Events may be *controllable* or not, and *predictable* or not.

<table>
<thead>
<tr>
<th></th>
<th>controllable</th>
<th>uncontrollable</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>predictable</em></td>
<td>loading a part</td>
<td>lunch</td>
</tr>
<tr>
<td><em>unpredictable</em></td>
<td>???</td>
<td>machine failure</td>
</tr>
</tbody>
</table>
Definitions

- **Scheduling is the selection of times for future controllable events.**

- Ideally, scheduling systems should deal with all controllable events, and not just production.
  - That is, they should select times for operations, set-up changes, preventive maintenance, etc.
Definitions

- Because of recurring random events, scheduling is an on-going process, and not a one-time calculation.
- Scheduling, or shop floor control, is the bottom of the scheduling/planning hierarchy. It translates plans into events.
This is the general paradigm for control theory and engineering.
In a factory,

- **State**: distribution of inventory, repair/failure states of machines, etc.
- **Control**: move a part to a machine and start operation; begin preventive maintenance, etc.
- **Noise**: machine failures, change in demand, etc.
• **Release**: Authorizing a job for production, or allowing a raw part onto the factory floor.

• **Dispatch**: Moving a part into a workstation or machine.

• **Release is more important than dispatch**. That is, improving release has more impact than improving dispatch, if both are reasonable. *(Wein observation.)*
Scheduling systems or methods should ... 

- deliver good factory performance.
- compute decisions quickly, in response to changing conditions.
Performance Goals

- To minimize inventory and backlog.
- To maximize probability that customers are satisfied.
- To maximize predictability (i.e., minimize performance variability).
Performance Goals

- For MTO
  - To meet delivery promises.
  - To make delivery promises that are both soon and reliable.
- For MTS
  - to have FG available when customers arrive; and
  - to have minimal FG inventory.
Objective is to keep cumulative production close to cumulative demand.
Performance Goals

- Complex factories
- Unpredictable demand (ie $D$ uncertainty)
- Factory unreliability (ie $P$ uncontrollability)
Basic approaches

- Simple rules — *heuristics*
  - Dangers:
    - Too simple — may ignore important features.
    - Rule proliferation.

- Detailed calculations
  - Dangers:
    - Too complex — impossible to develop intuition.
    - Rigid — had to modify — may have to lie in data.
- Deterministic optimization.
  - Large linear or mixed integer program.
  - Re-optimize periodically or after important event.
- Scheduling by simulation.
Nervousness or scheduling volatility (fast but inaccurate response):

★ The optimum may be very flat. That is, many very different schedule alternatives may produce similar performance.

★ A small change of conditions may therefore cause the optimal schedule to change substantially.
- Slow response:
  - Long computation time.
  - Freezing.
- Bad data:
  - Factory data is often very poor, especially when workers are required to collect it.
  - GIGO
A **heuristic** is a proposed solution to a problem that *seems* reasonable but cannot be rigorously justified.

In reentrant systems, heuristics tend to favor *older* parts.

★ This keeps inventory low.
Heuristics

- Good heuristics deliver good performance.
- Heuristics tend to be simple and intuitive.
  - People should be able to understand why choices are made, and anticipate what will happen.
  - Relevant information should be simple and easy to get access to.
  - Simplicity helps the development of simulations.
Heuristics

- It is often desirable for people to make decisions on the basis of local, current information.
  - Centralized decision-making is most often bureaucratic, slow, and inflexible.
- Most heuristics are naturally decentralized, or can be implemented in a decentralized fashion.
Heuristics

Material/token policies

Performance evaluation

- An operation cannot take place unless there is a token available.
- Tokens authorize production.

- These policies can often be implemented either with finite buffer space, or a finite number of tokens. Mixtures are also possible.
- Buffer space could be shelf space, or floor space indicated with paint or tape.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Heuristics

Material/token policies

Performance evaluation

- Tradeoff between service rate and average cycle time.
Heuristics

Material/token policies

Finite buffer

- Buffers tend to be close to full.
- Sizes of buffers should be related to magnitude of disruptions.
- Not practical for large systems, unless each box represents a set of machines.
Heuristics

Material/token policies

Kanban

- Performance slightly better than finite buffer.
- Sizes of buffers should be related to magnitude of disruptions.
Heuristics

- **Constant Work in Progress**
- Variation on kanban in which the number of parts in an area is limited.
- When the limit is reached, no new part enters until a part leaves.
- Variations:
  - When there are multiple part types, limit work hours or dollars rather than number of parts.
  - Or establish individual limits for each part type.
- If token buffer is not empty, attach a token to a part when $M_1$ starts working on it.
- If token buffer is empty, do not allow part into $M_1$.
- Token and part travel together until they reach last machine.
- When last machine completes work on a part, the part leaves and the token moves to the token buffer.
Heuristics

- Infinite material buffers.
- Infinite token buffer.
- Limited material population at all times.
- Population limit should be related to magnitude of disruptions.
Heuristics

Material/token policies

CONWIP

- Claim: $n_1 + n_2 + \ldots + n_6 + b$ is constant.
Define \( C = n_1 + n_2 + \ldots + n_5 + b \).

