Lean Software Development

"Fat processes are out. Thin is in" –
Jim Highsmith

Rigid or Lean

- In 1990, was the decade of process for IT.
  - Plan the work, and work the plan.
  - The list of companies who were riding the CMM ladder early in 1990s, were the same companies who were downsizing by the end of 1990s.

We prostrated before ISO and CMM.

It wasn't enough to do things right, we also had to say in advance exactly what we intended to do and do that exactly that\

Process rigor was simply not the right recipe, when change is the only constant.

Lean Thinking

- Capitalizes on the intelligence of frontline workers, believing that they are the ones who should determine and continually improve the way they do their jobs.
- First emerged in manufacturing.

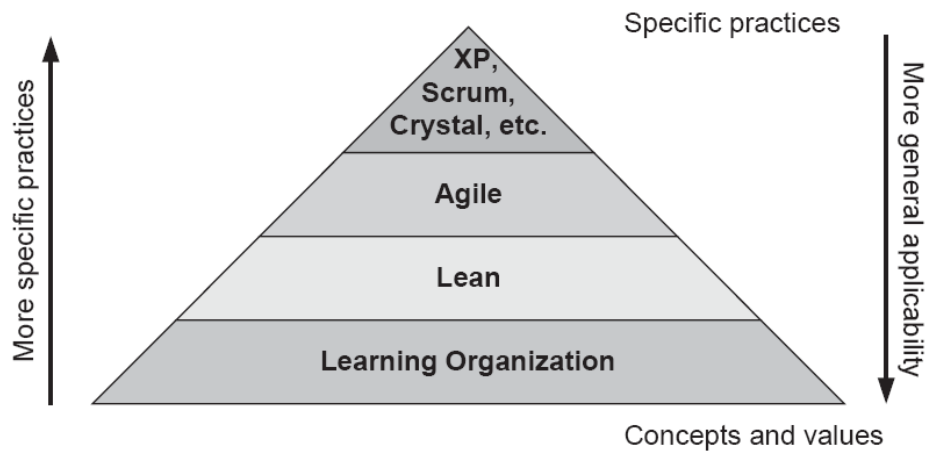Mary and Tom Poppendieck transferred the principles from manufacturing to software development.

Stop the line culture. If a team thinks that it is not going to meet the Sprint backlog, abort the Sprint.
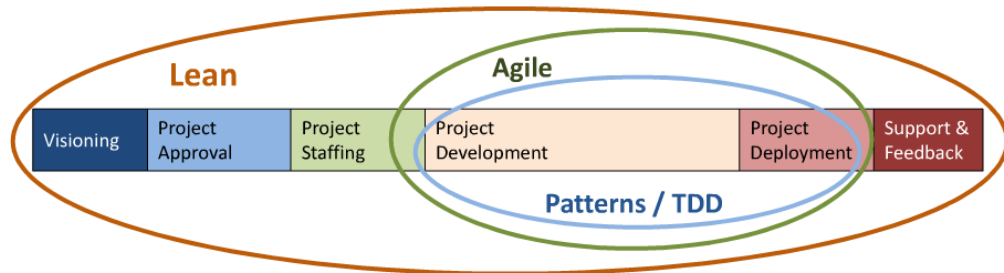
Lean Compared

Use of Lean

Seven Lean Principles

- Eliminate Waste
- Build Quality in
- Create Knowledge
- Learn before Commitment
- Deliver as fast as possible
- Respect People
- Optimize the whole

1 – Eliminate Waste

- Anything we do that does not add customer value is waste.
  - Any delay that keeps customers from getting value when they want it is also waste.

Increase speed by reducing the waiting time. So quality improves and cost reduces. Traditional methods try to reduce the cost. Usually quality deteriorates.

Code Less

"The task then is to refine the code base to better meet customer need. If that is not clear, the programmers should not write a line of code. Every line of code costs money to write and more money to support. It is better for the developers to be surfing than writing code that won't be needed. If they write code that ultimately is not used, I will be paying for that code for the life of the system, which is typically longer than my professional life. If they went surfing, they would have fun, and I would have a less expensive system and fewer headaches to maintain."
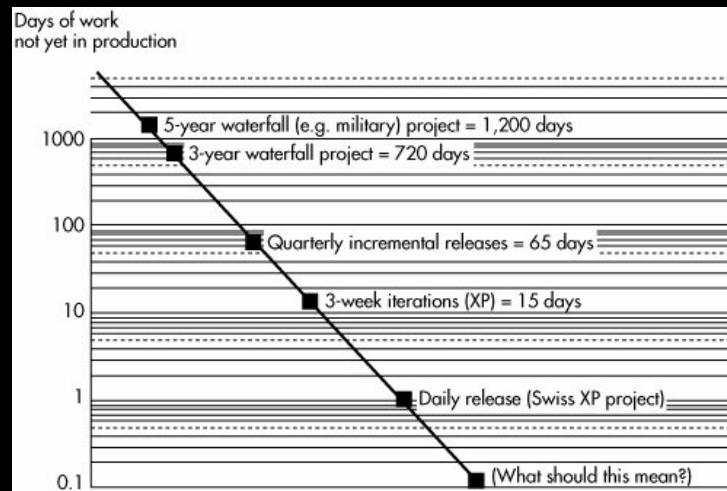
Jeff Sutherland, CTO PatientKeeper

