

IN5140 – Smart processes and agile methods in software engineering

Lecture 8 September 2025:

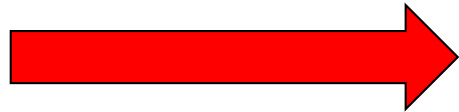
Lean and Agile Software Engineering



Dag Sjøberg

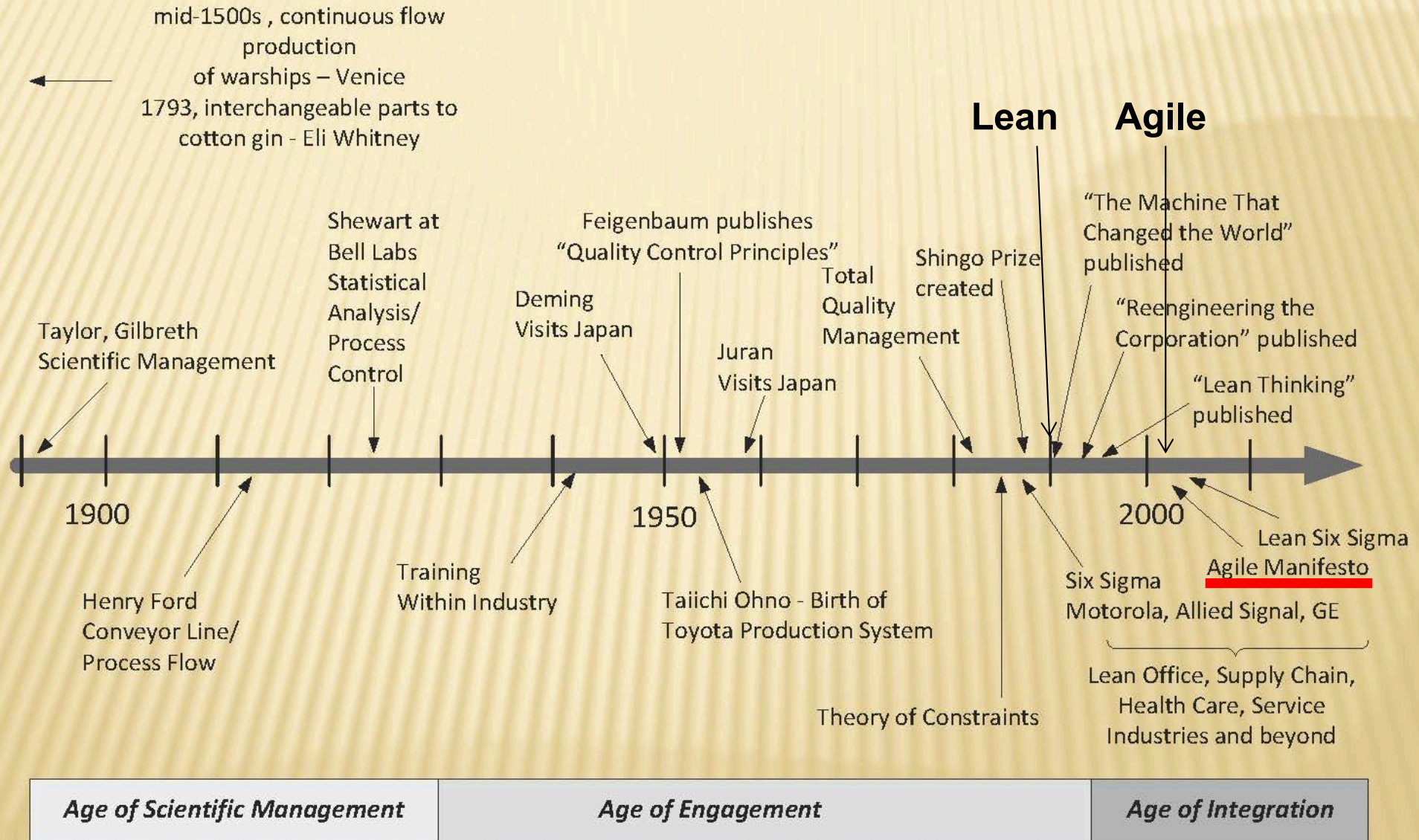
Clear relationship between

- Process improvement**
- Lean**
- Agile**

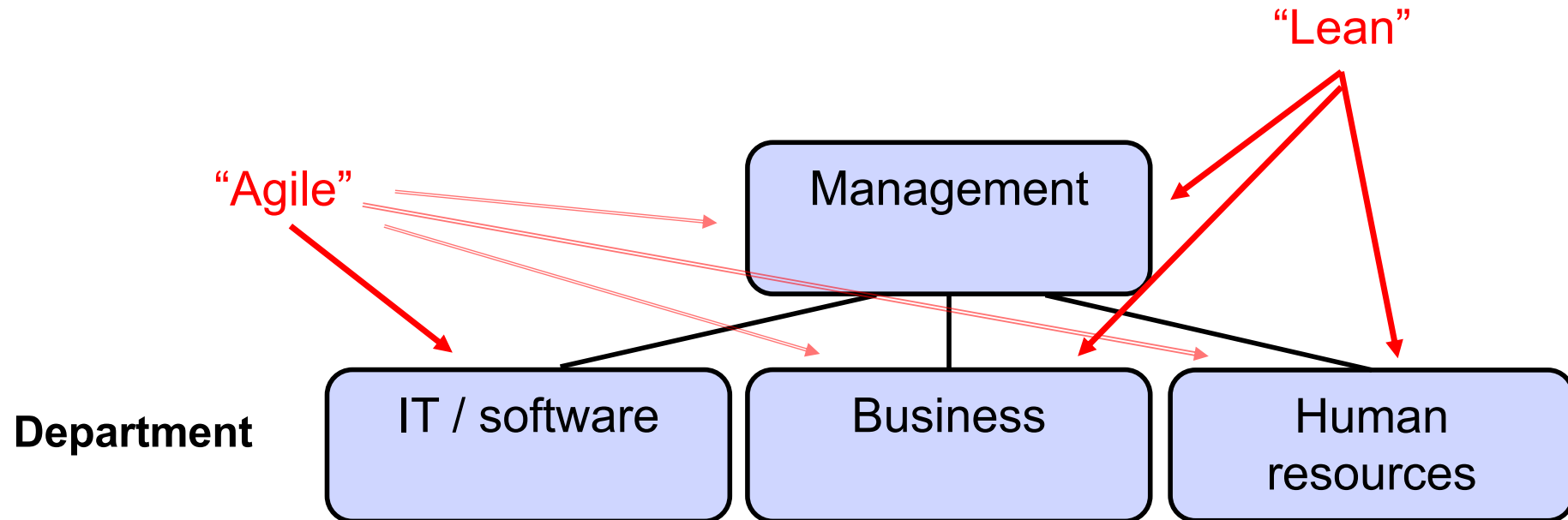


It's about working smarter

HISTORY OF PROCESS IMPROVEMENT



Lean versus Agile



- Some agile principles come from Lean (Lean predated agile), but Lean is not an “agile method” (as indicated by the structure of Ch. 9 in the book), but “rather a philosophy made of a set of general observations”, book p. 135
- Crystal is not important; Section 9.5 is not syllabus

