



l e a n
software development

Lean Software Development

What is Waste?

What is Waste?

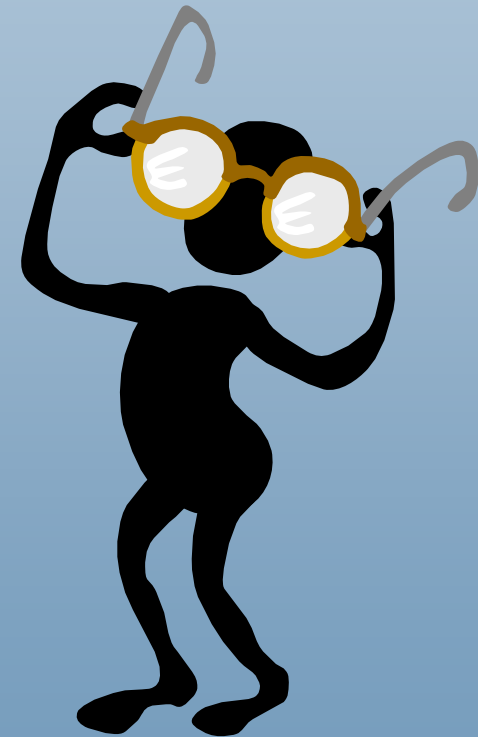


Waste is...

Anything that depletes resources of time, effort, space, or money without adding customer value.

Value is...

Seen through the eyes of those who pay for, use, support, or derive value from our systems.



Put on Customer Glasses

What is Waste?



Put yourself in the shoes
of an airline executive:

Are empty seats waste?

✗ What do you think?

Airline Statistics:

Available Seat Miles (ASM)

✓ A Capacity Measurement

Revenue Seat Miles (RSM)

✓ A Production Measurement

Load Factor = RSM/ASM

✓ A Utilization Measurement

Re-phrasing the question:

*Will a higher load factor
increase profits?*



Case Study



Customers Love Southwest
 Excellent Customer Service
 Reliable Performance
 Point-to-Point Routing
 Consistent Low Fares
 No “Nuisance Charges”
 Lots of Flight Options

*By the Numbers**

Most Punctual
 Lost the Least Bags
 Had the Fewest Complaints
 Rated “Most Admired” US Airline
 35 Consecutive Years of Profitability
 Operates at the Lowest Load Factor

* Sources:
 FAA Statistics
 SEC Filings
 Fortune Rating

Carrier	Flights	On-Time Performance		Load Factors				Cancelled	Diverted
	2007	2007	2006	2007	2006	2005	2004	2007	2007
WN - Southwest Airlines	1,164,906	81.72%	84.66%	72.60%	73.10%	70.70%	69.50%	0.43%	0.12%
DL -Delta Air Lines	568,862	75.65%	75.72%	80.60%	78.50%	76.50%	74.70%	1.18%	0.21%
CO - Continental Airlines	411,105	72.95%	76.54%	81.40%	80.70%	78.90%	76.90%	0.92%	0.35%
NW - Northwest Airlines	480,382	71.23%	79.02%	83.90%	84.00%	81.50%	79.20%	2.04%	0.13%
UA - United Airlines	594,488	69.90%	74.79%	82.70%	82.10%	81.50%	79.30%	2.28%	0.10%
US - US Airways	402,568	68.70%	78.51%	75.30%	78.30%	75.50%	75.10%	1.65%	0.08%
AA - American Airlines	792,404	68.34%	78.33%	81.50%	80.00%	78.60%	74.80%	2.73%	0.09%

