

# Project Management Course Book

T++ Technical Skills Training Program

CUNY Institute for Software Design & Development (CISDD)

New York Software Industry Association (NYSIA)

Aron Trauring

Class 5 October 19th, 2004

# Class Agenda

**1:30-2:45** Agile Management Theory - Part I (Presentation)

**2:45-3:00** Break

**3:00-3:45** The Making of Spirited Away - Video

**3:45-4:15** Exercise 5 — What Makes Hayao Miyazaki an Agile Project Manager?

**4:00-4:15** Break

**4:15-5:30** Breakout Presentations and Analysis

# Agile Management Theory - Part I

## The Outsourcing Threat

“Jobs are at stake!...The answer isn't that software developers must work harder if they want to keep their jobs...The answer is that management techniques must improve, in order to improve competitiveness.”

—David Anderson

- Outsourcing = The Mythical Man Month
- Knowledge work isn't like manufacturing
- Learning Organizations — culture of continuous improvement

## The Problem with Agile Methods

- Go against analogy of engineering - counter intuitive
- Radical change is scary
- Must provide believable business metrics
- Must address real business benefits — improved value-added and ROI

## The Theory of Constraints (TOC)

- A production facility is only as fast as the slowest process in the chain
- The capacity of the weakest link is the current system constraint (bottleneck)
- One weakest link at any given time
- New material is introduced into the system only at the rate it can be consumed by the bottleneck.

## Just-in-Time Inventory (JIT)

- Inventory delivered to the point it is consumed just before it is needed
- Reducing inventory improves profitability by reducing investment costs
- Reducing inventory improves profitability by reducing operating expenses

## Quality Assurance (QA)

- Improved quality reduces rework (waste) and so improves profitability
- Deming — Statistical Process Control
- Six Sigma — improve quality through reducing variance in defects

**Lean Production**

- Toyota Production System — Kanban System
- Minimize waste
- Conformance to specification

**Agile Manufacturing**

- Lower priced goods
- Higher quality
- Higher profitability
- Improved competitiveness — faster to react to market conditions
- Scientific manufacturing

**Three Phases of Scientific Development — Eli Goldratt**

1. Classification — nomenclature
2. Correlation — corroborating evidence for theory
3. Effect-Cause-Effect - understanding cause so you can predict future effects

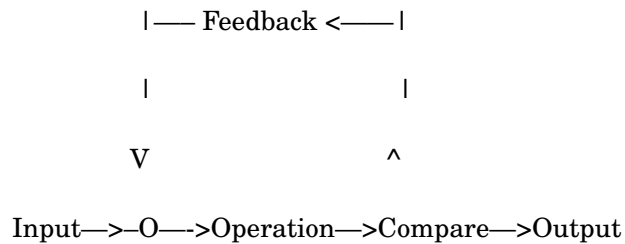
**Empirical vs. Defined**

- Empirical processes — no underlying theory — Phase 2
- Empirical measurement — calendar
- Defined processes — underlying theory — Phase 3
- Defined measurement — getting a rocket to moon

**Where is Software Development?**

- FourM — Inherent Complexity and No Silver Bullet
- Agilist — Software engineering is at best empirical
- Chaordic (Highsmith) — chaos and order that defy predictive planning
- Chaordic implies software engineering is not predictable
- Convergent — under some form of control a process will converge to predictable results
- “Empirical processes can be convergent and therefore predictable” — Anderson
- Even in defined processes we recognize uncertainty
- Uncertainty means processes can be convergent but only to an approximation
- Agile management — learning to cope with uncertainty

## Systems Thinking



- Repeat & Rinse
- Nonlinear — no linear relationship between effort expended in operation and input or output
- Output is proportional to input
- Feedback loops introduce non-linearity
- Adaptive behavior — learning through feedback to create desired behavior
- Emergent behavior — artifacts, by-products, internally observable behavior
- Self-organizing — systems can't be controlled by detailed rules
- Feedback is used for learning how to improve processes
- Simple rules produce complex behavior
- Convergence — getting the system to 'settle down' and produce desired behavior

## Managing Complex Adaptive Systems

- Organizations are complex adaptive systems with feedback loops
- Set the goals — i.e. define desired output
- Measure feedback and goals
- Create an environment for self-organization and convergent adaptive behavior
- Set goals which lead to global optima not local efficiencies
- Detail complexity — something with many parts
- Detail complexity can be dealt with through decomposition relatively easy
- Inherent complexity — system with many nested feedback loops
- Inherent complexity requires understanding how varying the rules will leverage system effects
- Inherent complexity far more difficult



**Profit and ROI**

- Net Profit = Throughput - Operating Expense
- ROI = Net Profit / Investment
- Increase Throughput — produce more value
- Decrease Investment — reduce inventory
- Decrease Operating Expenses — reduce waste
- Nowhere near a predictive metric
- T — predicting sales harder than software and internally not really understood
- I — not even measured
- OE — Hard to guess

**Throughput Accounting vs. Cost Accounting**

- OE is best known hence most focused on
- Costs accounting sees labor and machinery as variable costs
- Inactive labor and machinery is put into overhead
- Distorts unit costs by focusing on local efficiency
- How many units can be transformed for a dollar of local value
- Throughput accounting focuses on delivered value
- How effective is the system in moving investment value through the system and converting it to throughput
- How many hours or minutes to earn a dollar of value added — reducing lead time
- Throughput accounting better fit to changeable nature of software

**Relative Importance of T, I and OE**

- OE is bounded — can't reduce below zero reducto ad absurdum
- Focuses on internal matters not customers — customer service rage
- T is unbounded — continuous and constant improvement
- T is customer focused — focus on value
- $I > OE$
- Reducing I also reduces OE by reducing carrying costs of borrowing
- Reducing I reduces Lead Time to delivering value — focus on flow
- Reducing OE is the least important for long term profitability

