l e a n

software development

Rethinking Scheduling

What if Computers are the Problem, not the Solution?

mary@poppendieck.com Mary Poppendieck www.poppendieck.com

1930
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

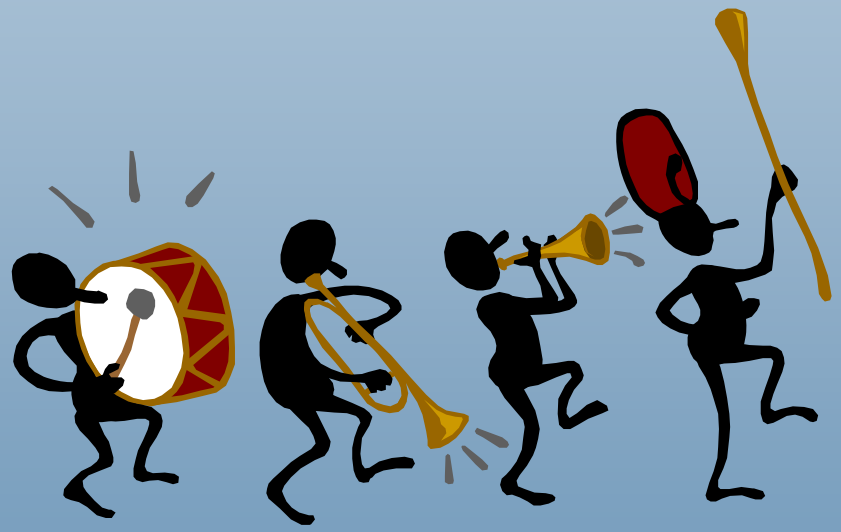
Exactly on time
18% under budget

One Year Earlier:

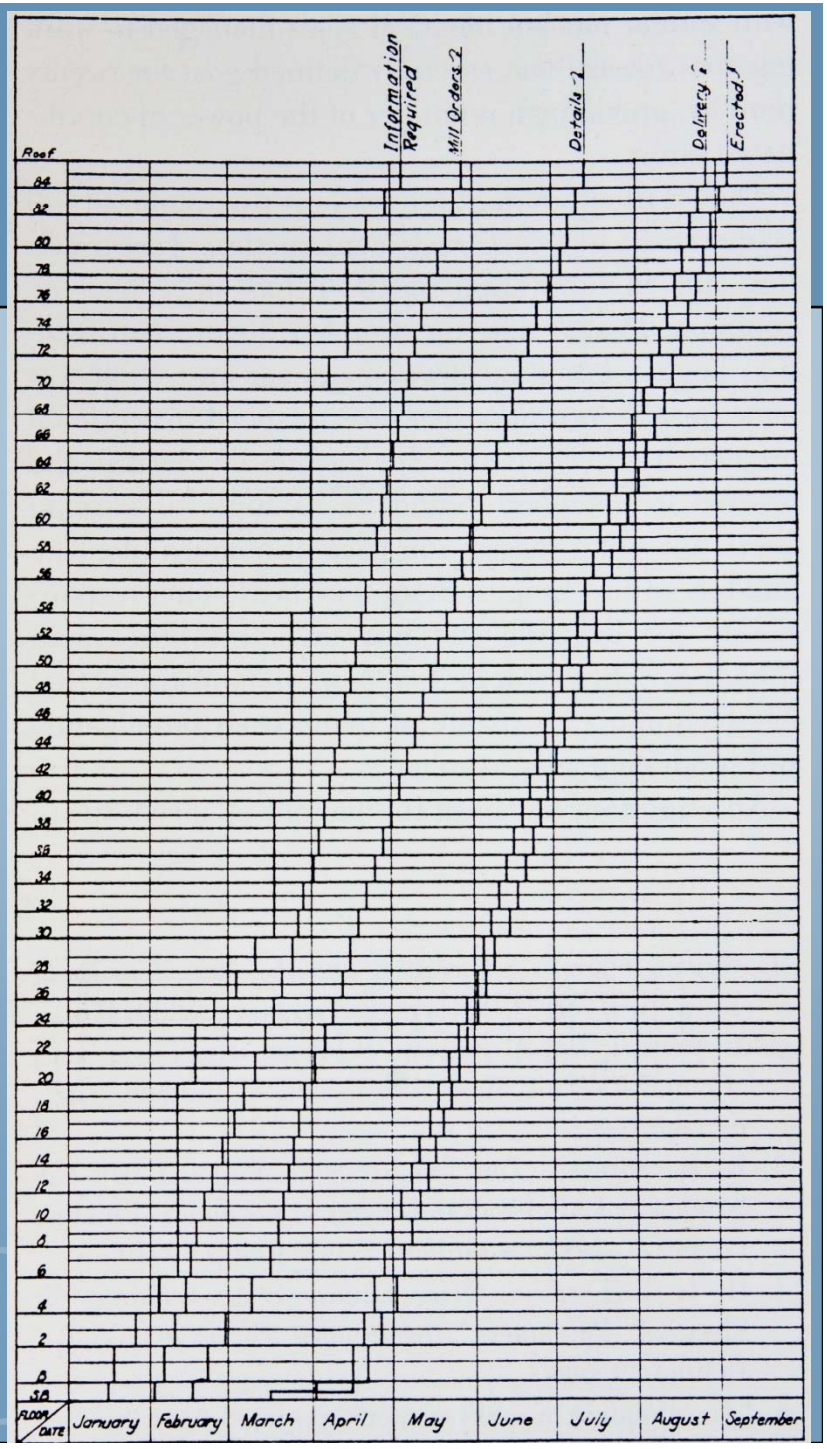
How did they do it? The key: Focus on FLOW.

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



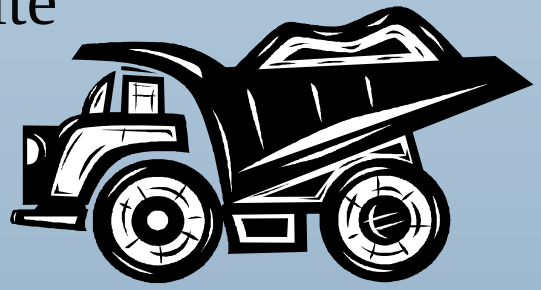
The Four Pacemakers

1. Structural Steel Erection
   - ✓ Completed September 22, 12 days early
2. Concrete Floor Construction
   - ✓ 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

Additional Success Factors

1. Focus on the constraint: Material Flow
   - ✓ 500 trucks a day – no storage on site
2. Eliminate all possible waste
   - ✓ Bricks dumped into hopper
   - ✓ Rail cars used to deliver material
   - ✓ Restaurant for workers
3. Teamwork of owner, architect, and builder
   - ✓ Fast-track Construction
4. Designed for construction
   - ✓ Example: Electrical connections
5. Break all possible dependencies
   - ✓ Steel ledge for stone, interior construction



Schedule the Constraint

What is the constraint?