Structure

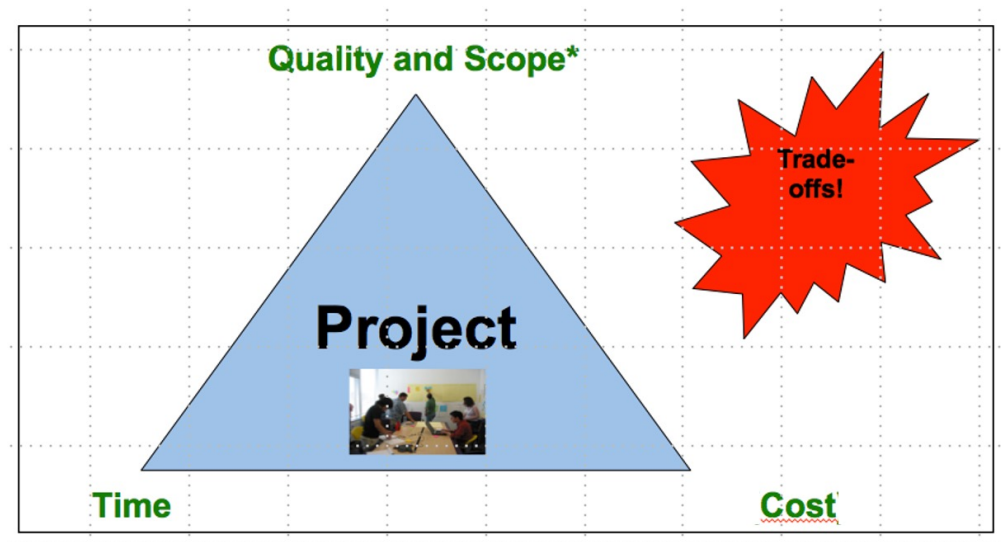
- Lean
- Comparing Lean and agile organizational principles



Purpose of Lean

- All parts of the organization should optimize their processes to achieve a common goal: **To create value for the customer**
- Value = what the customer wants and is willing to pay for

Customer value



Project/Magic/Iron triangle

Systems thinking

Systems thinking is the process of understanding how things, regarded as systems, influence one another within a whole. In nature, systems thinking examples include ecosystems in which various elements such as air, water, movement, plants, and animals work together to survive or perish. In organizations, systems consist of people, structures, and processes that work together to make an organization "healthy" or "unhealthy".

[From Wikipedia]

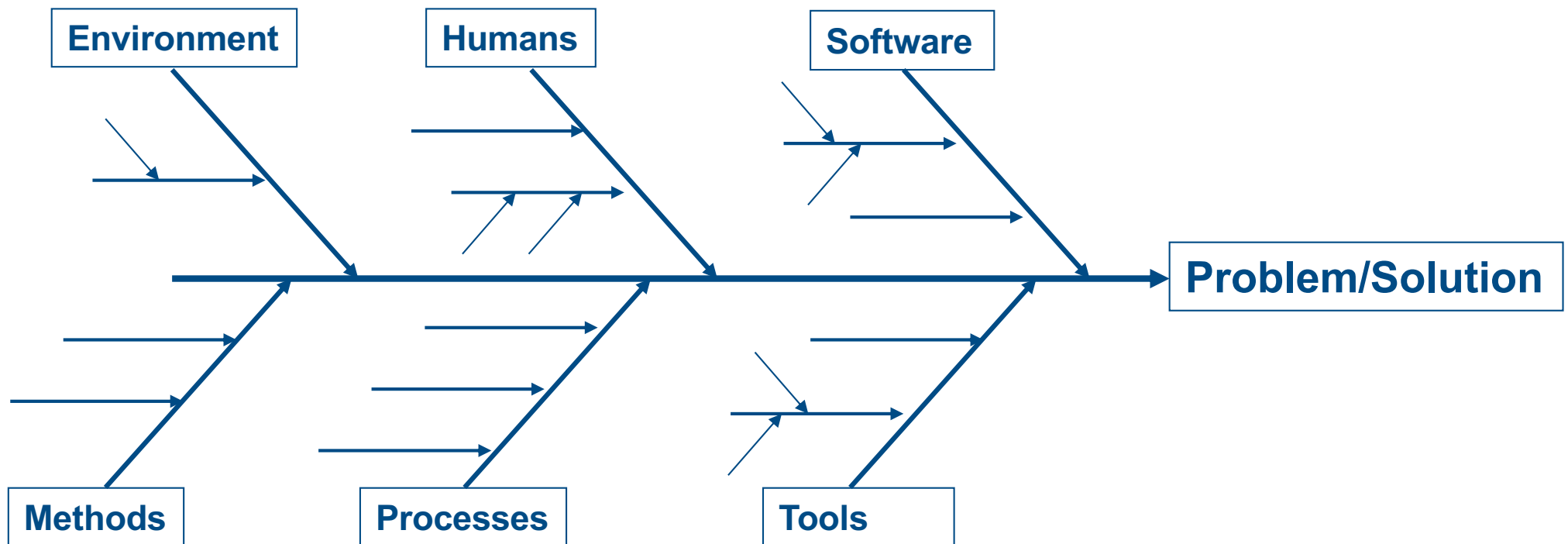
Lean production

- “The Japanese school”, primarily Toyota
 - If failure discovered, stop the assembly line and find the reason for the failure, instead of only collecting and fixing the failures (**root cause analysis** and **root cause fix**)
 - “**Just-in-time**” (JIT) principle: don’t produce before somebody demands it
 - Tempo in production determined by **pull** from customer or next element in the production chain rather than push internally to produce as much as possible
 - **Removal of temporary storage**
 - **Component-based production** (same chassis, bumper, etc. on different car models). Production geared towards the customer, components can be outsourced to third-party vendors
 - **Continuous learning** and improvement (Kaizen)
- Result: few failures and fast production
- The most productive factories spent least resources on management and administration (“lean management”)



Technique for root cause analysis

- Ishikawa diagram* indicates relationship between (un)wanted effects and their possible causes/preconditions



*After the Japanese professor Kaoru Ishikawa – also called "fish bone diagram"

Lean stems from Japanese car industry, but...

- Now widespread in all kinds of knowledge-intensive industry and administration
 - Large companies
 - Hospitals
 - Governments
 - Universities
 - ...



Lean has become a generic name for all kinds of process improvement

Delta på Lean Forums årskonferanse 11.-12. november

Jobb smartere – kultur, lederskap og menneskene



Akinori Hyodo

Akinori Hyodo er en av verdens mest kjente innen Toyota Production System - med over 40 års karriere hos Toyota Japan. Vi møter Akinori i både plenum og parallellsesjoner under årskonferansen.

[Les mer](#)

Effektivisering handler ofte om investering i ny teknologi og er ofte det som får mest oppmerksomhet. Det å ta i bruk ny teknologi er viktig, men å jobbe smartere handler om utvikling av folk, bygge kompetanse, for mer effektivt samspill menneske/maskin.

På årskonferansen ser vi nærmere på hvordan du kan bygge en sterk kultur, utvikle lederskap som løfter, og skape medarbeidere som kan, vil og tør i en verden i stadig raskere endring.

