



# *Lean Software Development*

Value

Flow

People

*Mary Poppendieck*

mary@poppendieck.com

www.poppendieck.com

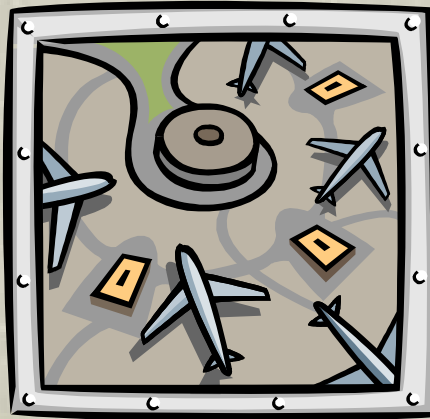
Author of: *Lean Software Development*

# *Value*

- ❖ Primary Focus of Lean Thinking
  - Create *Value* for the customer
  - Improve the *Value Stream*
- ❖ Waste
  - Anything that does not create value for the customer
  - The customer would be equally happy with the software without it

# *The Efficiency Paradox*

- ❖ We make our organization more efficient
- ❖ But customer value is decreased
- ❖ So we have not eliminated waste

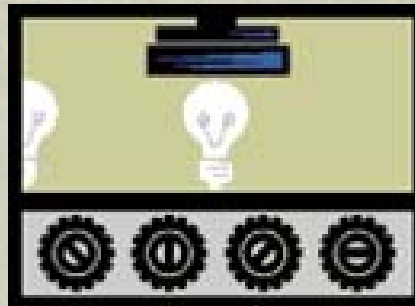


## *Example 1: Gating*

- ❖ We have limited resources
  - We let customer requests accumulate
  - Occasionally we review, prioritize and release work for action
- ❖ Customer value is decreased
  - Delay in realizing value
  - Priorities are not set by customers

# *The Lean Alternative*

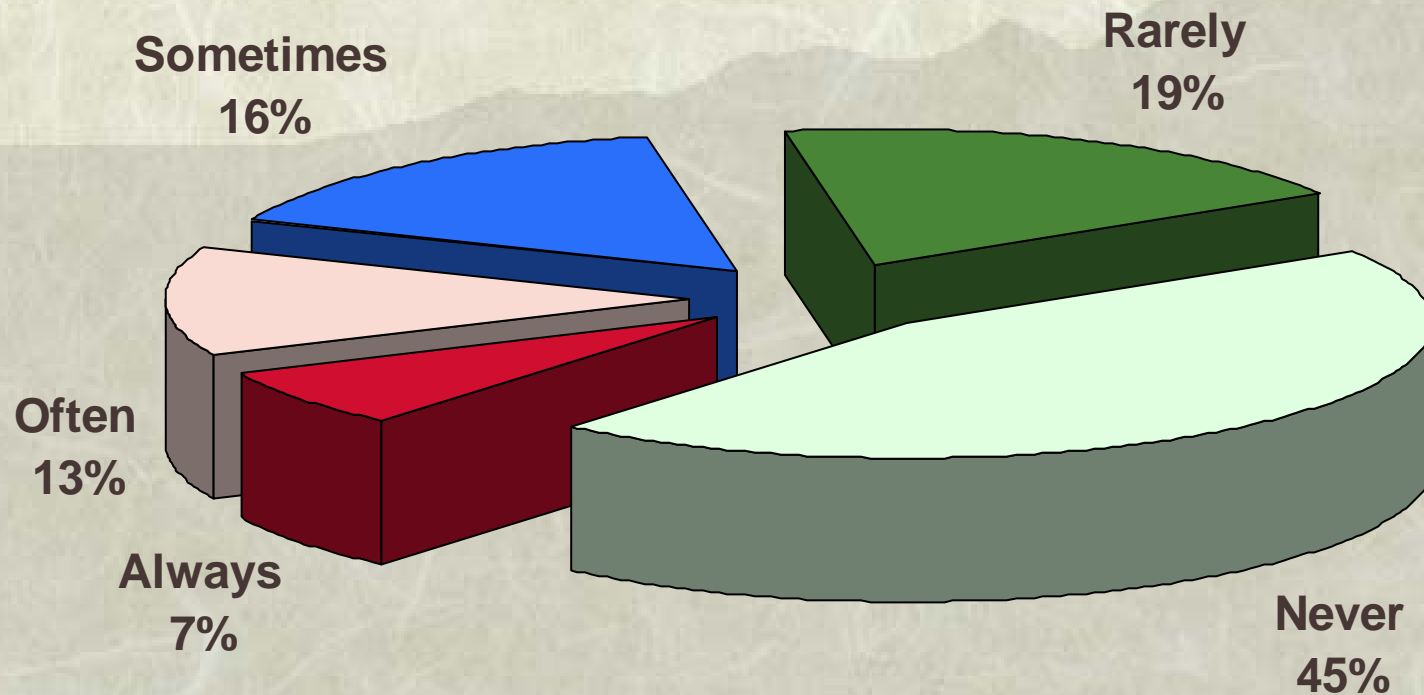
- ❖ No queues
- ❖ Small batches
- ❖ Continuous flow
- ❖ Customers set priorities



