



lean

software development

Thrashing

What it is, what causes it, & what to do about it.



Buried in Work?



You are not alone!

Good organizations that have solved most of their problems always seem to have one problem left:

Thrashing l e a n



What is Thrashing?

Thrashing is the term used to describe a degenerate situation on a computer where increasing resources are used to do a decreasing amount of work.

Usually it refers to two or more processes accessing a shared resource repeatedly such that serious system performance degradation occurs because the system is spending a *disproportionate* amount of time just *accessing* the shared resource.

Resource access time may be considered wasted, since it does not contribute to the advancement of any process.





What are you doing about it?



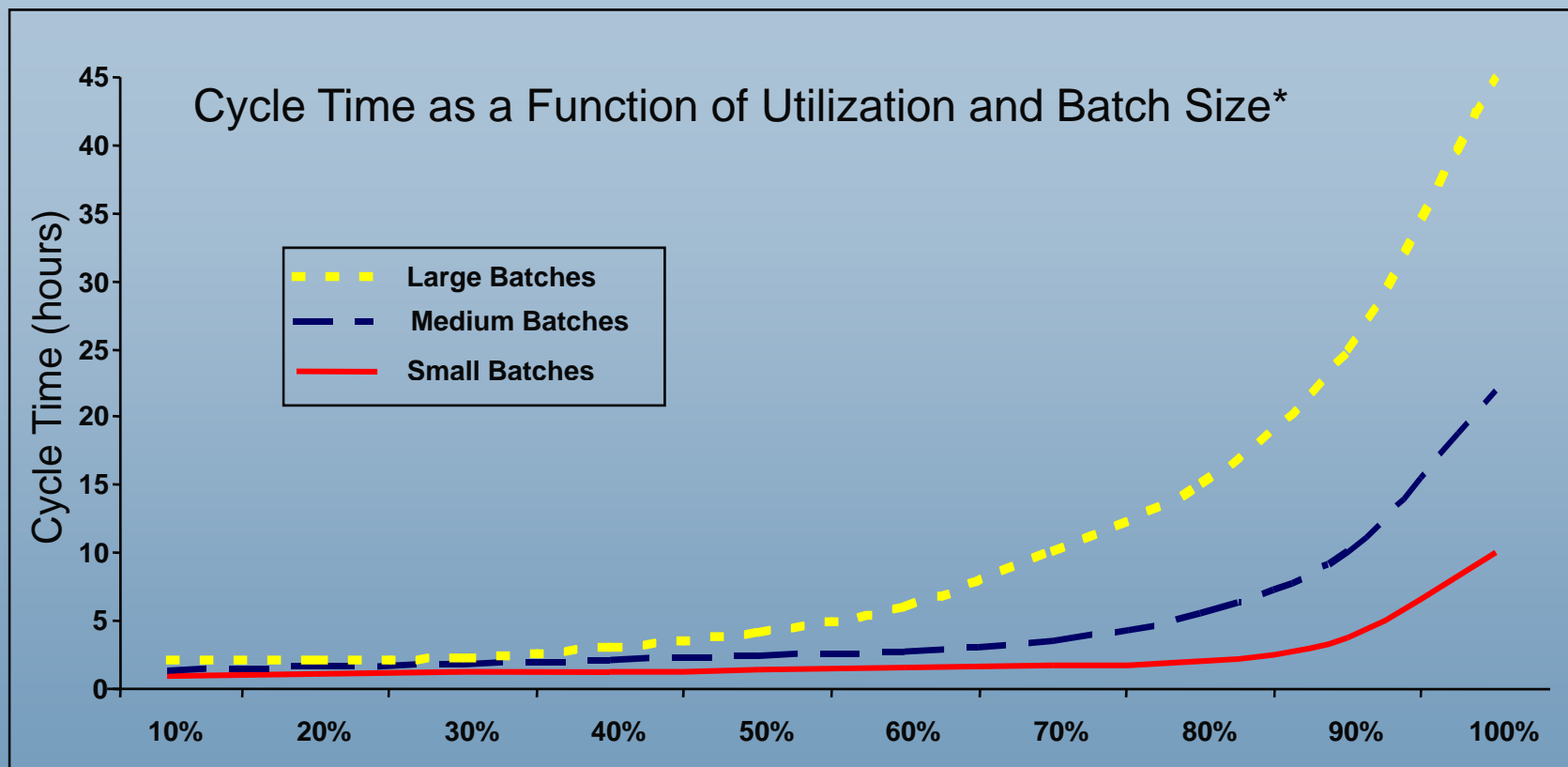
Ignore it and hope it will go away?