- Information about the problem to be solved?
- Understanding the best solution to the problem?
- Available Developers?
- Available Equipment?
- What is YOUR constraint?

Use a pacemaker schedule for the constraint.

Early 1980's
From MRP to Kanban

Crisis at our Video Cassette Plant

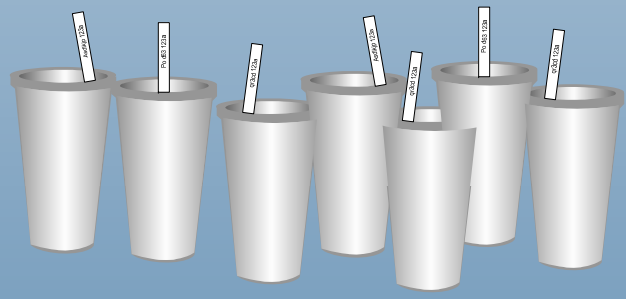
- ✓ Competition selling cassettes for less than we could make them

Our Response

- ✓ Just-in-Time (Lean) Production

The Great Coffee Cup Simulation

- ✓ Plant Manager, Materials Control Manager, IT Manager (me)
- ✓ Add Production General Managers
- ✓ Add General Supervisors
- ✓ Add Shift Supervisors
- ✓ Add Shift Workers



The Result

- ✓ Entire plant switched to pull scheduling over one weekend
- ✓ Pack-out went from 60% to 95% the first week
- ✓ Employees said this was the best thing that happened to the plant

Case Study: Development is the Constraint

Speed to market

- ✓ 45 releases a year – for 4 years
  - × Maintenance releases take a week
  - × New feature releases take a month
  - × New applications take a quarter

patientkeeper

Jeff Sutherland
CTO PatientKeeper

Predictable Delivery

- ✓ Every iteration ends in a release
  - × Live at all targeted customer sites
- ✓ Never a late release in 4 years

Work is Limited to Capacity

- ✓ All work is specified by product managers in fine-grained "stories"
- ✓ Work on stories starts when it is assigned to a release by a product manager
- ✓ No more than 3 releases are active at once: Maintenance, Feature, App.
- ✓ Time to complete each story is re-estimated every day by developers
- ✓ If work exceeds available time, FEATURES MUST BE REMOVED!
- ✓ Priorities realigned weekly by CEO with product & account management

*New Car Development at Toyota**

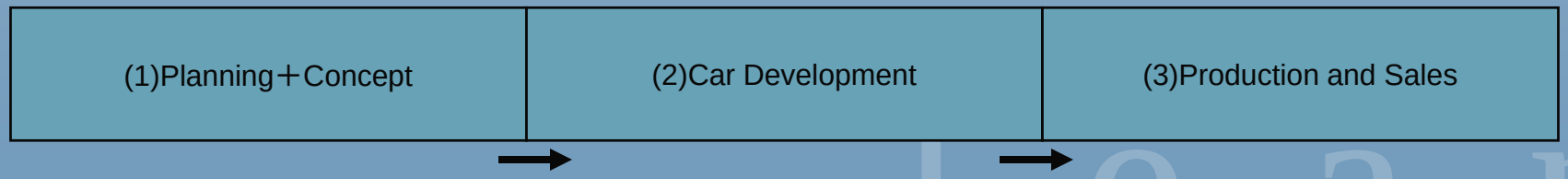
Nobuaki Katayama – Chief Engineer of Lexus/SC, IS, & Altezza

- ✓ Developers Summit 2008 in February in Tokyo

Three phase of development:

1. Planning and concept development
   - ✓ Concept, Style, Market Research, Pre-development, Cost & Profit Targets
2. Real car development
   - ✓ Designing, Prototyping, Evaluation
3. Production and sales

* Presented by Kenji Hiranabe Agile 2008, Toronto, 8/2008



Learn First

Phase 1: Planning and Concept Development

Concept, Style, Market Research,
Pre-development, Cost & Profit Targets

- ✓ Project Backbone
- ✓ Clear appealing points
- ✓ Refine until satisfaction
  - × "Perfect" concept of the product
- ✓ Discuss multiple options from wide point of view
- ✓ Walk through the whole
- ✓ Rules to avoid immature "go"
- ✓ Calm evaluation with a third party

From "New Car Development at Toyota" by Nobuaki Katayama – Toyota Chief Engineer – Developers Summit 2008, Tokyo, 2/2008

Presented by Kenji Hiranabe Agile 2008, Toronto, 8/2008

Large Products

Phase 1 -Planning/Concept

- ✓ "Do it until it's done"

Phase 2: Car Development

- ✓ "Do it until it's time"

From "New Car Development at Toyota" by Nobuaki Katayama – Toyota Chief Engineer – Developers Summit 2008, Tokyo, 2/2008

Presented by Kenji Hiranabe Agile 2008, Toronto, 8/2008

Development Process Management

- ✓ Strong dedicated leader
  - × And organizations that support him
- ✓ Clear milestones
  - × Support people to achieve them
  - × Avoid proceeding before achieving them
- ✓ Mechanisms to avoid "too late"
  - × Honest communication
  - × Bad news first
- ✓ Early and reliable backup plan

