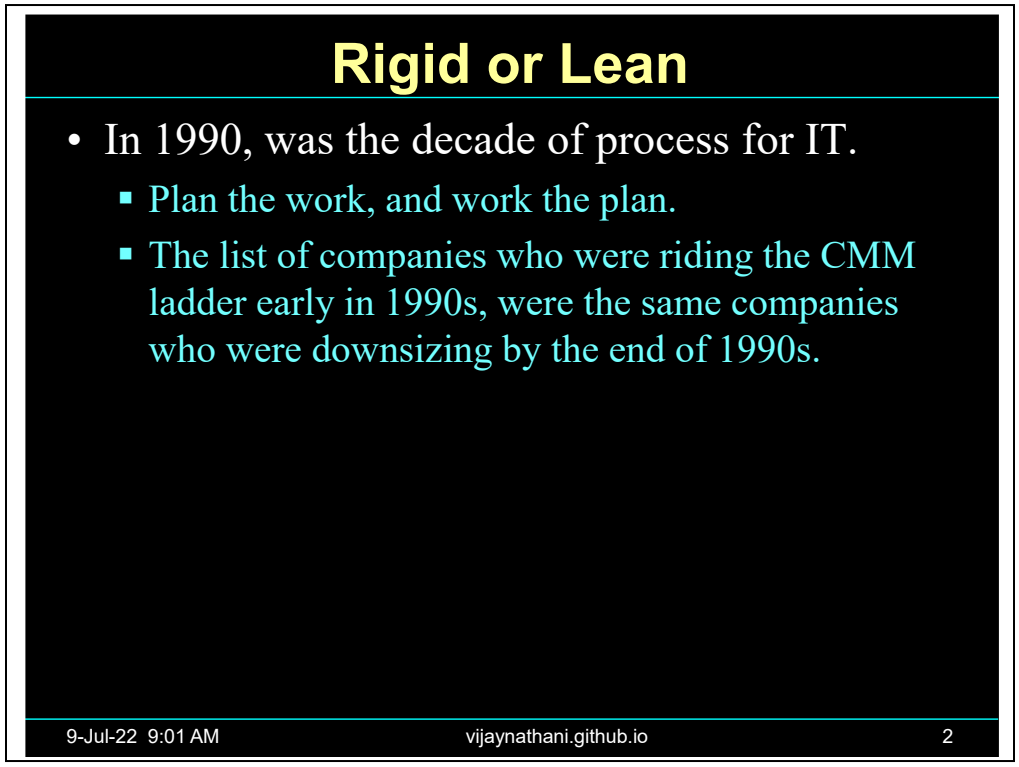
Lean Software Development

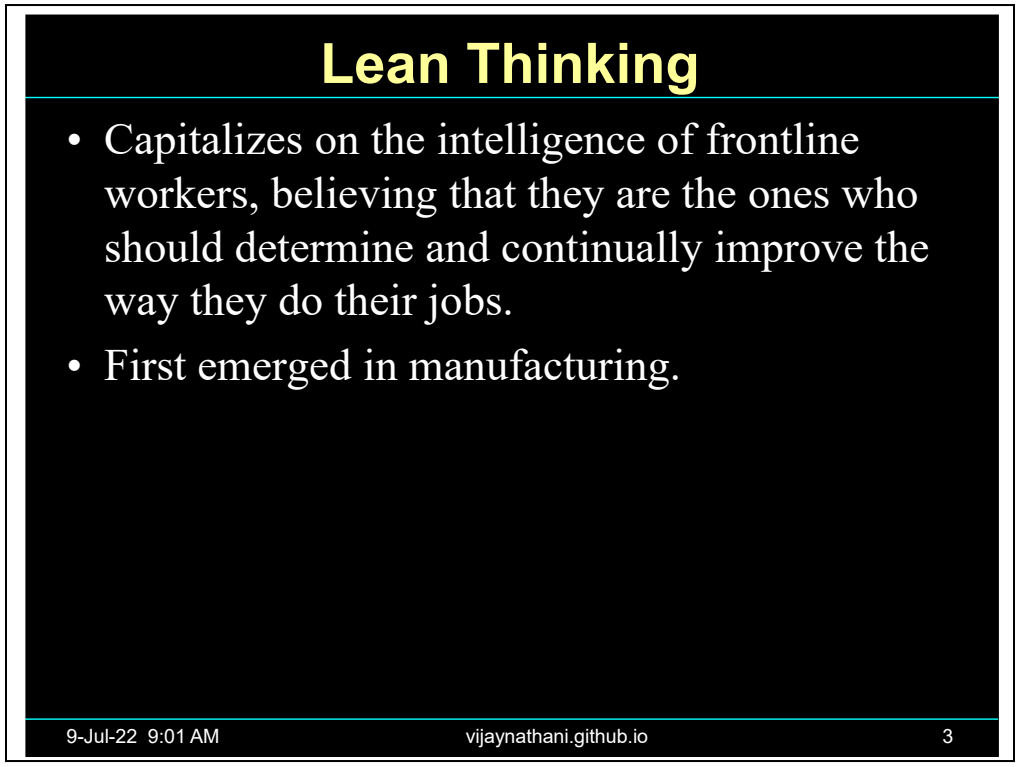
"Fat processes are out. Thin is in" –
Jim Highsmith



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\

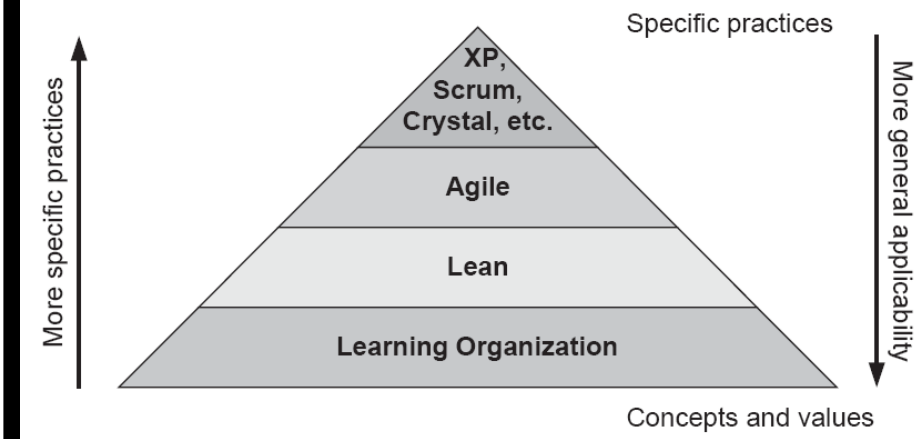
Process rigor was simply not the right recipe, when change is the only constant.



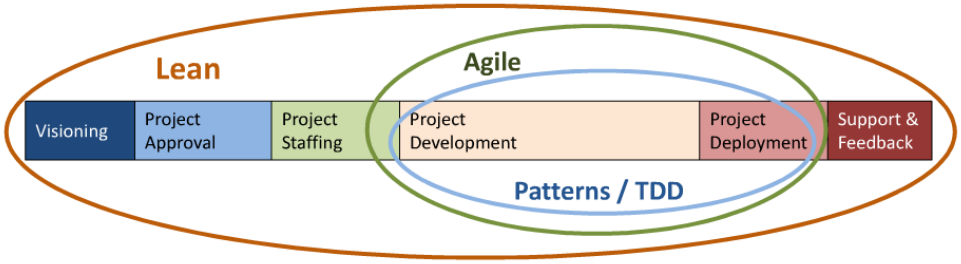
Mary and Tom Poppendieck transferred the principles from manufacturing to software development.

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

Lean Compared

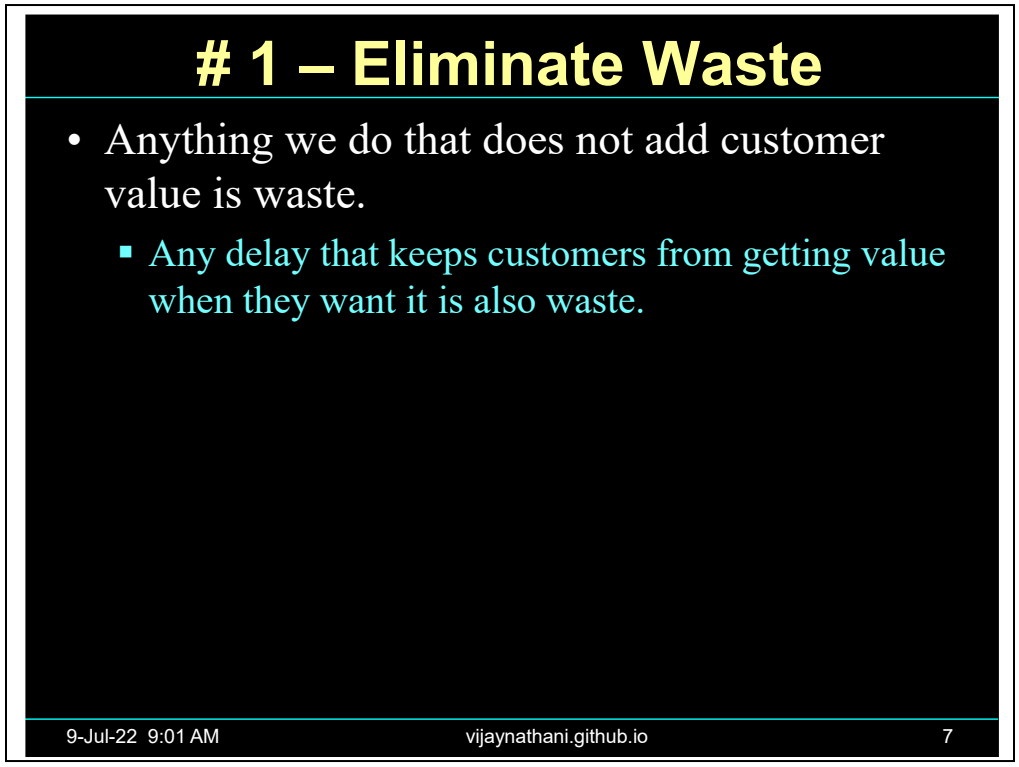


Use of Lean


Lean
Agile
Visioning
Project Approval
Project Staffing
Project Development
Project Deployment
Support & Feedback
Patterns / TDD

Seven Lean Principles

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



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."

