# Lean Software Development

*Alexandre Boutin*

Responsable Stratégie International Développement Logiciel chez Yahoo

*Scrum Master & Practitioner Certifié – Coach Agile*

*Blog : www.agilex.fr*

*Président du Club Agile Rhône Alpes*

## Agenda

- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

## Overview - LEAN

- *LEAN, at its core, is a management approach for streamlining production systems by*
  - *Streamlining the value chain (even across companies)*
  - *Eliminating waste from the flow*
  - *Being disciplined about “when” decisions are made*
  - *Leveraging people as the most flexible resource in the system,*
- *LEAN offers a set of tools to challenge our beliefs and find better way to deliver product*
- *Mary and Tom Poppendieck have transferred principles and practices from the manufacturing environment to the software development*
- *Mary said: “There is nothing directly relating the LEAN and AGILE  concepts, yet they fit together nicely in a software organization.”*

## Agenda

- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

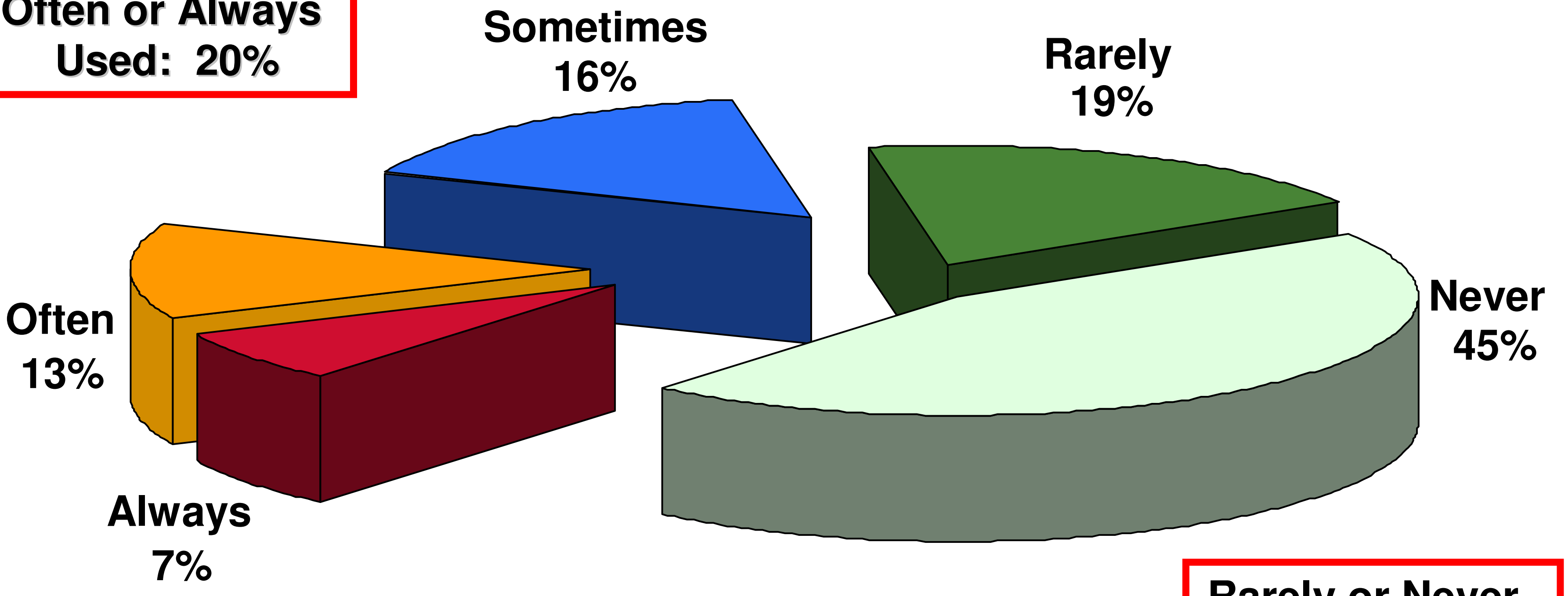
## The Seven Wastes

| The Seven Wastes of Manufacturing - Shigeo Shingo | The 7 Wastes of software Development |
|---|---|
| 1. Inventory | 1. Partially Done Work |
| 2. Overproduction | 2. Extra Features |
| 3. Extra Processing | 3. Extra Processes |
| 4. Motion | 4. Task Switching |
| 5. Transportation | 5. Handoffs |
| 6. Waiting | 6. Delays |
| 7. Defects | 7. Defects |