Think Products, not Projects

Projects

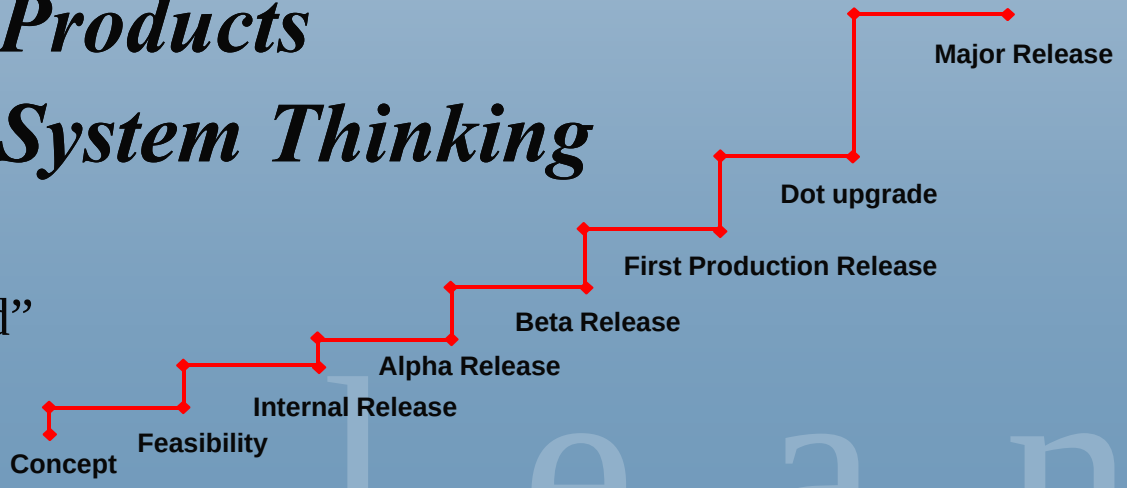
- ✓ Up-front funding
- ✓ Scope fixed at onset
- ✓ Success = cost/schedule/scope
- ✓ Projects have an "end"
  – Successful Software does not

Batch Funding →
Batch Thinking

Products

- ✓ Incremental funding
- ✓ Scope is expected to evolve
- ✓ Success = profit/market share
- ✓ Successful Products don't "end"
- ✓ Team usually stays with the Product

System Thinking



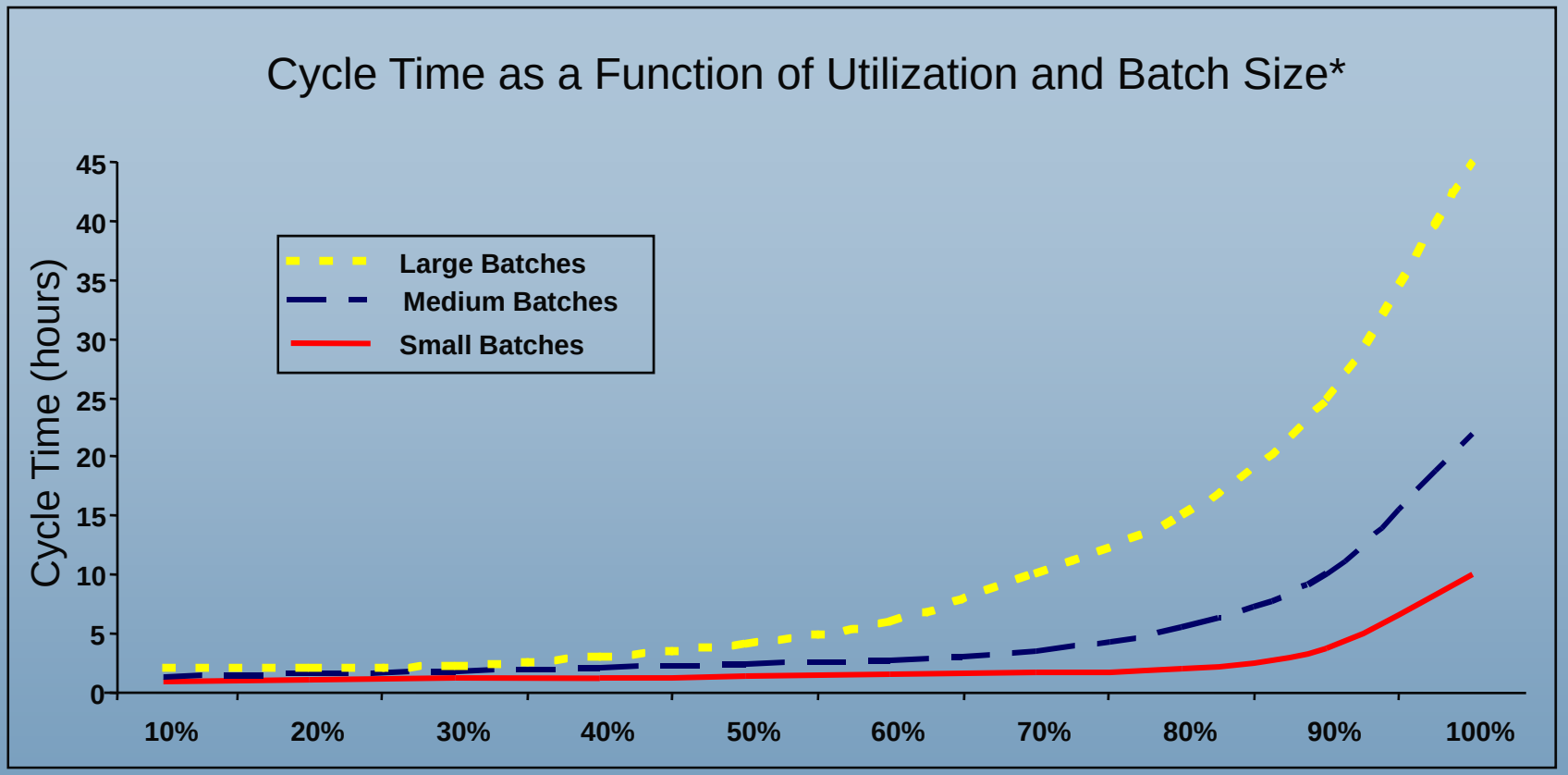
Queuing Theory

Little's Law

Time Through the System =

$$\frac{\text{Number of Things in Process}}{\text{Average Completion Rate}}$$

The Utilization Paradox



High Performance ***Thrashing***

*This assumes batch size is proportional to variability.