## *Example 2: Requirements Sign-off & Scope Control Systems*

- ❖ We need to get a handle on what we are doing ahead of time
  - We develop a list requirements and get customers to sign-off on them
  - Changes require written approval
- ❖ Customer value is decreased
  - Customers are constrained to their original, imperfect view of their problem
  - They aren't supposed to learn or change

# *The Biggest Cost of Early Specification: Extra Features*



Features and Functions Used in a Typical System

*Standish Group Study Reported in 2000 Chaos Report.*

## *The Lean Alternative*

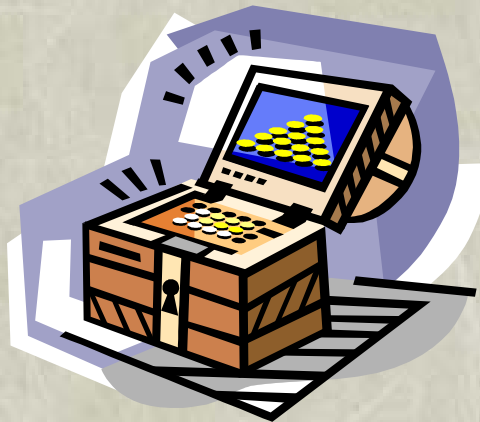
- ❖ Recognize that sign-off / scope control motivate customers to inflate scope
- ❖ Recognize that an early rush to detail interferes with good design
- ❖ Control scope by overall mission
- ❖ Control development with feedback
- ❖ Stop when you run out of resources
- ❖ Use Target-Cost contracts

## *Example 3: Maintenance Policies*

- ❖ We want to realize revenue from maintenance
  - We rush to declare we are in maintenance mode
  - We limit customer access to technical support
- ❖ Customer value is decreased
  - Customers resist acceptance because there are always more features that could be added
  - Customers worry that they won't be able to get problems resolved easily

# *The Lean Alternative*

- ❖ Become involved in the customer's value stream
  - Assist with deployment
  - Be sure customer realizes value
- ❖ Look for ways to increase value
  - This is the most reliable revenue stream



# *Lean Principle #1: Eliminate Waste*

Is each step in the process

- ❖ Valuable
  - Or would the customer be equally happy with the result if the step were left out?
- ❖ Adequate
  - Is there capacity to perform the step whenever the value stream requires it?
- ❖ Capable
  - Does it produce reliable results each time?
- ❖ Flexible
  - Can it quickly shift to satisfying a new request without compromising adequacy and capability?

# *Flow*

- ❖ Work *flows* from one step to the next with no delay.
- ❖ Work flows only at the *pull* of a customer request.
- ❖ *Feedback* is rapid and reliable.