Structure

- Lean
- Comparing Lean and agile organizational principles



Agile Manifesto – 12 principles*

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals.
Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile principles

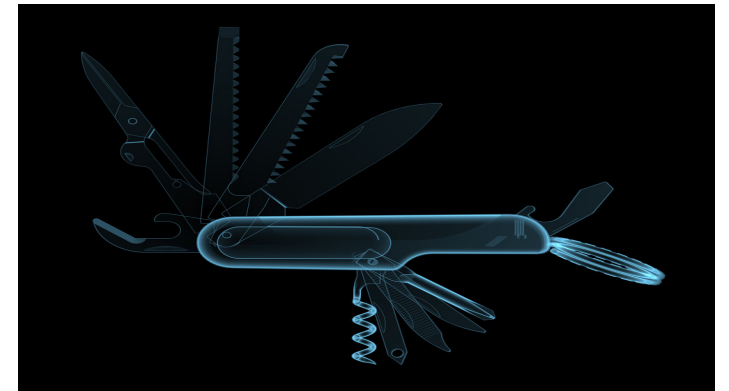
- Divided by Meyer, book Ch. 4, into:
 - *Organizational principles*
 1. Put the customer at the centre
 2. Let the team self-organize
 3. Work at a sustainable pace
 4. Develop minimal software
 5. Accept change
 6. Continuous learning (not included in the book)
 - *Technical principles*
 1. Develop iteratively
 2. Treat tests as a key resource
 3. Do not start any new development until all tests pass
 4. Test first
 5. Express requirements through scenarios
- Compared with Lean principles by Dag S.

1. Put the customer at the centre

Agile manifesto principle 1: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

Agile manifesto principle 4: Business people and developers must work together daily throughout the project

Lean: All parts of the organization should optimize their processes to achieve a common goal: To create *value* for the customer



Who is the customer or stakeholder?

One who

- pays for the system,
- uses the system or
- gets value from the system
 - Consider the whole value chain, including customers of the customer
- may be internal or external



Expectations

- **Owners:** “not too expensive”?
- **Users:** “prioritize everything” or are they more realistic?
- **Management:** “consider overall goals, please no surprises”
- **Developers:** “top technical quality, a fancy system”?
- **Maintainers:** “few bugs, understandable and well documented code”?
- **Other stakeholders** (law makers, other systems)

How to prioritize expectations among stakeholders?

- “Money rules”
- Difficult to ignore the management



How to communicate with the customer?

- Include Product Owner in the team
 - How representative for the users is the product owner?
 - See lecture 1 September
- BizDevOps teams
 - Business, development and operation staff work closely together, see later lectures

Waste in Lean

Waste is everything that requires resources and does not give value to the customer

- time
- work effort
- space
- equipment
- money



Examples of waste

[Larman & Vodde 2012]

Non-Value-Adding Action	Example or Comment
1. Overproduction of solutions or features, or of elements ahead of the next step; duplication	• features or services the customer doesn't really want • large engineering documents, more detailed designs than can be quickly implemented • duplication of data
2. Waiting, delay	• ...for clarification, documents, approval, components, other groups to finish something
3. Handoff, conveyance, moving	• giving a specification from an analyst to an engineer • giving a component to another group for testing
4. Extra processing (includes extra <i>processes</i>), relearning, reinvention	• forced conformance to centralized process checklists of 'quality' tasks • recreating something made
5. Partially done work, work in progress (WIP) or design in progress (DIP)	• designs documented but not built • things built but not integrated or tested
6. Task switching, motion between tasks; interrupt-based multitasking	• interruption • multitasking on 3 projects • partial allocation of a person to many projects
7. Defects, testing and correction after creation of the product	• testing and correction at-the-end to find and remove defects is not a value action; it may be a <i>temporarily necessary</i> waste
8. Under-realizing people's potential and varied skill, insight, ideas, suggestions	• people only working to single-speciality job title, or ...? • do people have the chance to change what they see is wasteful?
9. Knowledge and information scatter or loss	• information spread across many separate documents • communication barriers such as walls between people, or people in multiple locations
10. Wishful thinking (for example, that plans, estimates, and specifications are 'correct')	• "The estimate cannot increase; the effort estimate is what we want it to be, not what it is now proposed." • "We're behind schedule, but we'll make it up later."

Waste versus Value

- **Value Added Work**
 - What the customer is willing to pay for
- **Waste = Non-value Added Work**
 - What the customer is NOT willing to pay for
- **Necessary but Non-value Added Work**
 - No value but is required by existing processes, for example, infrastructure
 - What about other investments? see, book p. 13:
“generalizing code for ease of extension and reuse, and developing tools to automate repetitive processes.”

Flow

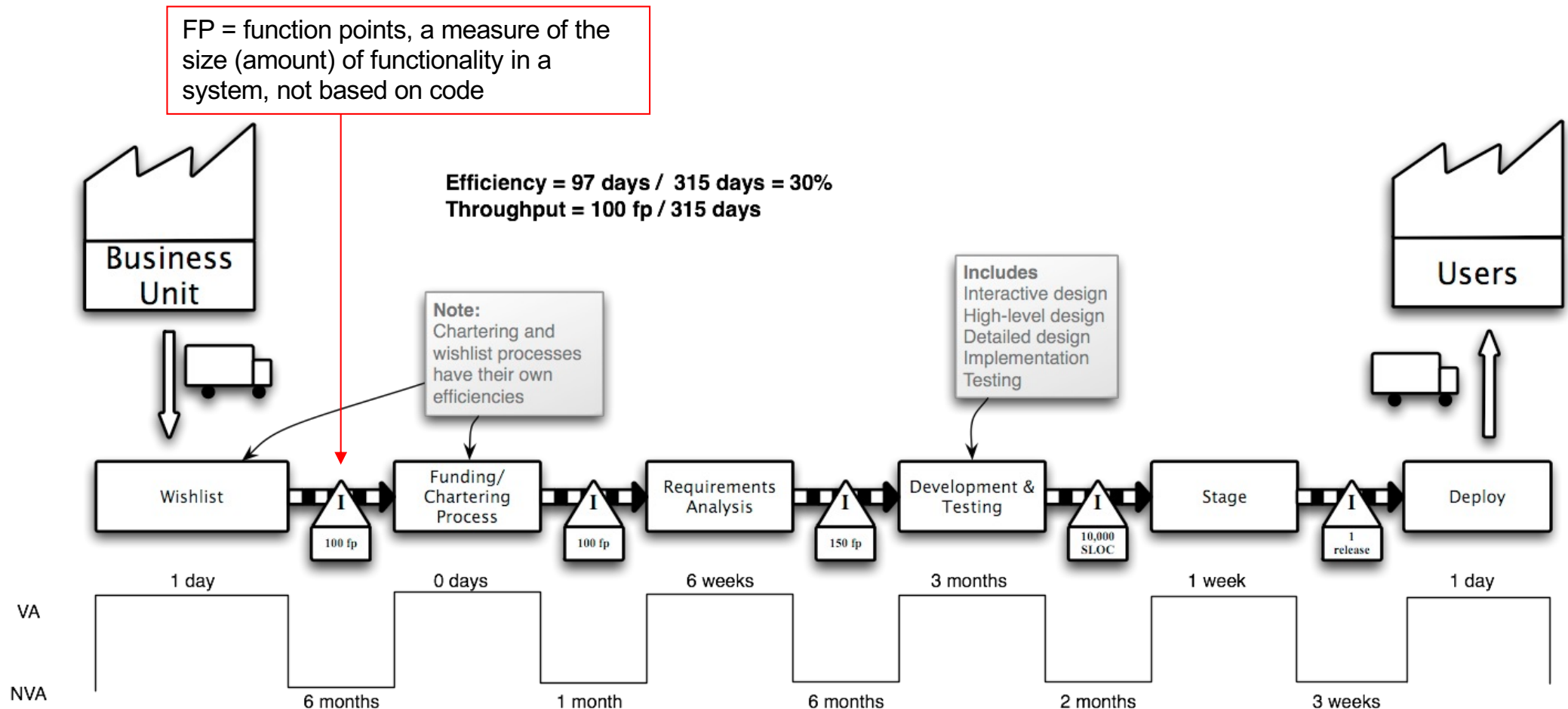
Fundamental in Lean, less focus in Agile