Work In Progress

- **Work in process** or **in-process inventory** consists of the unfinished products in a production process.
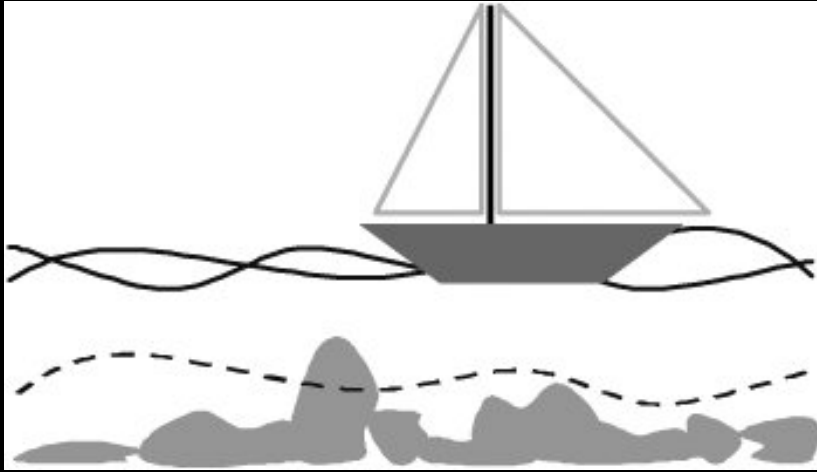
Inventory

Days of work
not yet in production

1000 — ■ 5-year waterfall (e.g. military) project = 1,200 days
■ 3-year waterfall project = 720 days

100 — ■ Quarterly incremental releases = 65 days

10 — ■ 3-week iterations (XP) = 15 days

1 — ■ Daily release (Swiss XP project)

0.1 — ■ (What should this mean?)

Less Inventory

Problems surface sooner

Do you have a tendency to work in batches?

If you had to mail 100 letters, how would you go about folding the letters, stuffing the envelopes, adding address labels and stamps?

Would you process one envelope at a time, or would you perform each step in a batch?

Why?

Try timing both ways and see which is faster.

If you have children, ask them how they would approach the problem.

#2 – Build Quality in

- Build quality in the code and not test it later.
- Our policy – Zero Known defects at end of every iteration
  - Myth: The job of testing is to find defects.
  - Reality in Agile: The job of testing is to prevent defects.

People forget how fast you did a job –
but they always remember how well you did it.

You don't focus on putting defects into a tracking system; you avoid creating defects in the first place.

Off shoring from a developed country to developing only reduces the cost by about 30%.

A company typically has to pay Indian developers about 30% of what it pays developers in USA. The catch is that it there is extra project management overhead. So it costs the company 70 cents for every dollar after off shoring.

Jobs aren't going in search of low salaries. Jobs are going in search of integrity and accountability.

If integrity and accountability can better be supplied by a separate company many time zones away, customers will pay the necessary price in difficult communication to get them.
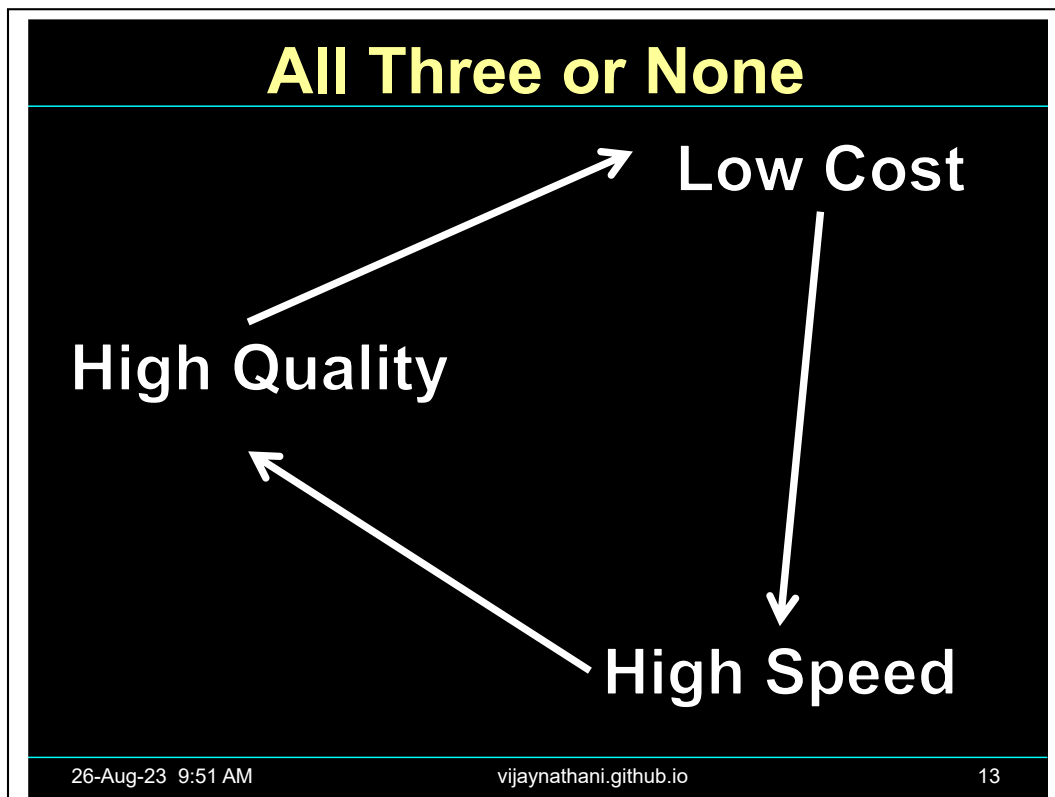
In the long run