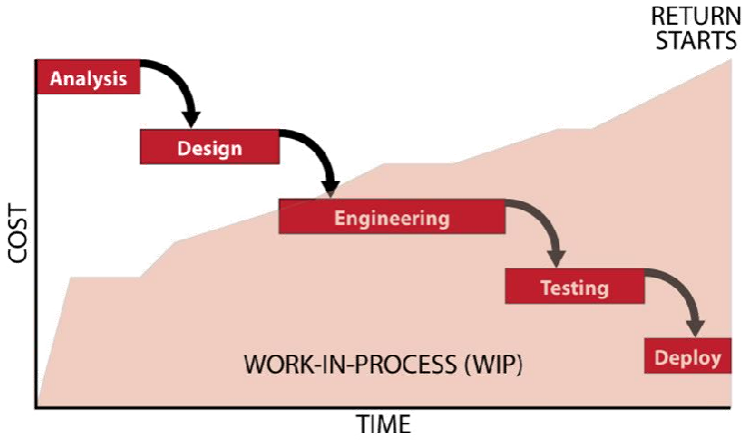
Jeff Sutherland, CTO PatientKeeper

Work In Progress

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

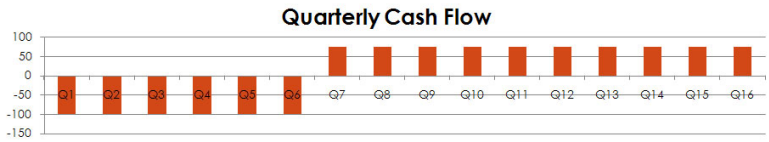
Capital Stuck

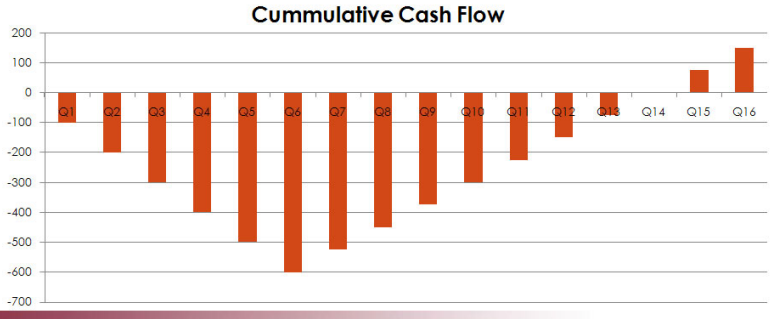
Traditional methods lockup capital



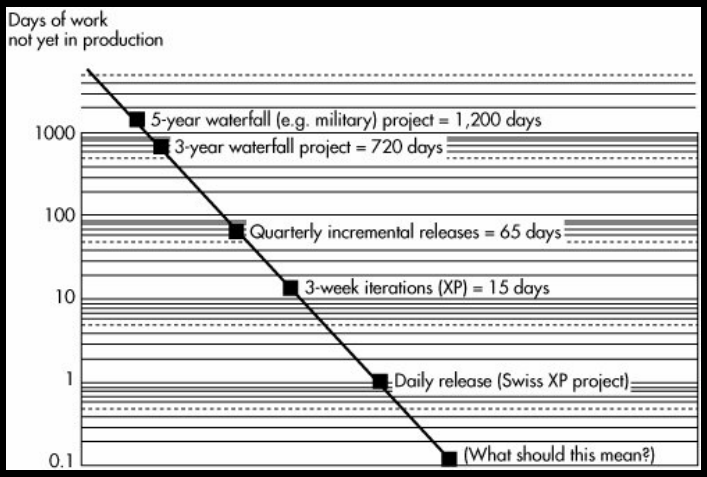
Predictive

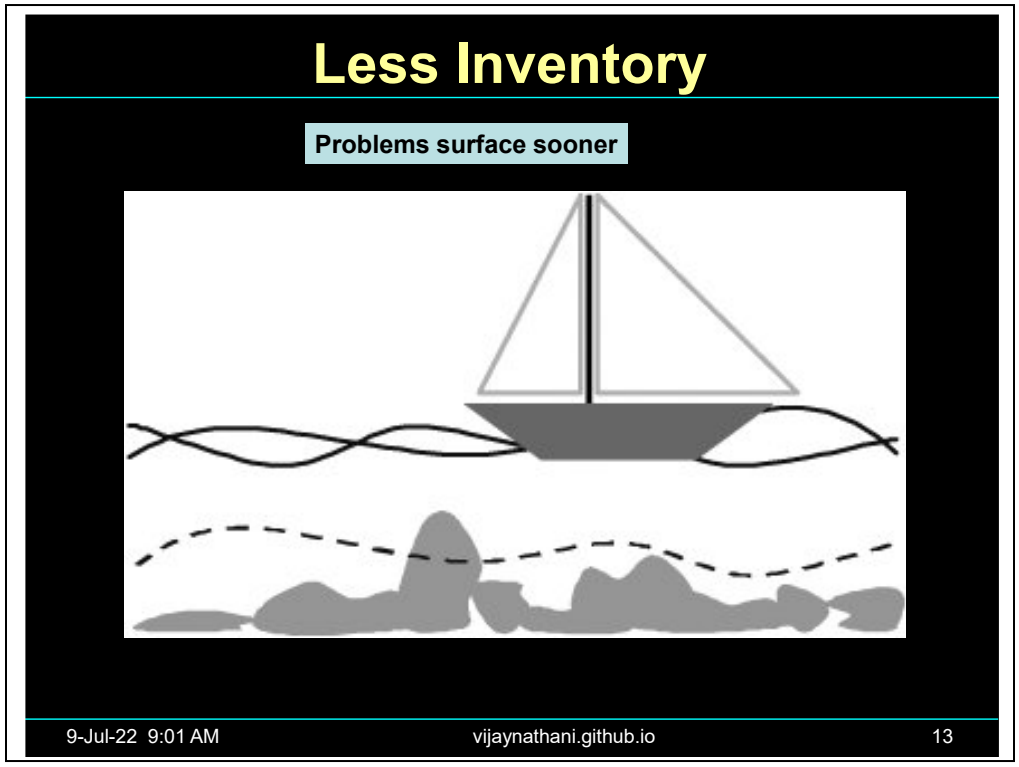
Typical IT project





Inventory


Days of work
not yet in production
1000
100
10
1
0.1
5-year waterfall (e.g. military) project = 1,200 days
3-year waterfall project = 720 days
Quarterly incremental releases = 65 days
3-week iterations (XP) = 15 days
Daily release (Swiss XP project)
(What should this mean?)



Do you have a tendency to work in batches?

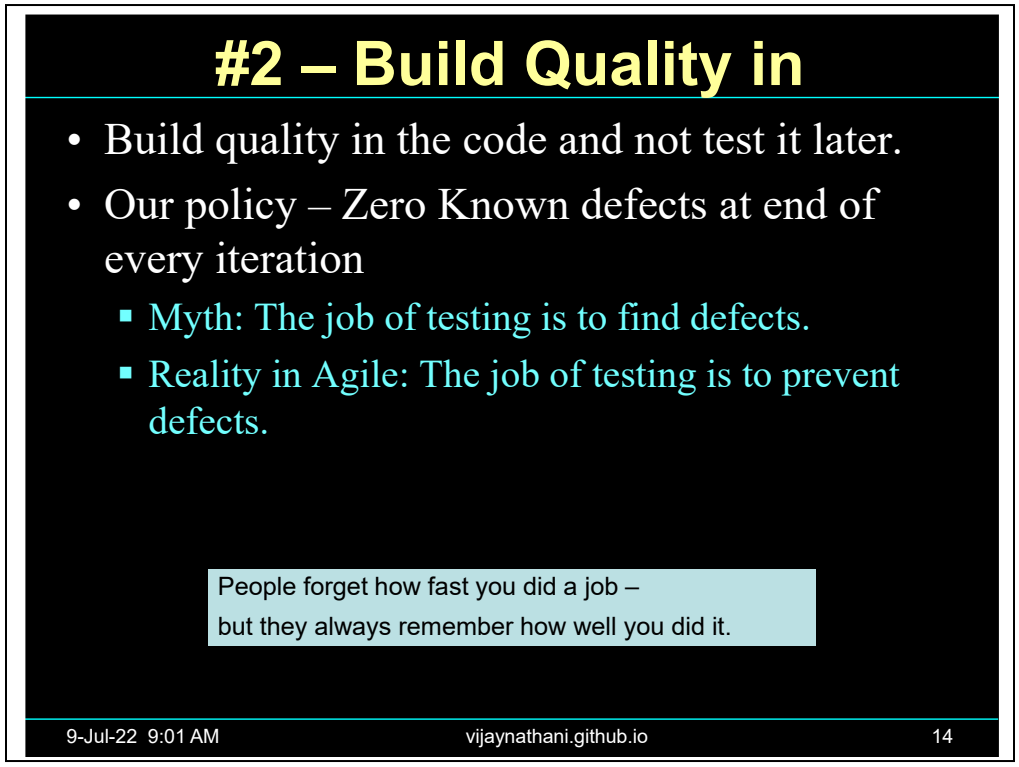
If you had to mail 100 letters, how would you go about folding the letters, stuffing the envelopes, adding address labels and stamps?