Total value stream – systems thinking

- What is the customer's purpose with the system?
- How to ensure flow of a customer's request or product through the whole stream until the customer has got some value?
- Value stream map of the organization to identify
 - waste
 - possibilities for optimizing the processes

Value stream mapping – Example



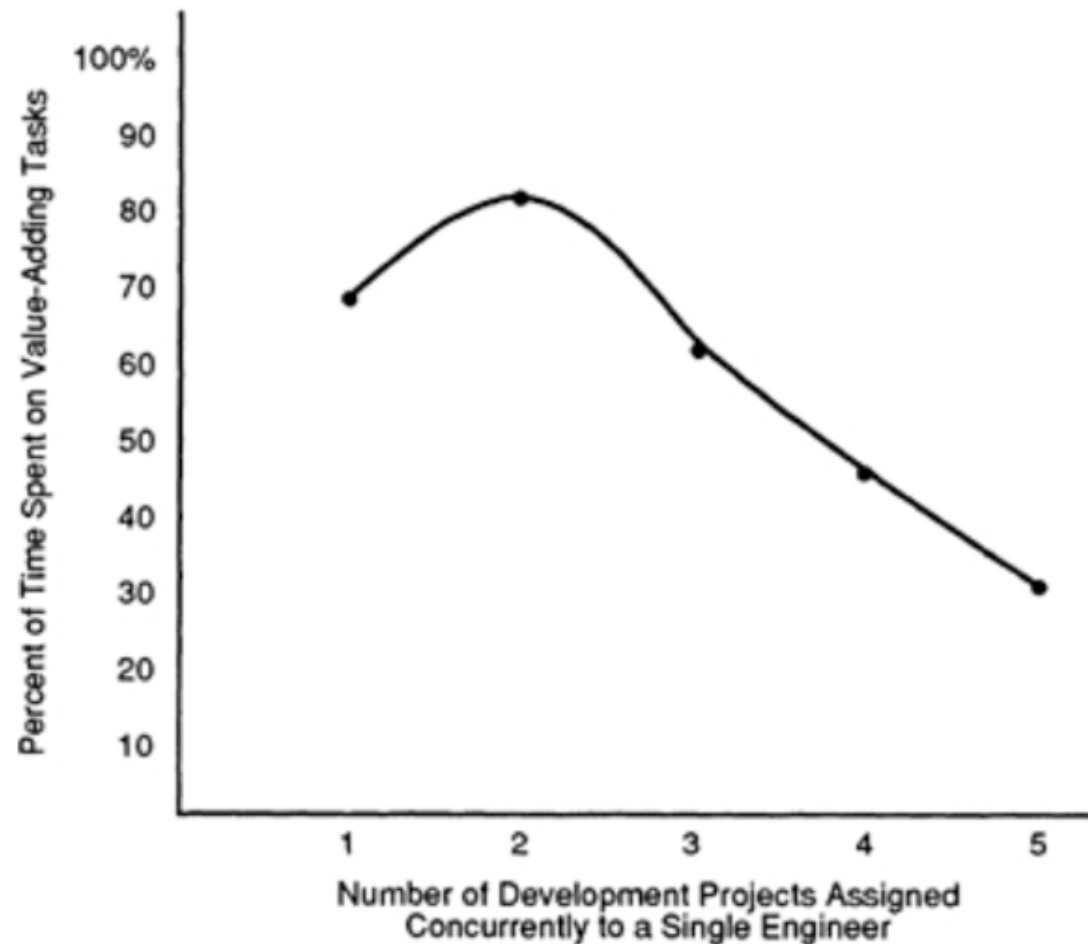
Join at menti.com | use code 8289 4006

 Mentimeter

How many activities in parallel make you most efficient over time?



Cost of context switching / multitasking



IN5140, 2023

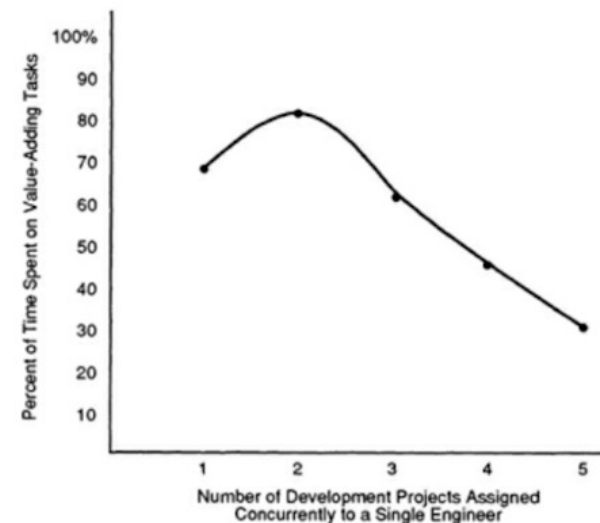
[K. B. Clark and S. C. Wheelwright, "Managing New Product and Process Development: Text and Cases," Free, New York, 1993.]

for such a key resource than to have the resource waiting for the next project. While that sounds appealing, it seldom works that way in practice.

The problem is that as the number of projects increases above one, productivity rises and then falls. Extensive proprietary studies by a major computer firm and our own observations of several medical electronics firms indicate a pattern like that shown in [Exhibit 4-2](#). When an engineer focused on a single project is given a second one, utilization often rises slightly because the engineer no longer has to wait for the activities of others involved in that single project. Instead, the engineer can move back and forth between the two projects. However, if a third, fourth, or even fifth project is added, the percentage of time spent on value-adding tasks drops rapidly, as an increasing fraction of valuable time is spent on non-value-added tasks—coordinating, remembering, or tracking down information, for example.² In addition, the engineer becomes the bottleneck on all of the projects to which he or she is assigned.

EXHIBIT 4-2

Productivity of Development Engineering Time^{*}



^{*}In the studies underlying this graph, the activities engaged in by development engineers were grouped into two categories—those that added value to a development project, and those that did not. This graph shows the percent of an engineer's activities in value-adding tasks.