## Extra Features

Features and Functions Used in a Typical System

**Often or Always Used: 20%**

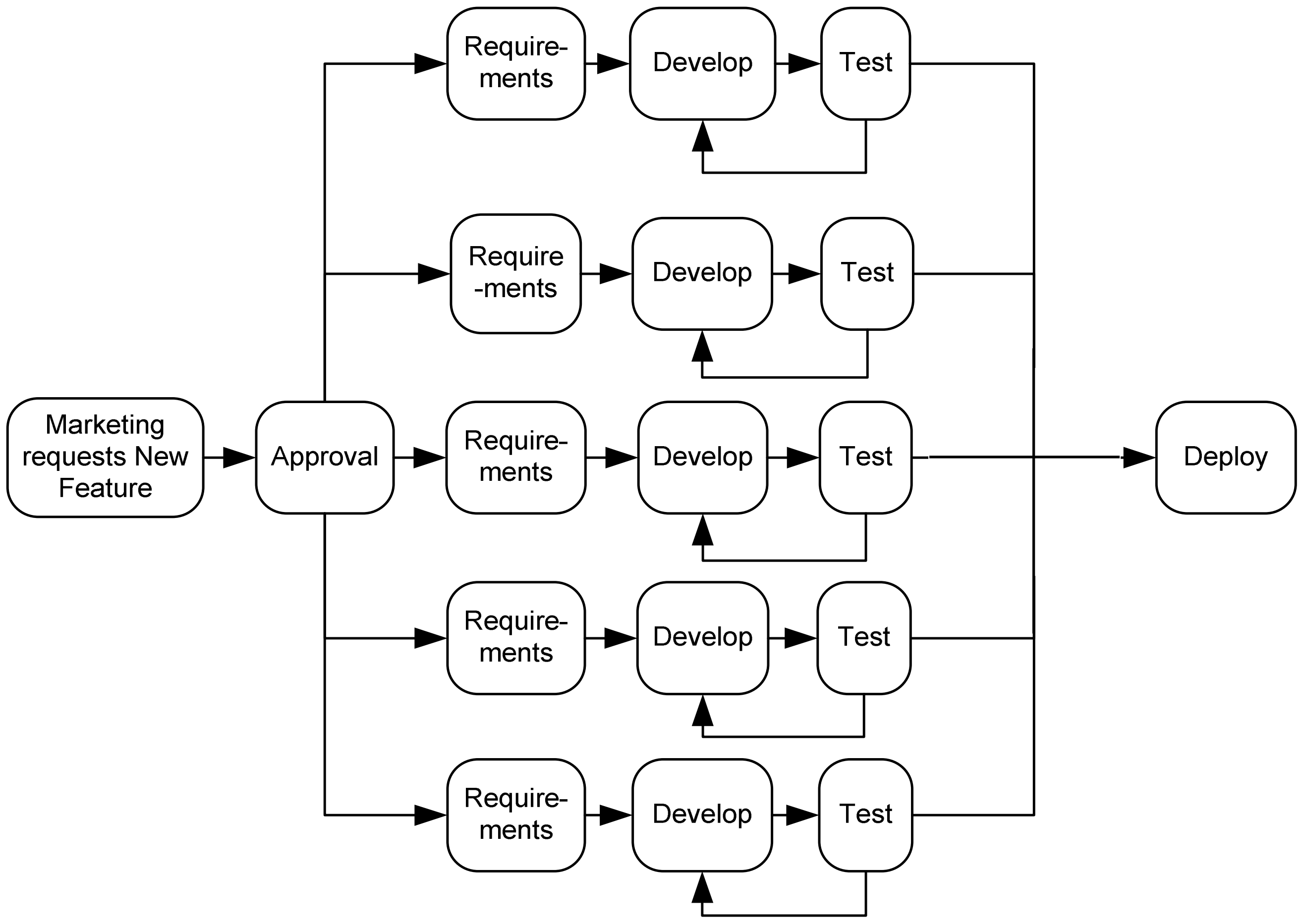


**Rarely or Never Used: 64%**

*Standish Group Study Reported at XP2002 by Jim Johnson, Chairman*

# Value Stream Mapping

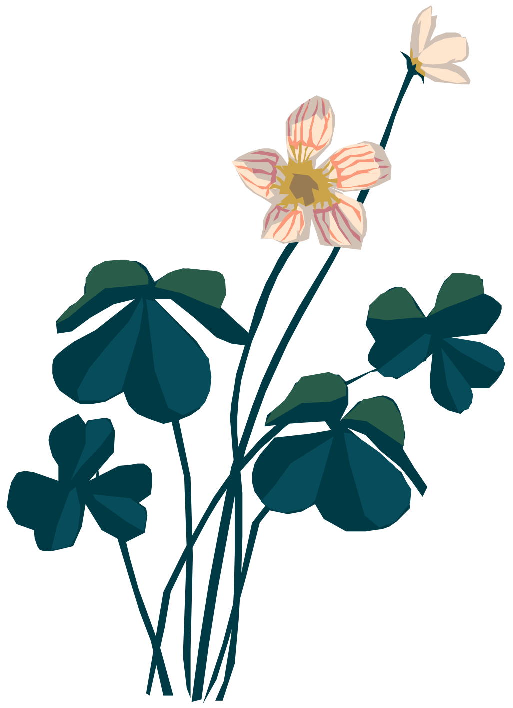
**Example**



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **value** | 1 week | | 1 week | | 6 weeks working together | | 1 hour |
| **waste** | | 1 week | | 1 day | 1 week | ½ week | |