*Insanity: Continuing to do the same thing
and expecting different results.*



What Causes Thrashing?

Overload



High Performance

Thrashing

*This assumes batch size is proportional to variability.



Reducing Cycle Time

Limit Work To Capacity

- ✓ Timebox, Don't Scopebox
- ✓ Pull – Don't Push



Queuing Theory 101



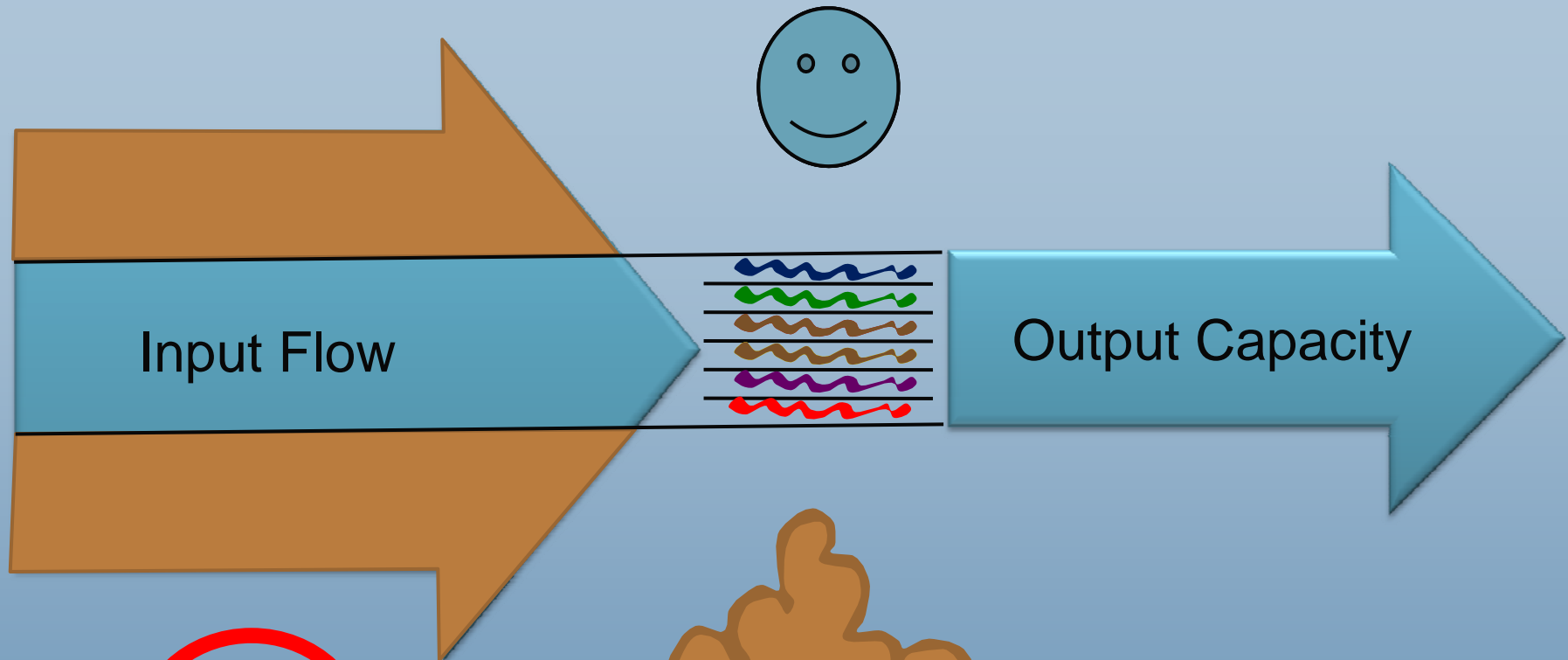
Timebox, Don't Scopebox



Ask NOT: How long will this take?

