IN5140 – Smart processes and agile methods in software engineering

Lecture 15 November 2023:

Lean and Agile Software Engineering

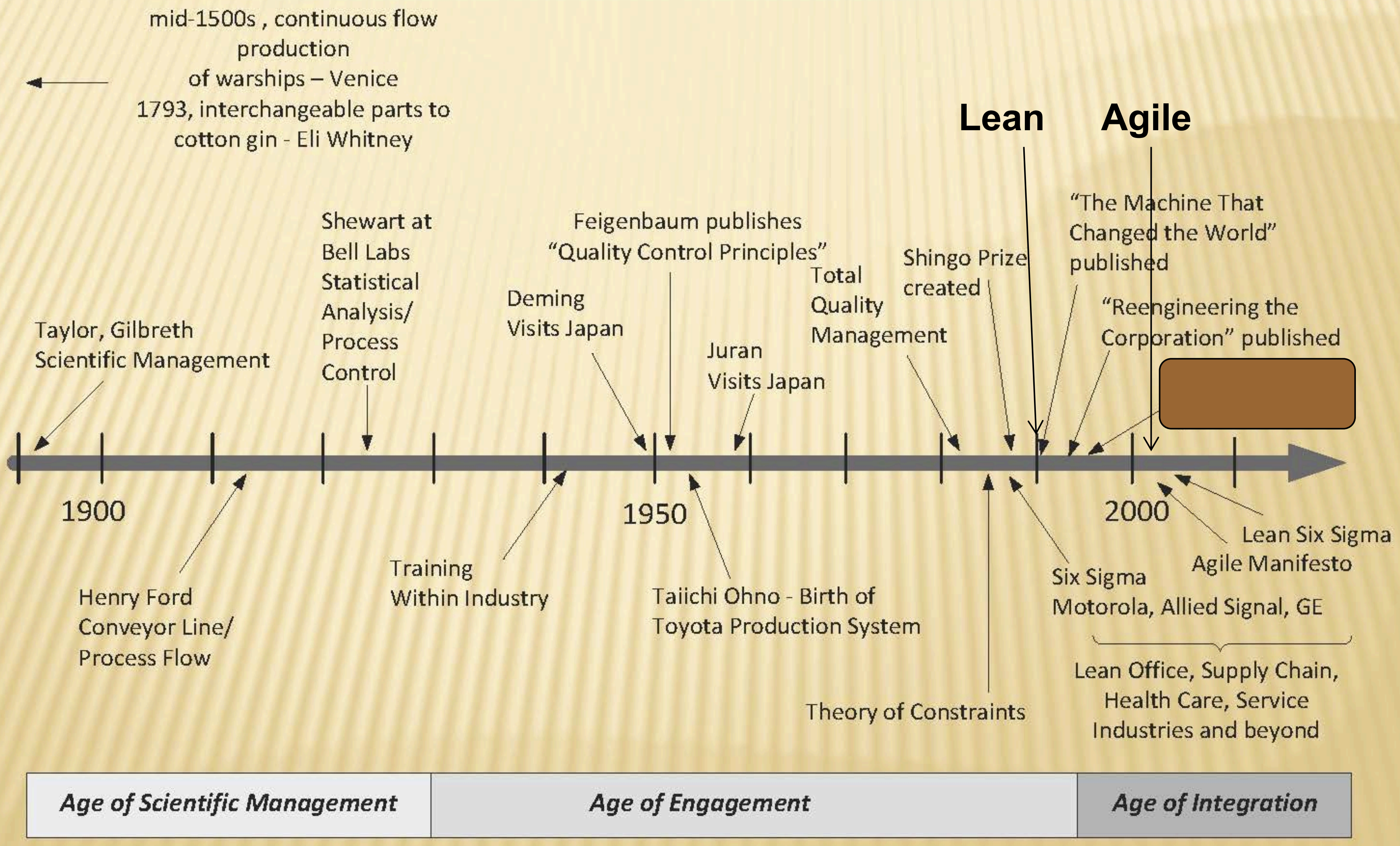
Dag Sjøberg

Clear relationship between

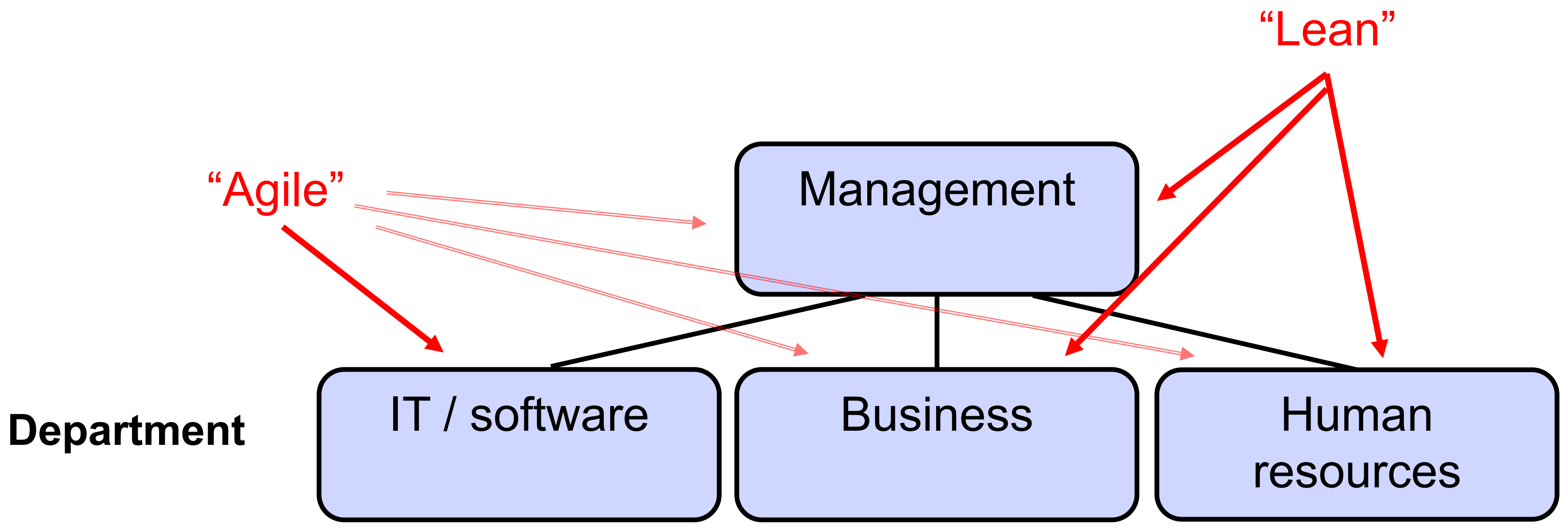
- **Process improvement**
- **Lean**
- **Agile**

→ **It's about working smarter**

HISTORY OF PROCESS IMPROVEMENT


mid-1500s , continuous flow production of warships – Venice
1793, interchangeable parts to cotton gin - Eli Whitney
Lean
Agile
Taylor, Gilbreth Scientific Management
Shewart at Bell Labs Statistical Analysis/ Process Control
Deming Visits Japan
Feigenbaum publishes "Quality Control Principles"
Juran Visits Japan
Total Quality Management
Shingo Prize created
"The Machine That Changed the World" published
"Reengineering the Corporation" published
1900
1950
2000
Henry Ford Conveyor Line/ Process Flow
Training Within Industry
Taiichi Ohno - Birth of Toyota Production System
Theory of Constraints
Six Sigma Motorola, Allied Signal, GE
Agile Manifesto
Lean Six Sigma
Lean Office, Supply Chain, Health Care, Service Industries and beyond
Age of Scientific Management
Age of Engagement
Age of Integration

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. Ch. 9 in the book), but "rather a philosophy made of a set of general observations", p. 135
- Crystal is not important; Section 9.5 is not syllabus