Eliminate Waste



The Seven Wastes of Manufacturing - Taiichi Ohno

1. Overproduction
2. Transportation
3. Work In Process
4. Extra Processing
5. Motion
6. Waiting
7. Defects

The 7 Wastes of Software Development

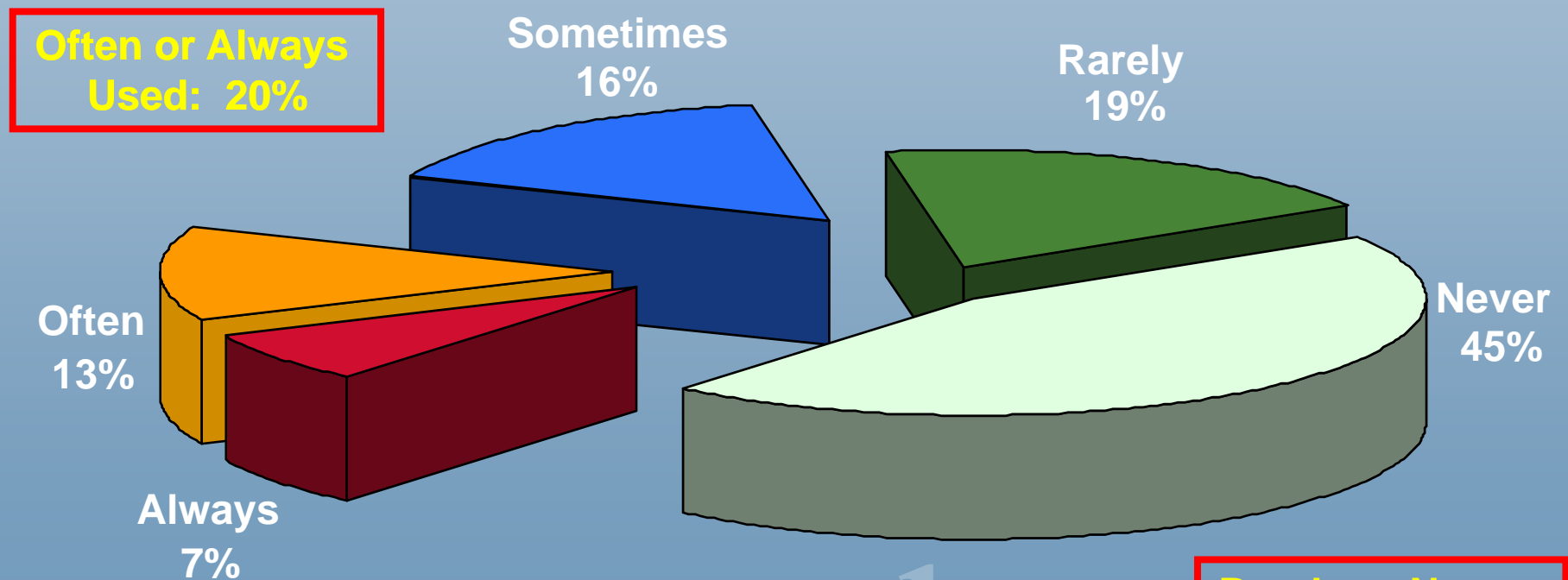
1. Extra Features
2. Handovers
3. Work In Process
4. Failure Demand
5. Task Switching
6. Delays
7. Defects

Extra Features



The Biggest Waste of Software Development

Features and Functions Used in a Typical Custom System



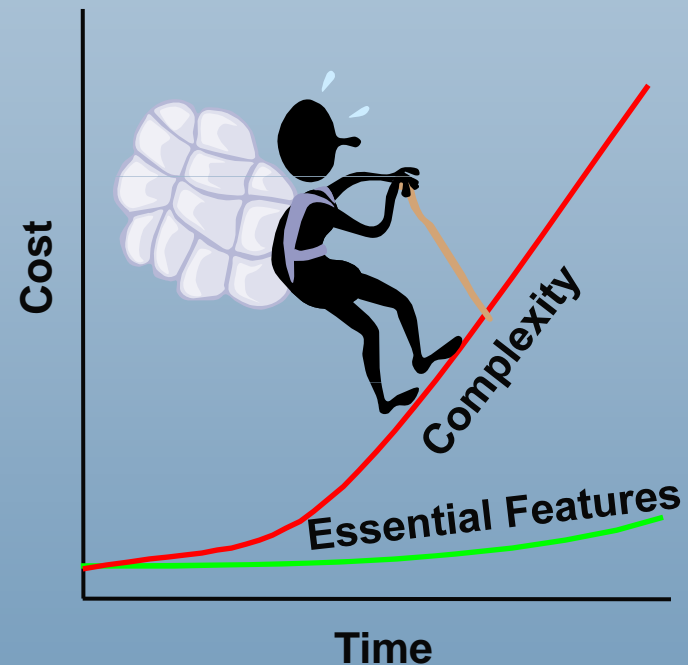
Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Complexity



The Silent Killer of Profits and Growth*

1. Justify Every Feature
Complexity Costs
2. Deploy Minimum
Useful Feature Sets
In Priority Order
3. Don't Automate Complexity
Simplify First!
4. Don't measure productivity in
lines of code or function points.
Measure Value!