Reducing Cycle Time

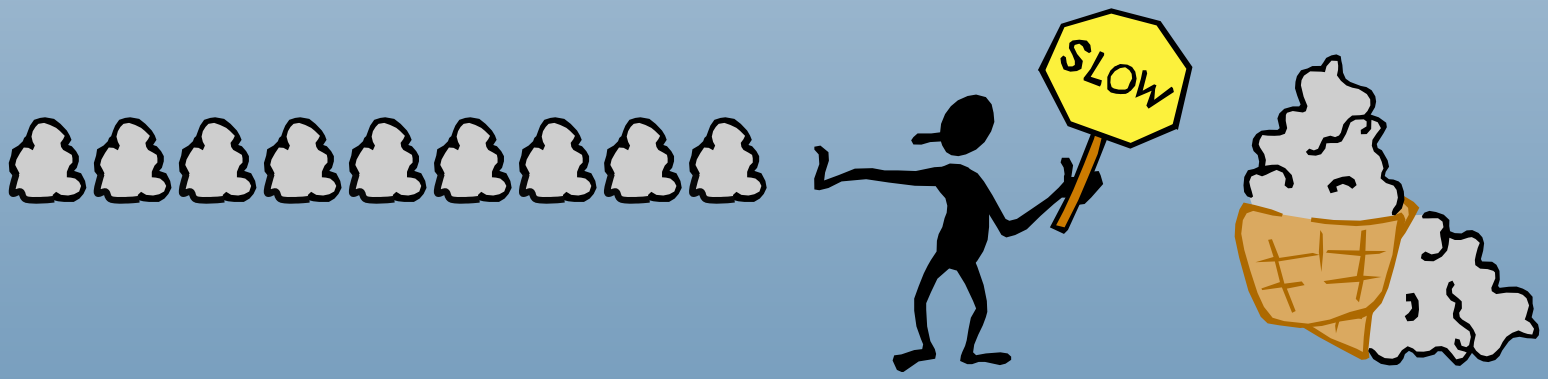
Level the Workload

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

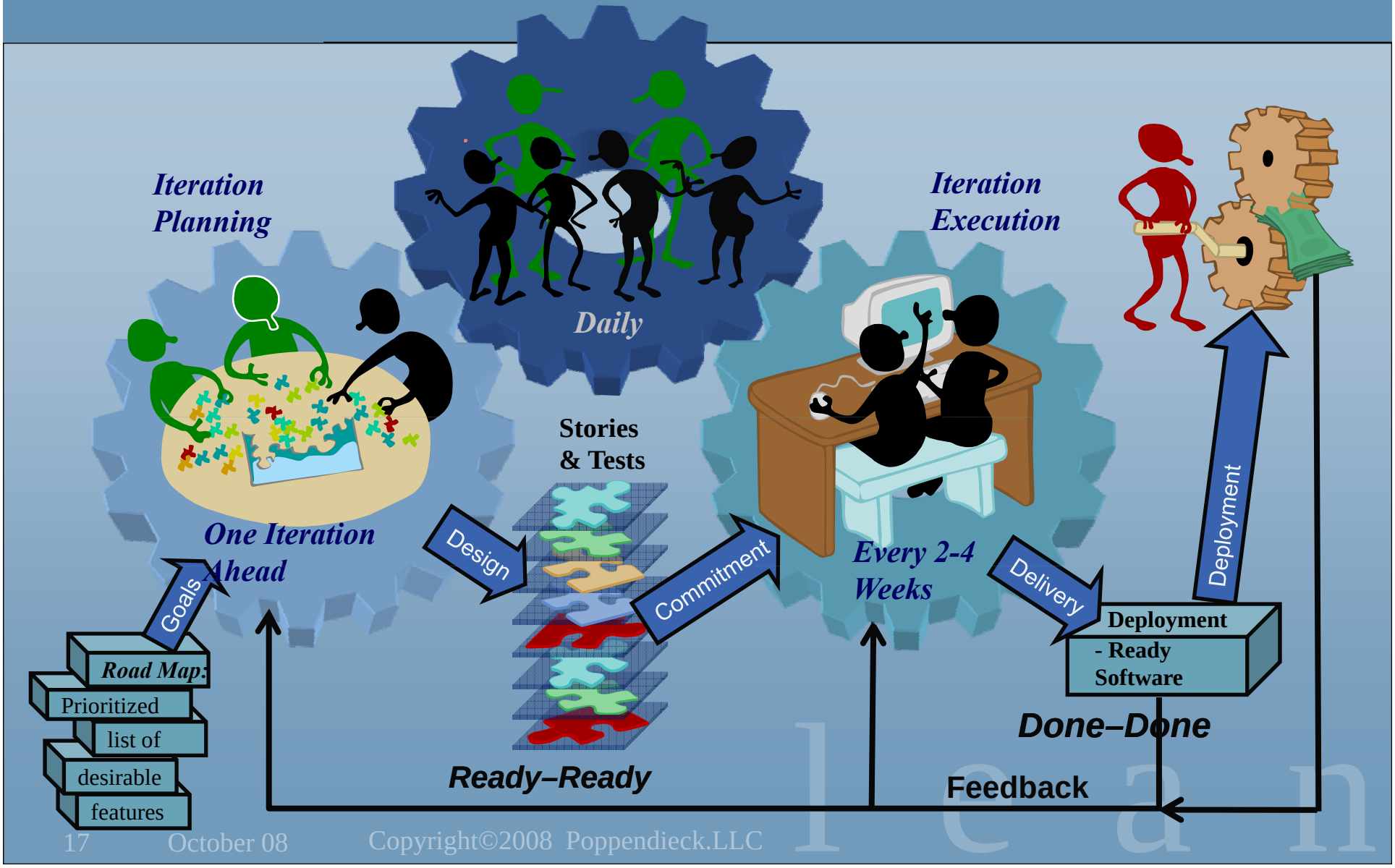
Queuing Theory 101

Even out the Arrival of Work

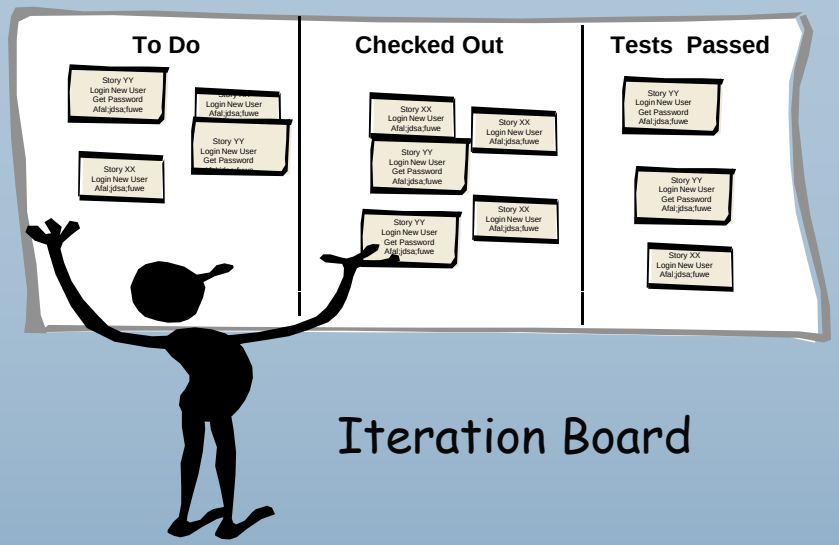
Most Budgeting, Planning, & Approval Processes Turn Even Demand into Lumpy Demand.



