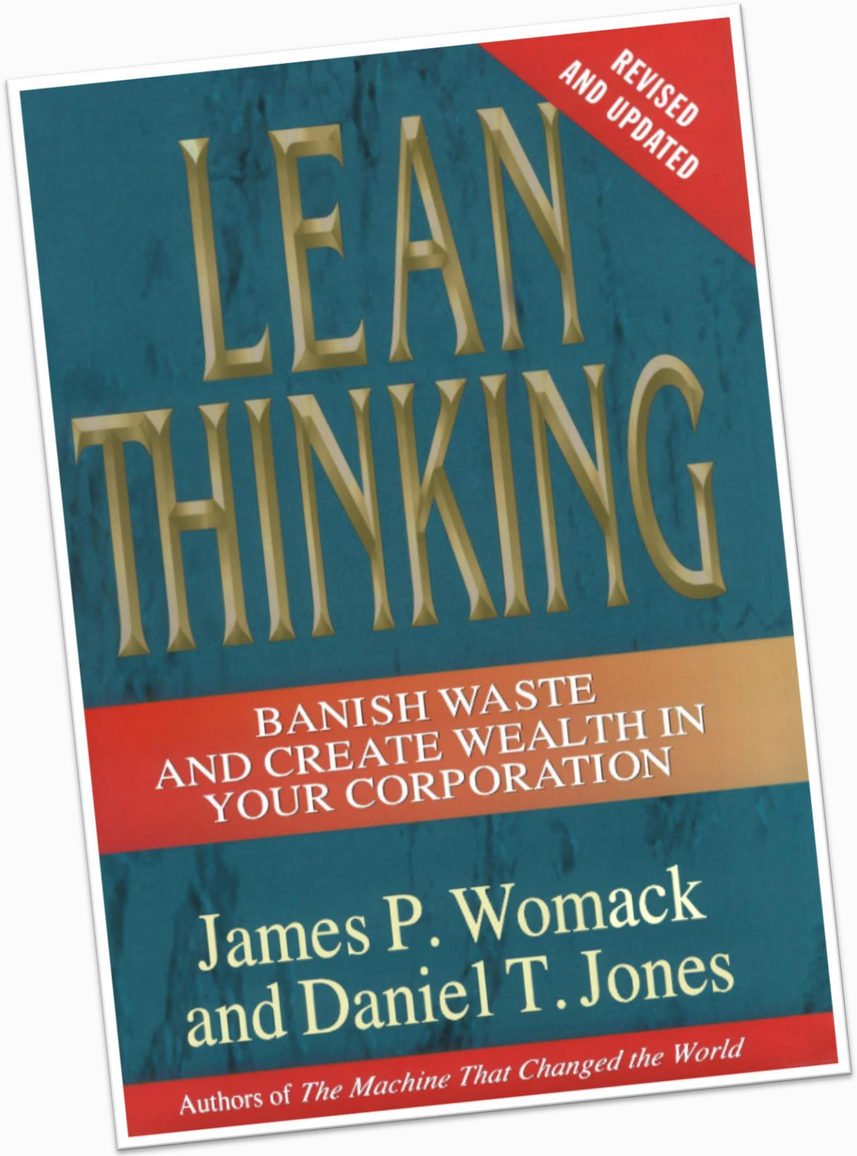
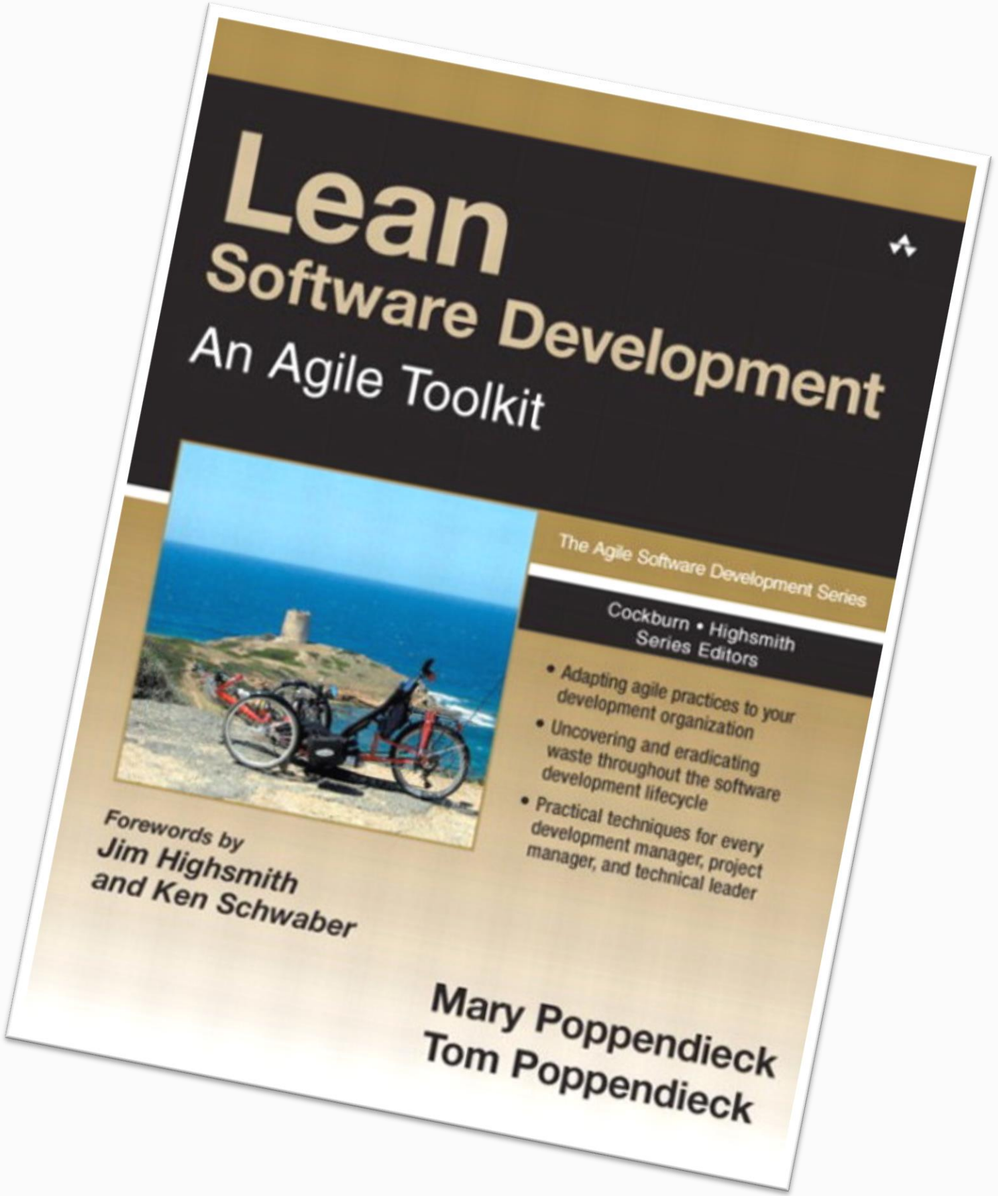
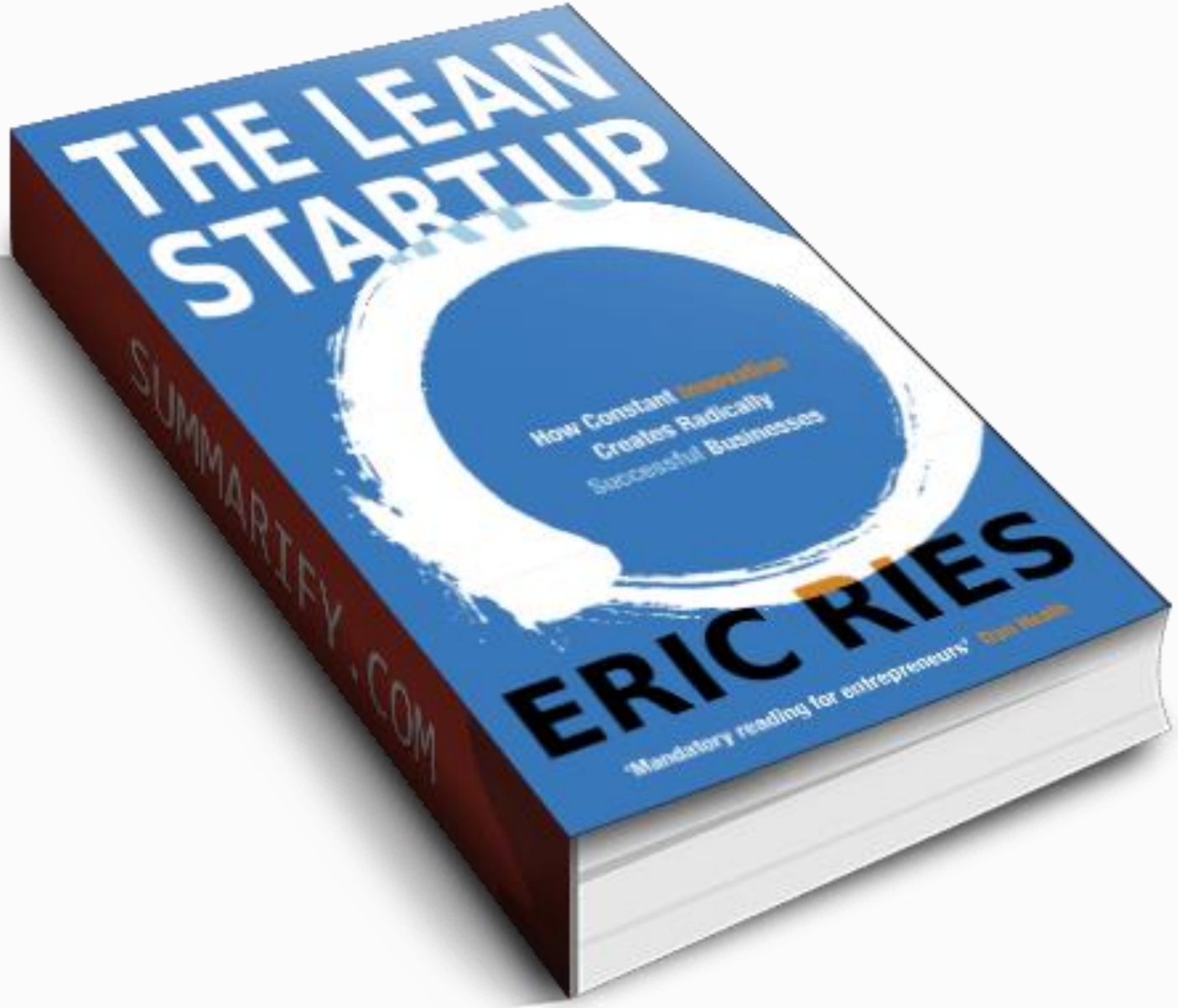
Lean software development