## Agenda

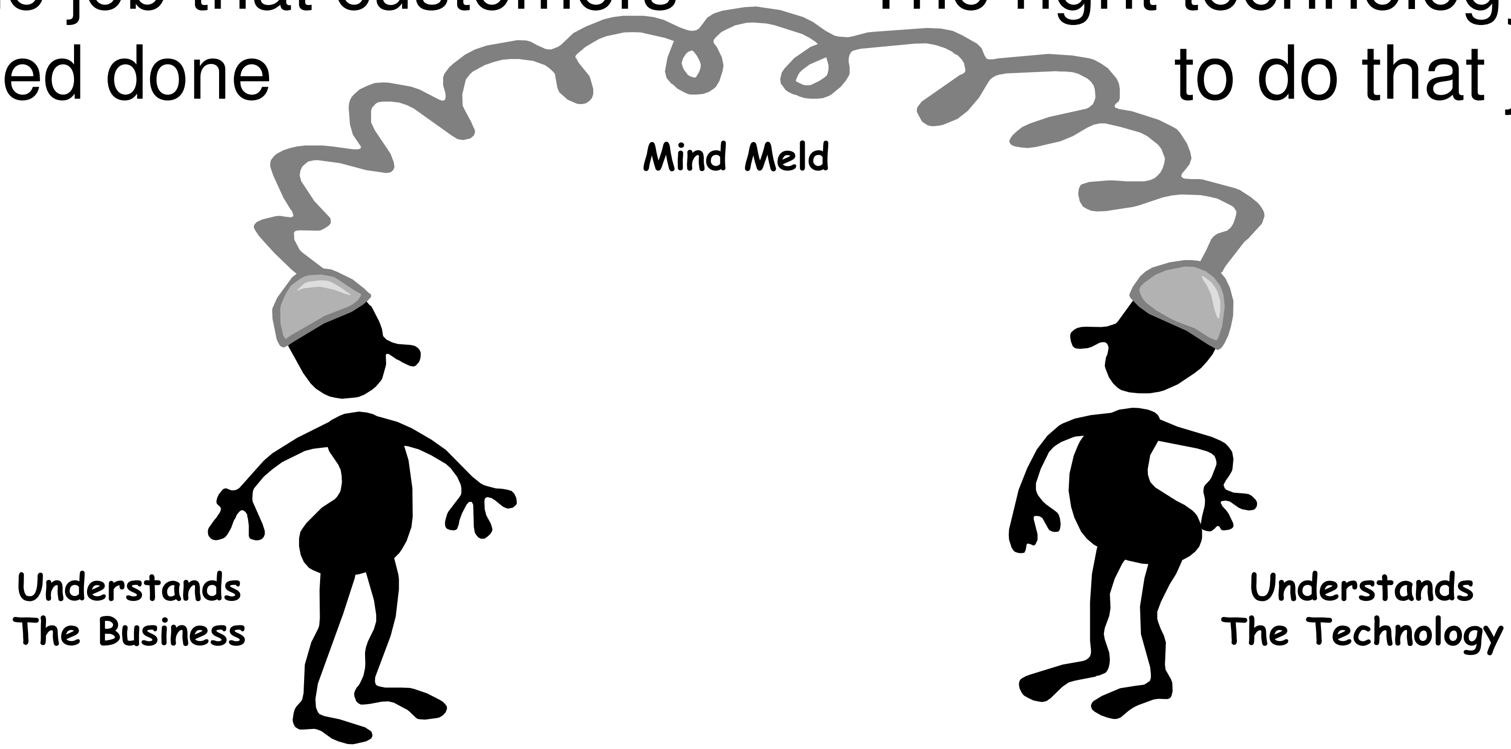
- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge



## Brilliant Products

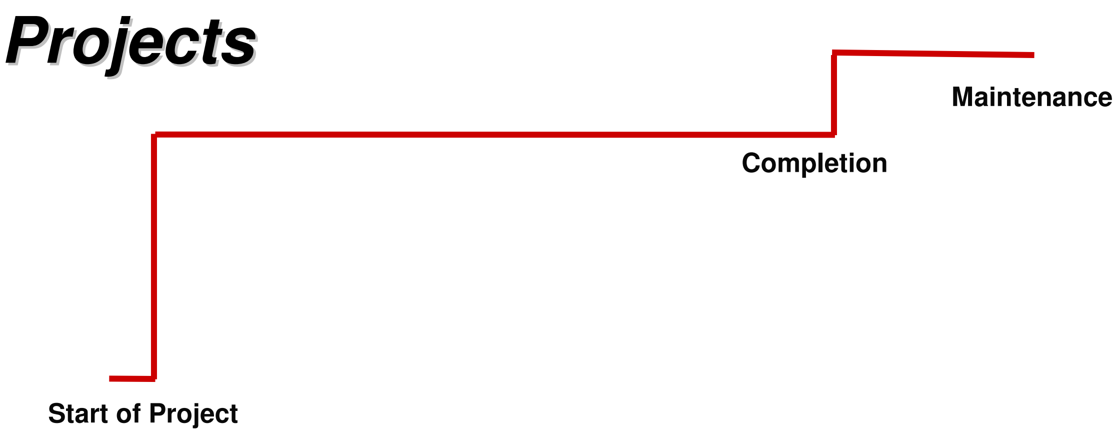
Breaking the Customer / Supplier model

- The job that customers need done
- The right technology to do that job

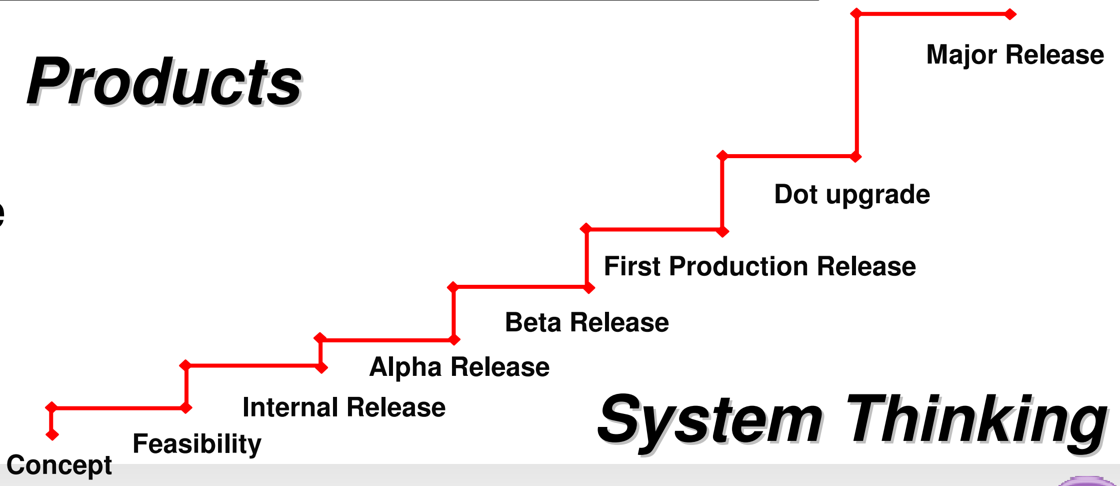


# Think Products, not Projects

- Up-front funding
- Scope fixed at onset
- Success = cost/schedule/scope
- Team disbands at completion
- Documentation tossed over-the-wall to maintenance