# *The Predictability Paradox*



- ❖ Predictability is highly valued in business
- ❖ The best way to increase predictability is to reduce the amount of guesswork that goes into a prediction

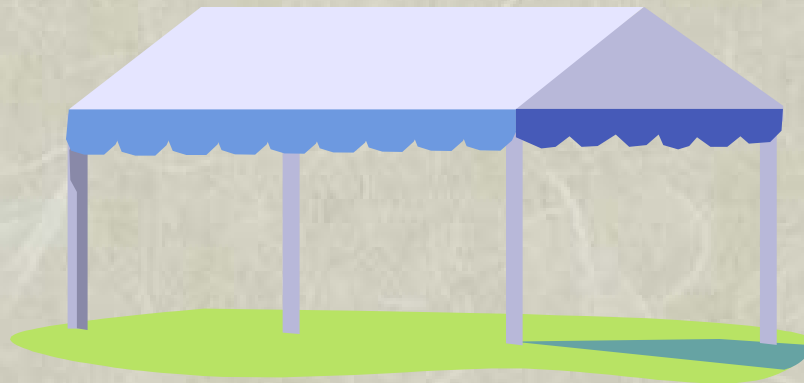


# *Example 1: Delaying Decisions Increases Predictability*

- ❖ We want to predict cost, schedule, & scope
  - We push for early decisions
  - We break the project down into detailed tasks
  - We add up the cost and schedule of each task
- ❖ But the best predictions are based on facts not forecasts
- ❖ Wait
  - For customers to ‘place the order’
  - For technology to ‘settle down’
  - For business needs to ‘gel’

## *Example 2: The Most Accurate Forecast Is No Forecast*

- ❖ We work hard to make accurate forecasts
- ❖ But the best approach is to reduce system response time so the system can respond to change rather than predict it
- ❖ Don't depend on forecasts  ? 
  - Rent a tent



## *Example 3: Don't Predict, Project From Experience*

- ❖ We must be able to predict performance so customers can decide what they want
- ❖ But we need to know what customers want in order to predict performance
- ❖ Break the cycle with iterations which implement increments of business value
  - Customers learn how to describe what they want
  - Developers learn to estimate what it will take
  - A history of accomplishment becomes the basis of accurate projections

# *Lean Principle #3*

## *Decide as Late as Possible*

- ❖ Delay Commitment Until The Last Responsible Moment
  - The point when failing to decide eliminates an important option
- ❖ Narrow Tolerances Gradually
  - Communicate Constraints
  - Not Solutions
- ❖ Keep Options Open
  - Make fact-based decisions

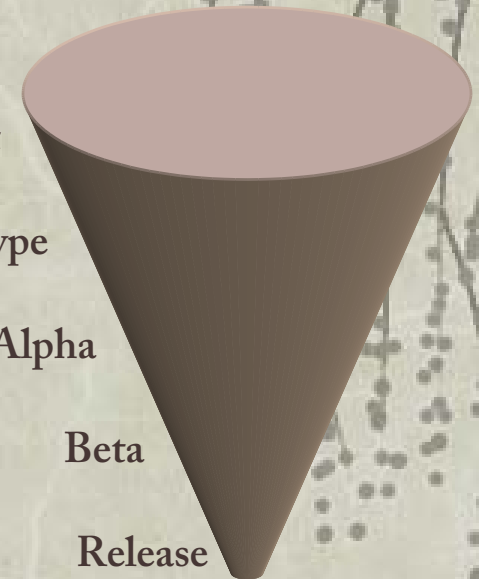
Concept

Prototype

Alpha

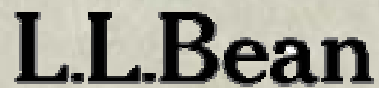
Beta

Release



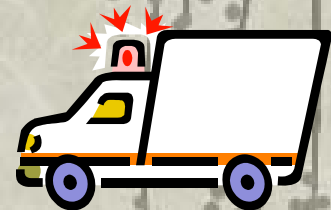
# *The Maturity Paradox*

- ❖ We think of maturity as taking the time to planning thoroughly before acting
- ❖ But breakthrough competitive advantage comes from speed