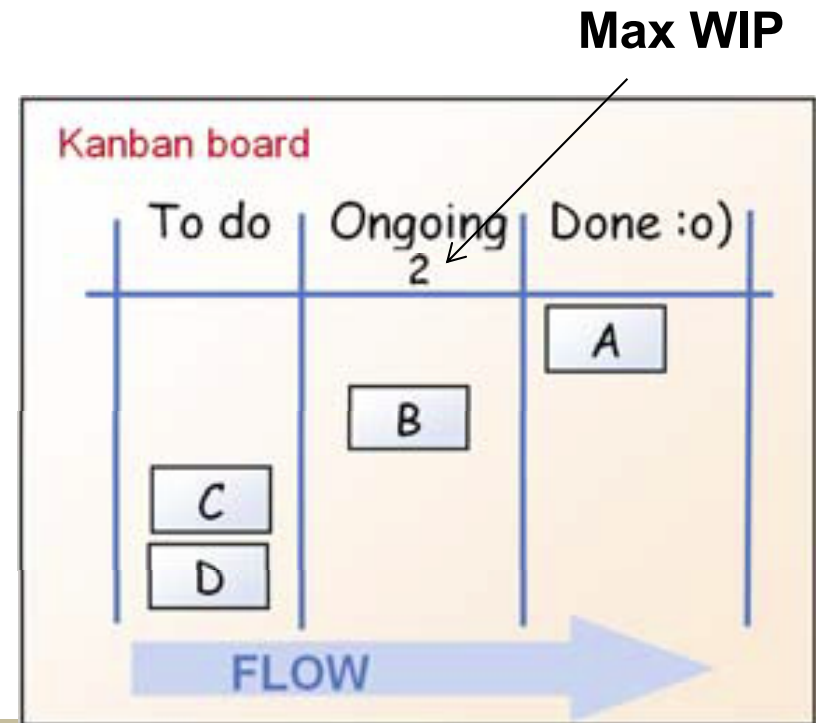
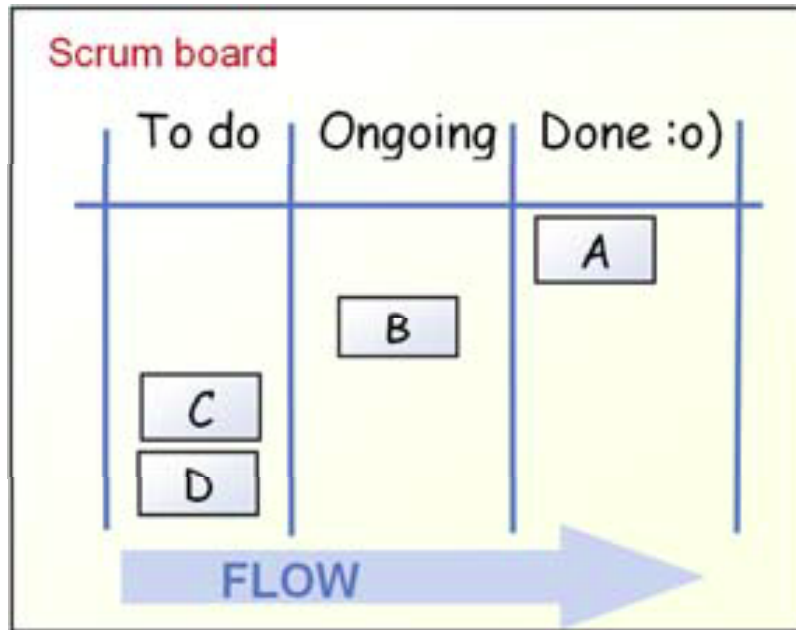
A Reactive Mix of Development Projects

For the senior managers at PreQuip, the news that their development organization was two to three times over-committed was bad enough. They also discovered, however, that their development resources were not focused on the critical issues for the business. After reviewing

WIP (Work in Progress/Process) limit

- Kanban limits the number of work items in a given state
- Scrum limits the number of work items in a sprint, but not per state

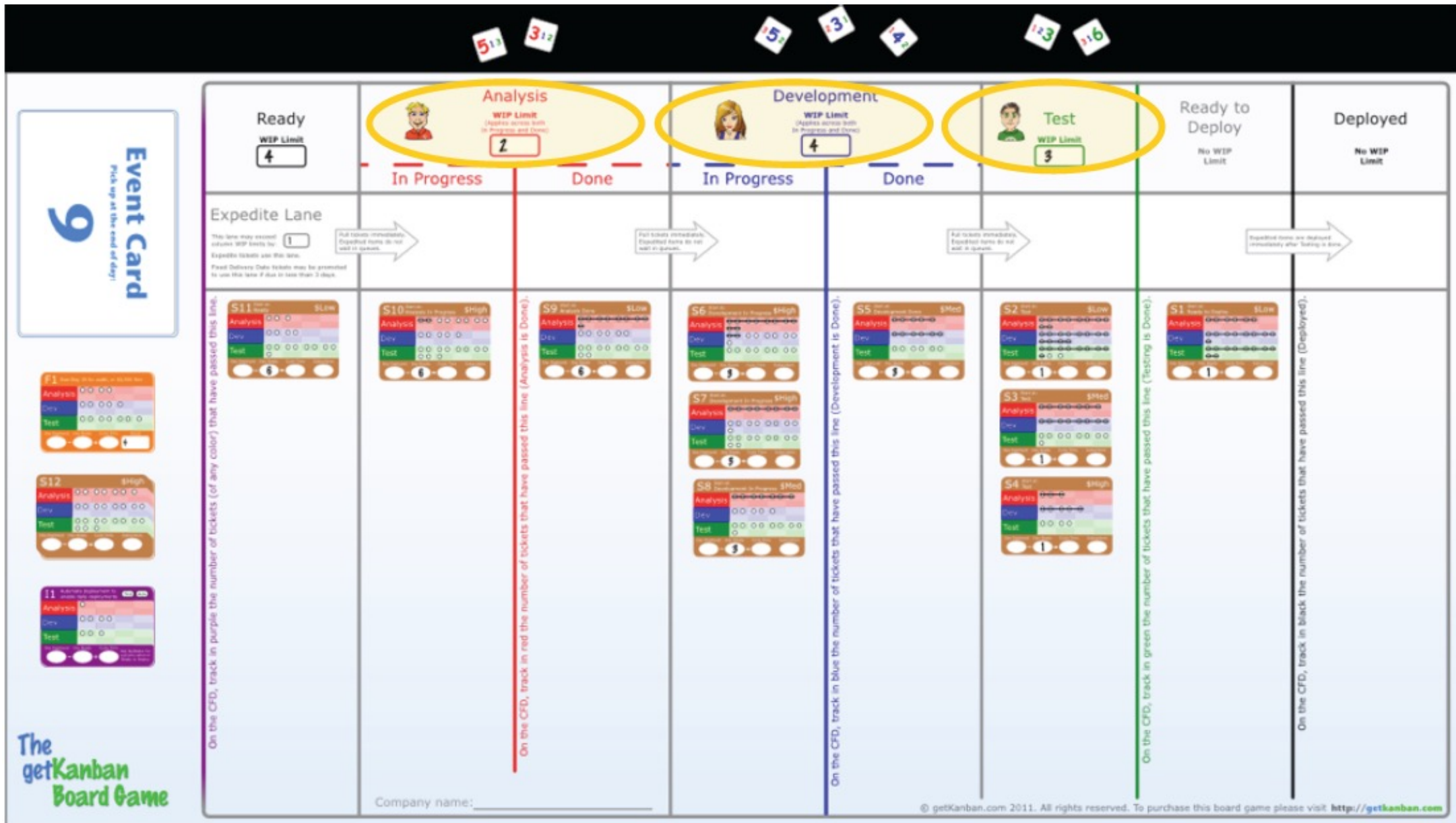
Scrum board versus Kanban board



From: Kanban and Scrum - making the most of both by Henrik Kniberg and Mattias Skarin on Dec 21, 2009