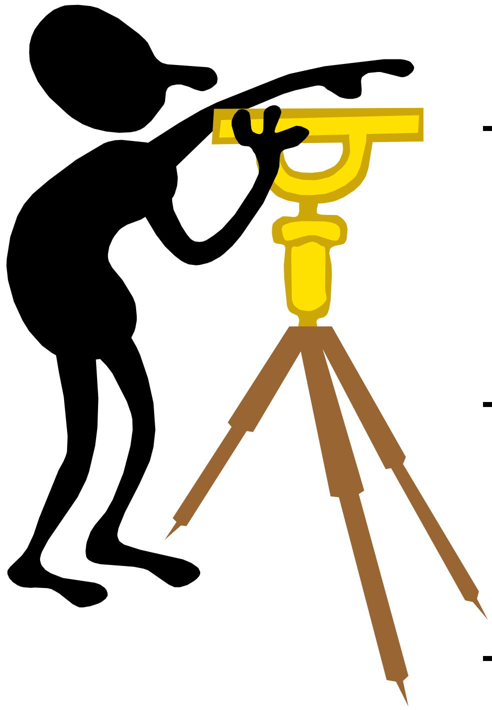
- Incremental funding
- Scope expected to evolve
- Success = profit/market share
- Team stays with product
- Team uses its own documentation



**_System Thinking_**

## Architecture

- The Role of Systems Design (Architecture):
  - Provide a foundation for growth
    - Create a common infrastructure
  - Enable incremental development
    - Minimize dependencies
    - Modularize potential change
  - Create space for teams to innovate
    - Design, code and test are different aspects of the same job and must be done concurrently
  - Leave room for the future
    - Evolve the architecture over time

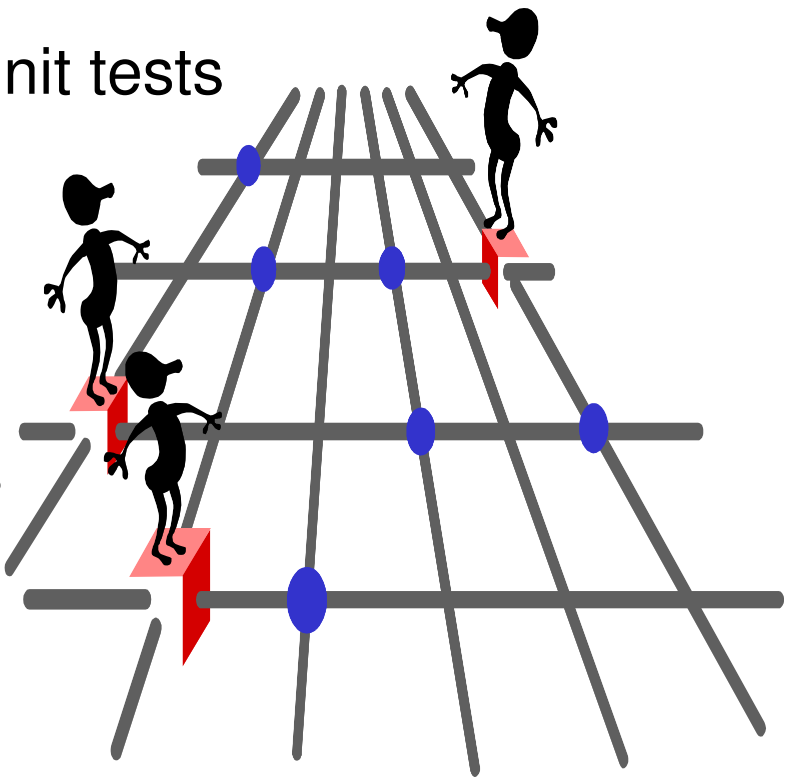


## Agenda

- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

## Continuous Integration

- Every few minutes
  - – Check in code, build and run unit tests
- Every day
  - – Run acceptance tests
- Every week
  - – Run more complete test suites
- Every iteration
  - – Deployment-ready code
- Every Release
  - – Deploy and run in production



## Technical Debt

Anything that makes code difficult to change increases the Technical Debt

- **Complexity**

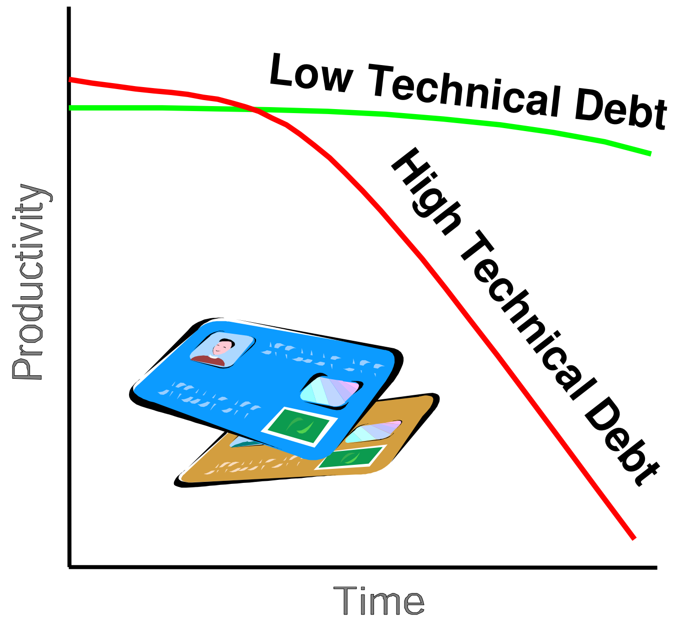
  The cost of complexity is exponential.
- **Regression Deficit**

  Every time you add new features the regression test grows longer!
- **Unsynchronized Code Branches**

  The longer two code branches remain apart, the more difficult merging will be.