Ask instead: What can be done by this date?

Pull Scheduling Small Requests





What Causes Thrashing?



Unevenness

Little's Law

Time Through the System –
Number of Things in Process

Average Completion Rate



Empire State Building

September 22, 1929
Demolition started

January 22, 1930
Excavation started

March 17, 1930
Construction started

November 13, 1930
Exterior completed

May 1, 1931
Building opened

Exactly on time
18% under budget



One Year Earlier:

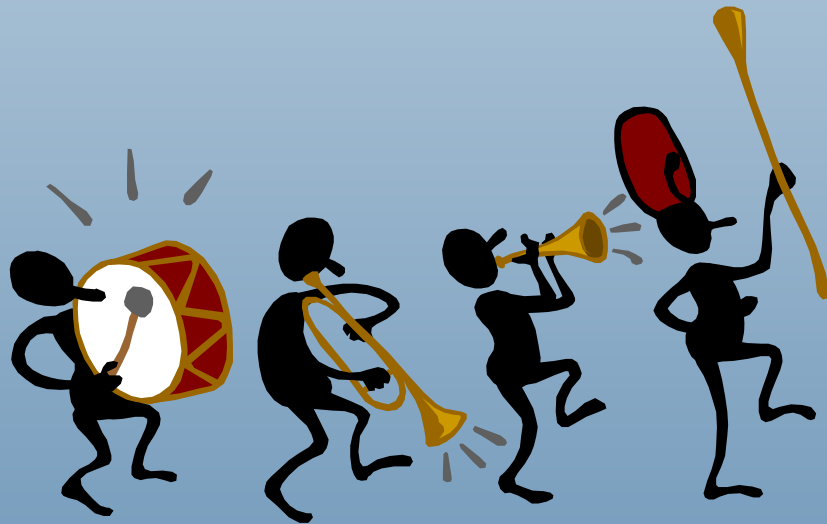


How did they do it? The key: Focus on FLOW.