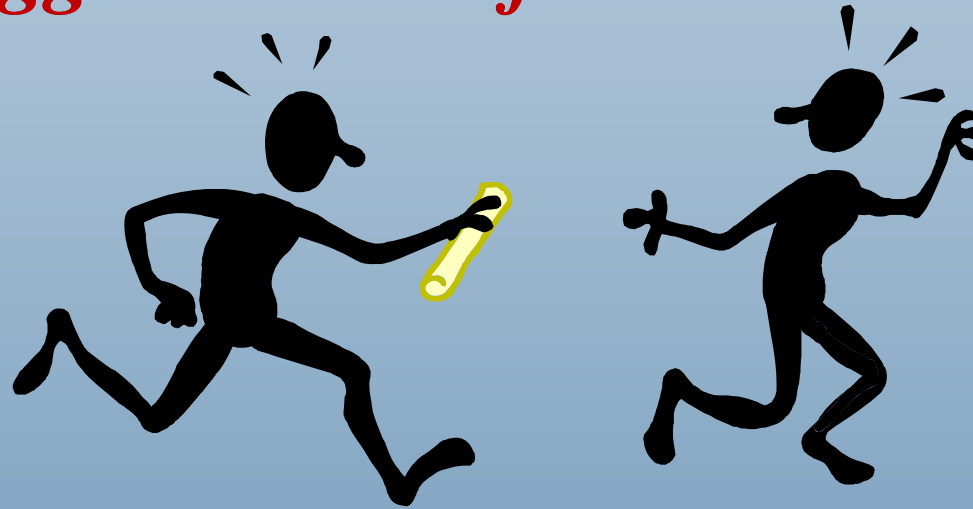
Keep it Simple!

*Michael George and Stephen Wilson,
"Conquering Complexity in Your Business"