#3 – Create Knowledge

- A development process focused on creating knowledge will expect the design to evolve during coding and will not waste time locking it down prematurely
    - As Yourdon points out, "A piece of program logic often needs to be rewritten three or four times before it can be considered an elegant, professional piece of work."
    - Why, he asks, do we object to revising programming logic when we are quite happy to rewrite prose three or four times to achieve a professional result?

The detailed design of software always occurs during coding, even if a detailed design document was written ahead of time.

An early design cannot fully anticipate the complexity encountered during implementation, nor can it take into account the ongoing feedback that comes from actually building the software.

Worse, early detailed designs are not amenable to feedback from stakeholders and customers.

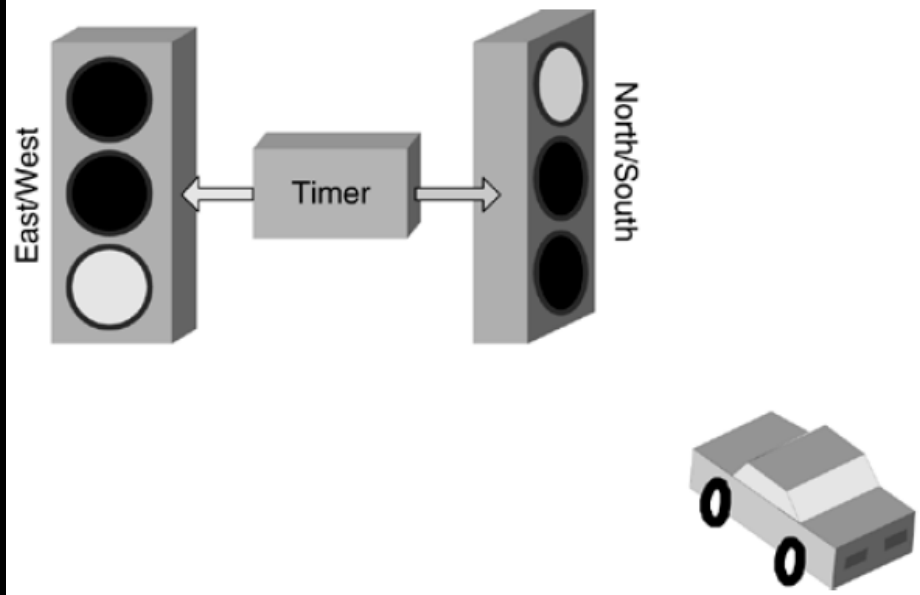Myth: Predictions create predictability

Good Practices

      Frequent releases to customers

      Daily builds and continuous integration

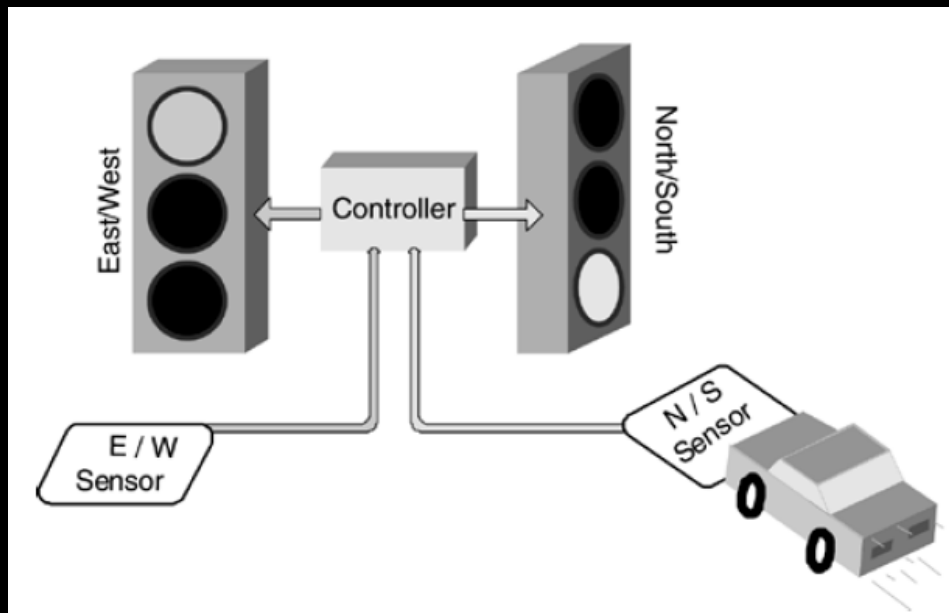   Continuous refactoring

      Pair Programming

Timed Traffic Light

26-Aug-23 9:51 AM   vijaynathani.github.io   15

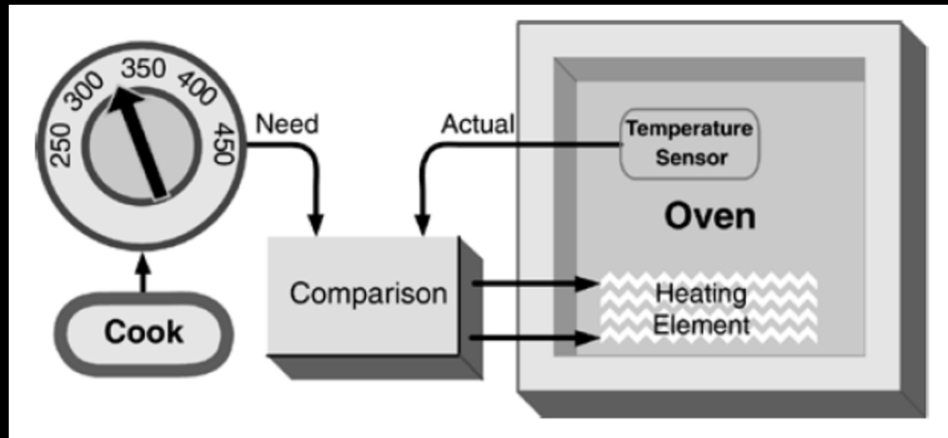**Traffic Light with Sensors**

Better Feedback

Oven

Continuously monitor the temperature and adjust

Estimate Accuracy
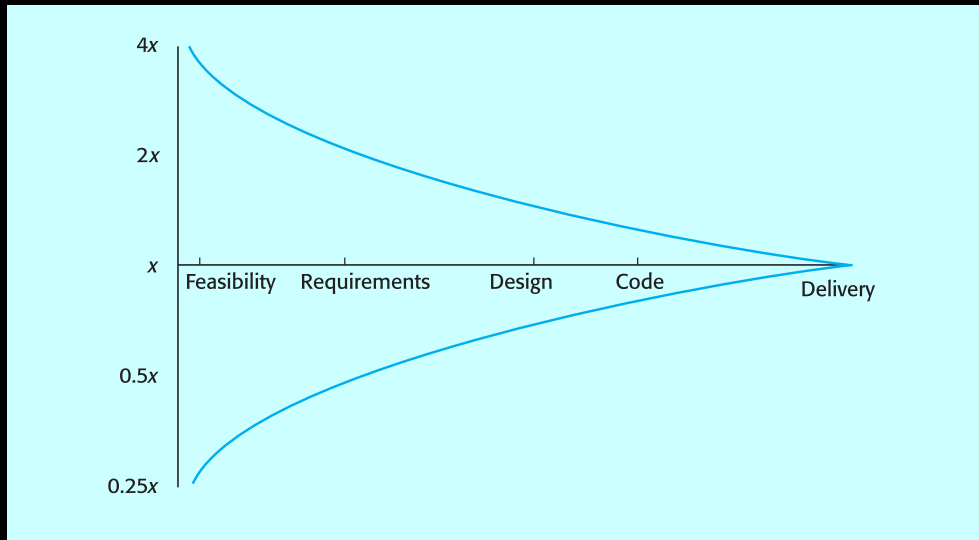
Product Burndown Chart