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.



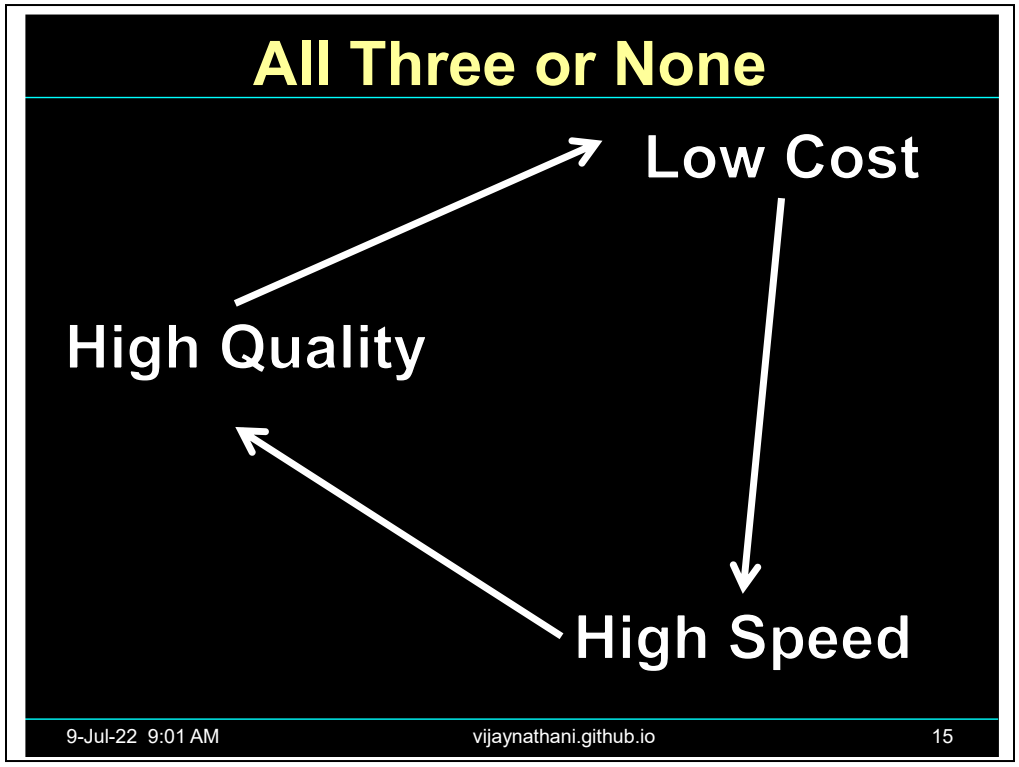
You don't focus on putting defects into a tracking system; you avoid creating defects in the first place.

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

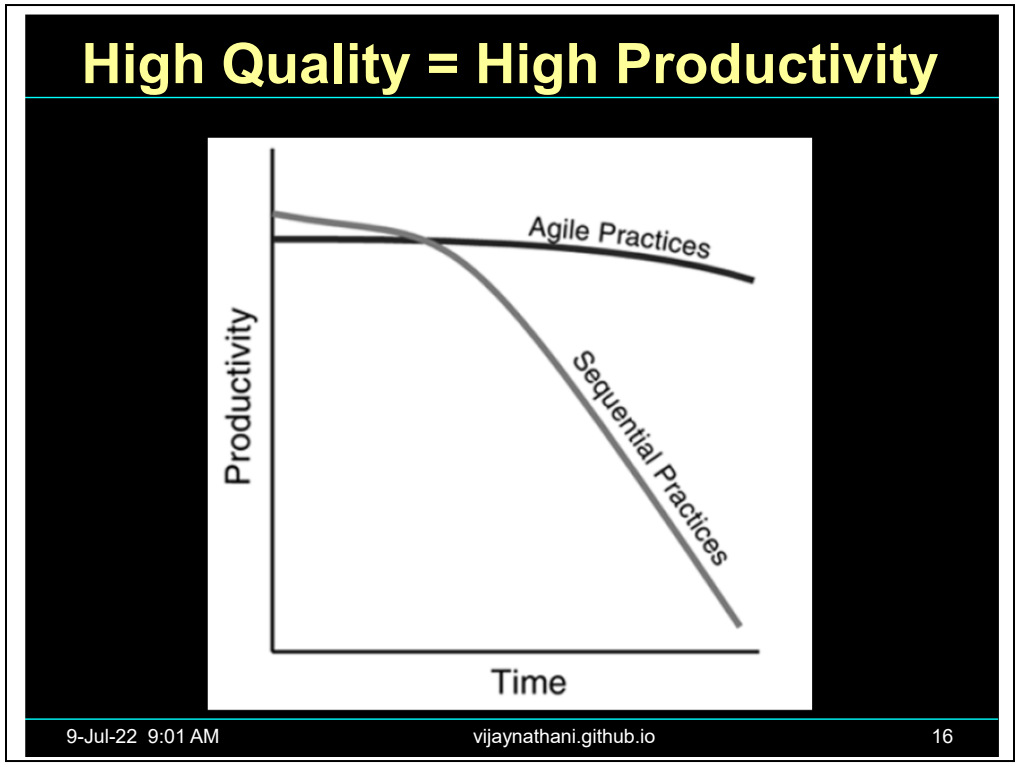
A company typically has to pay Indian developers about 30% of what it pays developers in USA. The catch is that it there is extra project management overhead. So it costs the company 70 cents for every dollar after off shoring.

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.



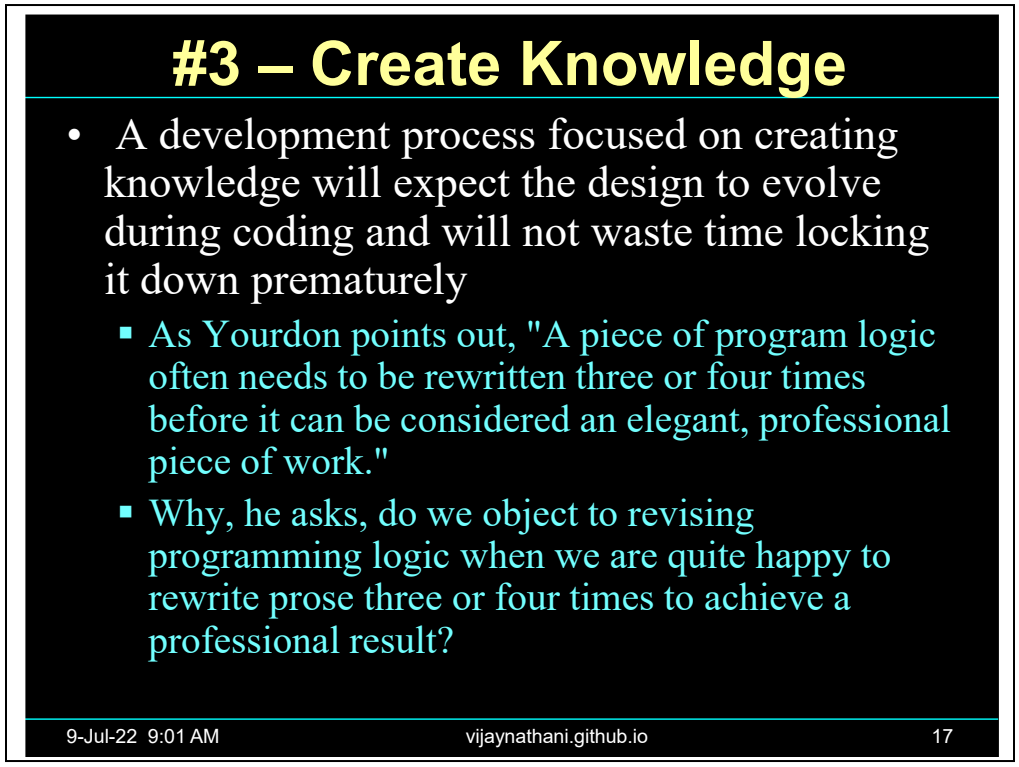
In the long run



You have an organization of intelligent people. Do these people make it their job to work around problems, or are problems considered a trigger to stop-the-line and find the root cause?