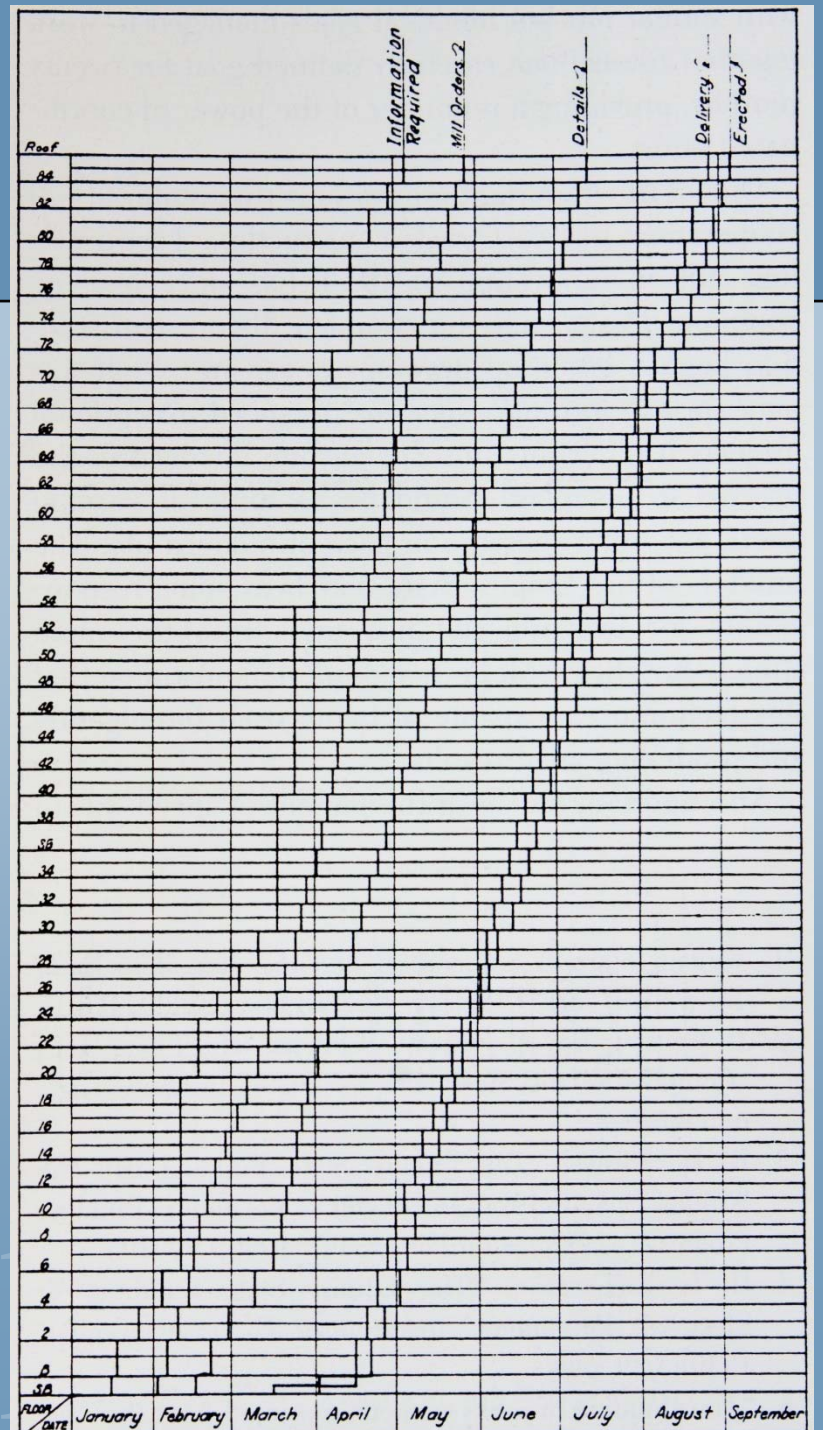


Steel Schedule

We thought of the work as if it were a band marching through the building and out the top.



From: "Building the Empire State"
Builders Notebook: Edited by Carol Willis





The Four Pacemakers

1. Structural Steel
 - ✓ Completed September 22, 12 days early
2. Concrete Floor
 - ✓ Completed October 22, 6 days early
3. Exterior Metal Trim and Windows
 - ✓ Completed October 17, 35 days early
4. Exterior Limestone
 - ✓ Completed November 13, 17 days early