REVISED AND UPDATED
LEAN THINKING
BANISH WASTE AND CREATE WEALTH IN YOUR CORPORATION
James P. Womack and Daniel T. Jones
Authors of The Machine That Changed the World


Lean Software Development
An Agile Toolkit
The Agile Software Development Series
Cockburn • Highsmith
Series Editors
• Adapting agile practices to your development organization
• Uncovering and eradicating waste throughout the software development lifecycle
• Practical techniques for every development manager, project manager, and technical leader
Forewords by Jim Highsmith and Ken Schwaber
Mary Poppendieck
Tom Poppendieck


THE LEAN STARTUP
How Constant Innovation Creates Radically Successful Businesses
ERIC RIES
'Mandatory reading for entrepreneurs' Ryan Hoole
SUMMARIFY.COM

$$\text{Lean} \overset{?}{\supset} \text{Agile}$$

A Lean History

- Lean is a *manufacturing & production* practice that considers the expenditure of resources for any goal other than the creation of **value for the end customer** to be wasteful, and thus a target for elimination

- "**value**" is defined as any action or process that a customer would be willing to pay for

A Lean History

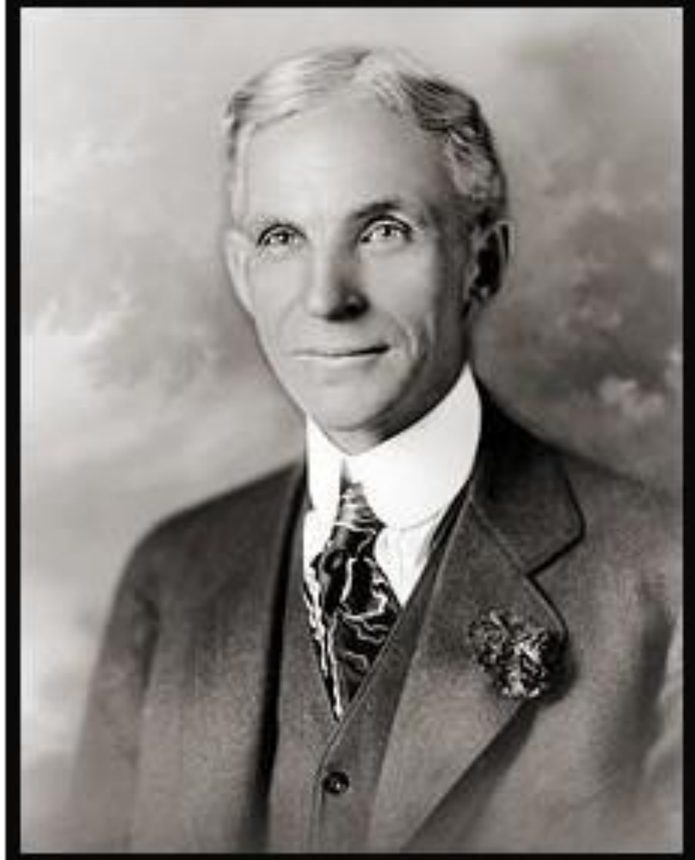
- Lean is centered around preserving value with less work
- Lean manufacturing is based on
  - optimizing flow,
  - increasing efficiency,
  - decreasing waste,
  - using empirical methods to decide what matters, rather than uncritically accepting pre-existing ideas
- Toyota was a leader in implementing lean practices in the 80s


Taiichi Ohno
Toyota Production System

The Toyota style is not to create results by working hard. It is a system that says there is no limit to people's creativity. People don't go to Toyota to 'work' they go there to 'think'.

— Taiichi Ohno —

AZ QUOTES



Any customer can have a car painted any colour that he wants so long as it is black.

(Henry Ford)