> Make a list of the Top 10 problems that occurred in your group in the last week.
>
> List after each problem the way it was resolved. Rank each problem on a scale of 0 to 5.
>
> > The rank of 5 means that you are confident that the cause of the problem has been identified and eliminated and it is unlikely to occur again. The rank of 0 means that there is no doubt the problem will crop up again.

What is your total score?



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

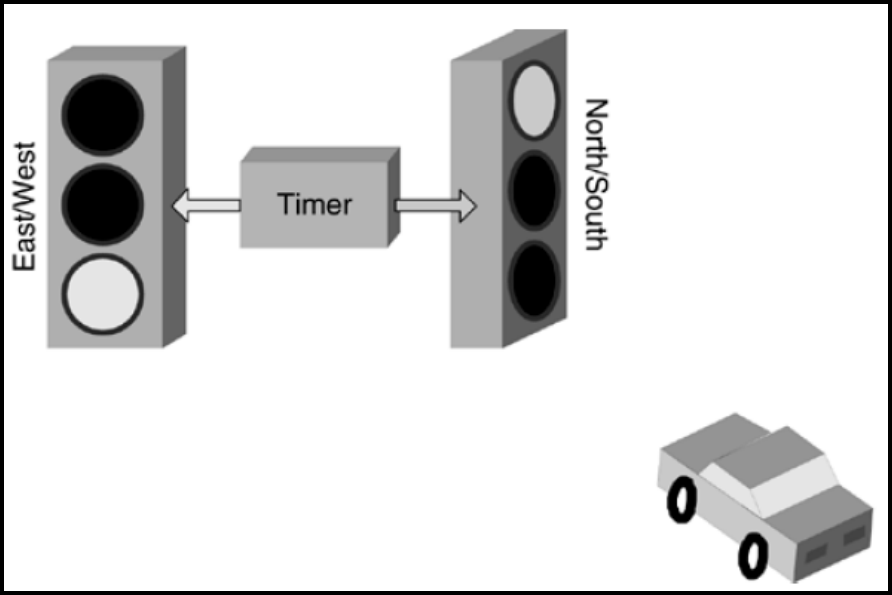
Frequent releases to customers

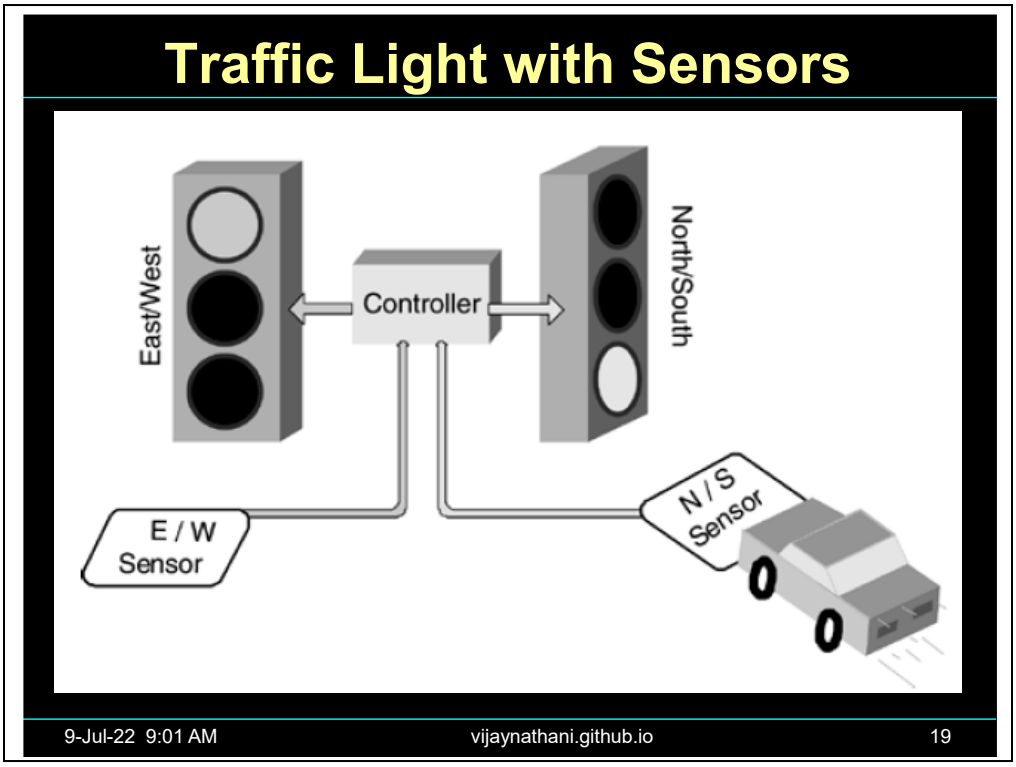
Daily builds and continuous integration

Continuous refactoring

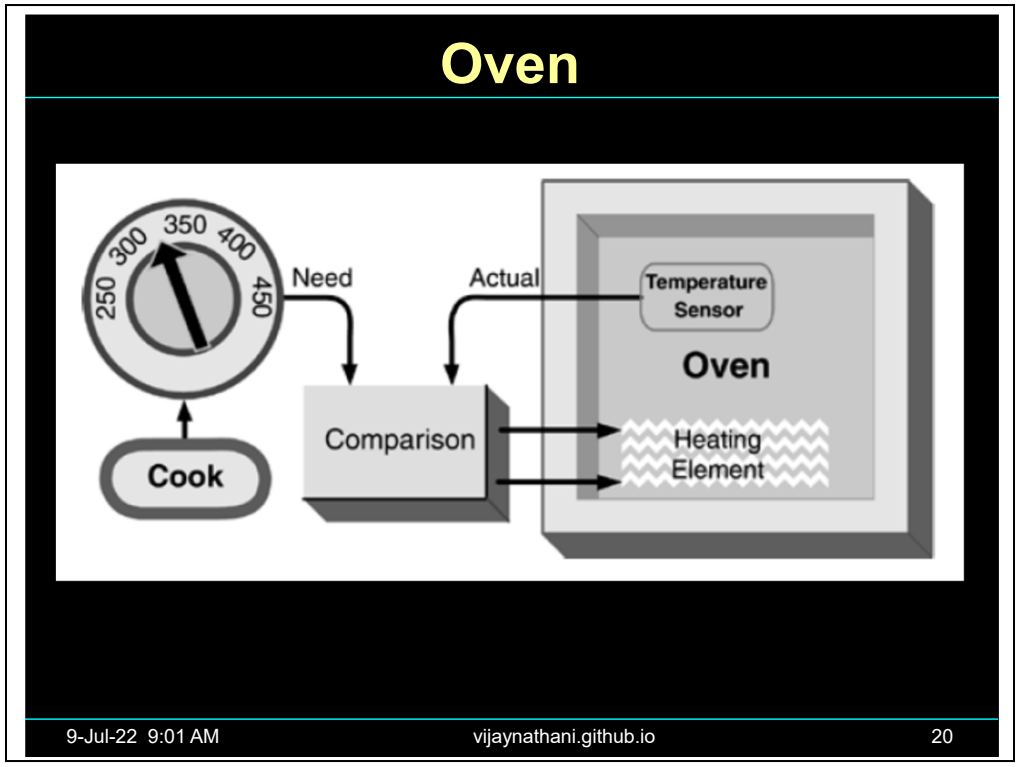