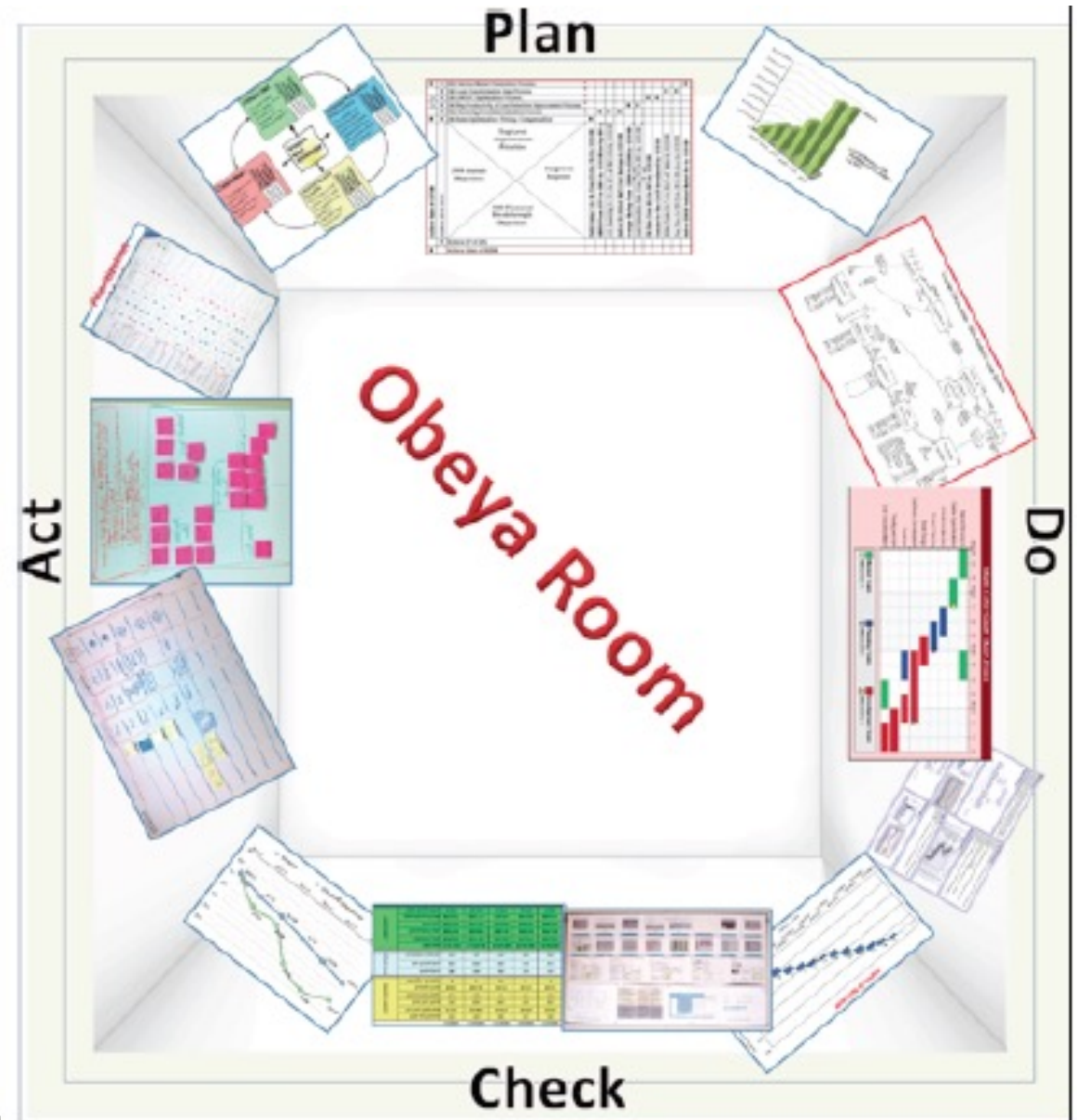


Visualization is important in Lean



War room (obeya)

- A form of project management, where all those involved in planning (of a project) gather in one room
- Visual charts and graphs depicting:
 - Vision/objectives
 - Plan with milestones and progress
 - Task board
 - Metrics
 - Timing or technical problems with countermeasures
- Reduces need for communication and thus saves time and costs



2. Let the team self-organize

Agile manifesto principle 11: The best architectures, requirements, and designs emerge from self-organizing teams

Lean: Central in both Lean and Norwegian/Nordic work life

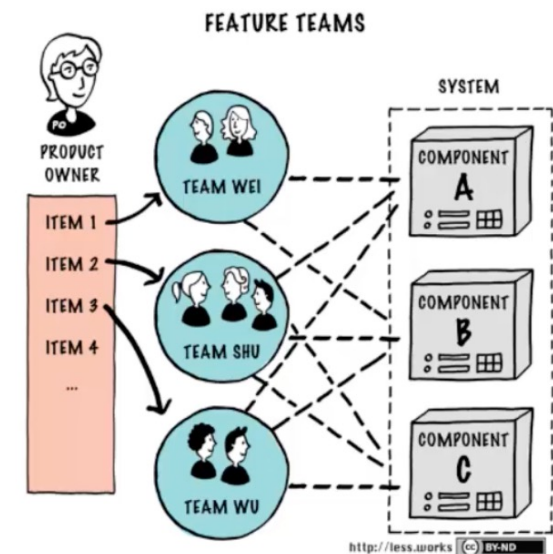
Self-organizing and cross-functional teams

- Decisions should be taken where the relevant competence is
- Cross-functional teams have all the competence needed to carry out all the tasks of an iteration
 - Include people who can test that the code works and domain experts who can test that the code does what it is supposed to do. By being in the same team, testing can be done continually, minimizing the amount of code waiting to be released.
- Requirements of self-organized team
 - Everybody should be heard

From lecture 1 September

Cross-functional teams (feature teams)

- Build features of the final product
 - *E.g.*, seat reservation, tickets, landing page etc.
- Team members not committed to one specialist area (i.e., they are full-stack)
 - Any developer can work on any task
- Should work together for a longer time period (preferably co-located)
- Ideally , 7 ± 2 people



Lean: local initiative

- Give people confidence to make decisions themselves. Then they become more motivated
- Lead teams to discover good processes themselves, instead of enforcing processes on them
- Don't document processes extensively
 - Newcomers should have a gentle introduction to processes, but should have freedom to continuously improve them

Work/process improvement in the “Norwegian/Nordic model”

- Self-governed (autonomous) teams
- Learning, redundancy/job-rotation
- Work environment
- ”Quality of life”
- ”Three-party model”: Co-operation among management, workers (trade unions) and government
- Norwegian Hydro, Volvo and many more

B. Gustavsen, T.U. Quale, B.A. Sørensen, M. Midtbø og P.H. Engelstad. Innovasjonssamarbeid mellom bedrifter og forskning – den norske modellen. Gyldendal 2010

3. Work at a sustainable pace

Agile manifesto principle 8: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Lean: Taken for granted, basis. Ensured through flow and avoiding waste.

Implies realistic schedules

- One of Deming's ten key principles: "Eliminate the use of slogans, posters and exhortations for the work force, demanding zero defects and **new levels of productivity**, without providing methods."
- **Lean**: To optimize flow, slack in the time schedule is OK. That is, an employee may have some waiting time to optimize overall flow.

4. Develop minimal software

Agile manifesto principle 10: Simplicity – the art of maximizing the amount of work not done – is essential