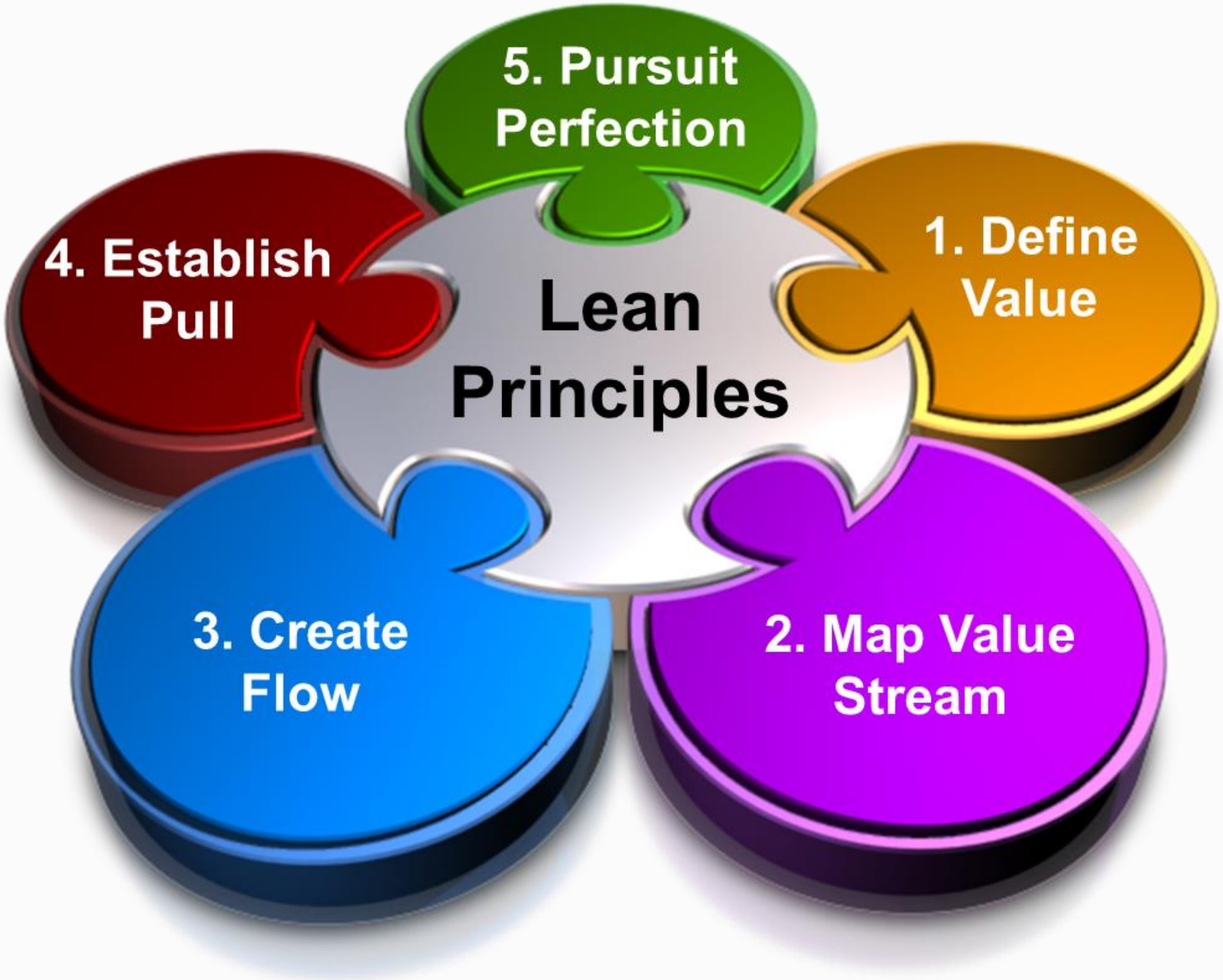
izquotes.com

Toyota Production System :

How could Toyota make cars in small quantities but keep them as inexpensive as mass-produced cars?

“Just-in-time” manufacturing

“Don't decide what to manufacture until you have a customer order; then make it as fast as possible”


5. Pursuit Perfection
4. Establish Pull
Lean Principles
1. Define Value
3. Create Flow
2. Map Value Stream

Lean Principles

- Define Value
- Map Value Stream
- Create Flow
- Establish Pull
- Pursuit Perfection

Value is always defined by the customer's needs for a specific product.

James P. Womack, Daniel T. Jones (1997)

Lean Principles

- Define Value
- Map Value Stream
- Create Flow
- Establish Pull
- Pursuit Perfection

Identify every step that does not create value and then find ways to eliminate it

James P. Womack, Daniel T. Jones (1997)

Lean Principles

- Define Value
- Map Value Stream
- Create Flow
- Establish Pull
- Pursuit Perfection

Be sure the remaining steps flow smoothly with no interruptions, delays, or bottlenecks

James P. Womack, Daniel T. Jones (1997)

Lean Principles

- Define Value
- Map Value Stream
- Create Flow
- Establish Pull
- Pursuit Perfection

the customer can "pull" the product as needed

James P. Womack, Daniel T. Jones (1997)

Lean Principles

- Define Value
- Map Value Stream
- Create Flow
- Establish Pull
- Pursuit Perfection

make lean thinking and process improvement part of corporate culture

James P. Womack, Daniel T. Jones (1997)

Case Study: Statewide Automated Child Welfare Information System (SACWIS)

- **Florida:** started in 1990, estimated 8 years and $32 million
  - In 2002 Florida spent $170 million and estimated to be completed in 2005 with $230 million
- **Minnesota:** started in 1999
  - completed in 2000 at cost of $1.1 million

- Why? Standardized infrastructure, minimized requirements, team of 8 capable people

Source: Standish Group

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Spend time only on what adds real customer value

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

When you have tough problems, increase feedback

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Keep your options open as long as practical, but no longer

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Deliver value to customers as soon as they ask for it

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Let the people who add value use their full potential

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Don't try to tack on integrity after the fact—build it in

Lean Principles for Software Development

1. **Eliminate waste**
2. **Amplify learning**
3. **Decide as late as possible**
4. **Deliver as fast as possible**
5. **Empower the team**
6. **Build integrity in**
7. **See the whole**

Beware of the temptation to optimize parts at the expense of the whole

Lean Principles are… just Principles

- **Eliminate waste** does not mean throw away all documentation.
- **Amplify learning** does not mean keep on changing your mind.
- **Decide as late as possible** does not mean procrastinate.
- **Deliver as fast as possible** does not mean rush and do sloppy work.
- **Empower the team** does not mean abandon leadership.
- **Build integrity in** does not mean big, upfront design.
- **See the whole** does not mean ignore the details.

1. Eliminate waste

If a development cycle has collected requirements in a book gathering dust, that is waste

If developers code more features than are immediately needed, that is waste

Whatever gets in the way of rapidly satisfying a customer need is waste.

Handing off development from one group to another is waste

The Seven Wastes of ~~Manufacturing~~ Software Development

• ~~Inventory~~	Partially Done Work
• ~~Extra Processing~~	Extra Processes
• ~~Overproduction~~	Extra Features
• ~~Transportation~~	Task Switching
• Waiting	
• Motion	
• Defects	

Shigeo Shingo, Toyota

1. Eliminate waste

WASTE

=

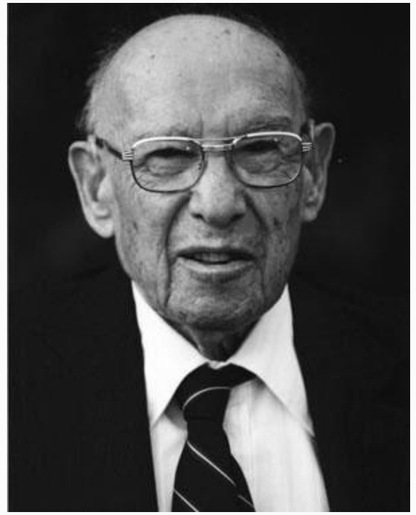
everything your organization does to develop

software that is not analysis or coding.