Handovers



*The Biggest Waste of Product Development**

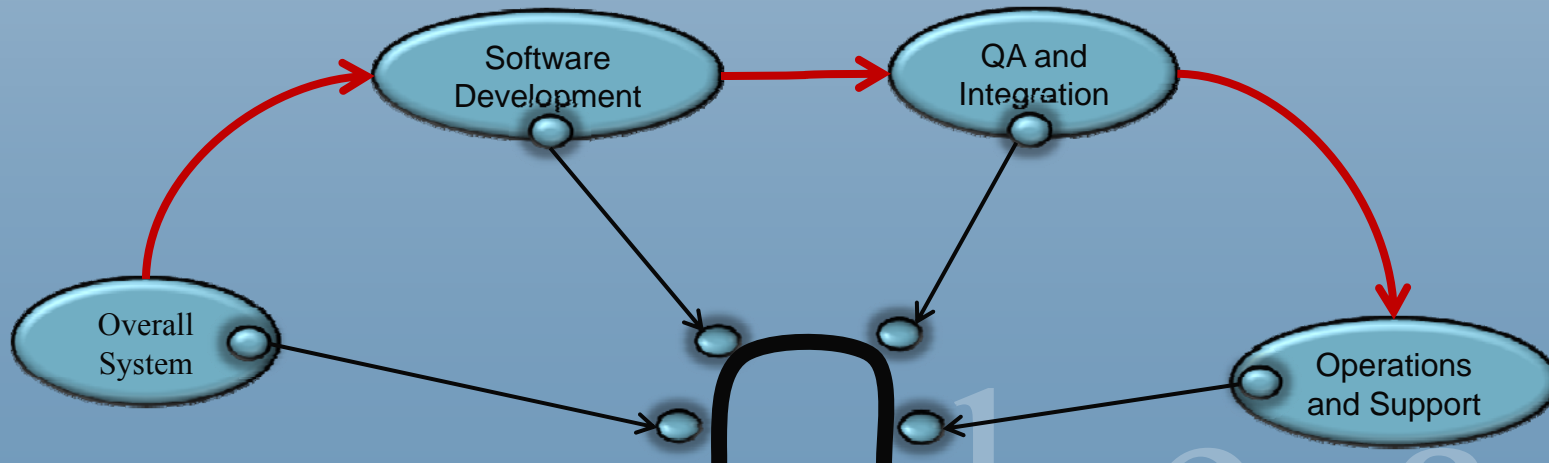
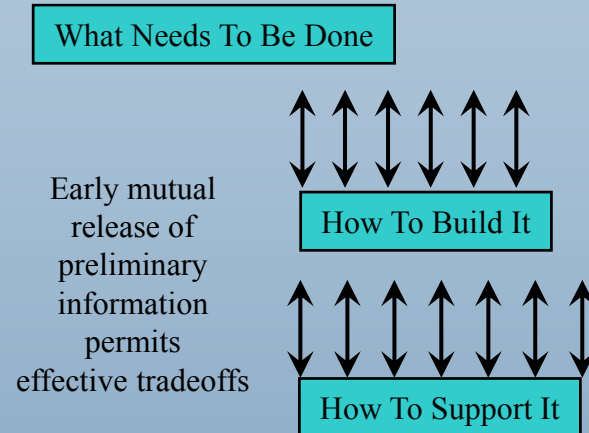
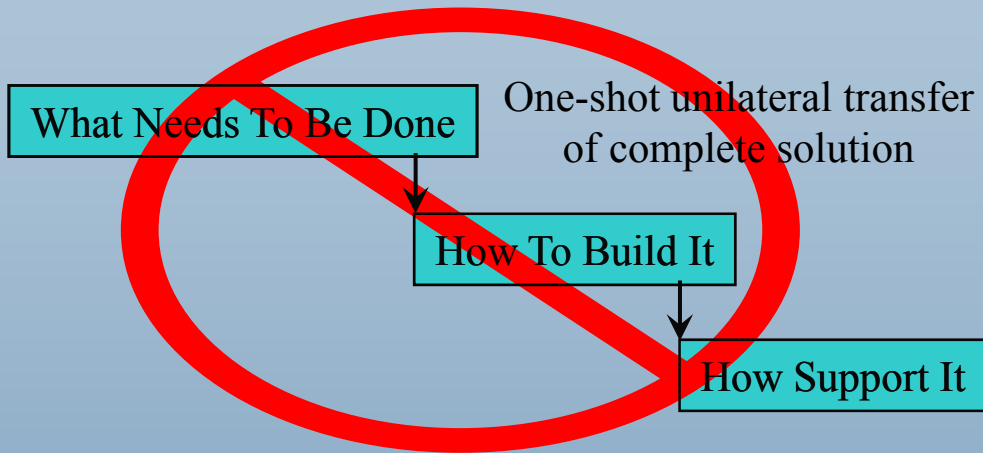


A handover occurs whenever we separate:*

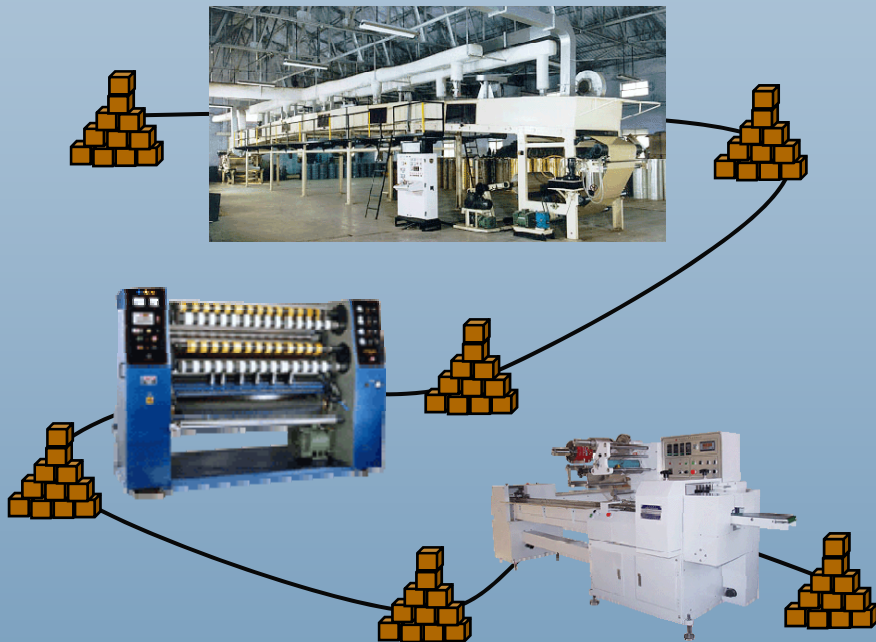
- ✓ Responsibility – What to do
- ✓ Knowledge – How to do it
- ✓ Action – Actually doing it
- ✓ Feedback – Learning from results