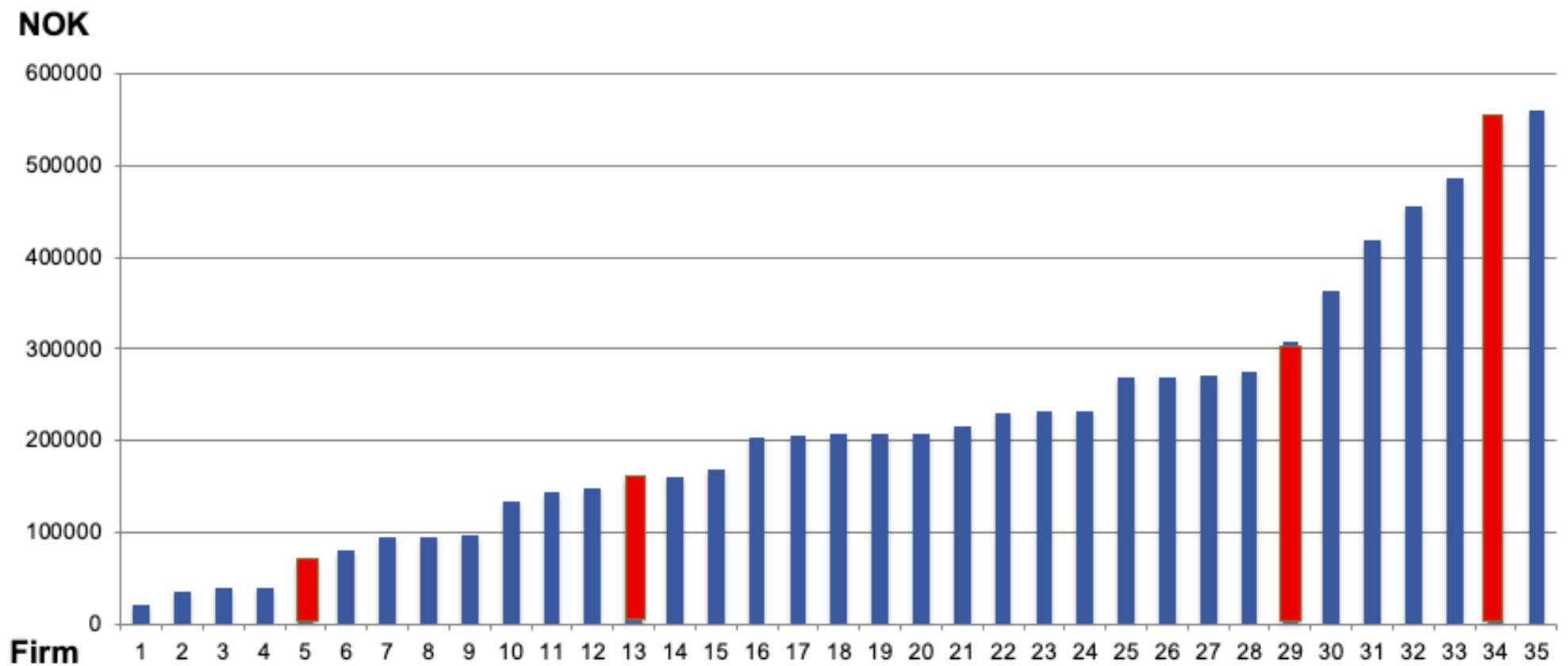
Simpler formulation:

Minimize the amount of necessary work done

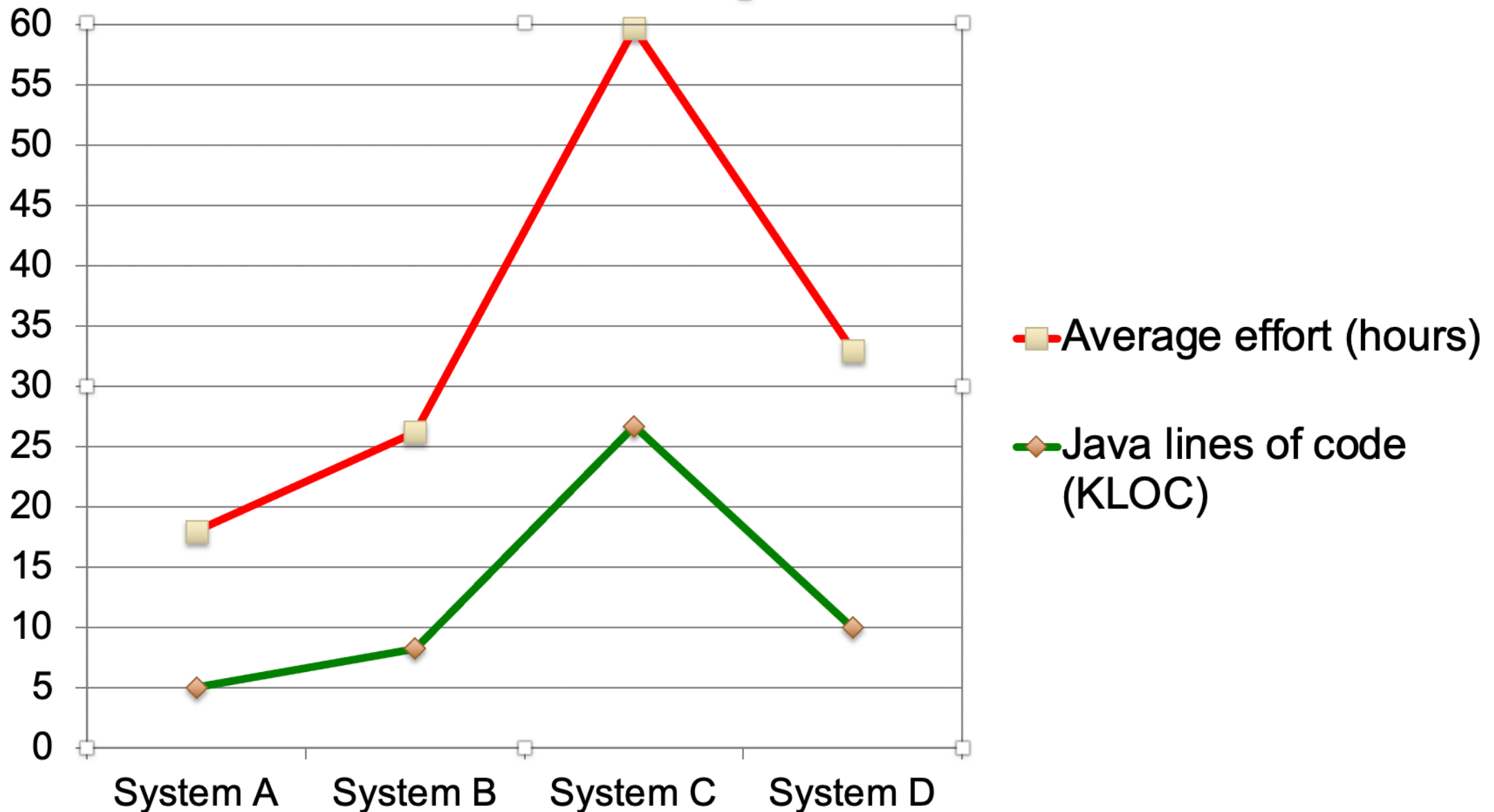
Lean: Complexity causes waste

Important point made by Meyer: The point is to develop minimal software, not to reduce the work done. It may require more effort to develop minimal than not so minimal software

Four companies developed the same system

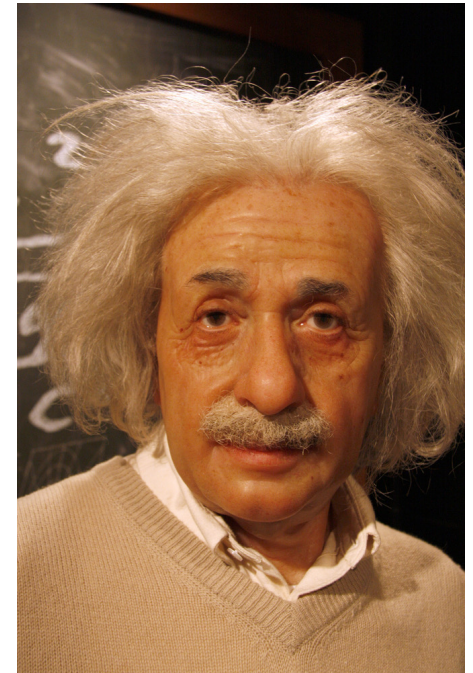


Code size versus maintainability



Consequence: “small is beautiful”