It is usually easier to see waste in a crisis

1. Eliminate waste

"There is nothing so useless as doing efficiently that which should not be done at all"



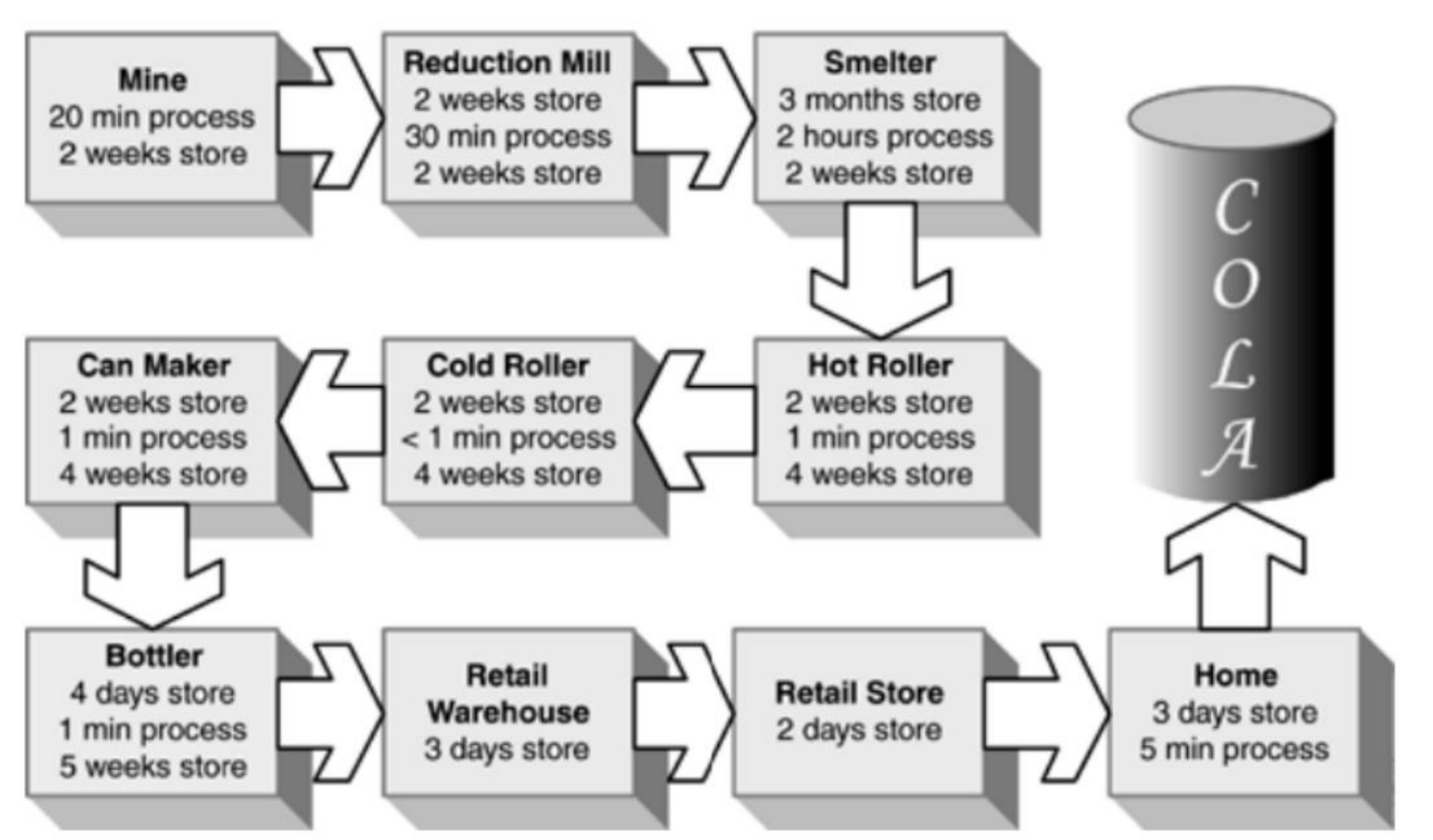
Peter Drucker

1. Eliminate waste

1. Implementing lean development is learning to see waste.
2. Uncover the biggest sources of waste and eliminate them.
3. Uncover the biggest remaining sources of waste and eliminate them.
4. Do it again.

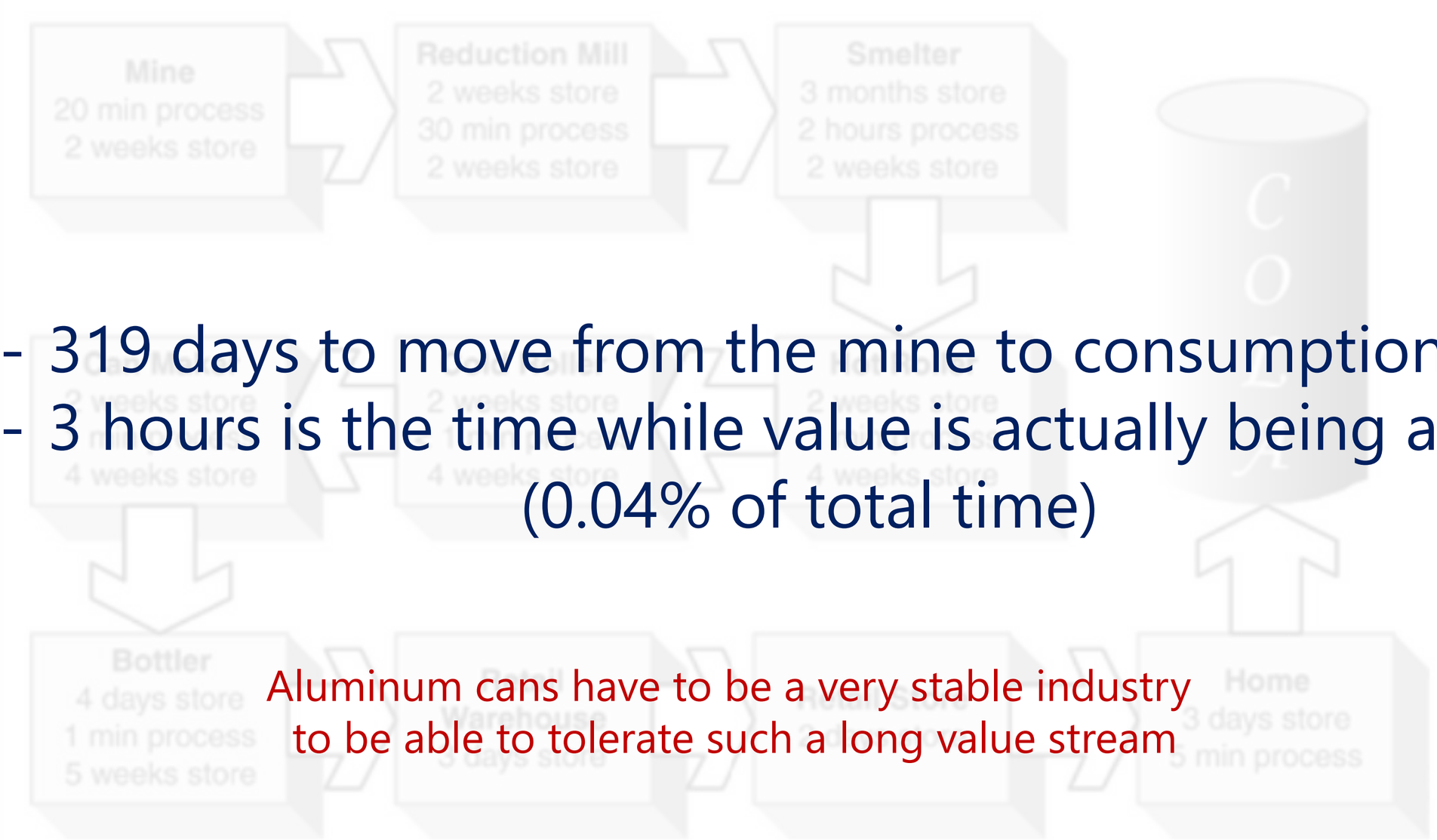
After a while, even things that seem essential can be gradually eliminated

Value Stream for Cola Cans



James P. Womack, Daniel T. Jones (1997)

Value Stream for Cola Cans



- 319 days to move from the mine to consumption
- 3 hours is the time while value is actually being added (0.04% of total time)