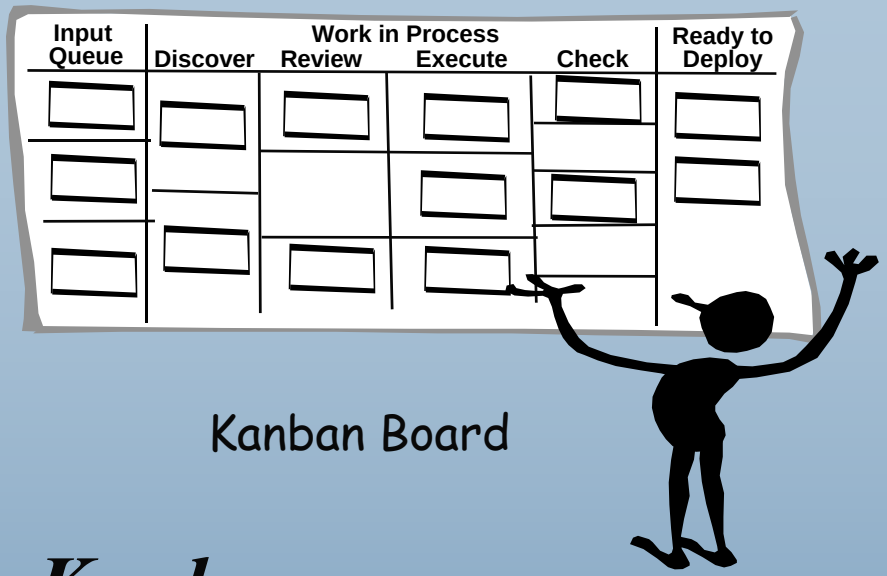
Leveling the Workload with Iterations



Iterations or Kanban?



Iteration Board



Kanban Board

Iteration

- ✓ Fixed iteration length: 2-4 weeks
- ✓ Team commits to finish a set of work based on its velocity.
- ✓ Change inside iterations discouraged
- ✓ Track commitment

Kanban

- ✓ Fixed release cycle: 1-2 weeks
- ✓ One piece flow
- ✓ Limit # of items in each phase
- ✓ Whatever is done is released
- ✓ Track average cycle time / item