“One principle that every software developer should follow is that “small is beautiful”. Developers should therefore put some extra effort in making their systems small. Our own, controlled studies show that smaller design and code, given other factors constant, lead to more understandable and maintainable systems. Also a large number of other studies show that size correlate negatively with many quality attributes. Note that this does not imply that the smallest possible system is the best. An analogy is written text—it should be minimal, but not less than minimal, or as Einstein stated: “Everything should be made as simple as possible, but not simpler.”*



[Walter Tichy: Empirical software research: an interview with Dag Sjøberg, University of Oslo, Norway.” *ACM Ubiquity*. (June 2011) <http://dl.acm.org/citation.cfm?id=1998374>]

See book Section 4.4.4 “Develop minimal software”

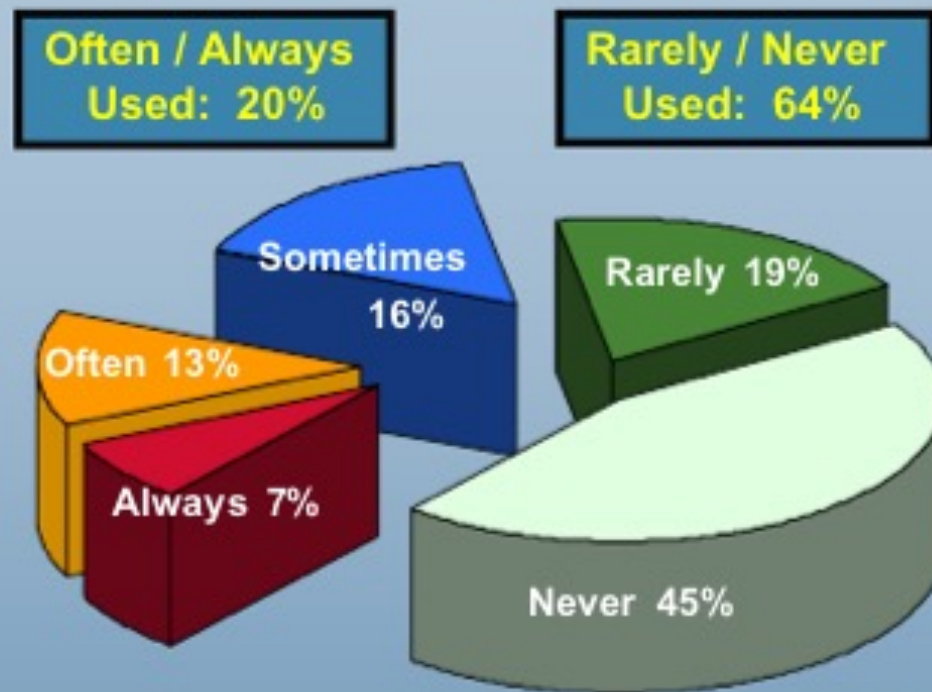
*Finnes ingen skriftlig kilde

Waste 1: Extra Features



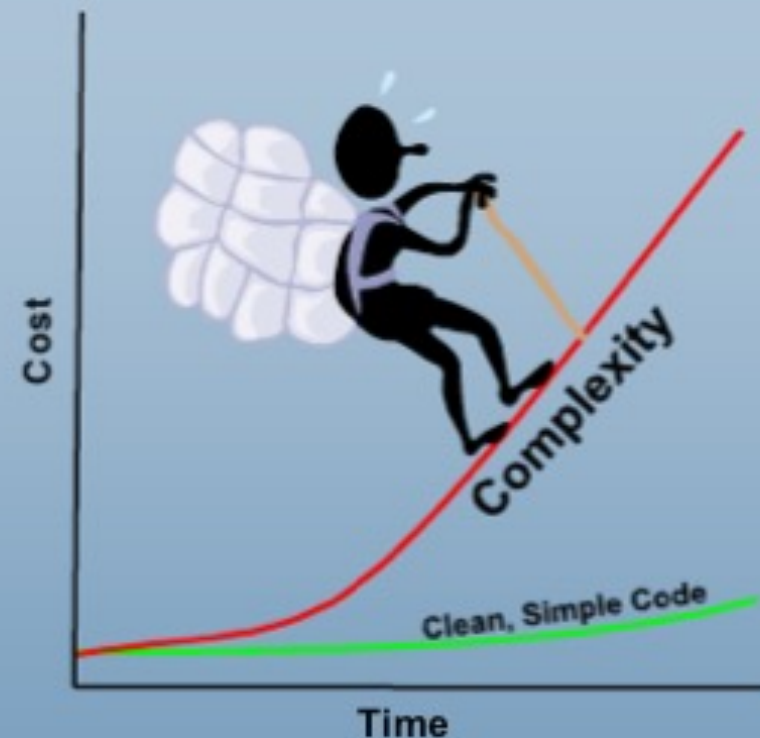
“Chaos reports”: Be a bit sceptic, cf. “catastrophism”, textbook p. 26

Features / Functions Used in a Typical System



Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Cost of Complexity



The Biggest opportunity for increasing Software Development Productivity: Write Less Code!

Lean principle: Just-in-Time (JIT)

- Make decisions as late as possible, that is, not before they are demanded
- By not developing functionality before it's needed, the need may disappear or changed, and then the size of the software is possibly reduced
- Still, don't exclude features of great value even though they are rarely used. Features should be considered from the combination of value and frequency of use

5. Accept change

Agile manifesto principle 2: Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Positive attitude towards change

- Change should not be considered a risk in itself
- Change is necessary to create better systems
- Agile teams don't oppose change



Lean principle: Just-In-Time (continued)

- Many details can be decided after fundamental aspects have been implemented
- Focus only on details that are necessary to implement the work items in process
- Waste is avoided:
 - Certain items may turn out not be needed at all
 - Items may be implemented in better ways than if they had been implemented earlier
 - Reduce bottlenecks

Good requirements specifications still important

- The possibility for change should be no excuse for not clarifying fundamental requirements early, see book p. 149 “deprecation of upfront tasks”
- Eliciting, analyzing and specifying requirements are as important as coding and testing
- Many methods and tools are available

6. Continuous learning*

Agile manifesto principle 12*: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

- In retrospectives (an agile practice Section 6.10 in book), the team should reflect on what went well and not in the last iteration (sprint). Problems are identified but often not solved.
- Related to Level 5 in CMMI: “Optimizing”
- An organization should facilitate continuous learning

*Not included as an organization principle in the book.
Agile manifesto principle 12 is considered a practice in the book (p. 50).

Lean: Continuous learning and improvement (Kaizen) – focus on details

- Improving processes requires improvements of many (small) elements
 - “[James Watt](#) made a critical improvement in 1764, by removing spent steam to a separate vessel for condensation, greatly improving the amount of work obtained per unit of fuel consumed.” [Wikipedia]
- Catastrophes often a result of sequences of small errors
- Consider problems as challenges – opportunities for learning, don't blame individuals
- Solutions require time but is an investment (improved quality and time saved later)

Continuous learning: related to the focus on quality in Lean and two Agile principles:

Agile manifesto principle 5*: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Agile manifesto principle 9*: Continuous attention to technical excellence and good design enhances agility.

*Called “platitudes” by Meyer

Summary

- Close relationship between Lean and Agile
- Principles like avoiding waste and ensuring flow are more prevalent in Lean than Agile
- The technical principles in Agile are more closely related to software development than general process improvement
- This lecture emphasized organizational principles; more about technical principles in later lectures