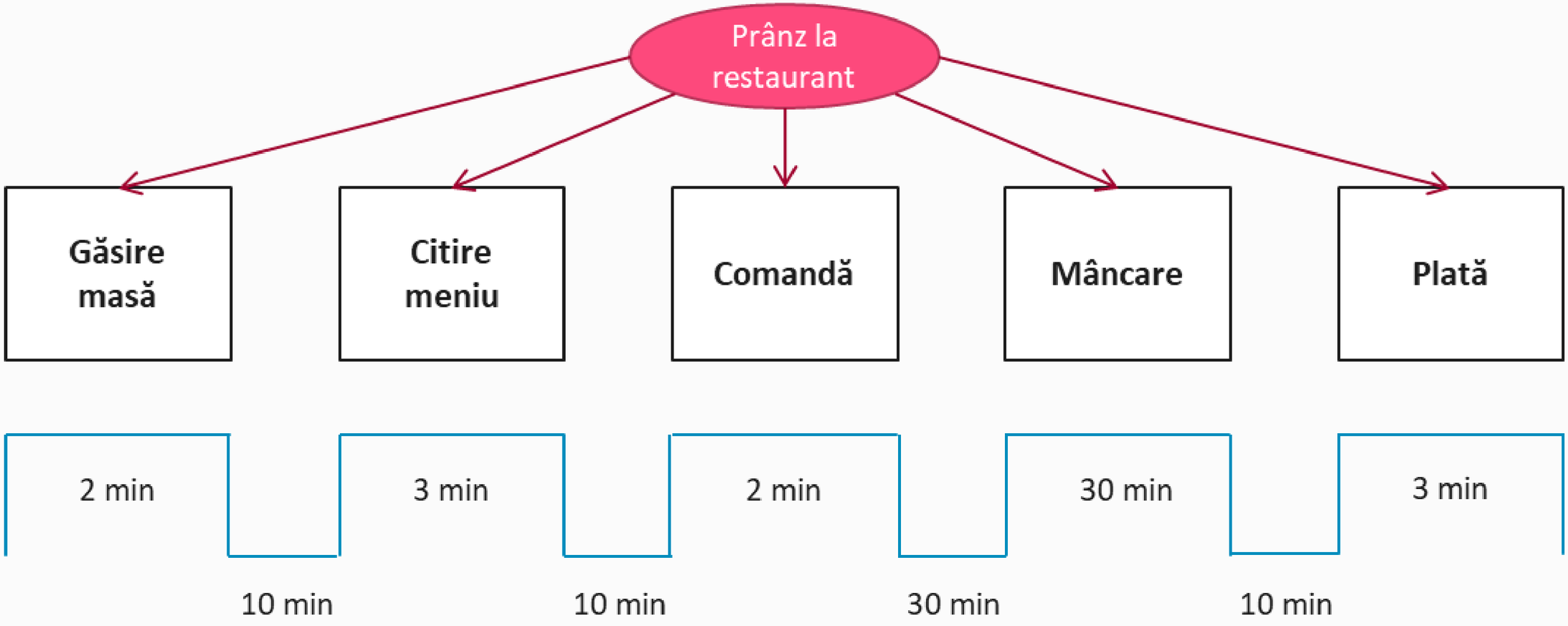
Aluminum cans have to be a very stable industry to be able to tolerate such a long value stream

James P. Womack, Daniel T. Jones (1997)

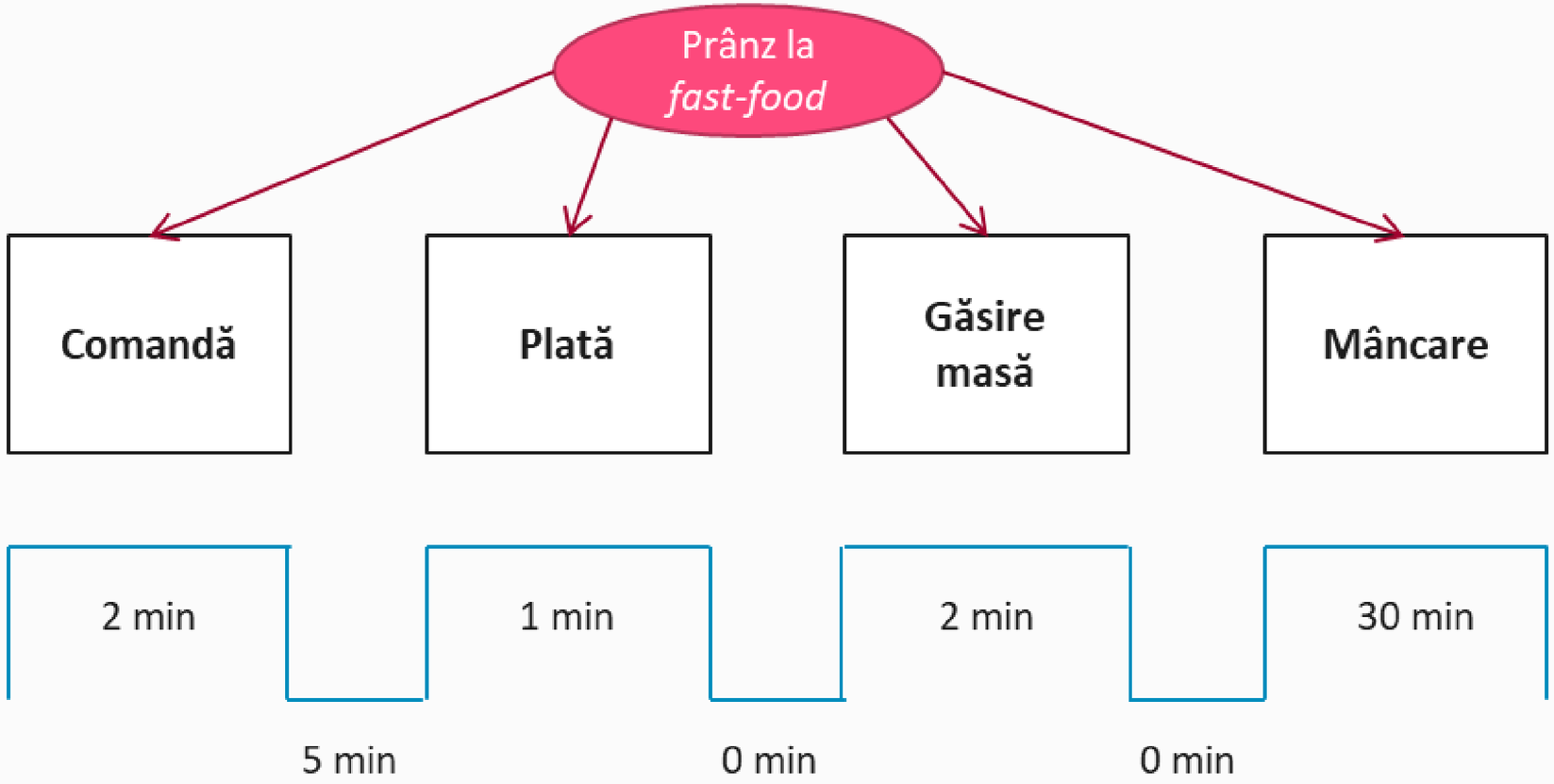
…not working for personal computers

Michael Dell

"8 days of inventory – competitors 40 days.
If Intel comes out with a new chip, I am going to get that to the market 32 days sooner."