Staff Days to Complete

Need to trim backlog
to meet August target

700
600
500
400
300
200
100
0

jan feb mar apr may jun jul aug sep oct nov dec

Staff Days to Complete

Project probably completes
by August target date

700
600
500
400
300
200
100
0

jan feb mar apr may jun jul aug sep oct nov dec
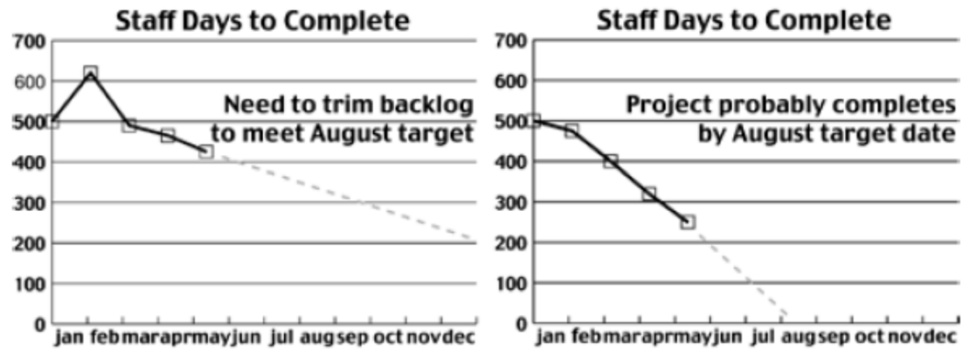
The project on the right will finish in Aug. The project on left will probably not finish, based on current scope

#4 – Learn before Commitment

- Schedule irreversible decisions for the last responsible moment, that is, the last chance to make the decision before it is too late.
  - We should try to make most decisions reversible, so they can be made and then easily changed.

Myth: Planning is commitment

Sticking to a detailed plan is not healthy, and measuring process capability against ones ability to do so is measuring the wrong thing.

Planning is not the same thing as making a commitment. We should plan thoughtfully and commit sparingly.

Agile

Reallocate funds to strong performers let's change the game

26-Aug-23 9:51 AM   vijaynathani.github.io   21

If in the middle of the game, our horse falls.

Can we give up 50% of our bet and bet the remaining on the horse that is leading?

Agile

Actively Manage Portfolio

- Reevaluate priorities constantly

- Reshuffle portfolio continuously

- Start/Stop projects

- Recognize increments of value

#5 – Deliver Fast

- Companies that compete on the basis of time often have a significant cost advantage over their competitors.
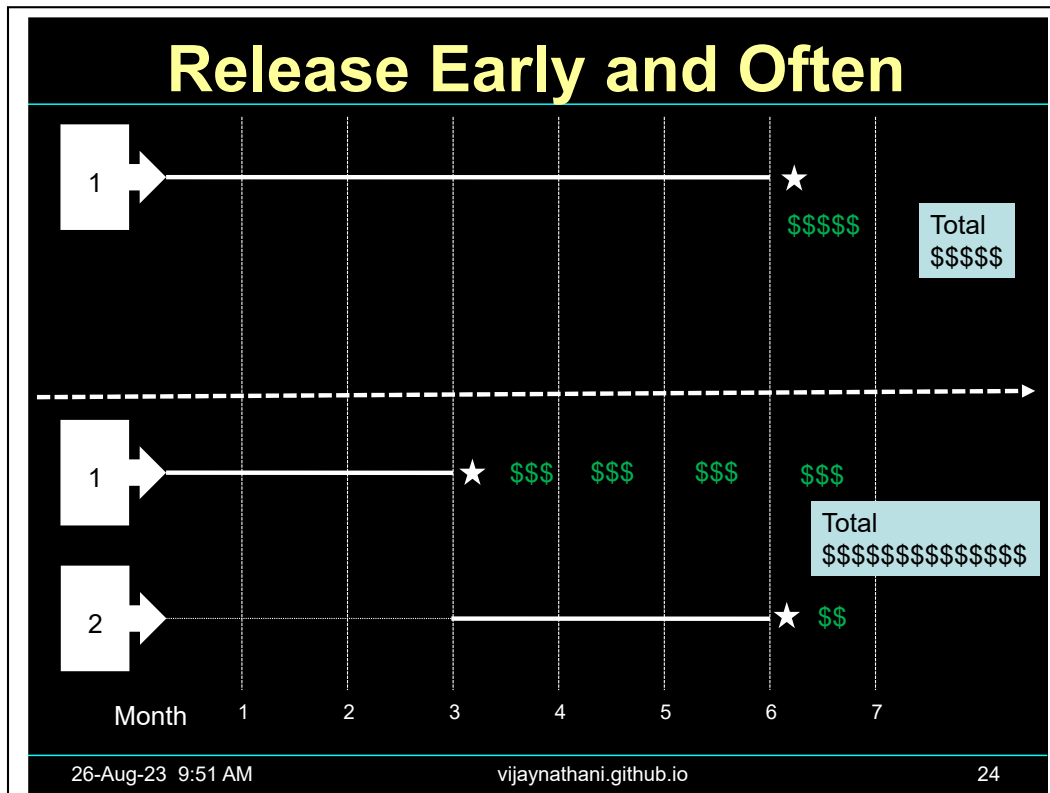
- Shorter the iterations, the better.

They have eliminated a huge amount of waste, and waste costs money. In addition, they have extremely low defect rates.