Reducing Cycle Time

Level the Workload

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

Limit Work To Capacity

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

Queuing Theory 101

Timebox, Don't Scopebox

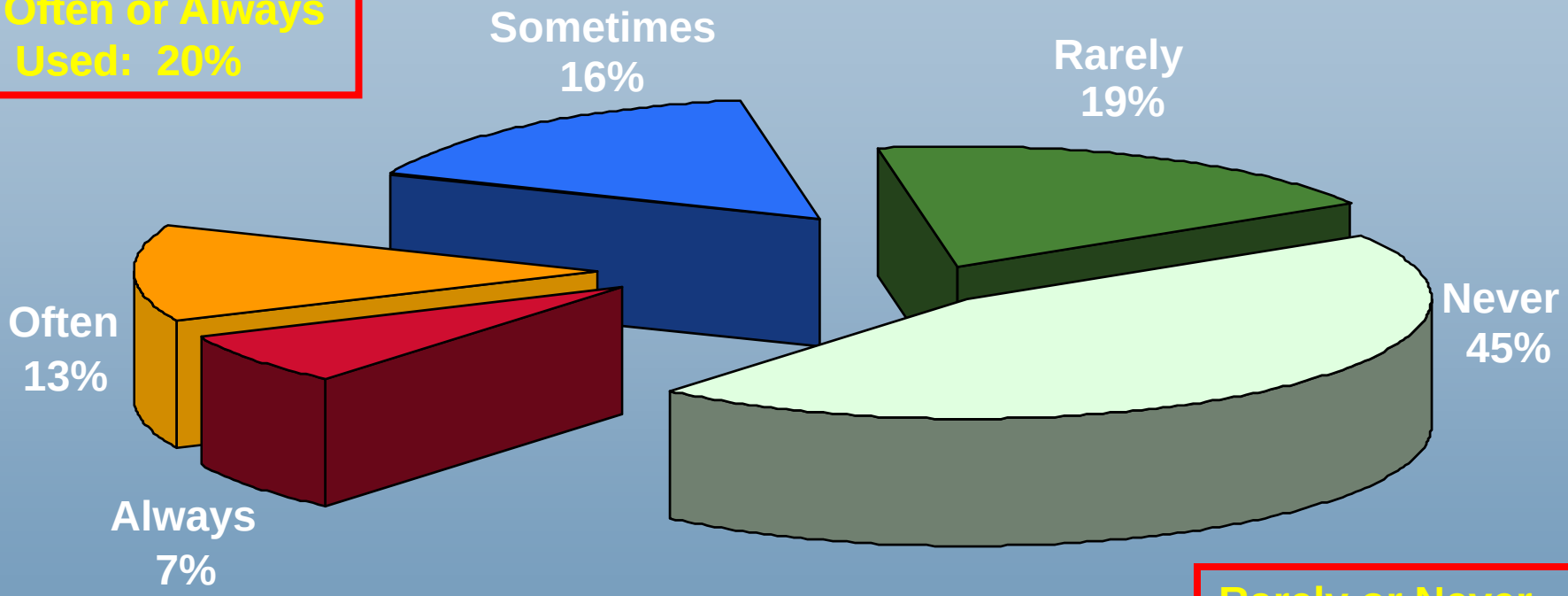
Ask NOT: ***How long will this take?***

Ask instead: ***What can be done by this date?***

Extra Features

Features and Functions Used in a Typical System

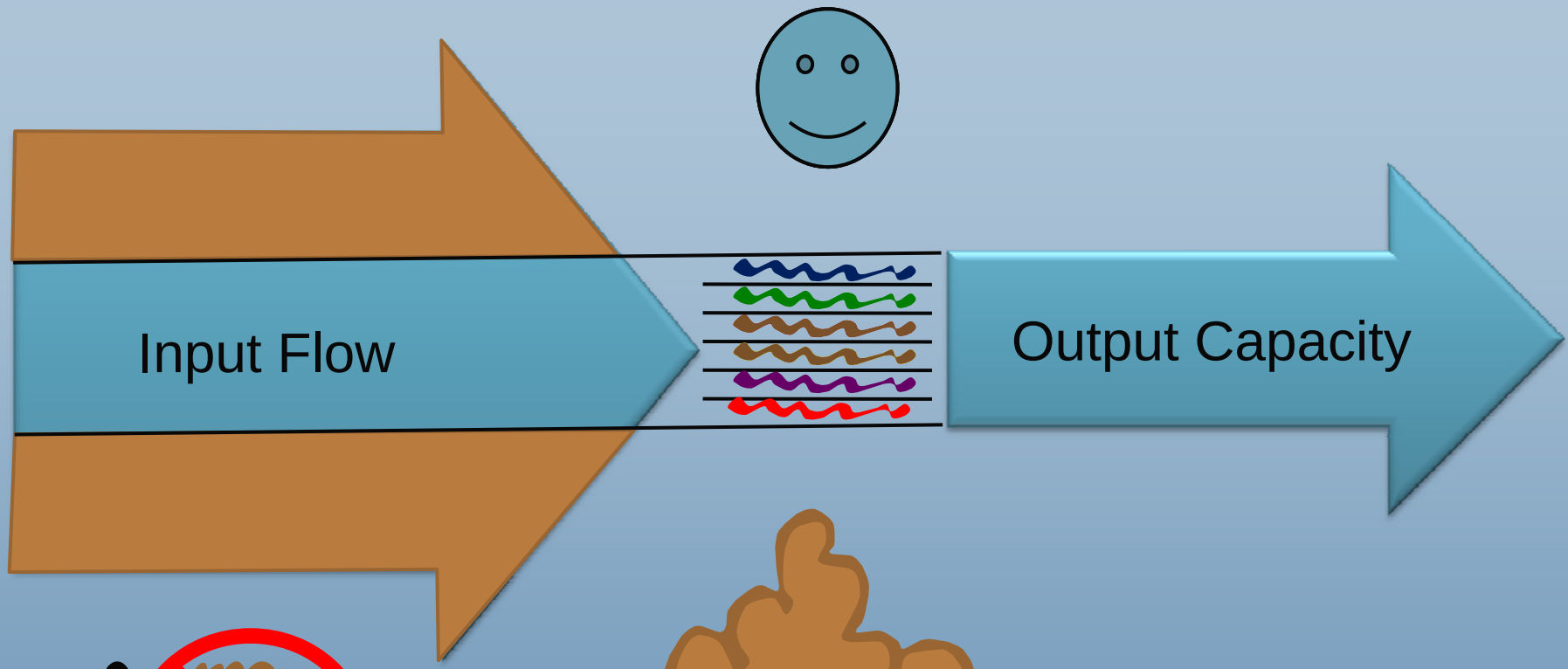
Often or Always Used: 20%



Rarely or Never Used: 64%

Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Pull Scheduling
Small Requests



Pull Scheduling: Larger Systems

8/1 Start

9/15 Release 1: Feature Set 1

11/1 Release 2 Feature Set 2

12/15 Release 3 Feature Set 3

2/1 Release 4 Feature Set 4

3/15 Release 5 Feature Set 5

5/1 Release 6 Feature Set 6

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Aug Sep Oct Nov Dec Jan Feb Mar Apr May

Start
Product Concept Approved

3 Months
Knowledge Review:
Customer Interest
Technical Approach

6 Months
Knowledge Review:
Proof of Concept
Alpha Release Decisions

9 Month Timebox
Knowledge Review:
Alpha Release
Beta Release Decisions

12 Month Timebox
Knowledge Review:
Beta Release
1st Release Decisions

15 Months
First Production Release

lean

Reducing Cycle Time

Level the Workload

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

Limit Work To Capacity

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

Optimize Throughput – Not Utilization

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

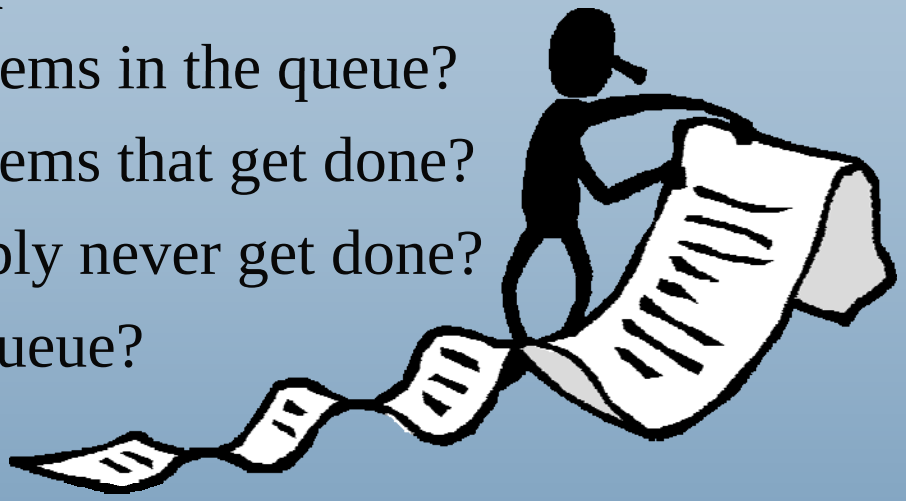
Queuing Theory 101

Minimize the Number of Things-in-Process

We have far too many things to do!