Pair Programming

Timed Traffic Light



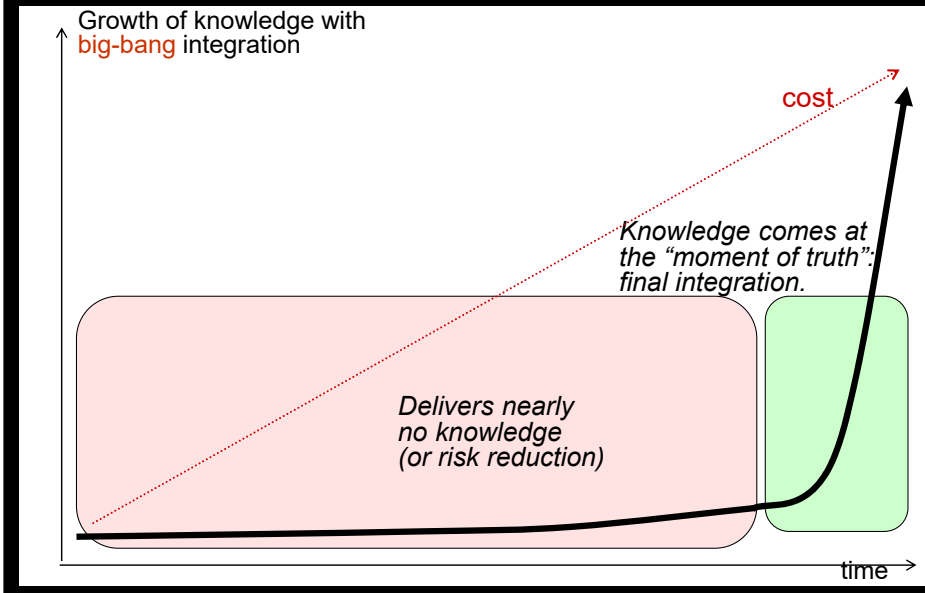


Better Feedback

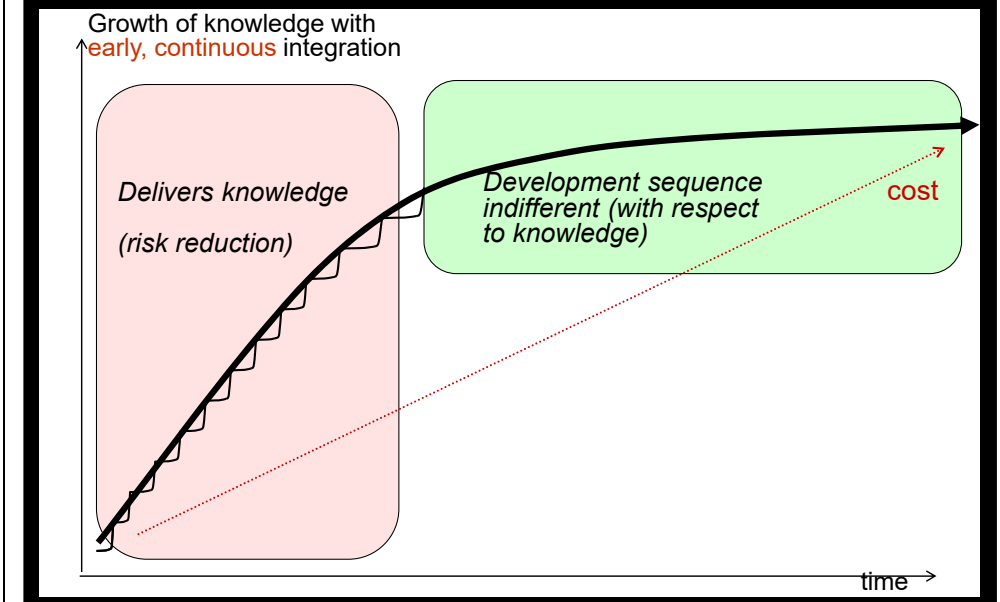


Continuously monitor the temperature and adjust

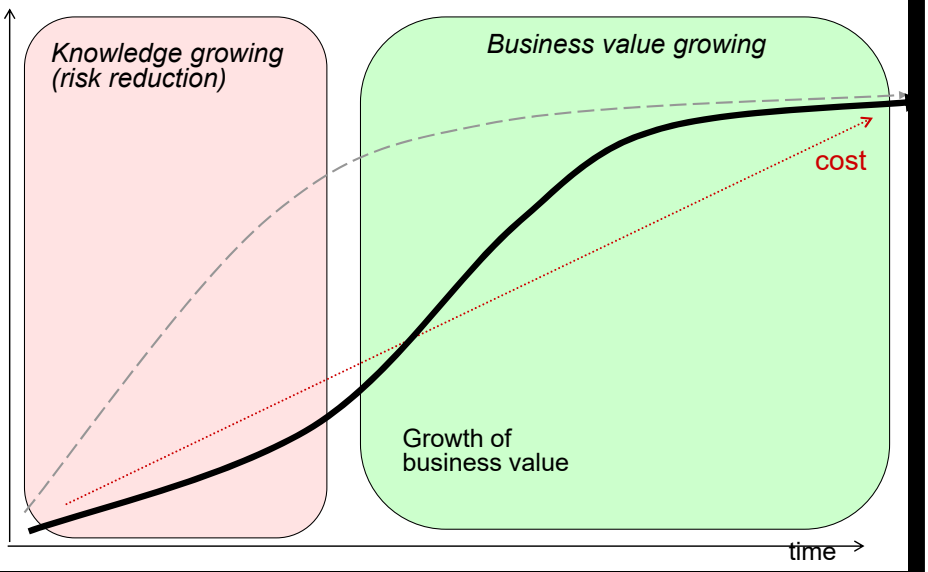
Waterfall is a late-learning Strategy



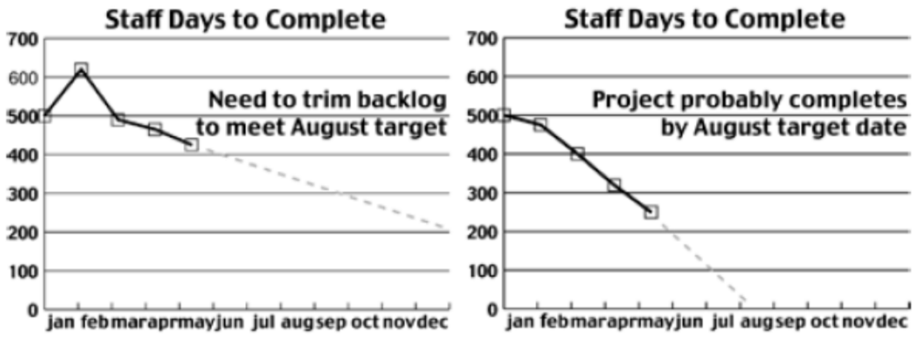
We can pay to *learn* early in the project



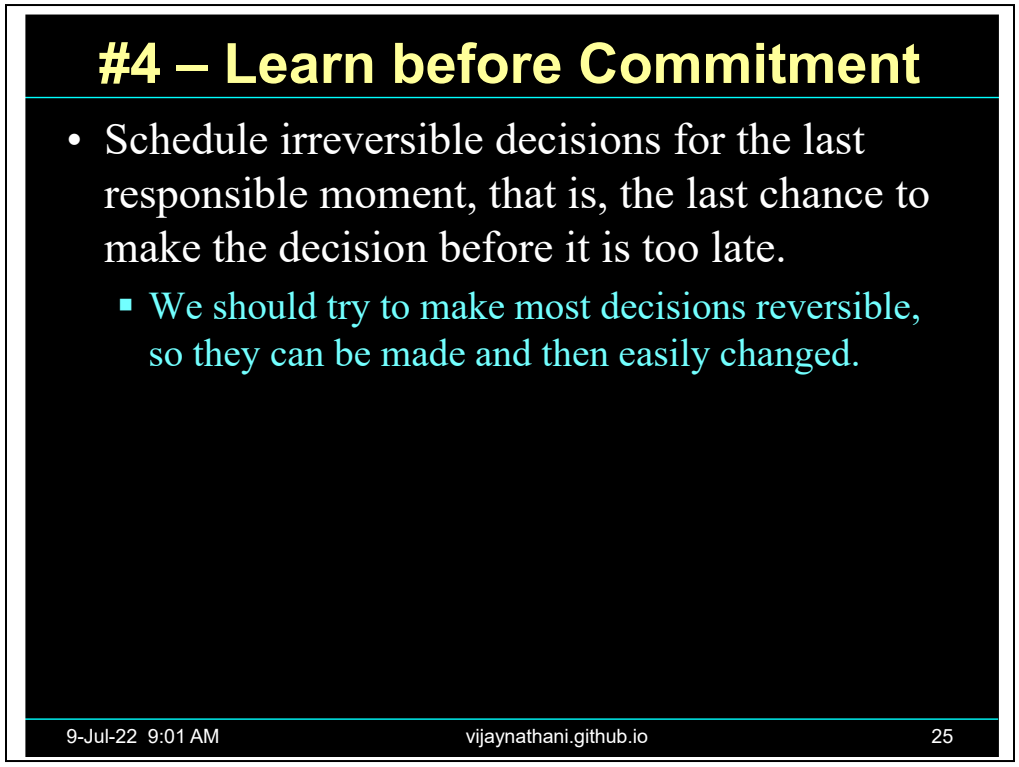
Develop for business value once risks are down



Product Burndown Chart



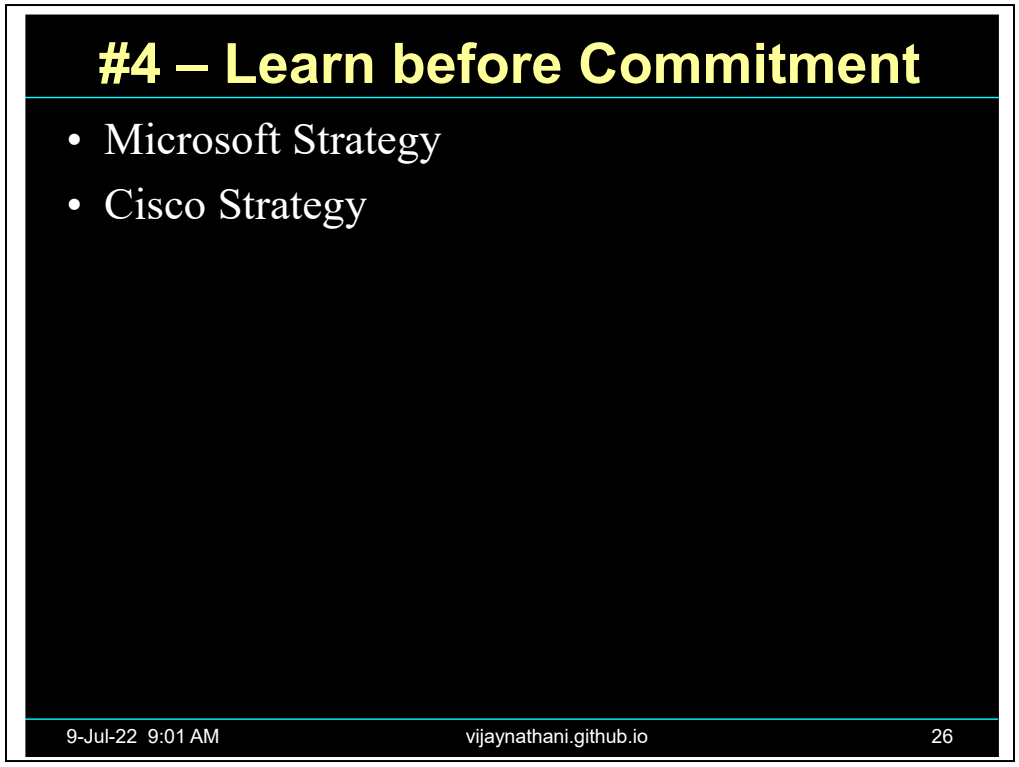
The project on the right will finish in Aug. The project on left will probably not finish, based on current scope



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.