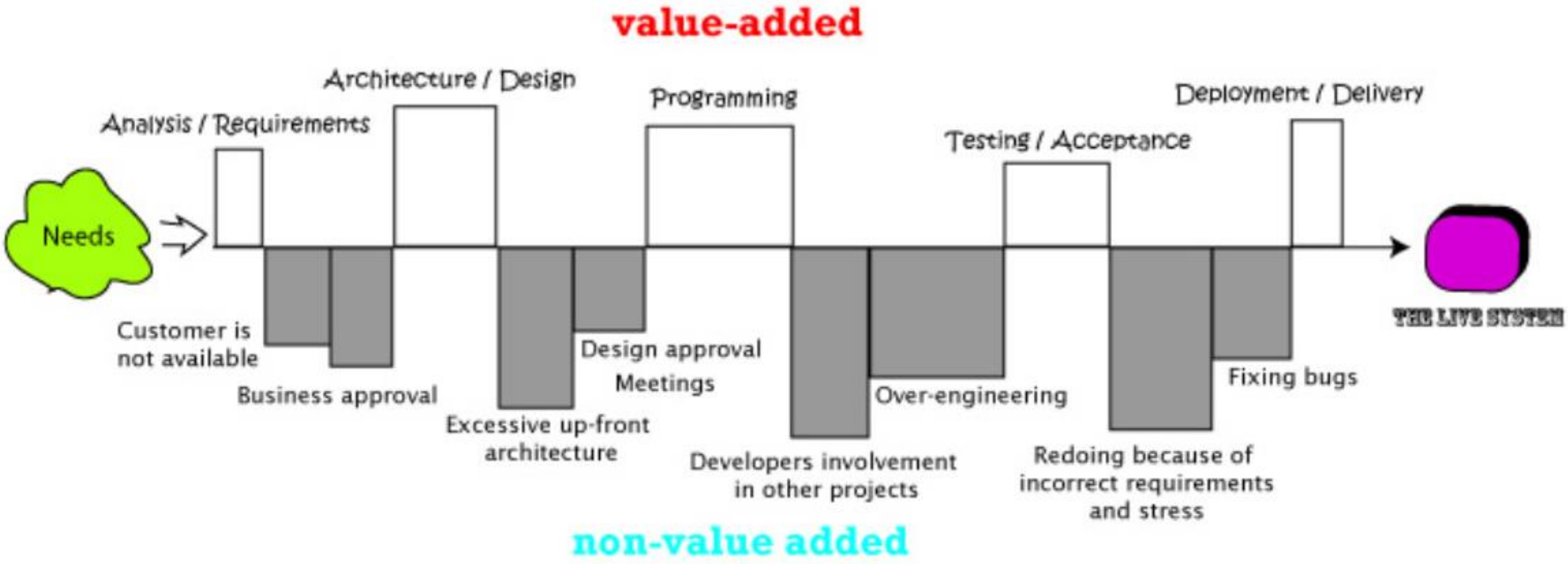


Timp total ciclu	= 110 min
Timp generator de valoare	= 40 min
Eficiența ciclului	= 40 / 110 = 36%



Timp total ciclu	= 40 min
Timp generator de valoare	= 35 min
Eficiența ciclului	= 35 / 40 = 88%

1. Eliminate waste



1. Eliminate waste

How to eliminate waste:

- Make a list of the 10 or 15 most important activities in your organization
- Rate 1-5 (1 customer do not care about , 5 customers value it highly)
- Develop a plan to cut those with 1 or 2 points

1. Eliminate waste

How to eliminate waste:

Develop a value stream map

Take the biggest cause of delay and plan to cut it in half

1. Eliminate waste

How to eliminate waste:

Seven meetings talk about the wastes in software development:

- Do you agree that this is waste? Why?
- How much time it consumes in avg / week
- What can we do to reduce that time

2. Amplify Learning

Planning is useful.
Learning is essential!

2. Amplify Learning

Development process

creating a recipe

vs

Production process

following a recipe

2. Amplify Learning

Development

Designs the Recipe
- Quality is fitness for use
- Variable results are good
- Iteration generates value

Production

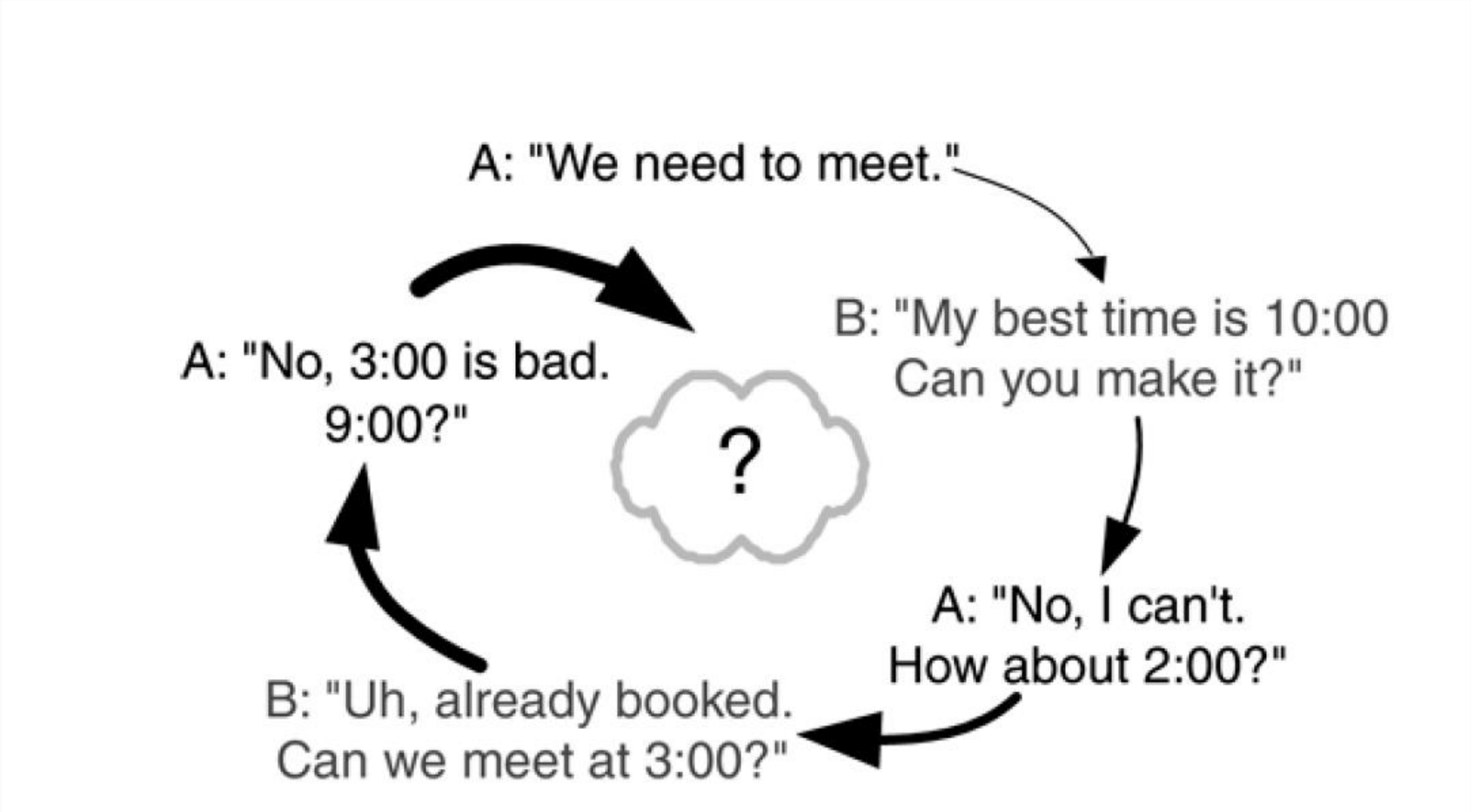
Produces the Dish
- Quality is conformance to requirements
- Variable results are bad
- Iteration generates waste (called rework)

2. Amplify Learning

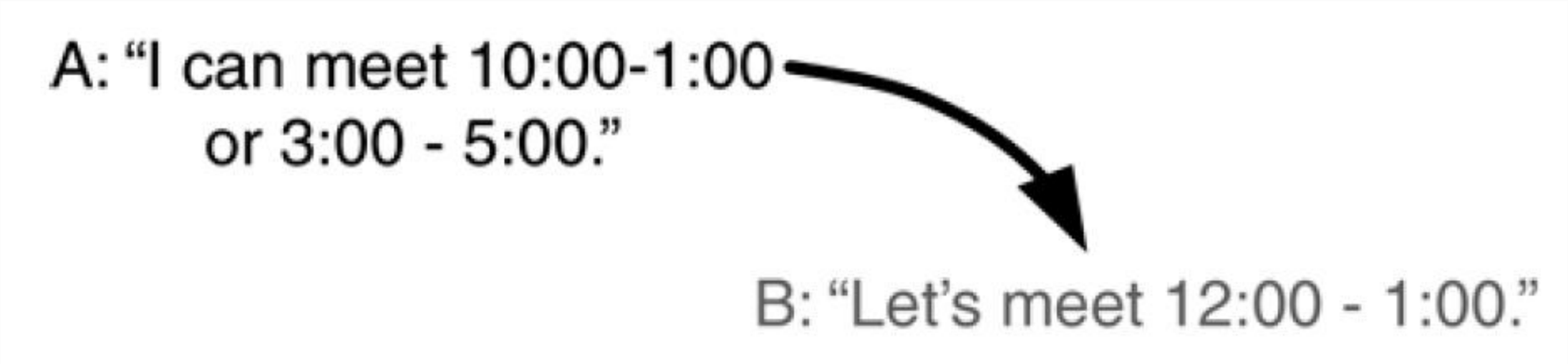
Development process

- observe,
- create a hypothesis,
- devise an experiment to test the hypothesis,
- run the experiment
- see if the results are consistent with the hypothesis

Point-based scheduling



Set-based scheduling



2. Amplify Learning

How to apply *set-based development* to software?