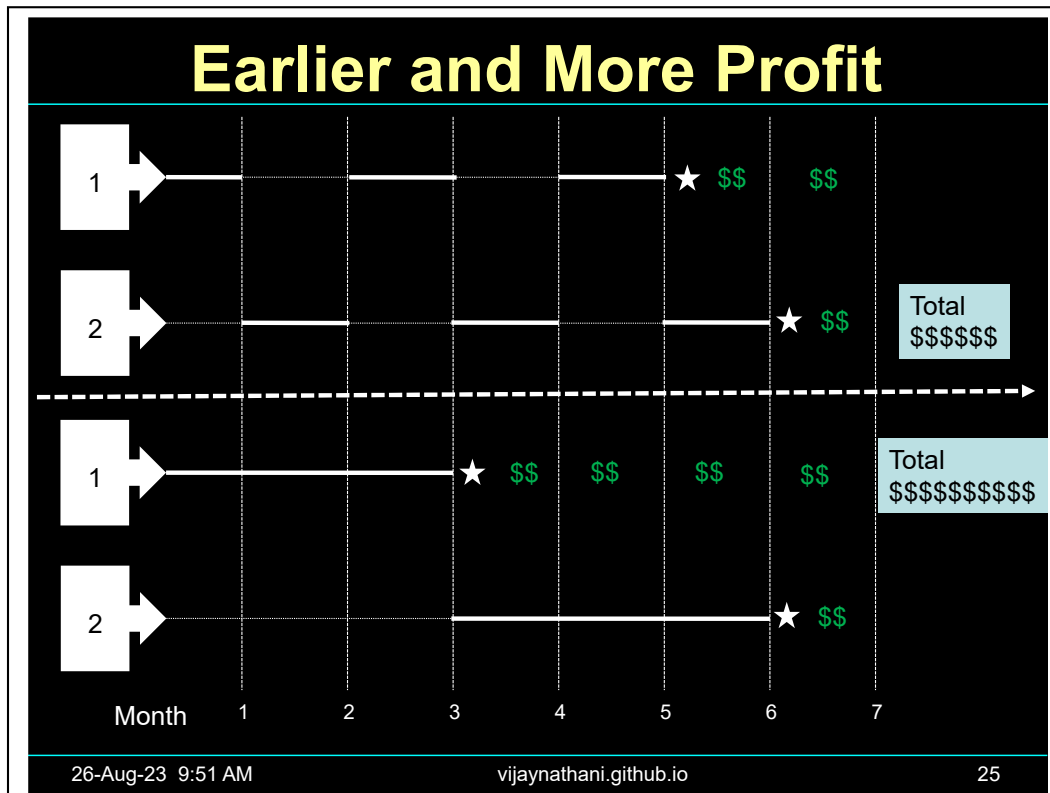Repeatable and reliable speed is impossible without superb quality.

Myth: Haste makes Waste

> Don't equate high speed with hacking. You can't sustain high speed unless you build quality in.

In first case, we release after 6 months.

In Second case, we release the important features after 3 months and remaining features in 3 months.
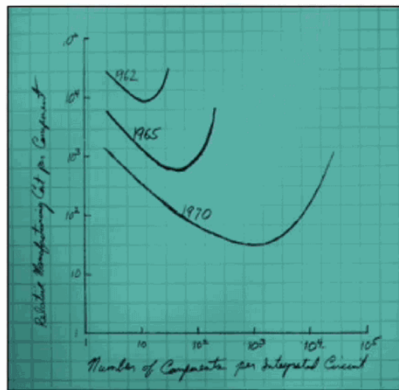
Assume we have two projects of high priority and which bring in same revenue when done.

Most companies feel good if both projects are going on together, but this is less profitable.

Lean: Finish one project and then do another. Avoid task switching for people.

Moore's Law in Software
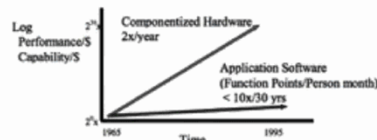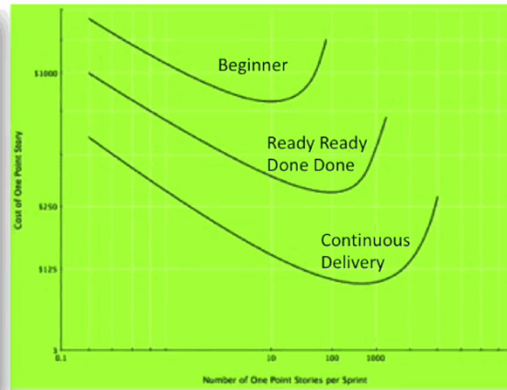
Transistors on a Chip — Stories in a Sprint

The emergence of a Business Object Component Architecture
Sutherland, J.
Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999.
(WET ICE '99) Proceedings. IEEE 8th International Workshops on
Year: 1999
Pages: 330 - 340, DOI: 10.1109/ENABL.1999.805223
Cited by: Patents (3)
IEEE Conference Publications

26-Aug-23 9:51 AM   vijaynathani.github.io   26

Moore's law: Number of transistors in IC doubles every two years.