From: "Building the Empire State"
Builders Notebook: Edited by Carol Willis



Reducing Cycle Time

Limit Work To Capacity

- ✓ Timebox, Don't Scopebox
- ✓ Pull – Don't Push

Level the Workload

- ✓ Even out the Arrival of Work
- ✓ Establish a Regular Cadence

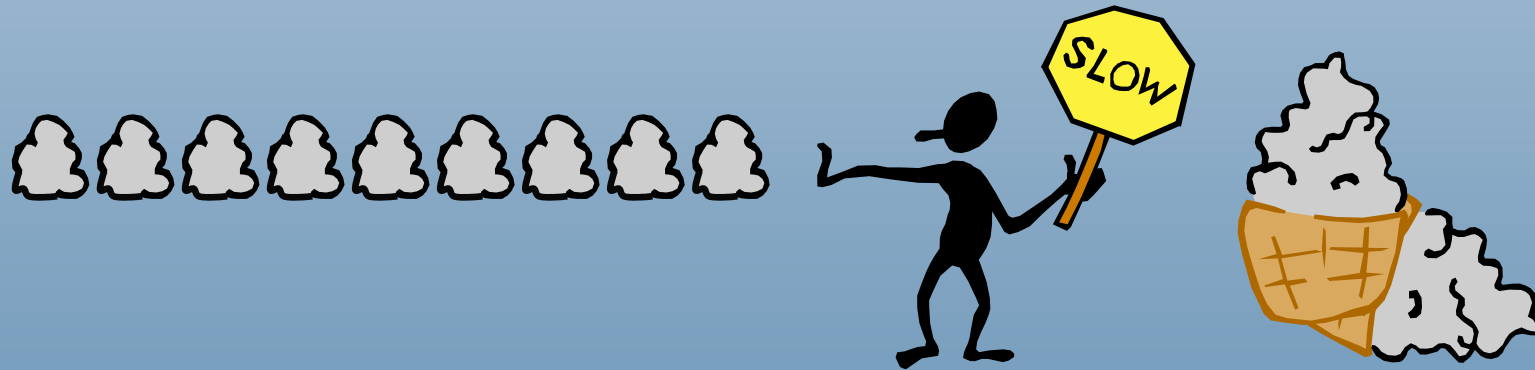


Queuing Theory 101

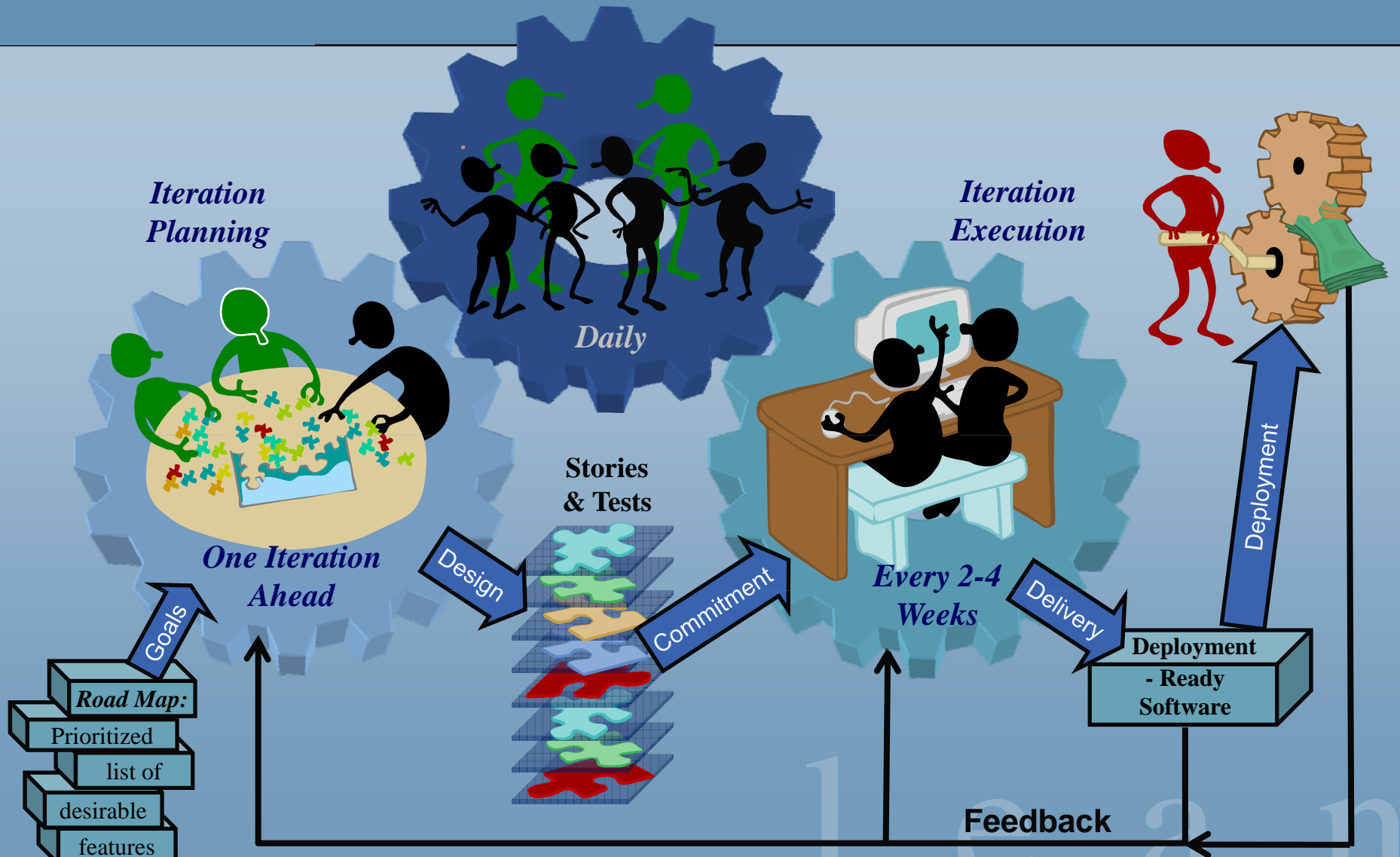


Even out the Arrival of Work

*Most Budgeting, Planning, & Approval Processes
Turn Even Demand into Lumpy Demand.*



Establish a Regular Cadence





Reducing Cycle Time

Limit Work To Capacity

- ✓ Timebox, Don't Scopebox
- ✓ Pull – Don't Push