# *Example 1: Planning*

- ❖ We are measured by our ability to develop a plan and then execute it
  - So we create a detailed plan and make sure everyone follows it
- ❖ Planning is a very good thing
  - But don't mix up predicting with planning
- ❖ Predictive planning doesn't work in complex / changing environments
  - Organize for local signaling & commitment
  - Make work self-directing



# *The Lean Approach*

- ❖ Plan at a mission and release level
- ❖ Plan at the boundaries, the interfaces, the failure modes, the high risk areas
- ❖ Plan details only for the current iteration
- ❖ Involve those who will develop and those who will use the system in the planning
- ❖ Communicate the goal, not how to get there
- ❖ Organize work to be self-directing

## *Example 2: Disaggregation*

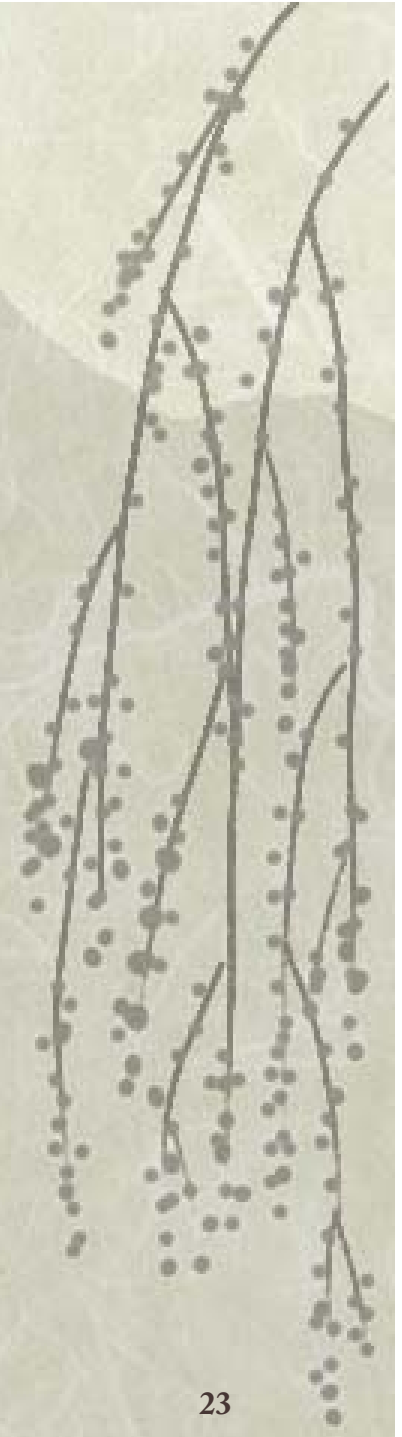
- ❖ Software capability maturity is measured by evaluating many KPA's (Key Process Areas)
  - We use many KPA's so nothing is overlooked
- ❖ But disaggregation does not assure we cover *everything* – aggregation works better
  - A high maturity assessment does not guarantee that customers will be satisfied
  - But routinely satisfied customers guarantee that the organization is highly disciplined

# *The Lean Approach*

- ❖ Measure *UP*
  - When measurements don't cover everything, measure one level higher
  - Aggregate, don't disaggregate
- ❖ Measure maturity by satisfied customers and financial success
- ❖ Disciplined people and disciplined thought are the foundation of maturity

## *Example 3: Speed*

- ❖ Maturity and speed don't seem to be compatible
  - Speed is associated with sloppy, ad-hoc development
  - Slow, deliberate processes are associated with discipline
- ❖ But the most disciplined organizations are those that respond to customer requests
  - Rapidly
  - Reliably
  - Repeatedly
- ❖ The measure of a mature organization is the speed at which it can repeatedly and reliably respond to customer demands