- develop multiple options, communicate constraints, and let solutions emerge.

Set-based development does not replace iterative development—it adds a new dimension. During early iterations, multiple choices are developed for key features; in later iterations, they are merged or narrowed to a single choice.

3. Decide as late as possible

In an evolving market, keeping design options open is more valuable than committing early.

How to avoid change penalties?

- Traditional: make the right design decision in the first place and avoid the need to change later

- Lean: Don't make irreversible decisions in the first place; delay design decisions as long as possible, and when they are made, make them with the best available information to make them correctly

3. Decide as late as possible

- The last responsible moment:
  - delay commitment until the last responsible moment, that is, the moment at which failing to make a decision eliminates an important alternative.

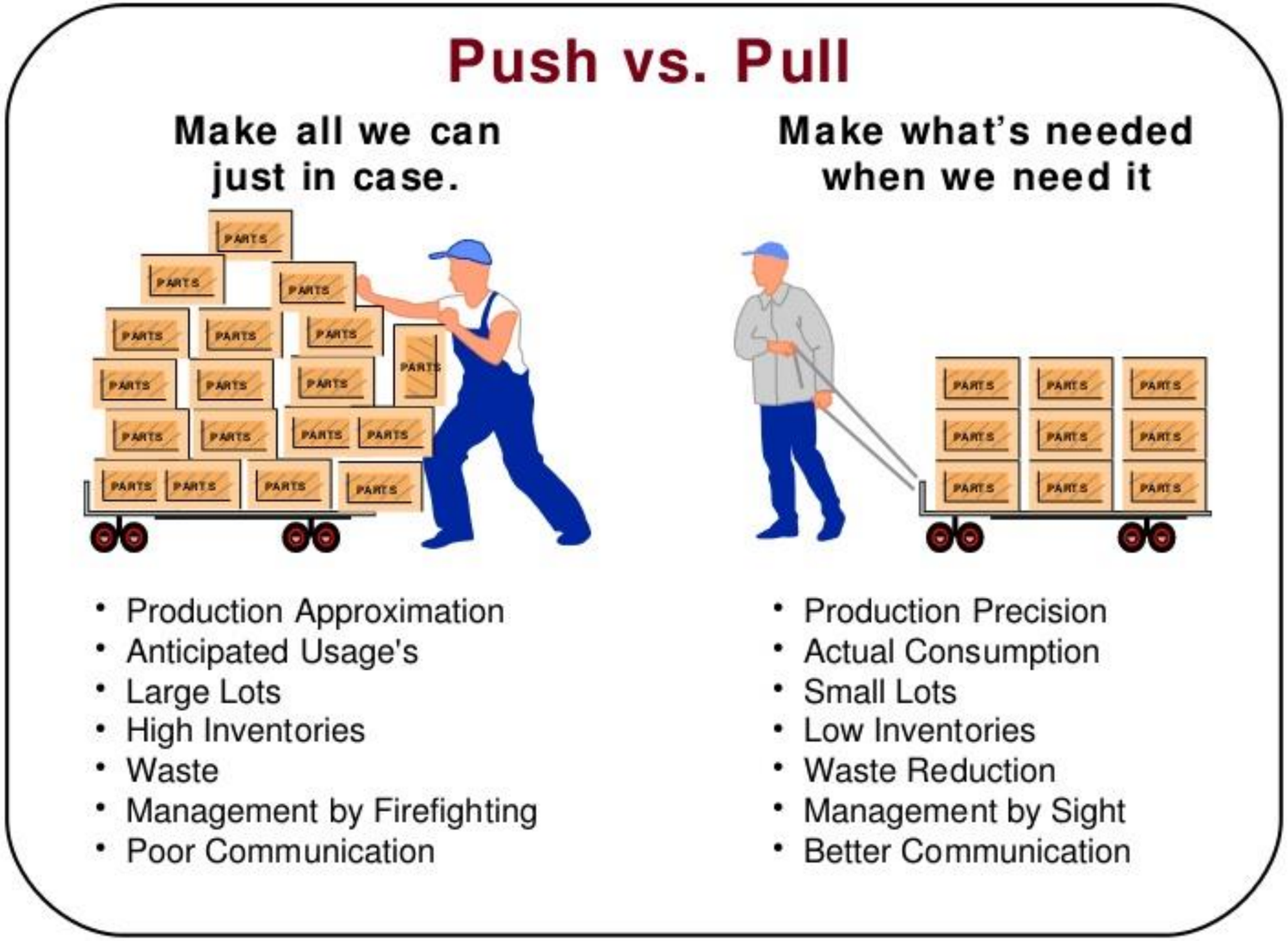
3. Decide as late as possible

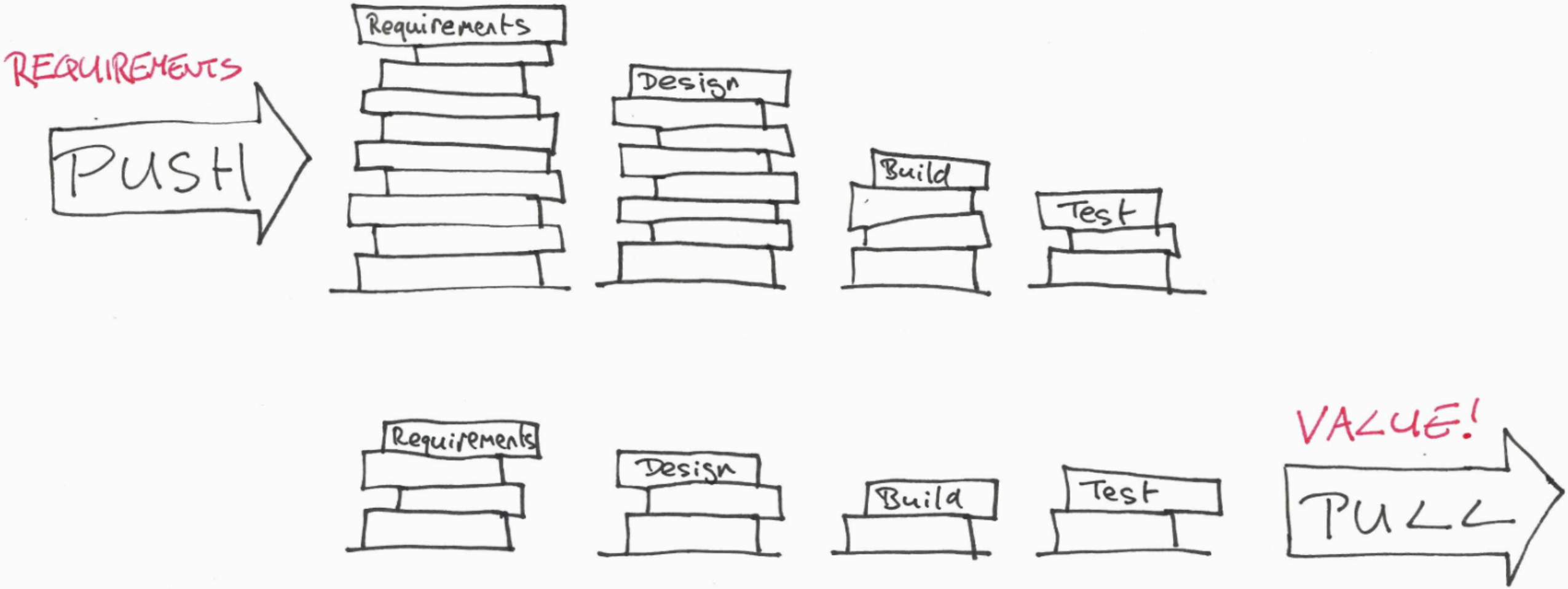
Depth First	Breadth First
Making early commitments	Delaying commitments
Needs an agreed to point to “zero” in on	Requires someone savvy to understand how the details will emerge and when it’s time to commit
Rational decision making	Intuitive decision making
Predictions and assumptions drives decisions	Real time information and feedback drives decisions