Level the Workload

- ✓ Even out the Arrival of Work
- ✓ Establish a Regular Cadence

Optimize Throughput – Not Utilization

- ✓ Minimize the Size of Things In Process
- ✓ Minimize the Number of Things In Process



Queuing Theory 101



Set-up Time Drives Batch Size

Manufacturing

Common Knowledge:

- ✓ Die changed have a huge overhead
- ✓ Don't change dies very often

Toyota Production System:

- ✓ Economics requires frequent die change
- ✓ *One Digit Exchange of Die*

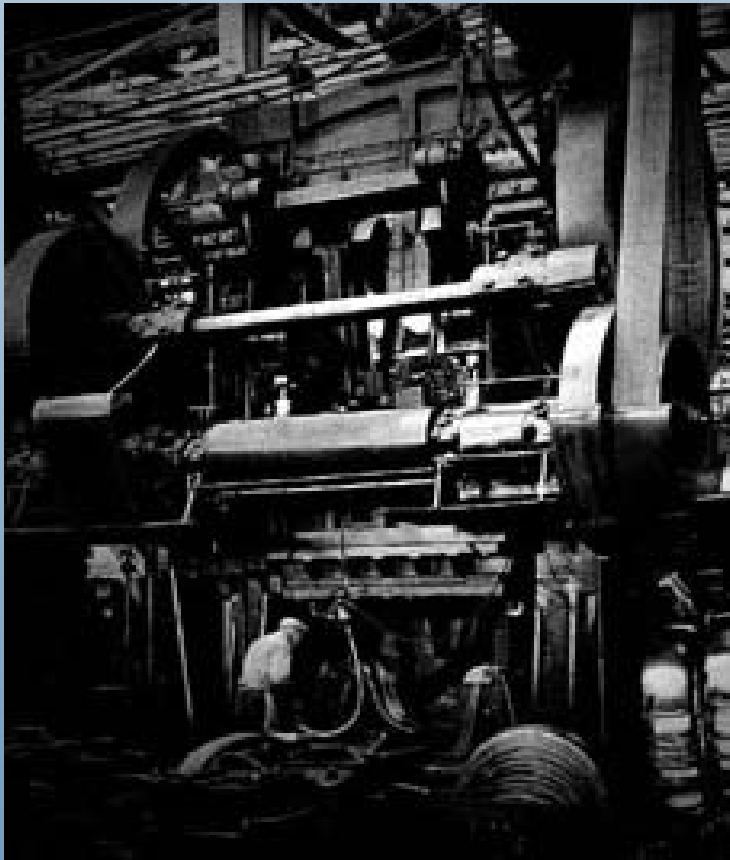
Software Development

Common Knowledge:

- ✓ Releases have a huge overhead
- ✓ Don't release very often

Lean:

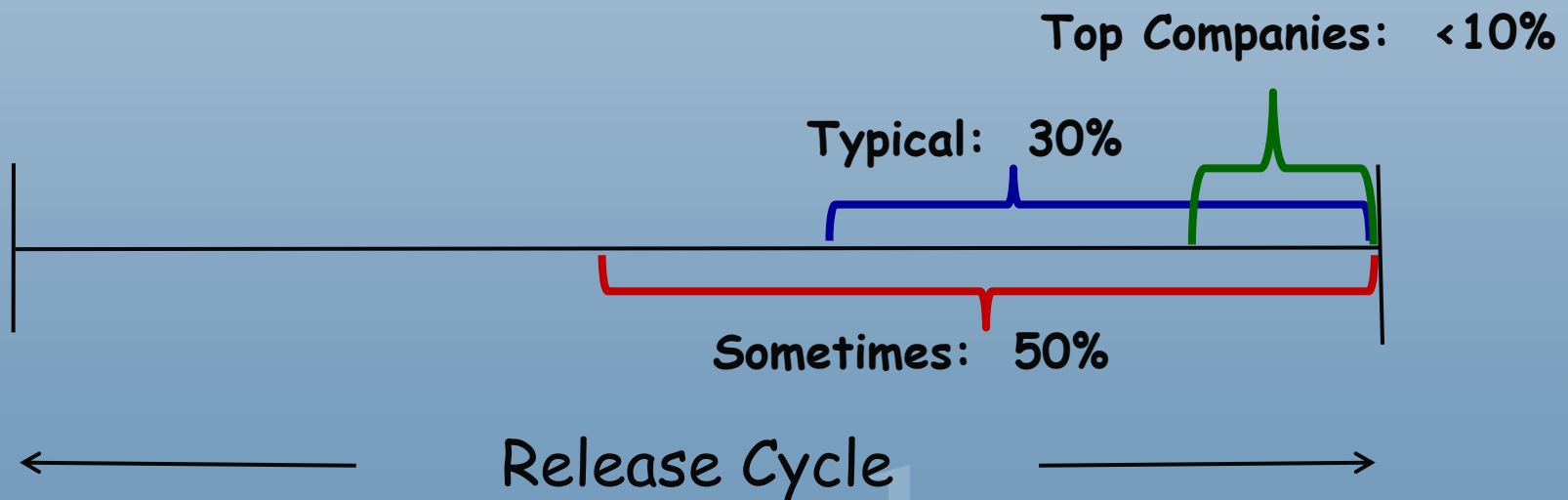
- ✓ Economics requires many frequent releases
- ✓ *Reduce the Overhead: One Digit Releases*





How Good are You?

When in your release cycle do you try to freeze code and test the system? What percent of the release cycle remains for this “hardening”?



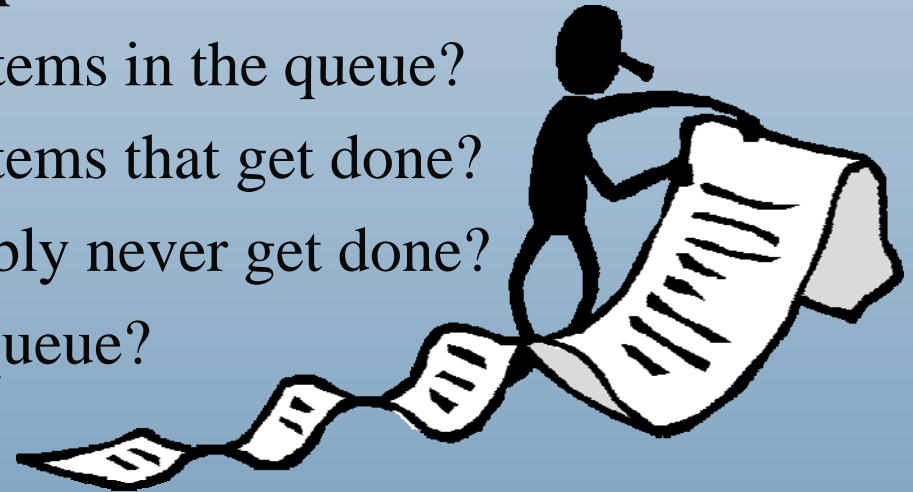


Don't Batch & Queue!

We have far too many things to do!

Date stamp items as the request comes in.

1. What is the average age of items in the queue?
2. What is the average age of items that get done?
3. How many items will probably never get done?
4. Why are these items in the queue?



Queues are buffers between departments that keep people from having to talk to each other!



l e a n

software development

Thank You!

More Information: www.poppendieck.com