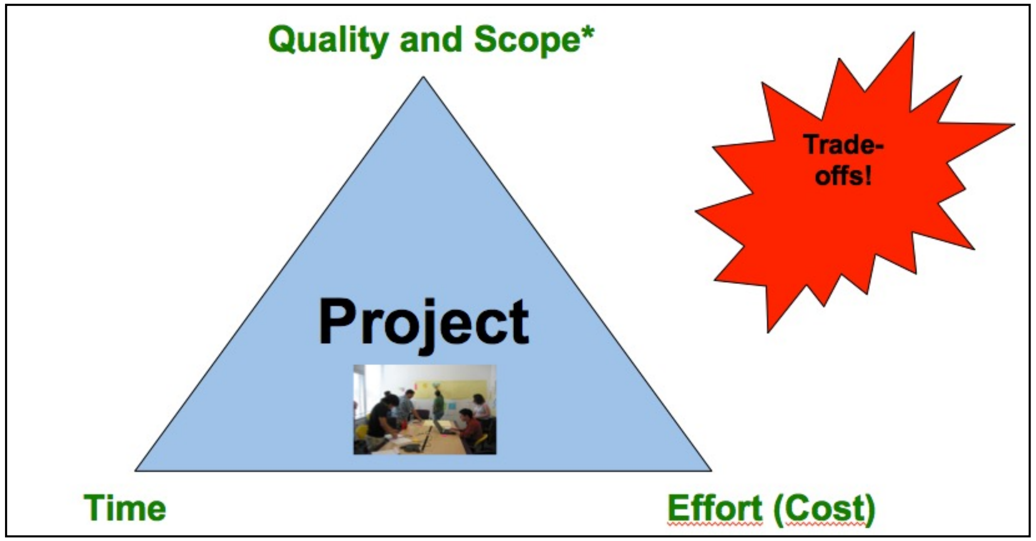
Structure

- 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 $\longrightarrow$



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".

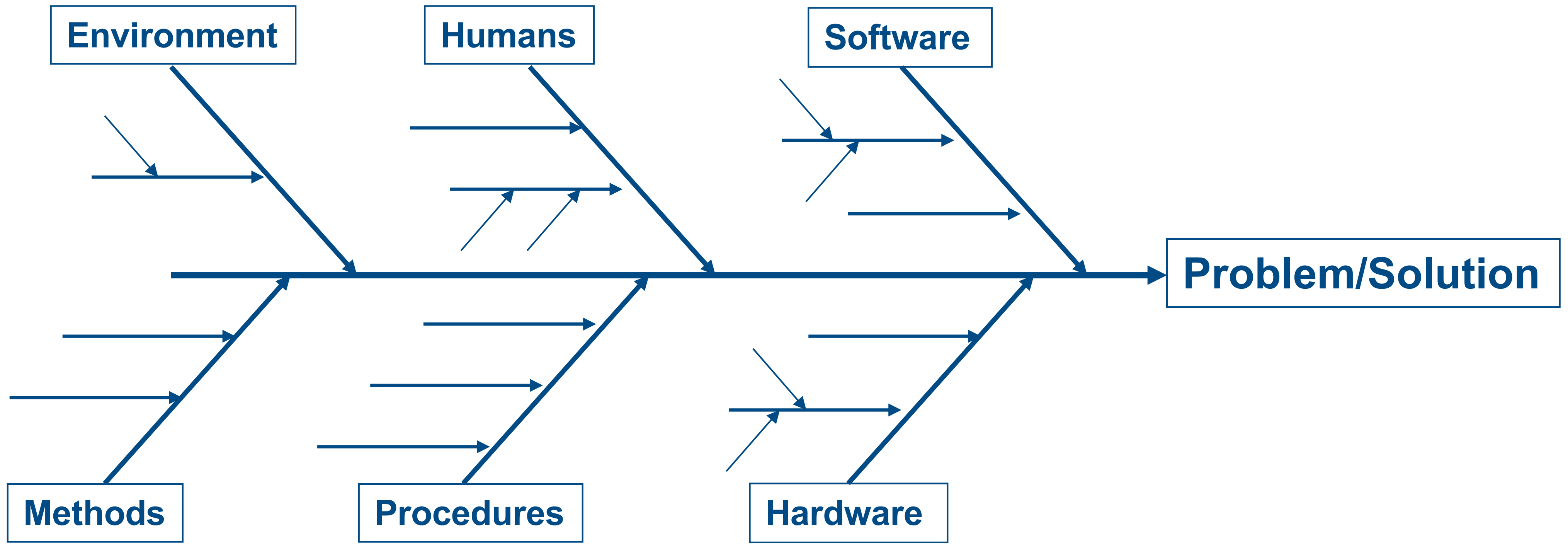
[From Wikipedia 2014]

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



***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 process improvement; it is an overall concept that covers may aspects