Whenever \( M_j \) does an operation, \( C \) is unchanged, \( j = 2, \ldots, 5 \).

\( \star \) ... because \( n_{j-1} \) goes down by 1 and \( n_j \) goes up by 1, and nothing else changes.

Whenever \( M_1 \) does an operation, \( C \) is unchanged.

\( \star \) ... because \( b \) goes down by 1 and \( n_1 \) goes up by 1, and nothing else changes.
Heuristics

Whenever $M_6$ does an operation, $C$ is unchanged.
  ... because $n_5$ goes down by 1 and $b$ goes up by 1, and nothing else changes.

That is, whenever *anything* happens,

$$C = n_1 + n_2 + ... + n_5 + b$$

is unchanged.

$C$ is an invariant.

Here, $C$ is the maximum population of the material in the system.
Heuristics

Finite buffers
Finite material population
Limited material population at all times.
Population and sizes of buffers should be related to magnitude of disruptions.
Heuristics

Material/token policies

CONWIP/Kanban Hybrid

- Production rate as a function of CONWIP population.
- In these graphs, total buffer space (including for tokens) is finite.
- Maximum production rate occurs when population is half of total space.
Heuristics

CONWIP/Kanban Hybrid

- When total space is infinite, production rate increases only.
Simple Policies

Material/token policies

Hedging point

- State: \((x, \alpha)\)
- \(x = \text{surplus} = \text{difference between cumulative production and demand}\)
- \(\alpha = \text{machine state.}\)
  \(\alpha = 1\) means machine up; \(\alpha = 0\) means machine is down.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Simple Policies

Material/token policies

Hedging point

- **Control**: $u$
- $u = \text{short term production rate.}$
  - ★ if $\alpha = 1$, $0 \leq u \leq \mu$;
  - ★ if $\alpha = 0$, $u = 0$. 

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Objective function:

$$\min E \int_0^T g(x(t)) dt$$

where

$$g(x) = \begin{cases} 
g_+x, & \text{if } x \geq 0 \\
-g_-x, & \text{if } x < 0 
\end{cases}$$
Simple Policies

Material/token policies

Hedging point

Dynamics:
\[ \frac{dx}{dt} = u - d \]
\[ \alpha \text{ goes from 0 to 1 according to an exponential distribution with parameter } r. \]
\[ \alpha \text{ goes from 1 to 0 according to an exponential distribution with parameter } p. \]
Simple Policies

Material/token policies

Hedging point

Cumulative Production and Demand

Solution:
- if $x(t) > Z$, wait;
- if $x(t) = Z$, operate at demand rate $d$;
- if $x(t) < Z$, operate at maximum rate $\mu$.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
- The hedging point $Z$ is the single parameter.
- It represents a trade-off between costs of inventory and risk of disappointing customers.
- It is a function of $d$, $\mu$, $r$, $p$, $g_+$, $g_-$. 
Operating Machine $M$ according to the hedging point policy is equivalent to operating this assembly system according to a finite buffer policy.
Simple Policies

- $D$ is a *demand generator*.
  - Whenever a demand arrives, $D$ sends a token to $B$.
- $S$ is a synchronization machine.
  - $S$ is perfectly reliable and infinitely fast.
- $FG$ is a finite finished goods buffer.
- $B$ is an infinite backlog buffer.
Simple Policies

Material/token policies

Basestock

- **Base Stock**: the amount of material and backlog between each machine and the customer is limited.
- Deviations from targets are adjusted locally.
Simple Policies

- Infinite buffers.
- Finite initial levels of material and token buffers.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Claim: \[ b_j + n_j + n_{j+1} + \ldots + n_{k-1} - b_k, \ 1 \geq j \geq k \]
remains constant at all times.
Simple Policies

- Consider $b_1 + n_1 + n_2 + \ldots + n_{k-1} - b_k$
- When $M_i$ does an operation ($1 < i < k$),
  - $n_{i-1}$ goes down by 1, $b_i$ goes down by 1, $n_i$ goes up by 1, and all other $b_j$ and $n_j$ are unchanged.
  - That is, $n_{i-1} + n_i$ is constant, and $b_i + n_i$ is constant.
  - Therefore $b_1 + n_1 + n_2 + \ldots + n_{k-1} - b_k$ stays constant.
- When $M_1$ does an operation, $b_1 + n_1$ is constant.
- When $M_k$ does an operation, $n_{i-1} - b_k$ is constant.
- Therefore, when any machine does an operation, $b_1 + n_1 + n_2 + \ldots + n_{k-1} - b_k$ remains constant.
Simple Policies

Material/token policies

Basestock Proof

- Now consider $b_j + n_j + n_{j+1} + ... + n_{k-1} - b_k$, $1 < j < k$

- When $M_i$ does an operation, $i \geq j$,
  $b_j + n_j + n_{j+1} + ... + n_{k-1} - b_k$ remains constant, from the same reasoning as for $j = 1$.