*Allen Ward “Lean Product and Process Development”

Cross-functional Teams

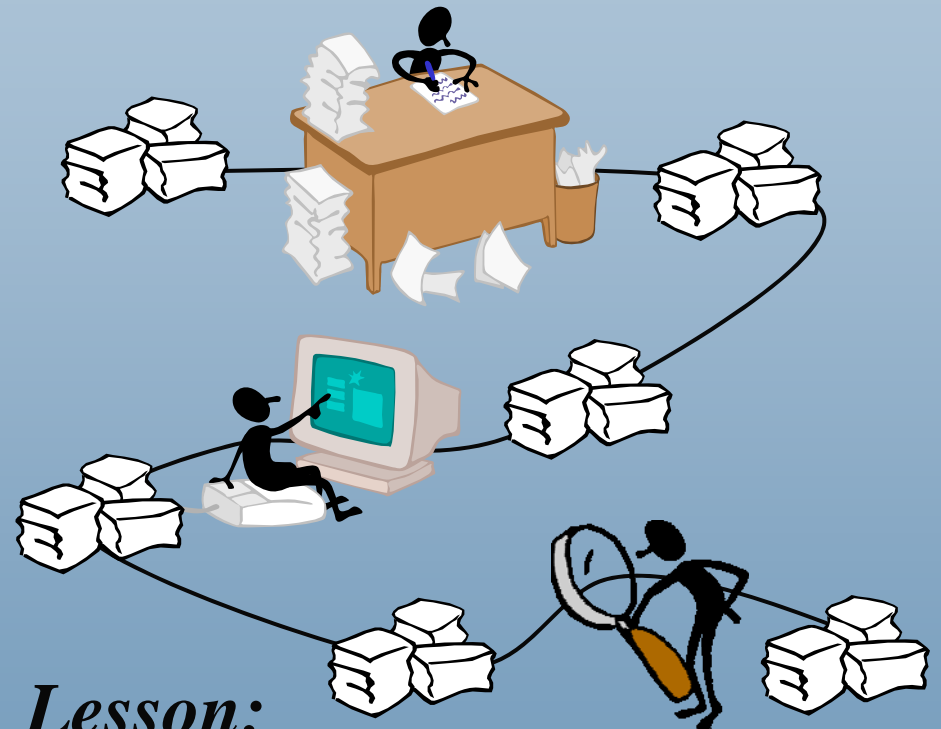


Work in Process



Lesson:

Stop trying to maximize machine productivity.



Lesson:

Stop trying to maximize “resource” utilization.

Work-In-Process



The Biggest Risk is Work-in-Process

- ✓ The **Big Bang** is Obsolete

Sources of Risk

- ✓ Un-coded specifications
- ✓ Un-tested code
- ✓ Un-integrated code
- ✓ Code that has not been used in production



The Best Risk Mitigation is Low Work-in-Process

- ✓ Test early, integrate often, fail fast.

Failure Demand



Value Demand

- ✓ Demand for work that adds value from a customer perspective
- ✓ The Goal: Delight Customers by Responding to Value Demand



Failure Demand

- ✓ Demand on the resources caused by your failures
 - ✗ Eg. Support Calls
- ✓ The Goal: Eliminate Failure Demand
 - ✗ Meanwhile, respond as fast as possible





Technical Debt



Anything that makes code difficult to change

- ✓ Sloppy / Un-testable Code
Code without a test harness is Legacy Code.