# *Lean Principle #4: Deliver as Fast as Possible*

- ❖ Nothing is done unless it is *pulled* in small increments by a customer demand
- ❖ Once the customer requirement is clear, development *flows* without delay to tested, integrated, deployable software
- ❖ Immediate customer feedback provides quality control, risk mitigation, immediate ROI, forecast reliability, among many other benefits
- ❖ None of this is possible without a highly disciplined (mature) organization

# *People*

- ❖ Everyone agrees with the slogan
  - ‘People are our most important asset’
- ❖ But are these empty words?
- ❖ Do we
  - Involve workers from the start?
  - Make maximum use of everyone’s capability?
  - Trust workers to solve our problems?

# *The Productivity Paradox*

- ❖ We want people to be as productive as possible, so we
  - Standardize each processes so anyone can do it
  - Have each person specialize in only one job
  - Hand off work from architects to analysts to designers to programmers to testers to users
- ❖ But the batch & queue approach to productivity was abandoned by manufacturing two decades ago
  - We know that rapid flow of small batches through work cells that produce complete products is far more productive
  - We know that cross functional teams design better products that are easier to manufacture
  - This works for software too!

# *Example 1: Involvement is More Productive Than Handoffs*

- ❖ If we ask
  - Why should high priced architects get involved in the details of implementation?
  - Why waste valuable testing time until the code is finished?
  - Why involve busy help desk people until the software is released?
- ❖ Then we don't realize that
  - Integrated Product Teams are the most productive, produce the highest quality and the greatest value
  - At each document handoff, more than 50% of all tacit knowledge is lost.
  - Short feedback loops are essential for complex systems

## *Example 2: Telling People What To Do Limits Their Productivity*

- ❖ We want to get started quickly and produce uniform results
  - So we develop standard processes
  - And train people to follow the instructions
- ❖ But then we are not leveraging the capability of front line workers
  - They know their process best
  - They know their problems best
  - They are the best ones to design & improve their own processes

## *Example 3: Front Line People Can Solve Management Problems*

- ❖ Problems fester because managers and specialists don't have the time or motivation to address them
  - But ignoring problems or applying superficial solutions is a downhill spiral
- ❖ Enlist front line workers and trust them
  - Kaizan Events
  - GE Work-Out
  - The Listening Game