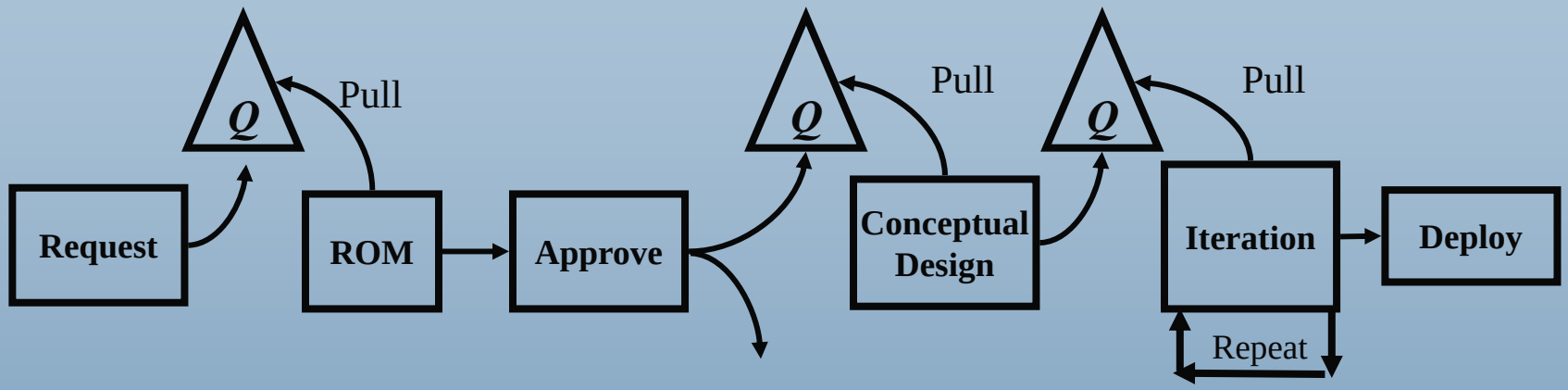
Date stamp items as the request comes in.

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!

Manage the Queues

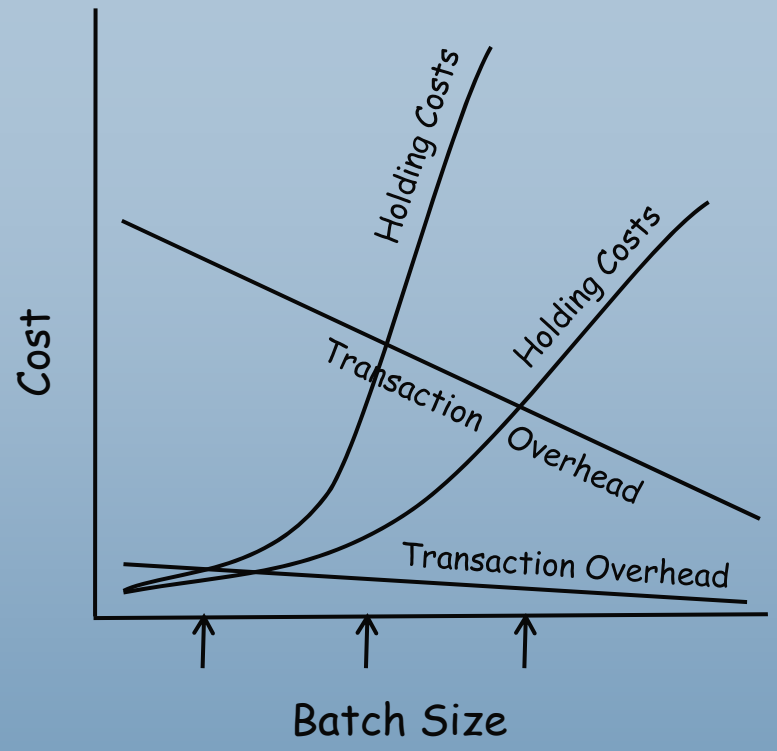


- Pull from Queues to limit work to capacity
- Keep Queues small to increase cycle time

Batch Size

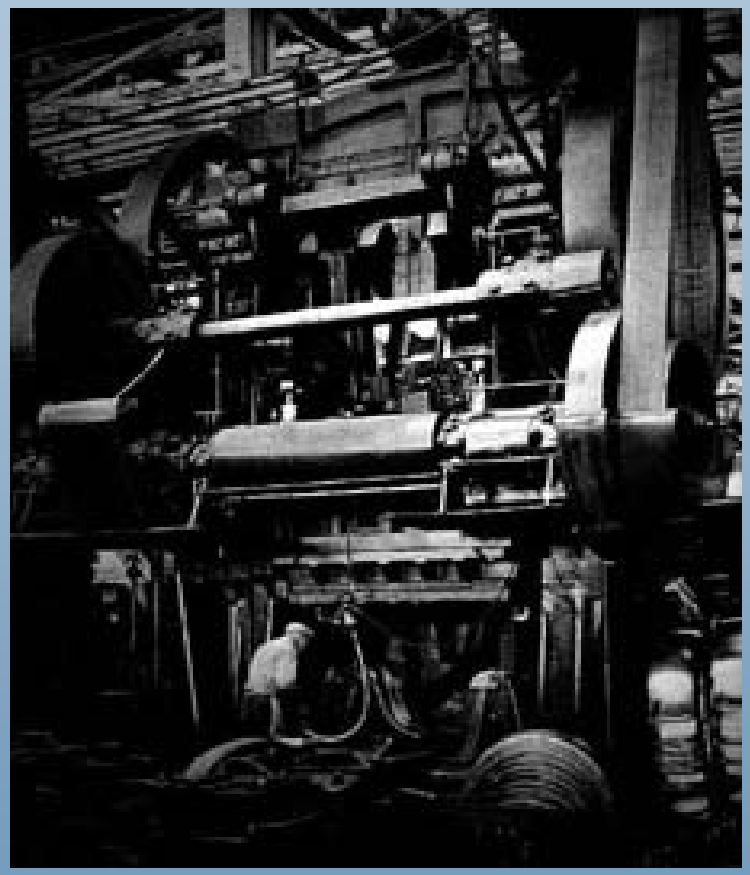
What is Batch Size?