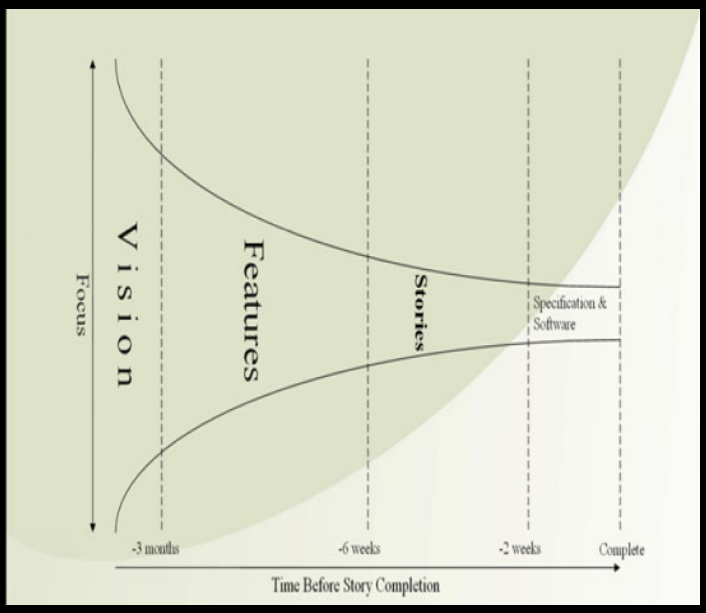
1988 Comdex trade show: All the big players were there with big booths: Apple was at the peak of its powers; IBM, Hewlett-Packard, DEC, Apollo, and Sun Microsystems were all touting their latest strategies. And then there was Microsoft, with a modest booth that "was more like a Middle Eastern bazaar than a trade-show booth." Microsoft showed its then current strength, DOS, along with an early version of Windows, OS/2 for IBM machines, a version of UNIX, and new releases of Word and Excel, which were a far distant second to Lotus and WordPerfect in the DOS environment but led the applications on Apple platforms. Beinhocker notes: "Along with confused customers, the press was also grumbling. Columnists claimed that Microsoft was adrift and Gates had no strategy."

In 1988 it was not at all clear which platform would win, and Gates did have a strategy—to cover all the bases. He wanted Windows to win but hedged his bets with DOS, OS/2, and even a version of UNIX. If Apple won the war, he would lose the operating system but win as the dominant application provider on that platform. In any case, he would develop expertise in both operating systems and applications. He played the options game and let the market emerge.

==========

In the 1990s Cisco Systems acquired companies with relevant technologies rather than maintaining a large research and development effort. This allowed Cisco to delay selecting technologies until both the market and the technology emerged, considerably reducing its risk. The cost of this options-based approach was the premium paid for the companies that had born the initial risk.

Agile



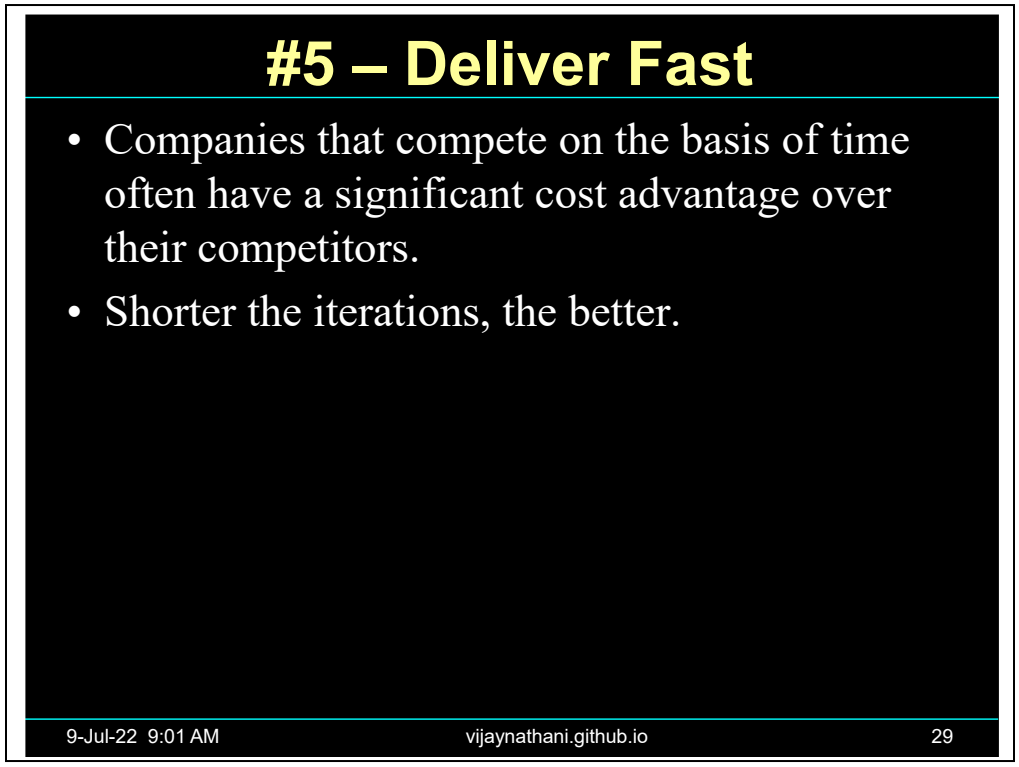
Plan at the last possible responsible moment not the last possible moment.

Agile

Reallocate funds to strong performers let's change the game

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?



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

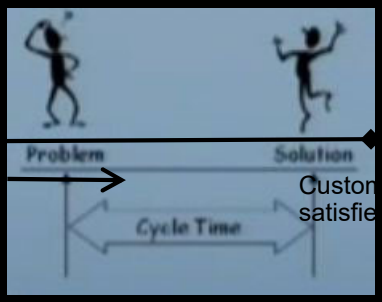
Repeatable and reliable speed is impossible without superb quality.

Myth: Haste makes Waste

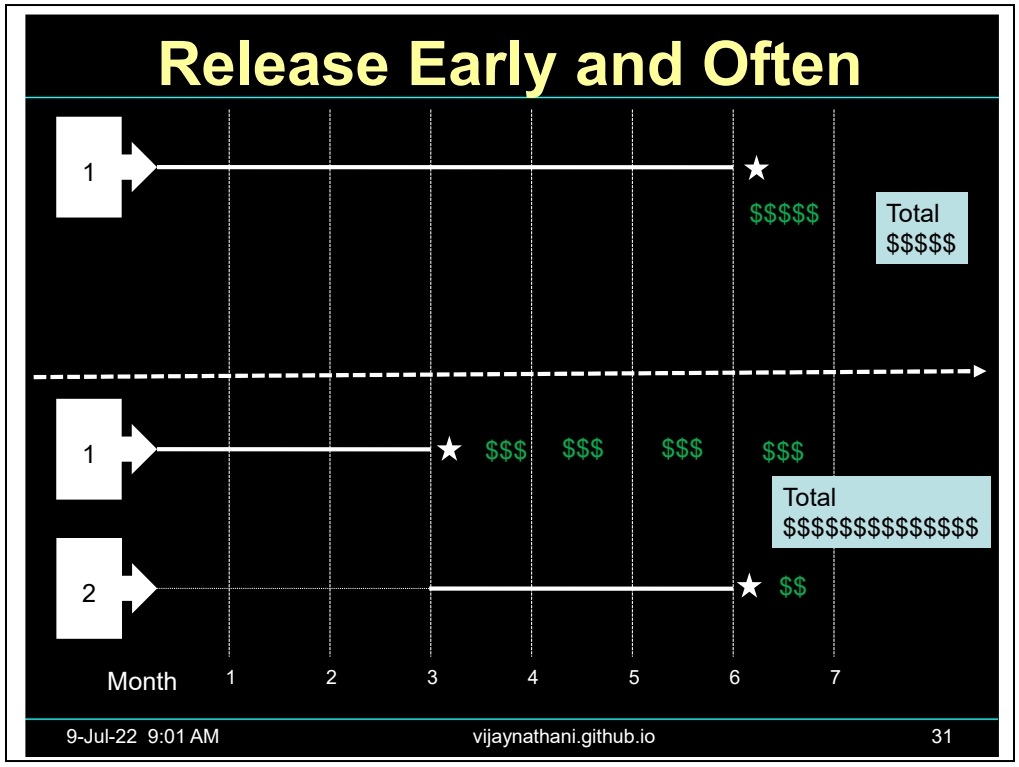
> Don't equate high speed with hacking. You can't sustain high speed unless you build quality in.