*You can pay full price for code when you build it or you can incur technical debt.*

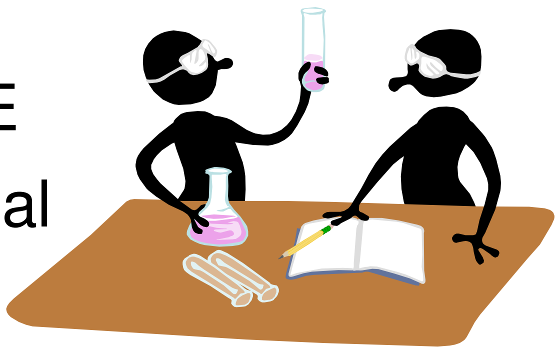
*But interests rates are very high.*



## Testing contribution to quality

### *Two Kinds of Inspection*

- Inspection to Find Defects – is WASTE
- Inspection to Prevent Defects – is Essential



### *The Role of Testing*

- The job of Testing is ***not*** to *find* defects
- The job of Testing is to *prevent* defects.
- A quality process builds quality into the code

  If you routinely find defects during verification

  ➔ Your process is defective.

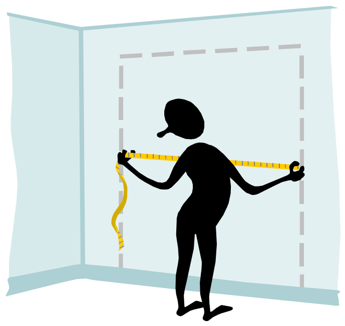
## Agenda

- Overview
  - Lean Software Development

- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

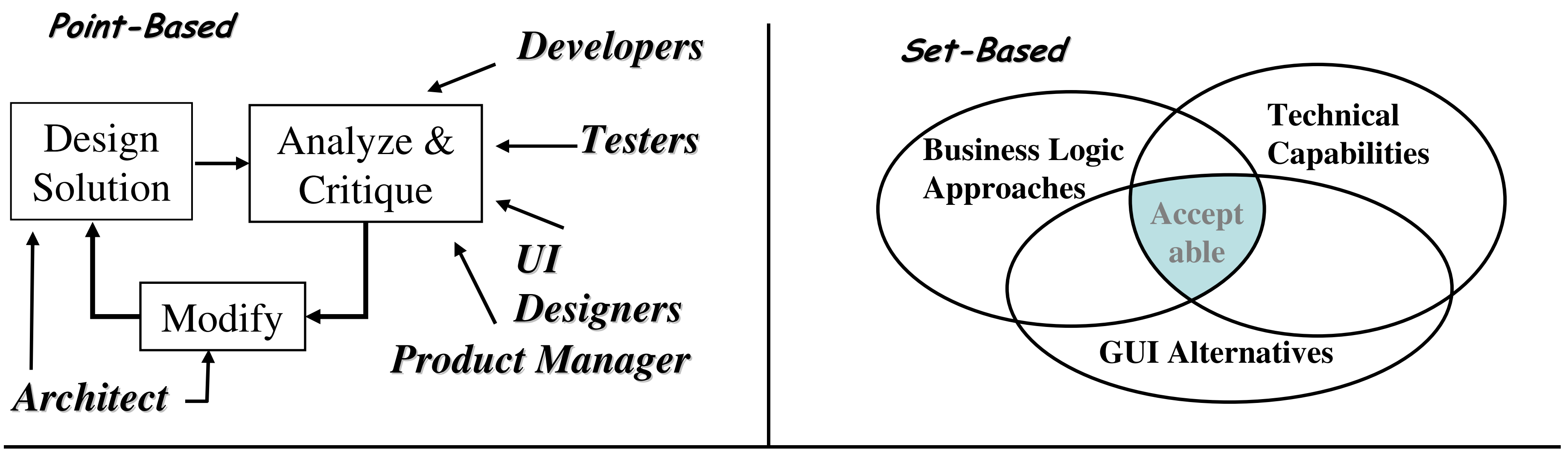
## Change Tolerant Software

- 60-80% of all software is developed after first release to production.



- A development process that anticipates change will result in software that tolerates change.
- System architecture should support the addition of any feature at any time
- Make decisions ***reversible*** whenever possible.
- Make ***irreversible*** decisions as late as possible.
  - Ex: When do you really need the user interface designed?

