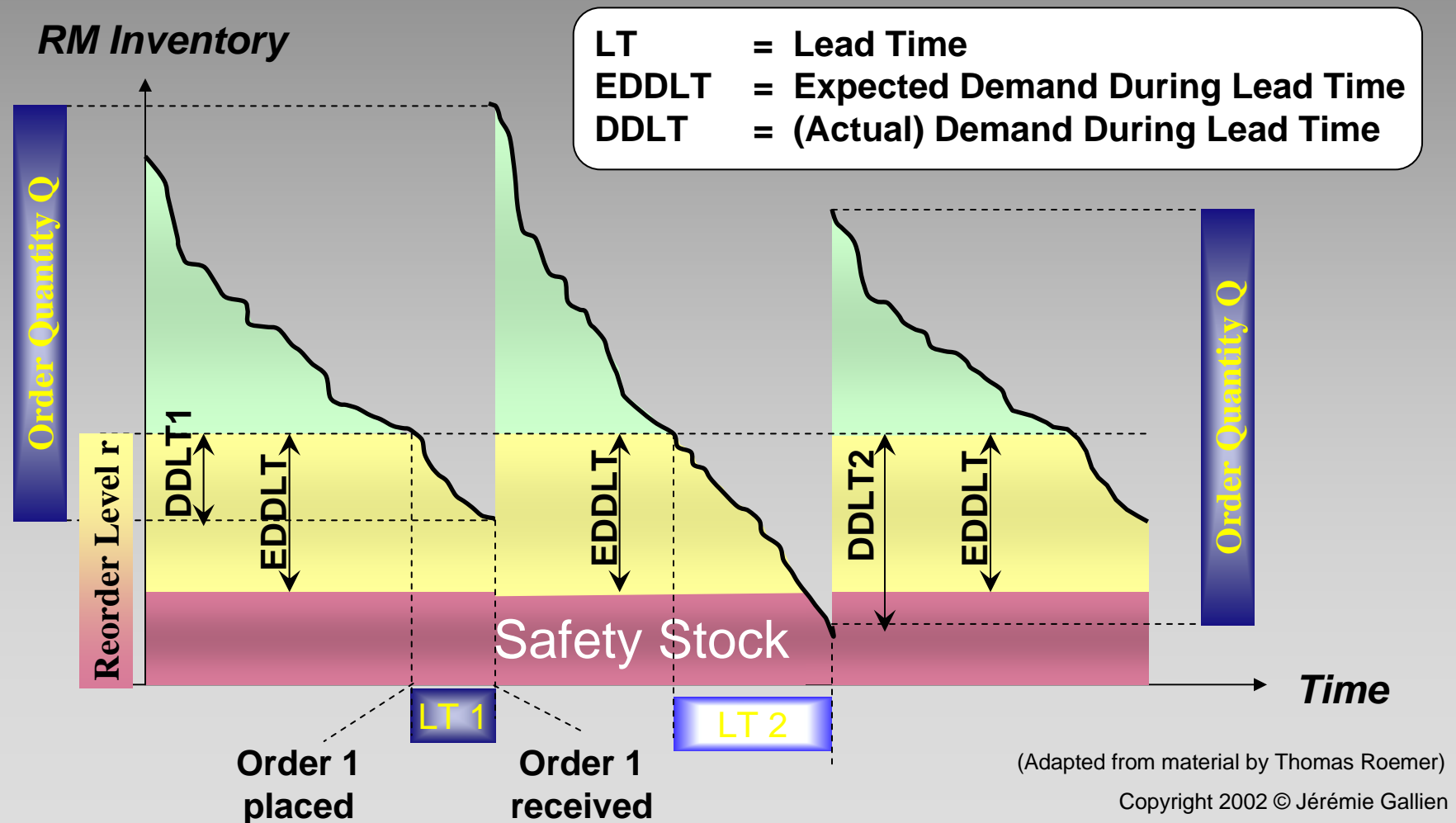


A Pull Production System



Reorder Point Policy

- Assume now a component replenishment lead-time LT (random) and a continuous (r, Q) policy:



Class 3 Wrap-Up

Industrial decision-making is interdisciplinary