- When $M_i$ does an operation, $i < j$,
  $b_j + n_j + n_{j+1} + ... + n_{k-1} - b_k$ remains constant, because it is unaffected.
Simple Policies

Material/token policies

Basestock Proof

- When a demand arrives,
  - $n_j$ stays constant, for all $j$, and all $b_j$ increase by one.
  - Therefore $b_j + n_j + n_{j+1} + \ldots + n_{k-1} - b_k$ remains constant for all $j$.

- **Conclusion:** whenever any event occurs, $b_j + n_j + n_{j+1} + \ldots + n_{k-1} - b_k$ remains constant, for all $j$. 

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Simple Policies

Comparisons

- Simulation of simple Toyota feeder line.
- We simulated all possible kanban policies and all possible kanban/CONWIP hybrids.
- The graph indicates the best of each.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
More results of the comparison experiment: best parameters for service rate = .999.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Buffer sizes</th>
<th>Base stocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite buffer</td>
<td>2  2  4  10</td>
<td>— — — —</td>
</tr>
<tr>
<td>Kanban</td>
<td>2  2  4  9</td>
<td>— — — —</td>
</tr>
<tr>
<td>Basestock</td>
<td>∞ ∞ ∞ ∞</td>
<td>1 1 1 12</td>
</tr>
<tr>
<td>CONWIP</td>
<td>∞ ∞ ∞ ∞</td>
<td>— — — 15</td>
</tr>
<tr>
<td>Hybrid</td>
<td>2  3  5  15</td>
<td>— — — 15</td>
</tr>
</tbody>
</table>
More results of the comparison experiment: performance.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Service level</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite buffer</td>
<td>0.99916 ± .00006</td>
<td>15.82 ± .05</td>
</tr>
<tr>
<td>Kanban</td>
<td>0.99909 ± .00005</td>
<td>15.62 ± .05</td>
</tr>
<tr>
<td>Basestock</td>
<td>0.99918 ± .00006</td>
<td>14.60 ± .02</td>
</tr>
<tr>
<td>CONWIP</td>
<td>0.99922 ± .00005</td>
<td>14.59 ± .02</td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.99907 ± .00007</td>
<td>13.93 ± .03</td>
</tr>
</tbody>
</table>
Simple Policies

- First-In, First Out.
- Simple conceptually, but you have to keep track of arrival times.
- Leaves out much important information:
  - due date, value of part, current surplus/backlog state, etc.
Simple Policies

- Earliest due date.
- Easy to implement.
- Does not consider work remaining on the item, value of the item, etc.
Simple Policies

- **Shortest Remaining Processing Time**
- Whenever there is a choice of parts, load the one with least remaining work before it is finished.
- Variations: include waiting time with the work time. Use expected time if it is random.
Simple Policies

- Widely used, but many variations. One version:
  - Define $\text{CR} = \frac{\text{Processing time remaining until completion}}{\text{Due date} - \text{Current time}}$
  - Choose the job with the highest ratio (provided it is positive).
  - If a job is late, the ratio will be negative, or the denominator will be zero, and that job should be given highest priority.
  - If there is more than one late job, schedule the late jobs in SRPT order.

Copyright ©2001 Stanley B. Gershwin. All rights reserved.
Simple Policies

- This policy considers a part’s due date.
- Define \textit{slack} = \text{due date} - \text{remaining work time}
- When there is a choice, select the part with the least slack.
- Variations involve different ways of estimating remaining time.
Simple Policies

- Due to Eli Goldratt.
- Based on the idea that every system has a bottleneck.
- **Drum**: the common production rate that the system operates at, which is the rate of flow of the bottleneck.
- **Buffer**: DBR establishes a CONWIP policy between the entrance of the system and the bottleneck. The buffer is the CONWIP population.
- **Rope**: the limit on the difference in production between different stages in the system.
- But: What if bottleneck is not well-defined?
Conclusions

- Many policies and approaches.
- No simple statement telling which is better.
- Policies are not all well-defined in the literature or in practice.
- My opinion:
  ★ This is because policies are not derived from first principles.
  ★ Instead, they are tested and compared.
  ★ Currently, we have little intuition to guide policy development and choice.