## Set-Based Engineering



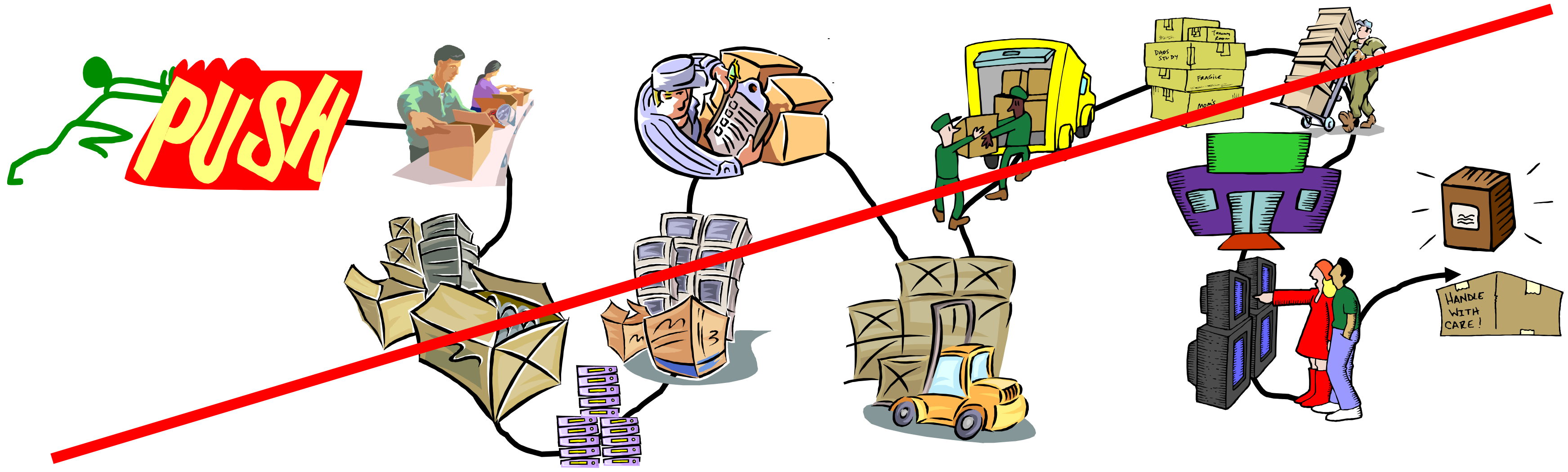
- Multiple options are prepared for the decision.
- There is always an option that will work.
- Paradox:

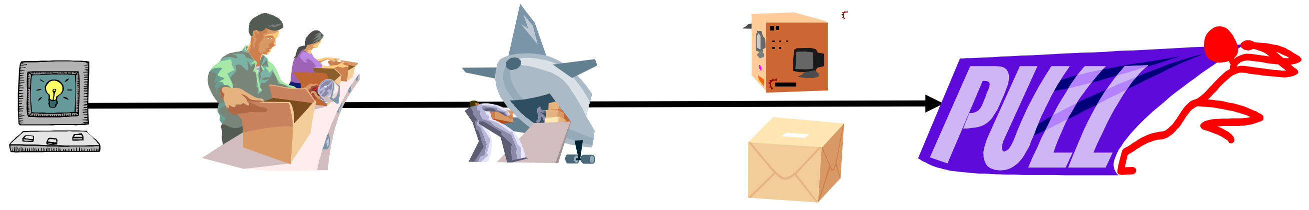
  **This is *not* waste!**

## Agenda

- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

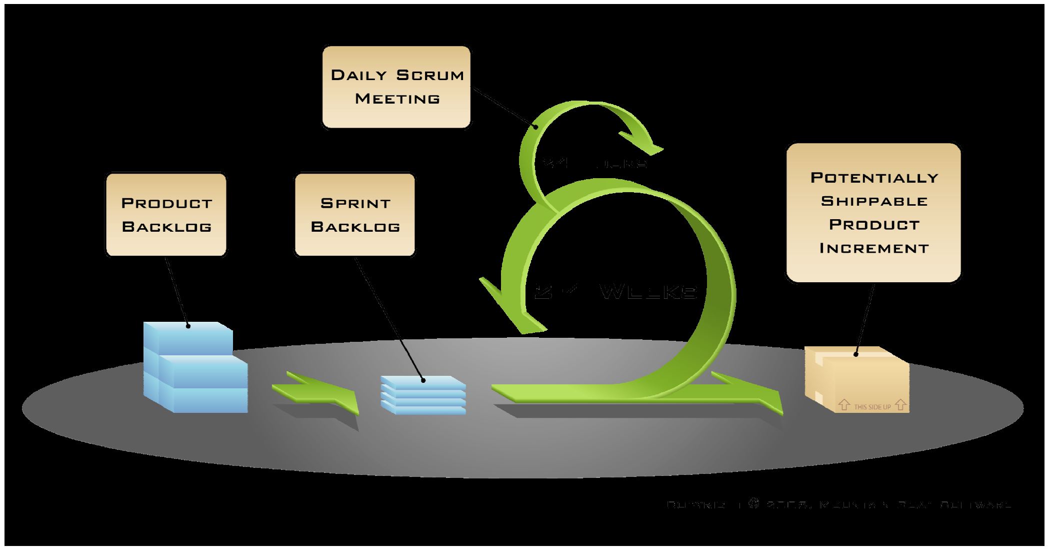
# Push vs Pull

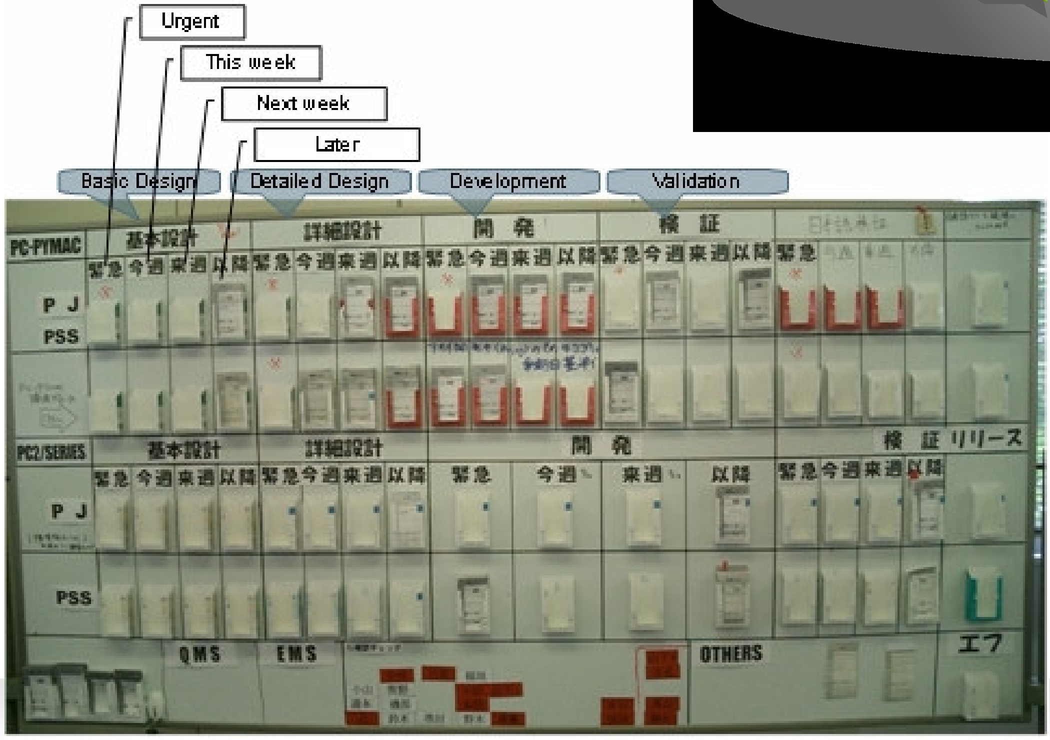




# Iterative Development

SCRUM





KANBAN

## Agenda

- Overview
  - Lean Software Development

- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge

## Environment

A TEAM

# Provide Effective Leadership

### *Marketing Leader*

- Business Responsibility
- Customer Understanding
- Roadmap Planning
- Tradeoffs

### *Technical Leader*

- System Architecture
  - At a high level
  - Work daily with those developing the details
- Technical Guidance
  - Integration
  - Tradeoffs

### *Process Leader*

- Build Block Disciplines
- Iterative Development
- Visible Workspace

### *Project Leader*

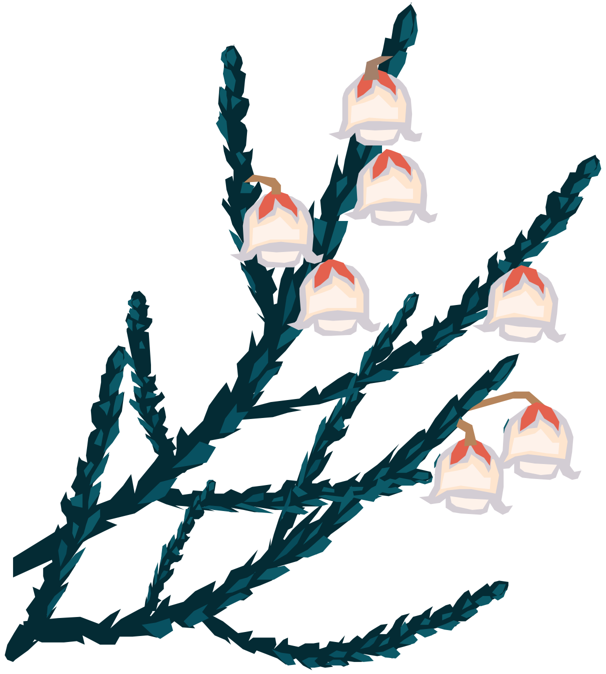
- Funding
- (Scheduling)
- Tracking

### *Functional Leader*

- Staffing
- Teaching
- Standards

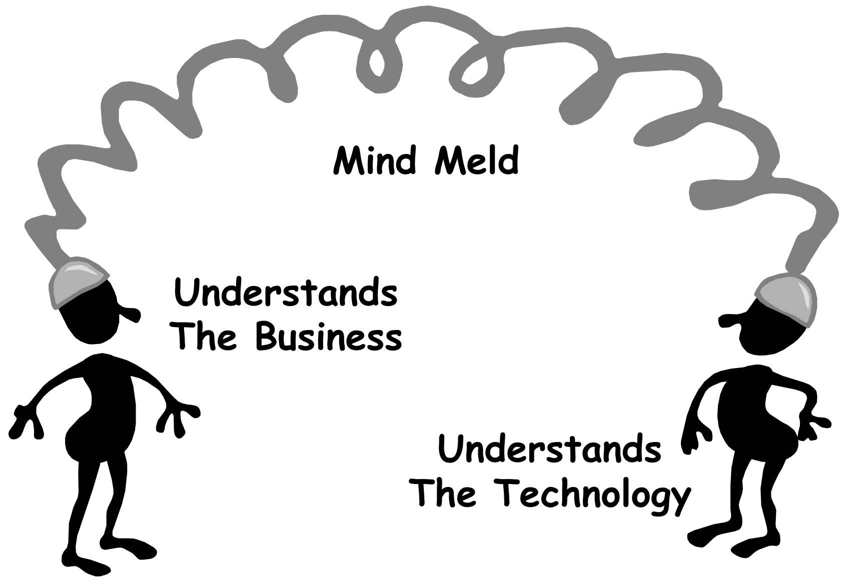
## Agenda

- Overview
  - Lean Software Development
- The 7 Lean Principles
  - Eliminate Waste
  - Improve the system
  - Build Quality In
  - Defer Commitment
  - Deliver Fast
  - Respect People
  - Create Knowledge