4. Deliver as fast as possible

- Customers like rapid delivery
- Rapid delivery means less time for customers to change their minds
- In-process, or partially done work can have undiscovered defects
- *Deliver as fast as possible* complements *decide as late as possible:* the faster you can deliver, the longer you can delay decisions.

4. Deliver as fast as possible




REQUIREMENTS
PUSH!
Requirements
Design
Build
Test
Requirements
Design
Build
Test
VALUE!
PULL

5. Empower the team

Because decisions are made late and execution is fast, it is not possible for a central authority to orchestrate activities of workers.

5. Empower the team

- How do you make sure that when people come in to work, they know how to spend their time in the most effective manner to achieve the goal at hand?

5. Empower the team

- Managers – Cope with complexity
  - Plan and Budget
  - Organize and Staff
  - Track and Control

- Leaders – Cope with change
  - Set Direction
  - Align People
  - Enable Motivation

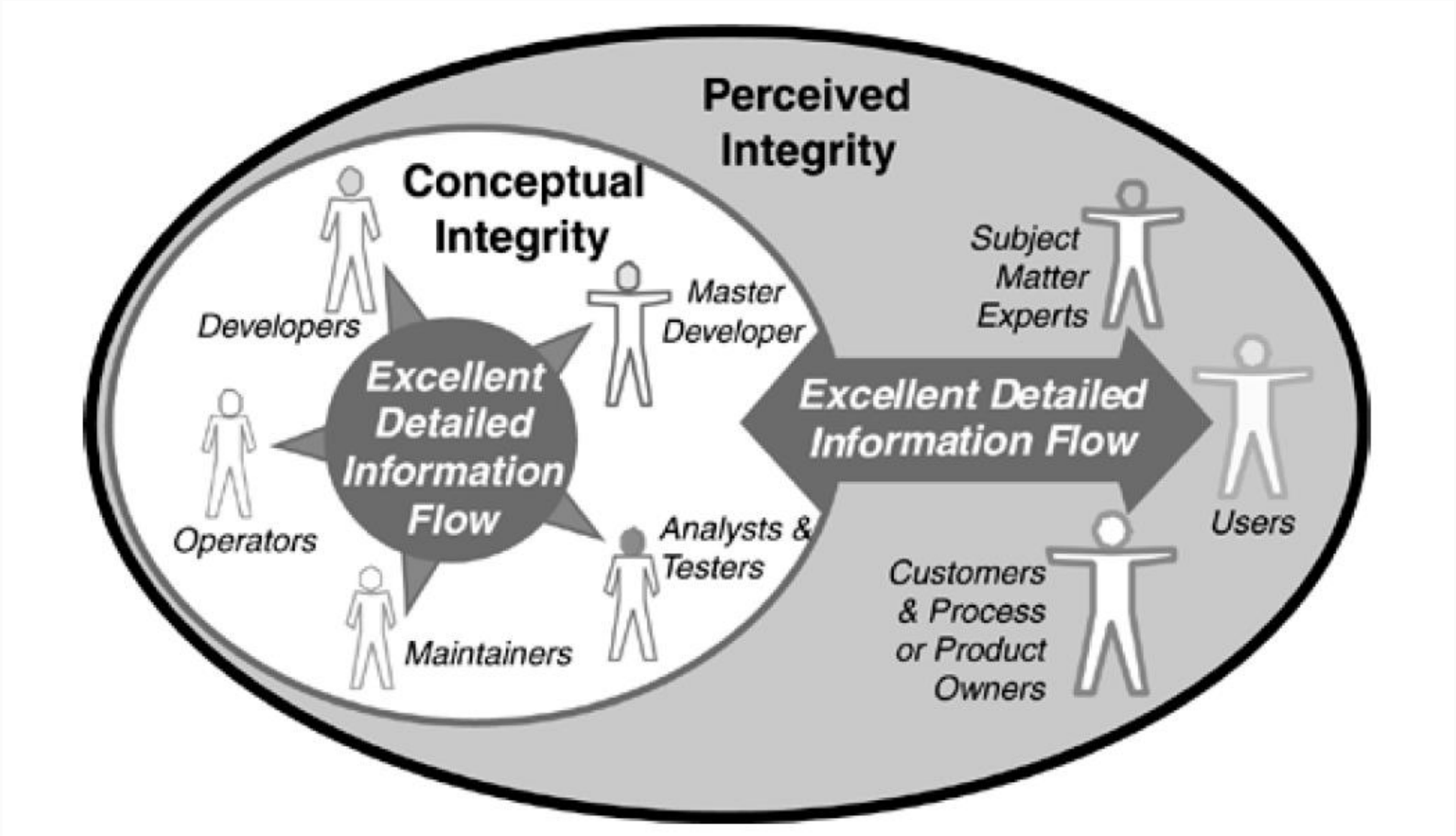
6. Build integrity in

Integrity comes from wise leadership, relevant expertise, effective communication and healthy discipline.

Processes, procedures, and measurements are not adequate substitutes!

6. Build integrity in

- ***Perceived integrity*** is affected by the customer's whole experience of a system: how it's advertised, delivered, installed, accessed; how intuitive it is to use; how well it keeps up with changes in the domain; how much it cost; how timely it is; how well it solves the problem.
- ***Conceptual integrity*** means that a system's central concepts work together as a smooth, cohesive whole.


Perceived Integrity
Conceptual Integrity
Developers
Master Developer
Operators
Analysts & Testers
Maintainers
Excellent Detailed Information Flow
Subject Matter Experts
Users
Customers & Process or Product Owners
Excellent Detailed Information Flow

7. See the whole

A Common Pattern

- **Limits of Growth:** even as one process produces a desired result, it creates a secondary constraint that eventually slows down the effect and limits growth.
- **The Theory of Constraints:** you can remove a constraint to growth in one place, but it will shift to another place. It’s an on going process.
- **Sub Optimization:** the more complex a system, the more you temptation it is to divide into parts
- **Shifting the Burden:** addressing the symptoms, instead of the root cause.

7. See the whole

Ask the 5 whys!

You have increasing defects on a project you ask:

Why #1:

A: New modules were added, that are causing new issues.

Why #2: Why did the new modules generate defects in other modules?

A: They were not tested

Why #3: Why where they not tested?

A: Developers where pressured to deliver before testing could occur

Why #4: Why was there so much pressure?

A: A Manager thought a hard deadline would work to motivate the developers

Why #5: Why did they think this approach was necessary?

A: A manager was worried about late delivery and schedule overruns

"Think big, act small, fail fast; learn rapidly"

Lean ? Agile

Agile

- Adaptive to change
- Shorter planning and commitment cycles
- Focus on collaboration and interaction

Lean

- System view of value stream
- Identify ways to eliminate waste
- Limit work queues

Agile & Lean Commonalities

- Improve quality
- Amplify learning
- Continuously improve
- Empower people