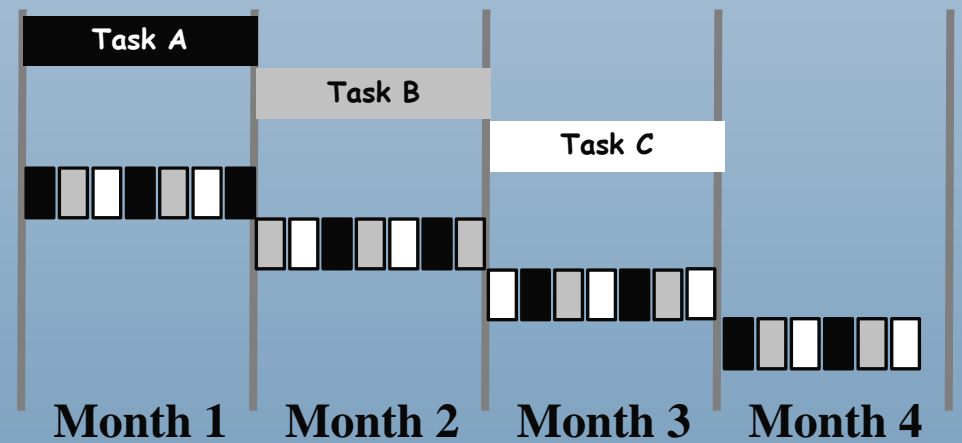
- ✓ Dependencies
High cohesion and low coupling are essential for testable code.



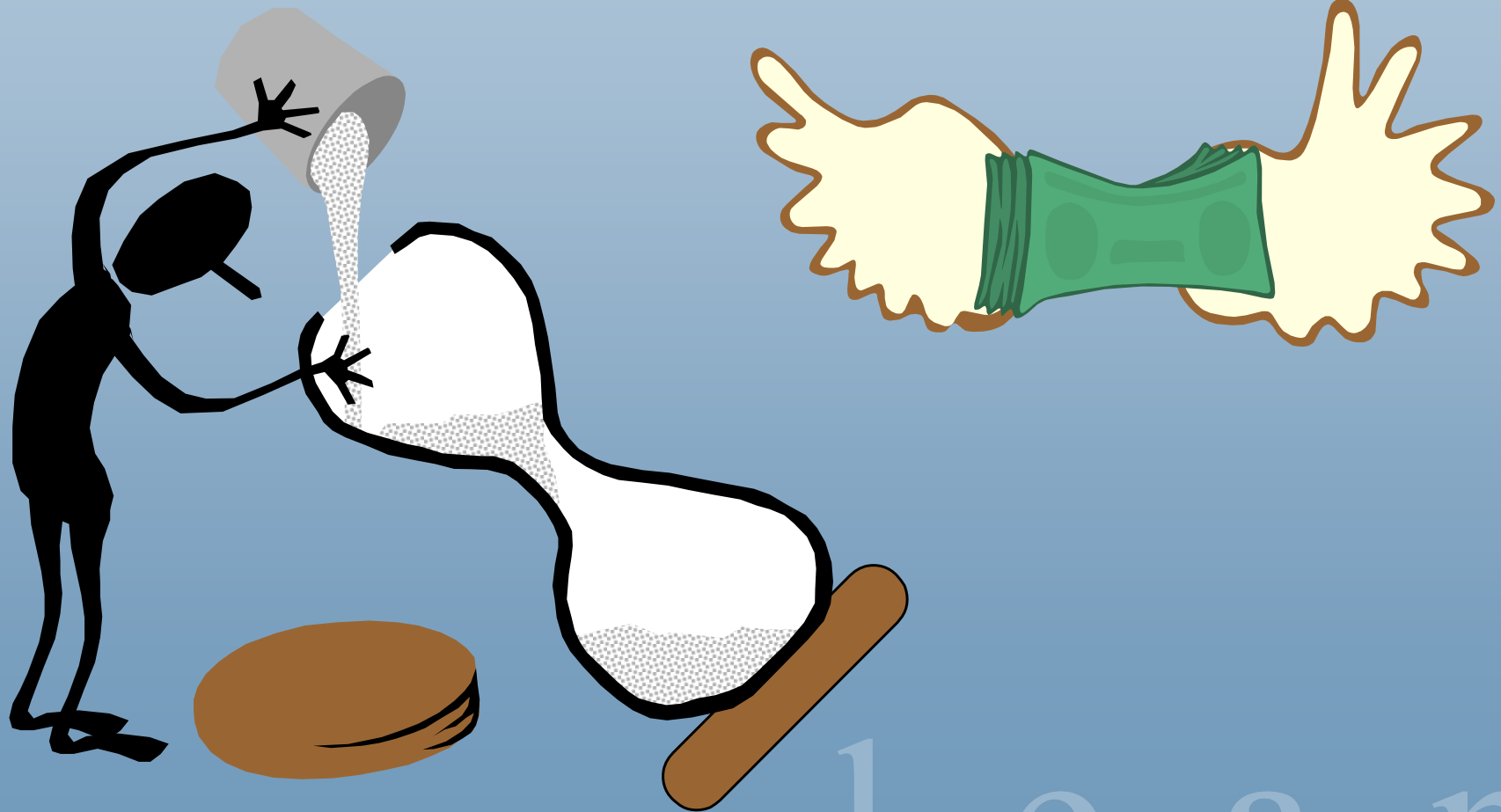
- ✓ Unsynchronized Code Branches
The longer two code branches remain apart, the more difficult they are to merge together.