On X-axis: Number of transistors in IC (log scale)

On Y-axis: Cost per transistor (log scale)

The cost decreases as transistor density increases with time. So Laptops become twice as fast, twice as powerful almost every 2 years.

Source: Scrum as Scale by Jeff Sutterland. GOTO Amsterdam 2015

The graph on the right shows why Scrum succeeds.

**Good Thinking.
Good Products**

Respecting people means that teams are given general plans and reasonable goals and are trusted to self-organize to meet the goals.

> Respect means that instead of telling people what to do and how to do it, you develop a reflexive organization where people use their heads and figure this out for themselves.

A scrumMaster would never sit in a separate cubicle. He would sit with the team. This is called "Go to the Gamba". Sit where the work is being done.

Instead of calling people to you. You go to the people.
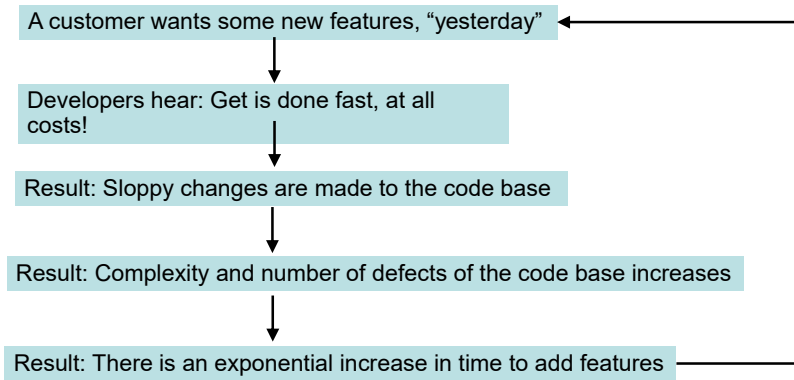
#6 – Respect for People

- If you put fences around people, you get sheep.
  - Give people the room they need."
- Myth: There is one best way.
  - There is no process that cannot be improved.
  - Our goal should be never-ending continuous improvement process.

7 – Optimize the Whole

- Software development is legendary for its tendency to sub optimize

- Myth: Optimize by decomposition

Vicious Cycle

A customer wants some new features, "yesterday"

↓

Developers hear: Get is done fast, at all costs!

↓

Result: Sloppy changes are made to the code base

↓

Result: Complexity and number of defects of the code base increases

↓

Result: There is an exponential increase in time to add features

Vicious Cycle

Testing is overloaded with work

Result: Testing occurs long after coding

Result: Developers don't get immediate feedback

Result: Developers create more defects

Result: Testing has more work. Systems have more defects

Result: Feedback to developers is delayed further

Stop The Line Mentality

- Detect abnormalities the moment they happen
- Don't fight fires!

  Stop!

- Find and fix the problem
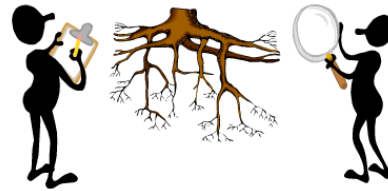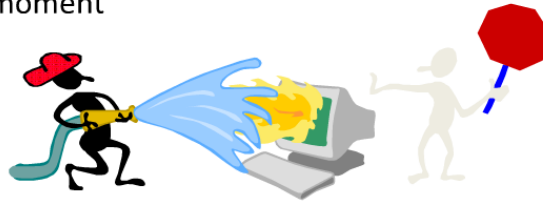 - Determine for the root cause
    Why? Why? Why? Why? Why?
  - Investigate Countermeasures
    Try several and see what works
  - Implement the best solution
    Keep on looking for a better way

Change the Process

- Developer didn't write the code correctly
  - Write Unit Tests
- Developer misunderstood what the customer wanted
  - Specify acceptance tests early
- Customer realized later that they mis-spoke the requirement
  - Specify acceptance tests early
- Customer spoke it properly, but realized they asked for the wrong thing once they saw what was delivered
  - Don't take a long time to show customers what you've done
- Customer spoke it properly, realized they got what they originally wanted, but they now have a better idea
  - Don't take a long time to show customers what you've done

Demming: Most of the time the process is at fault and not the people.