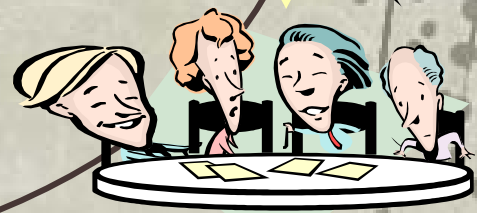
# THE LISTENING GAME



*Bring people together*



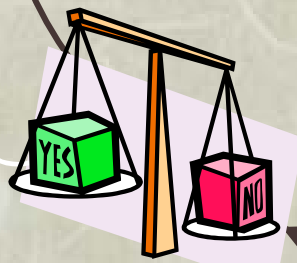
*Give them a challenge*



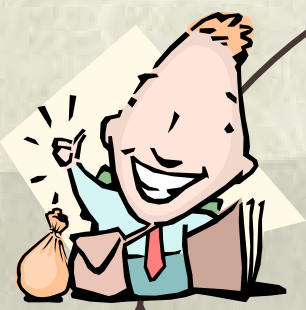
*Brainstorm solutions*



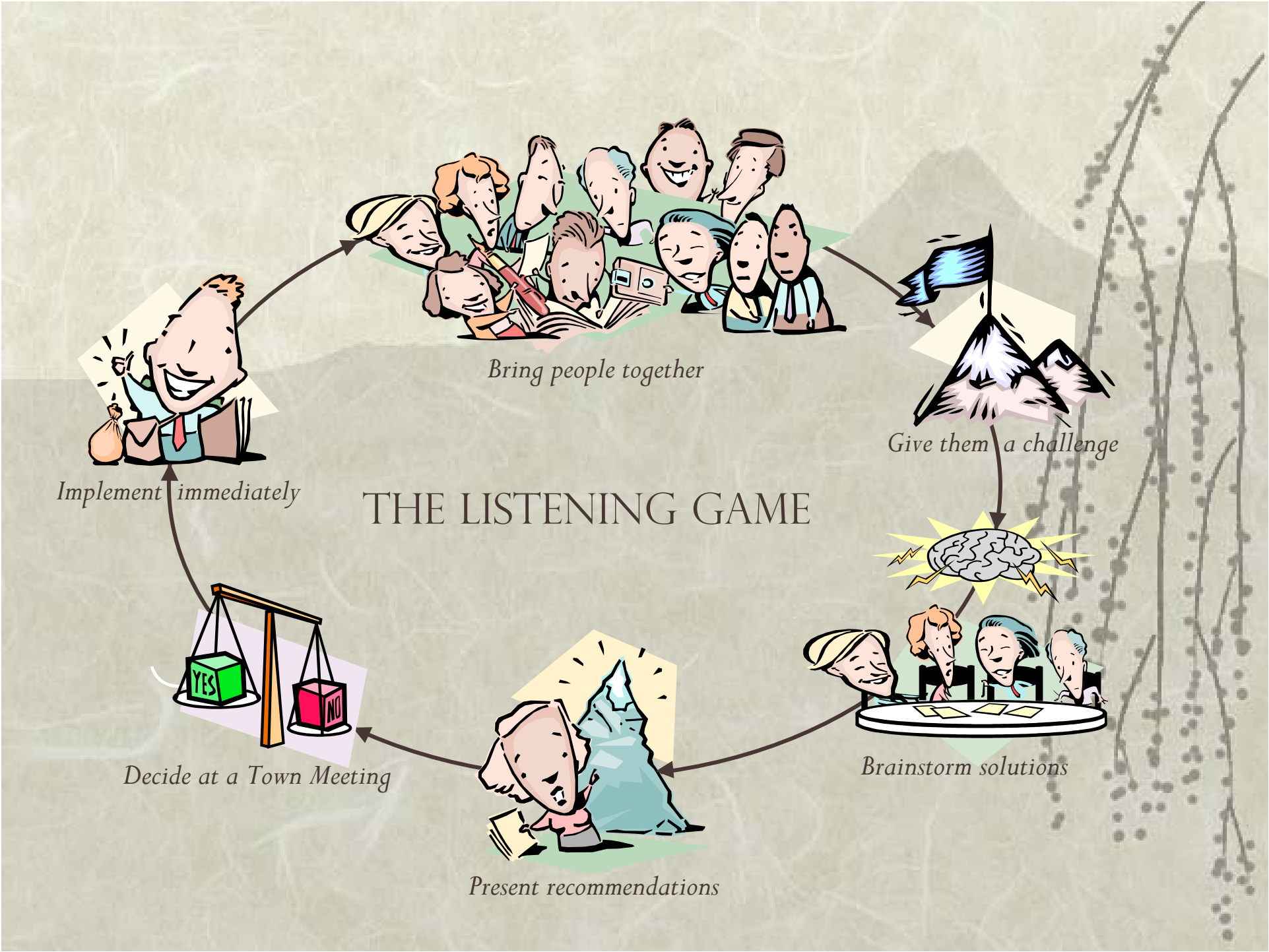
*Present recommendations*



*Decide at a Town Meeting*



*Implement immediately*



# *Lean Principle # 5: Empower The Team*

Provide

- ❖ Self Determination
- ❖ Purpose
- ❖ Leadership
- ❖ Expertise

and the team will meet the challenge

# *Principles of Lean Thinking*

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