Cycle Time

- Average end to end process time
- For software: Time taken reliably and repeatedly to translate a customer's need to deployed software.
- Begins and ends with the customer

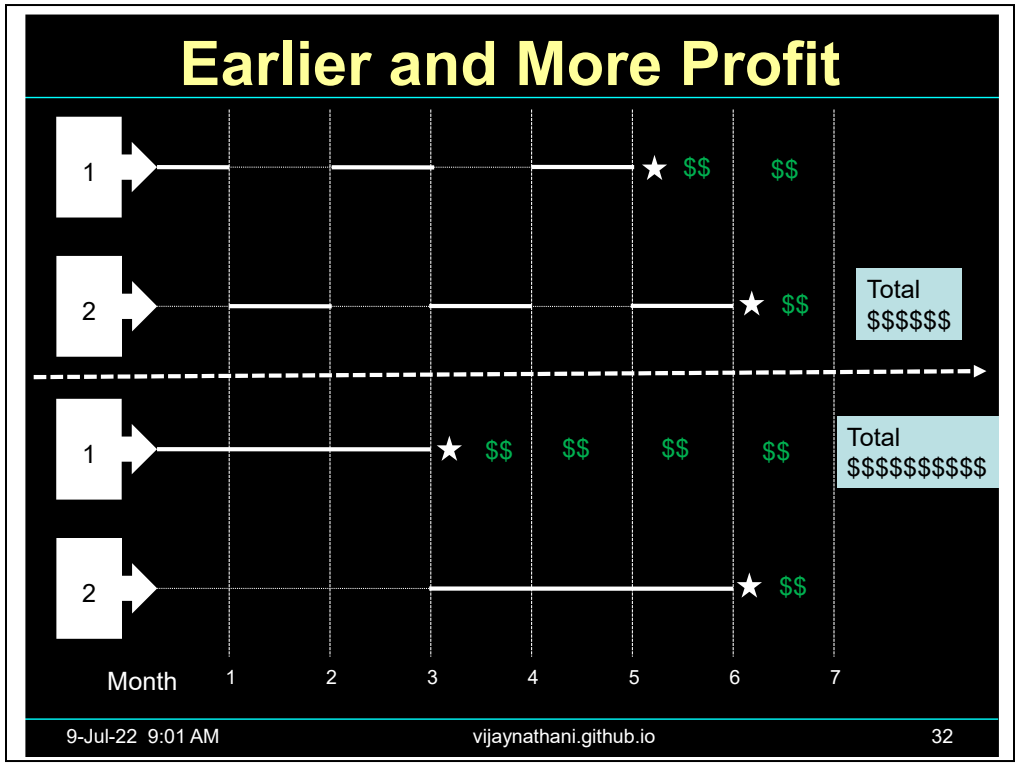


Little law Cycle time = Number of items in the process / average completion time per item



In first case, we release after 6 months.

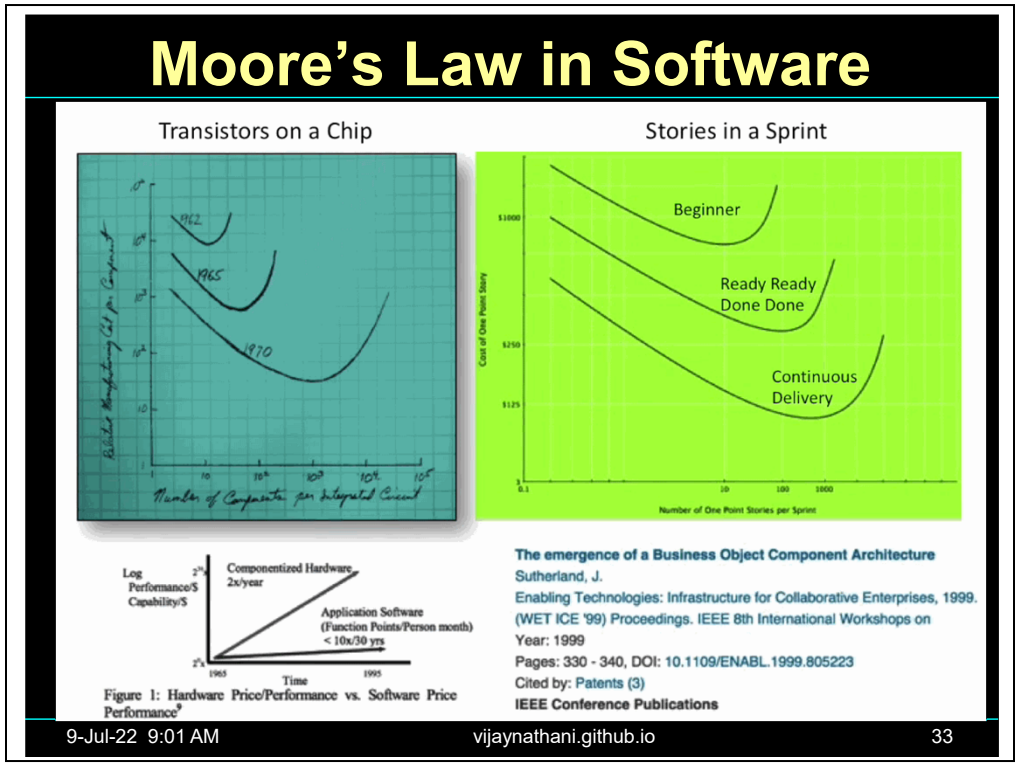
In Second case, we release the important features after 3 months and remaining features in 3 months.



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: 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.

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.

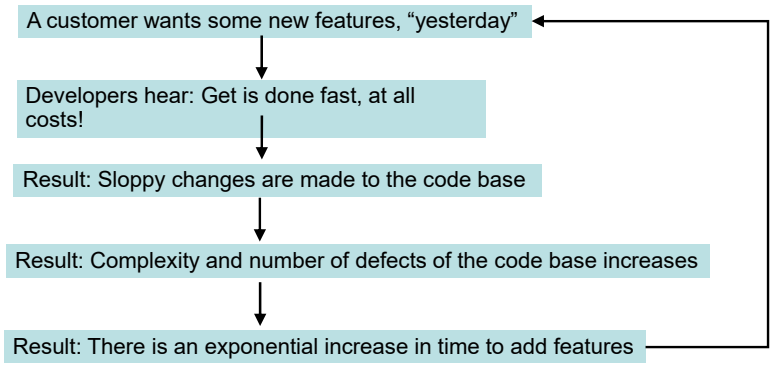
#6 – Respect for People

- 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.

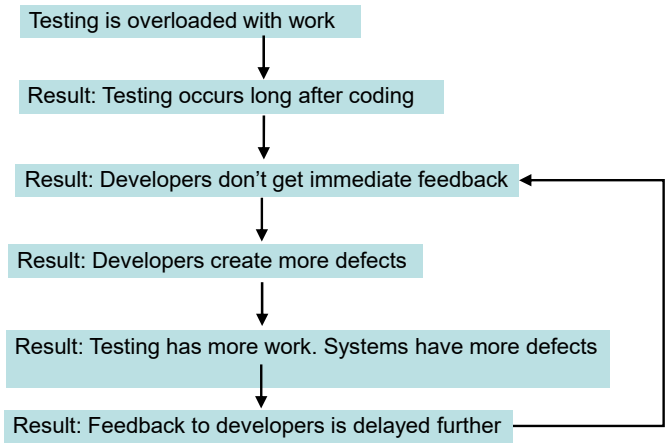
7 – Optimize the Whole

- Software development is legendary for its tendency to sub optimize
- Myth: Optimize by decomposition