Change is Difficult

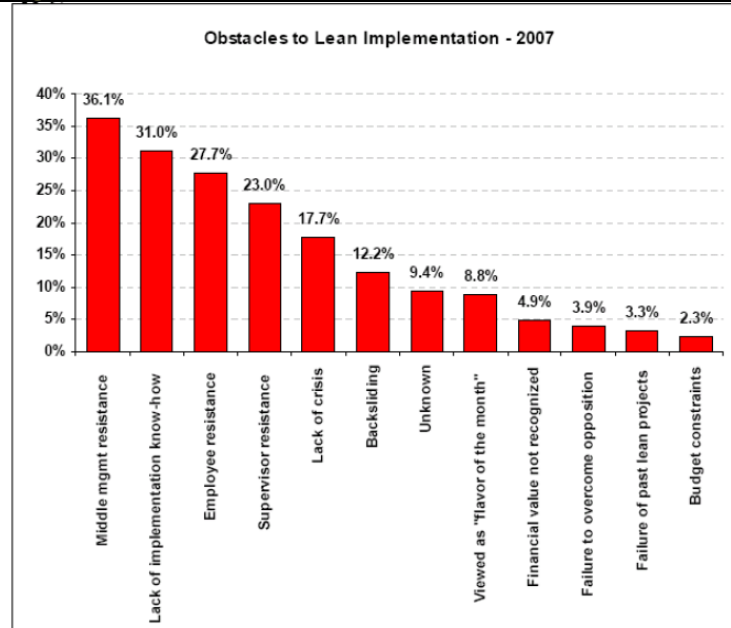"People only change their values in light of a miracle or near death experience"

LERCH
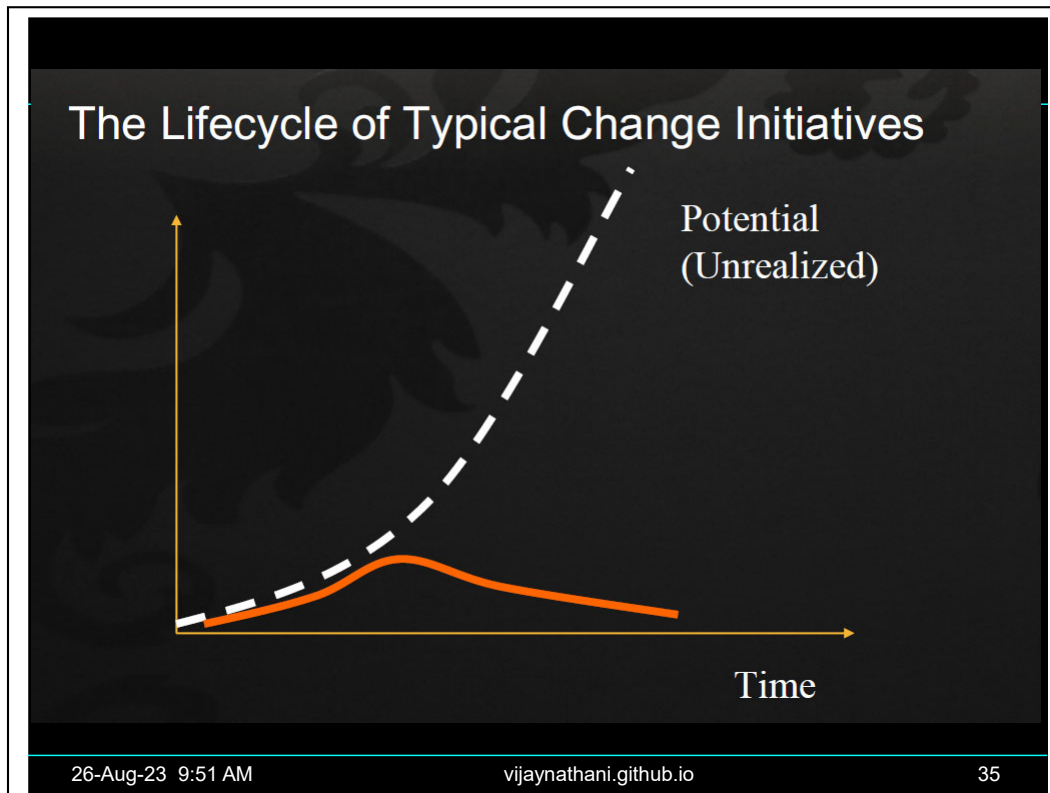
Y-Axis is benefit.

Companies have been adopting lean since 1990. Mostly non-software at start.

Majority of attempts to implement end in disappointing outcomes.

Organization culture is difficult to change

Obstacles to Lean Implementation - 2007

Obstacle	Percentage
Middle mgmt resistance	36.1%
Lack of implementation know-how	31.0%
Employee resistance	27.7%
Supervisor resistance	23.0%
Lack of crisis	17.7%
Backsliding	12.2%
Unknown	9.4%
Viewed as "flavor of the month"	8.8%
Financial value not recognized	4.9%
Failure to overcome opposition	3.9%
Failure of past lean projects	3.3%
Budget constraints	2.3%

Source: Lean Enterprise Institute, 2007 Survey

26-Aug-23 9:51 AM   vijaynathani.github.io   36

Reason for limited benefit.

Although everyone must be prepared for changes in behaviors and even in values, managers establish conditions that enable such change. If managers are not authentic in their convictions and sincere in their behavior there will be little trust and little safety for the reflection that leads to authentic change. – Peter Senge. The dance of change.