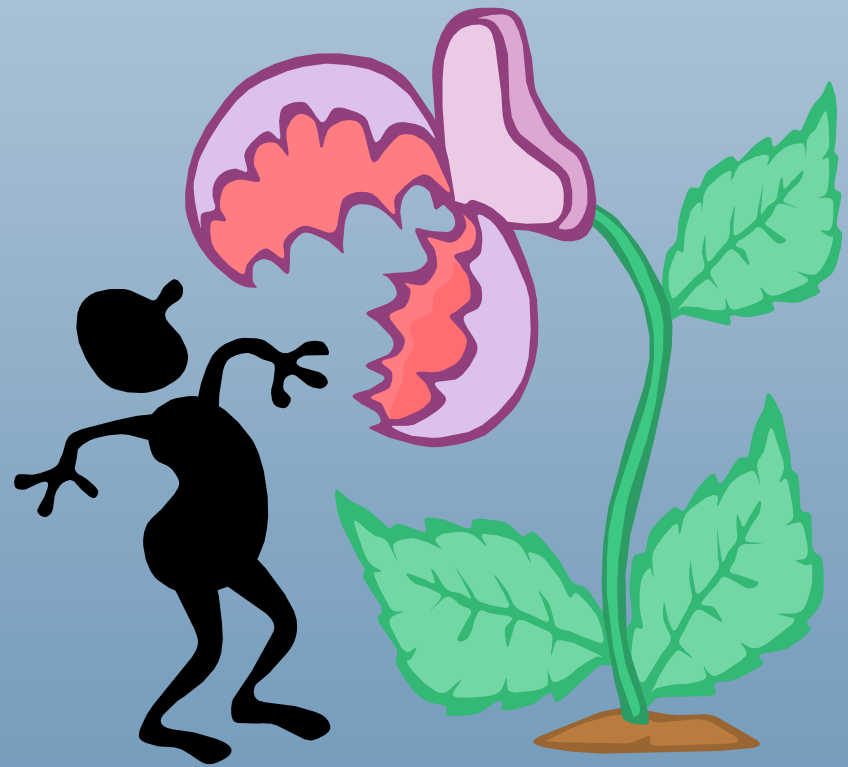
Task Switching



Delays



Defects



Build Quality In

A Quality Process Builds Quality IN.

- ✓ Rather than trying to test quality in later.

*90% of all defects caused by the System**

- ✓ If you find defects at the end of your process...
- ✓ Your process is defective!

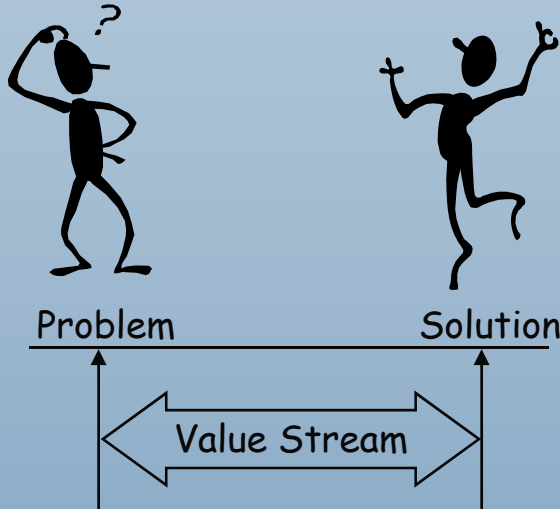
Quality by Construction

- ✓ Low Dependency Architecture
- ✓ Code that reveals its intentions
- ✓ Design/code reviews
- ✓ Immediate, automated testing
- ✓ Continuous, nested integration
- ✓ Escaped defect analysis & feedback