Vicious Cycle



Vicious Cycle

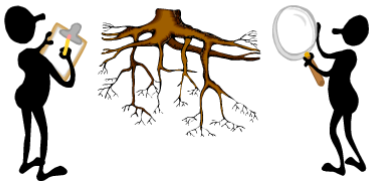


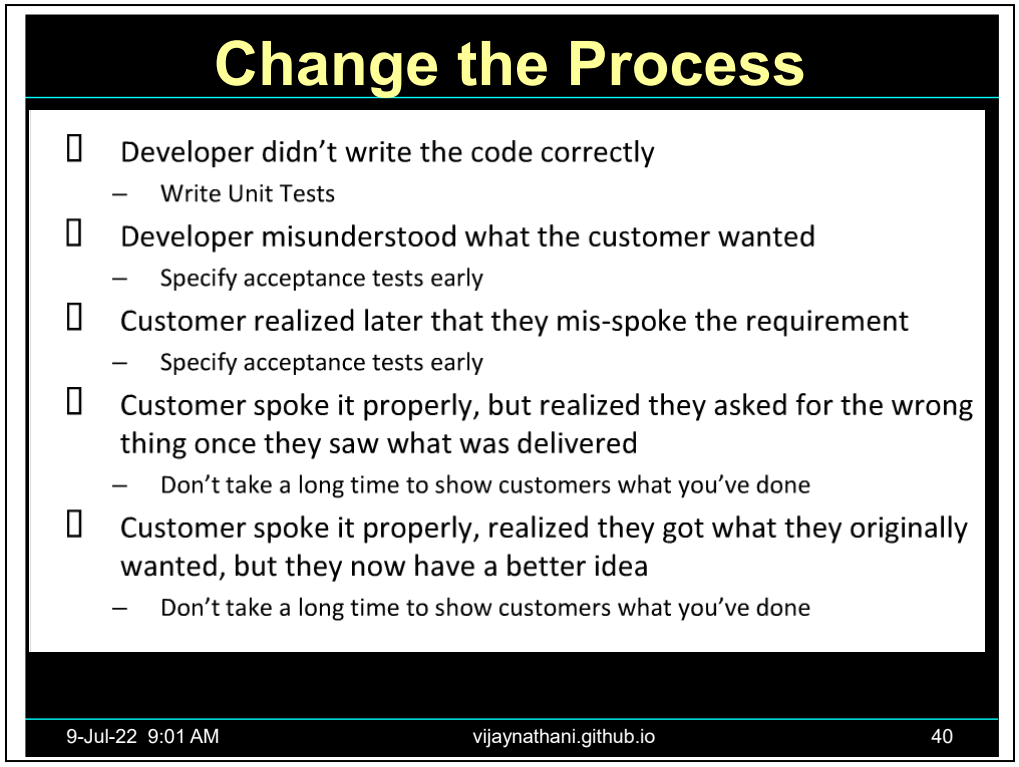
Stop The Line Mentality

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

Stop!

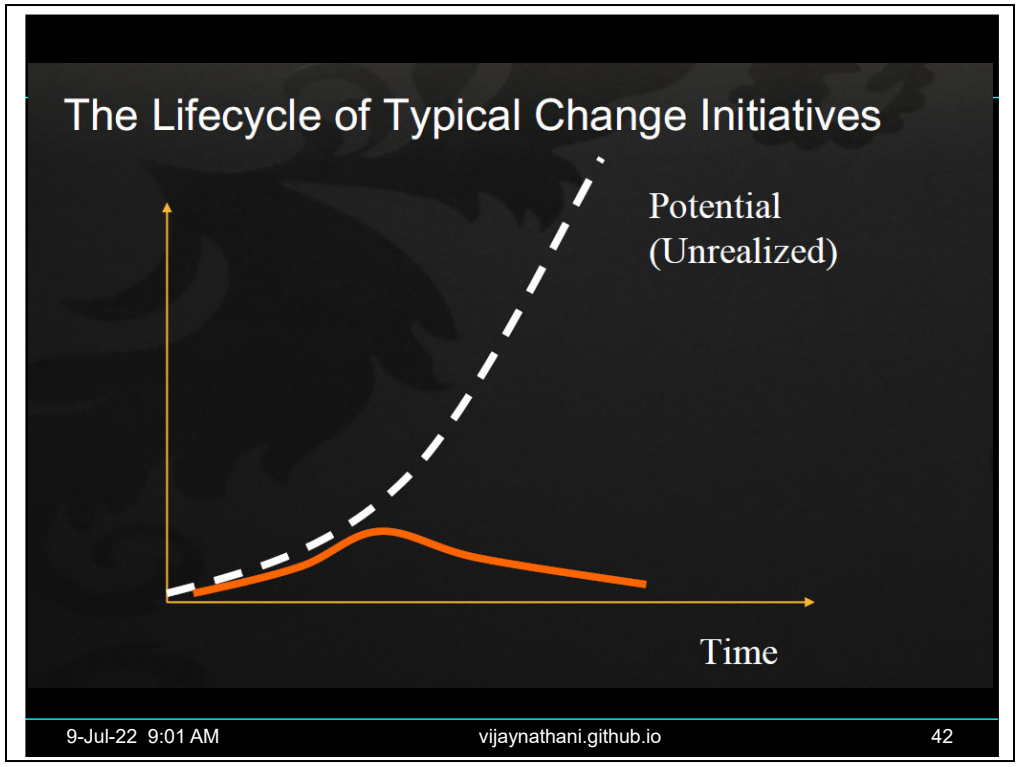
- Find and fix the problem
  - Determine for the root cause
    Why? Why? Why? Why? Why?
  - Investigate Countermeasures
    Try several and see what works
  - Implement the best solution
    Keep on looking for a better way





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

Change is Difficult

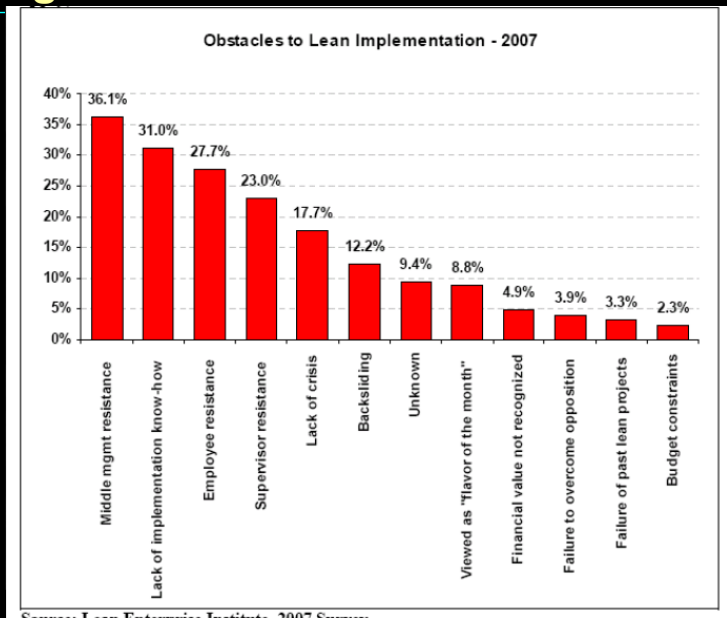


Y-Axis is benefit.

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



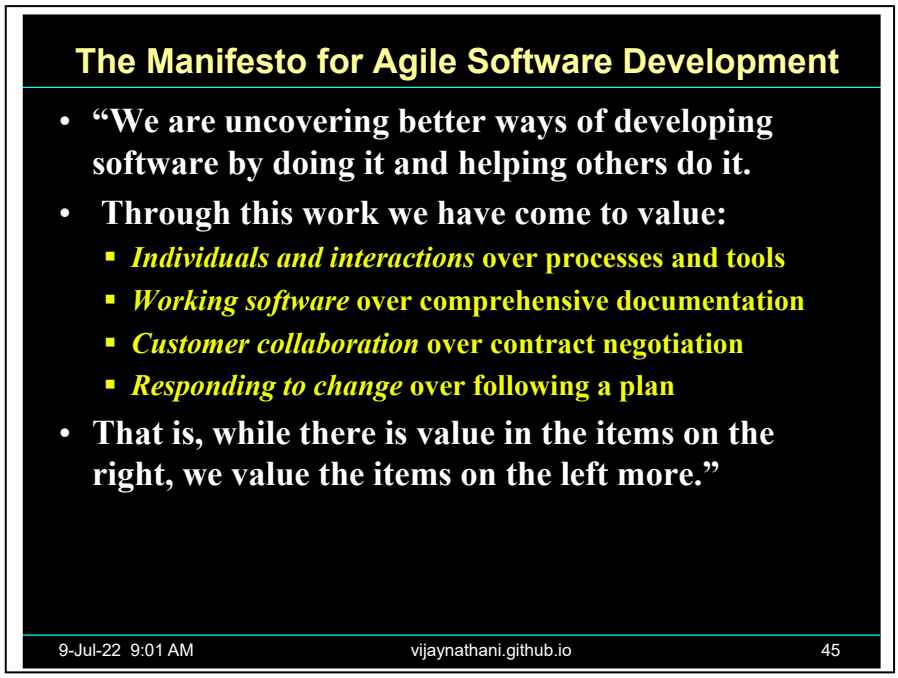
Reason for limited benefit.

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.

Agile Manifesto & Principles

A guy, on his first date, announces his wedding date, number of kids she will have, their birth time, …

This is a traditional project planner.



These days agile seems to be about

- improving productivity,

- reducing work in process,

- Increasing velocity in any way possible,

- holding teams accountable for finishing everything they say they will, and,

- doing just enough that an organization can call itself agile without really being agile.

Many of these are good things. But they are the candy manufacturer version of agile: sweeter and easier to sell.

But, ultimately, these are as unrelated to the original meaning of agile as sweets is to the original meaning of Diwali.

Principles behind the Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Working software is the primary measure of progress.
- Deliver working software frequently i.e. minimum length of sprint.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Principles (continued)

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The End

I cannot imagine any condition which would cause this ship to founder. Modern shipbuilding has gone beyond that.

– E. J. Smith, Captain of the Titanic ship.