## Predictable performance is driven by feedback

- Set Up the feedback Loop
  - The job that customers need done
  - The right technology to do that job
  - Do it often and regularly



- Stop asking for
  - More documentation, more details in requirements, more plans, more commitments …
- Deliver!
  - Prototype, Minimum Features set, Draft document …
- Then ask for Feedback

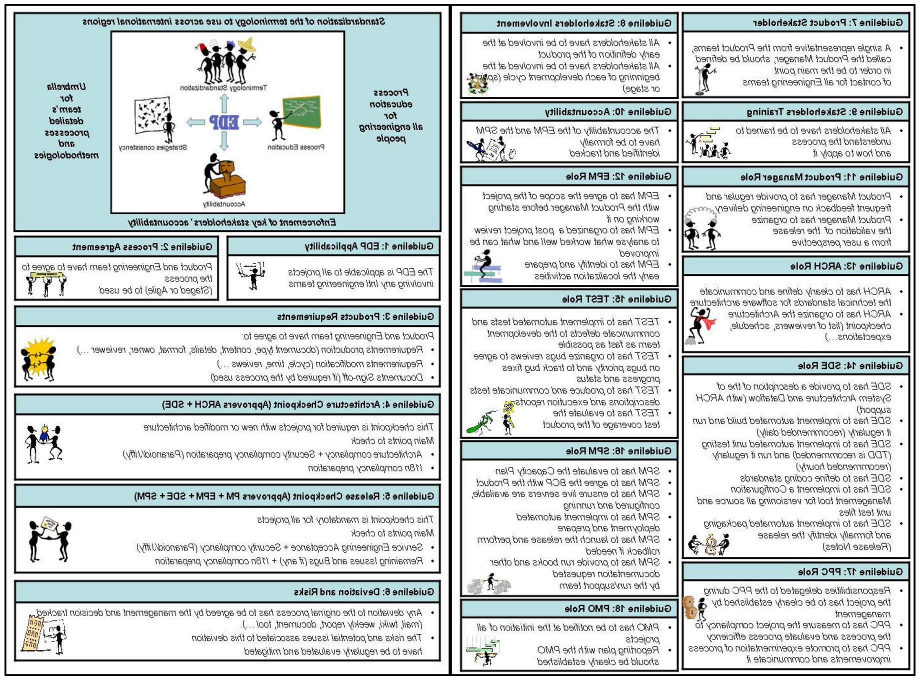
## Capturing Knowledge

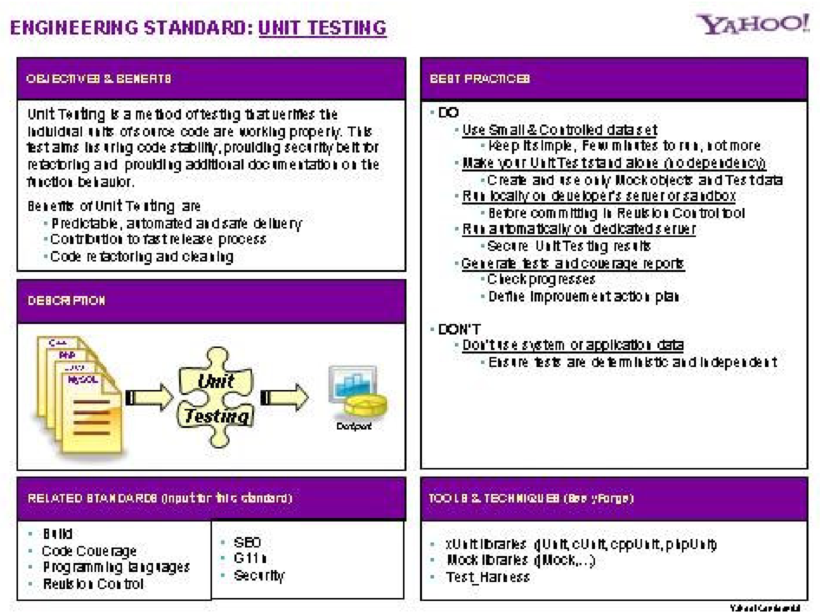
### *The A3 Report*

Two sheets of letter paper

### *Standards*

1 A4 page





ENGINEERING STANDARD: UNIT TESTING

OBJECTIVES & BENEFITS

Unit Testing is a method of testing that verifies the individual units of source code are working properly. This test aims insuring code stability, providing security belt for refactoring and providing additional documentation on the function behavior.

Benefits of Unit Testing are
- Predictable, automated and safe delivery
- Contribution to fast release process
- Code refactoring and cleaning

DESCRIPTION

BEST PRACTICES

- DO
  - Use Small & Controlled data set
    - Keep it simple, Few minutes to run, not more
  - Make your Unit Test stand alone (no dependency)
    - Create and use only Mock objects and Test data
  - Run locally on developer's server or sandbox
    - Before committing in Revision Control tool
  - Run automatically on dedicated server
    - Secure Unit Testing results
  - Generate tests and coverage reports
    - Check progresses
    - Define improvement action plan
- DON'T
  - Don't use system or application data
    - Ensure tests are deterministic and independent

RELATED STANDARDS (input for this standard)

- Build
- Code Coverage
- Programming languages
- Revision Control
- SEO
- G11n
- Security

TOOLS & TECHNIQUES (See yForge)

- xUnit libraries (jUnit, cUnit, cppUnit, phpUnit)
- Mock libraries (jMock,...)
- Test_Harness

Yahoo! Confidential

Thank You!