* Dr. W. Edwards Deming

Improve Capability: Visualize End-to-End Flow



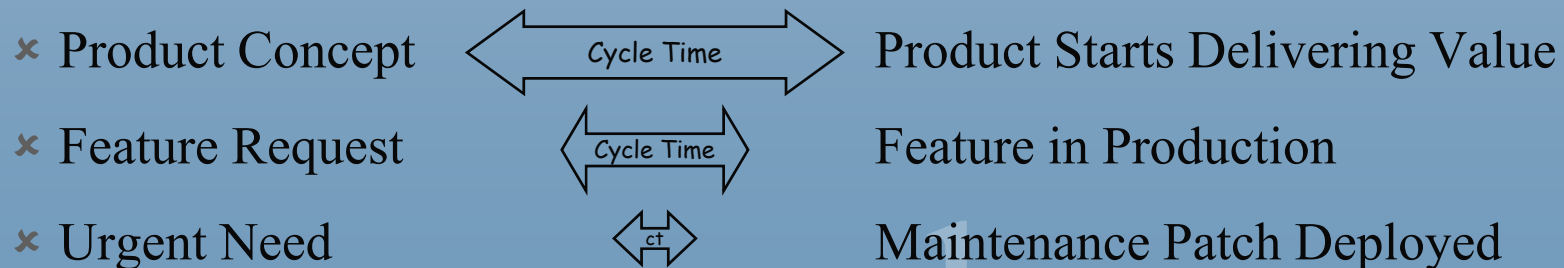
Value Stream

- ✓ The flow of activities that starts with a customer in need, and ends when that customer's need is satisfied.

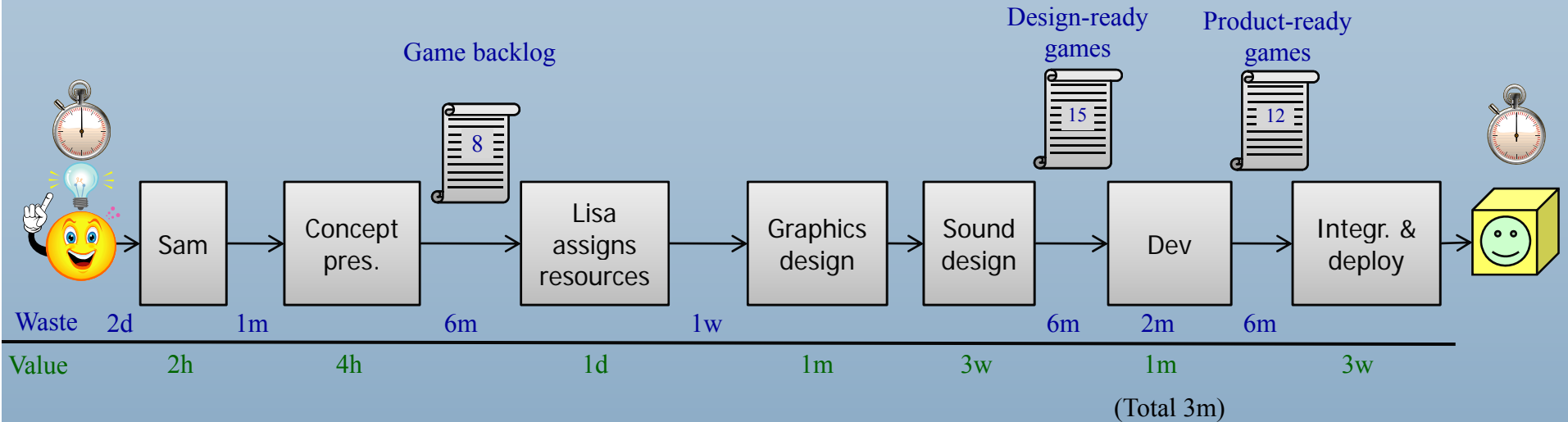
Process Capability:

- ✓ The reliable, repeatable cycle time from customer need until that need is satisfied.

Multiple Value Streams



Value Stream Map



Thanks to: Henrik Kniberg, of
Crisp, Stockholm
Used with Permission

$\frac{3 \text{ m value added time}}{25 \text{ m cycle time}} = 12\%$ Process cycle efficiency

Games out of date
⇒ Missed market windows
⇒ Demotivated teams
⇒ Overhead costs

What would you do?

l e a n



l e a n

software development

Thank You!

More Information: www.poppendieck.com