- ✓ The amount of information transferred at one time
  - × The % of specifications completed before development begins
  - × The amount of code tested in a system test
- ✓ Compare:
  - × Cost of transaction overhead (linear)
    - ❖ Test set-up and execution
  - × Cost of holding (hyperbolic)
    - ❖ Find/fix defects long after injection
- ✓ Holding costs are:
  - × Usually hidden & delayed
  - × Often larger than expected
- ✓ The Lean Approach:
  - × Recognize true waiting costs
  - × Drive down transaction overhead



From Don Reinertsen
www.roundtable.com/MRTIndex/LeanPD/ART-reinertsen-INT2-1.html

Small Batches Need Short Set-up Times



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:

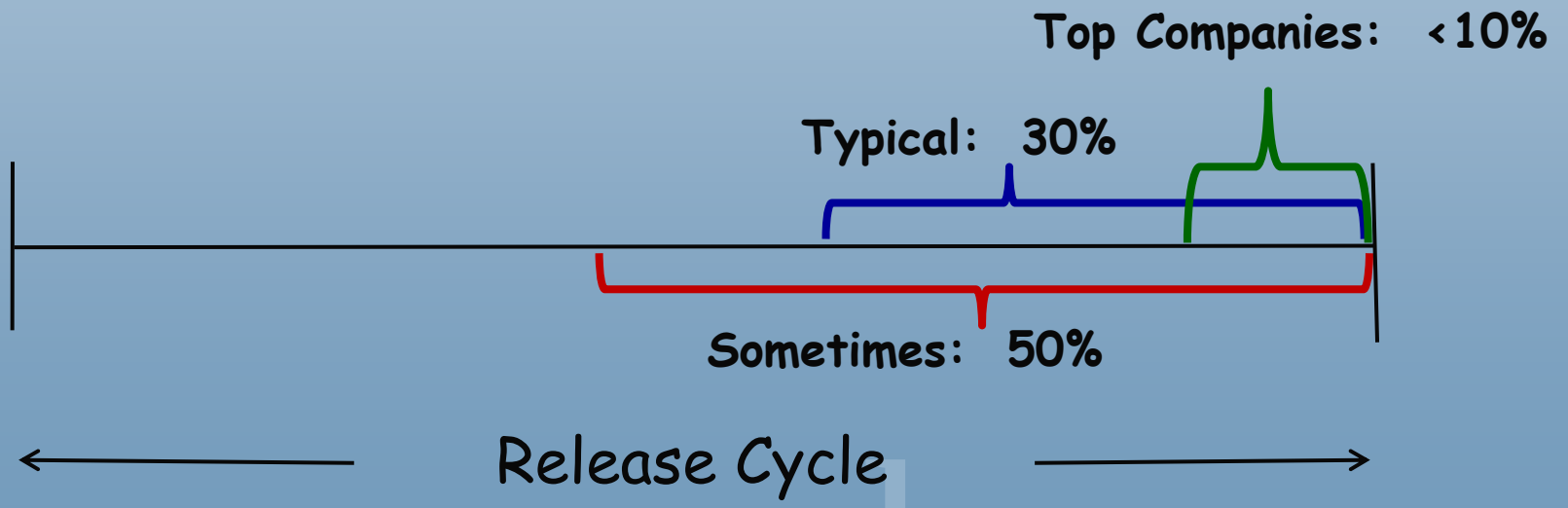
- ✓ Releases have a huge overhead
- ✓ Don't release very often

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 Tolerate Defects!

A Quality Process Builds Quality IN

- ✓ Rather than trying to test quality in later.

Where do we get the idea that it is okay to find defects at the end of the development process?

- ✓ *We categorize & track defects because we find them in big batches – why not find them as they are injected?*

Change The System

Mistake-Proof Every Step

- ✓ Detect defects the moment they occur

Automated Test at the Functional Level with Continuous Integration are REQUIRED!

Relentless Improvement

KAIZEN

Kai

改

Change

Zen

善

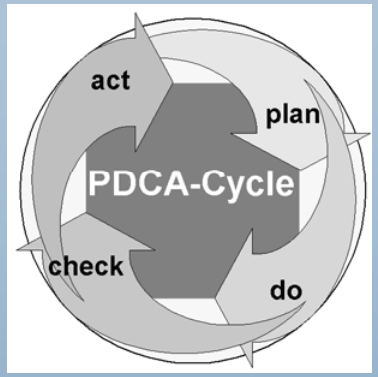
Good

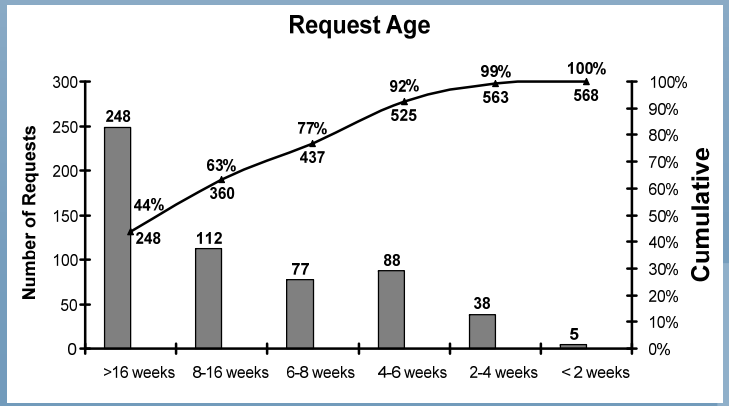
Solve One Problem at a Time

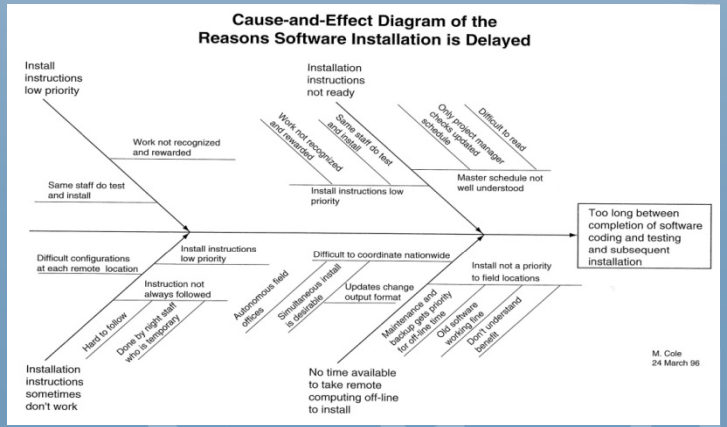
Data-Based Problem Analysis

- ✓ What's Important?
  - ✗ Pareto Analysis
- ✓ Root Cause Analysis
  - ✗ Ishikawa (Fishbone) Diagram
  - ✗ Five Why's?

Many Quick Experiments







l e a n

software development

Thank You!

More Information: www.poppendieck.com

mary@poppendieck.com Mary Poppendieck www.poppendieck.com