Quite far from Japanese car industry

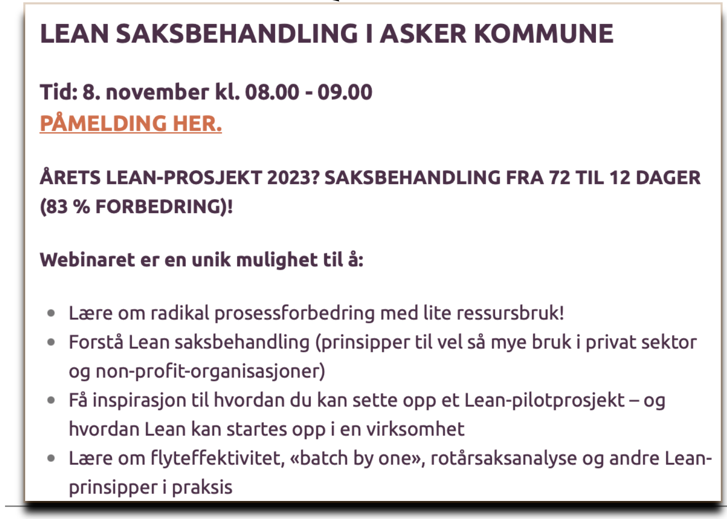
LEAN FORUMS ÅRSKONFERANSE
14.-15. november 2023

- FRA BESTE TIL NESTE PRAKSIS

LEAN FORUMS ÅRSKONFERANSE
14.- 15. november 2023
Hotel Quality Expo Fornebu
FRA BESTE TIL NESTE PRAKSIS
Møt Akinori Hyodo, en av verdens mest kjente innen Toyota Production System.

Kontinuerlig forbedringsarbeid er nøkkelen til suksess. Vi lever i en tid med stor uro og stadig skiftende krav. Det er viktigere enn noensinne å finne nye veier og tenke utenfor boksen. Vi må utfordre oss selv på produktivitet, og å tenke grønnere og smartere.

Forbedringsarbeid handler om mer enn å fjerne sløsing av tid og å standardisere prosesser. Det handler om å strekke seg mot det neste nivået, å utvikle seg, og å finne nye og innovative måter å jobbe på. Derfor inviterer vi til årskonferansen 2023 hvor vi setter fokus på en mer effektiv og bærekraftig virksomhet. En stor utfordring, men også nettopp her det største potensialet ligger.


LEAN SAKSBEHANDLING I ASKER KOMMUNE

Tid: 8. november kl. 08.00 - 09.00
PÅMELDING HER.

ÅRETS LEAN-PROSJEKT 2023? SAKSBEHANDLING FRA 72 TIL 12 DAGER (83 % FORBEDRING)!

Webinaret er en unik mulighet til å:

- Lære om radikal prosessforbedring med lite ressursbruk!
- Forstå Lean saksbehandling (prinsipper til vel så mye bruk i privat sektor og non-profit-organisasjoner)
- Få inspirasjon til hvordan du kan sette opp et Lean-pilotprosjekt – og hvordan Lean kan startes opp i en virksomhet
- Lære om flyteffektivitet, «batch by one», rotårsaksanalyse og andre Lean-prinsipper i praksis

Structure

- Lean
- Comparing Lean and agile organizational principles

Agile Manifesto – 12 principles*

See lecture by Marthe Nordengen Berntzen 6 September

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.

***http://agilemanifesto.org/principles.html**

Agile principles

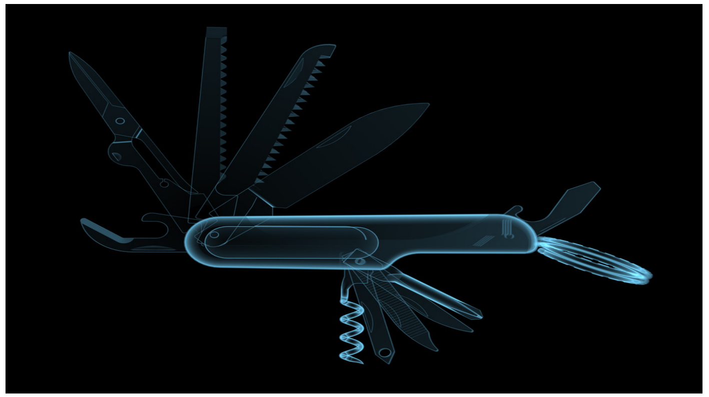
- Divided by Meyer, book Ch. 4, into:
  - *Organizational principles*
    1. Put the customer at the centre
    2. Let the team self-organize
    3. Work at a sustainable pace
    4. Develop minimal software
    5. Accept change
    6. Continuous learning
  - *Technical principles*
    1. Develop iteratively
    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?
- BizDevOps teams
  - Business, development and operation staff work closely together, see later lectures

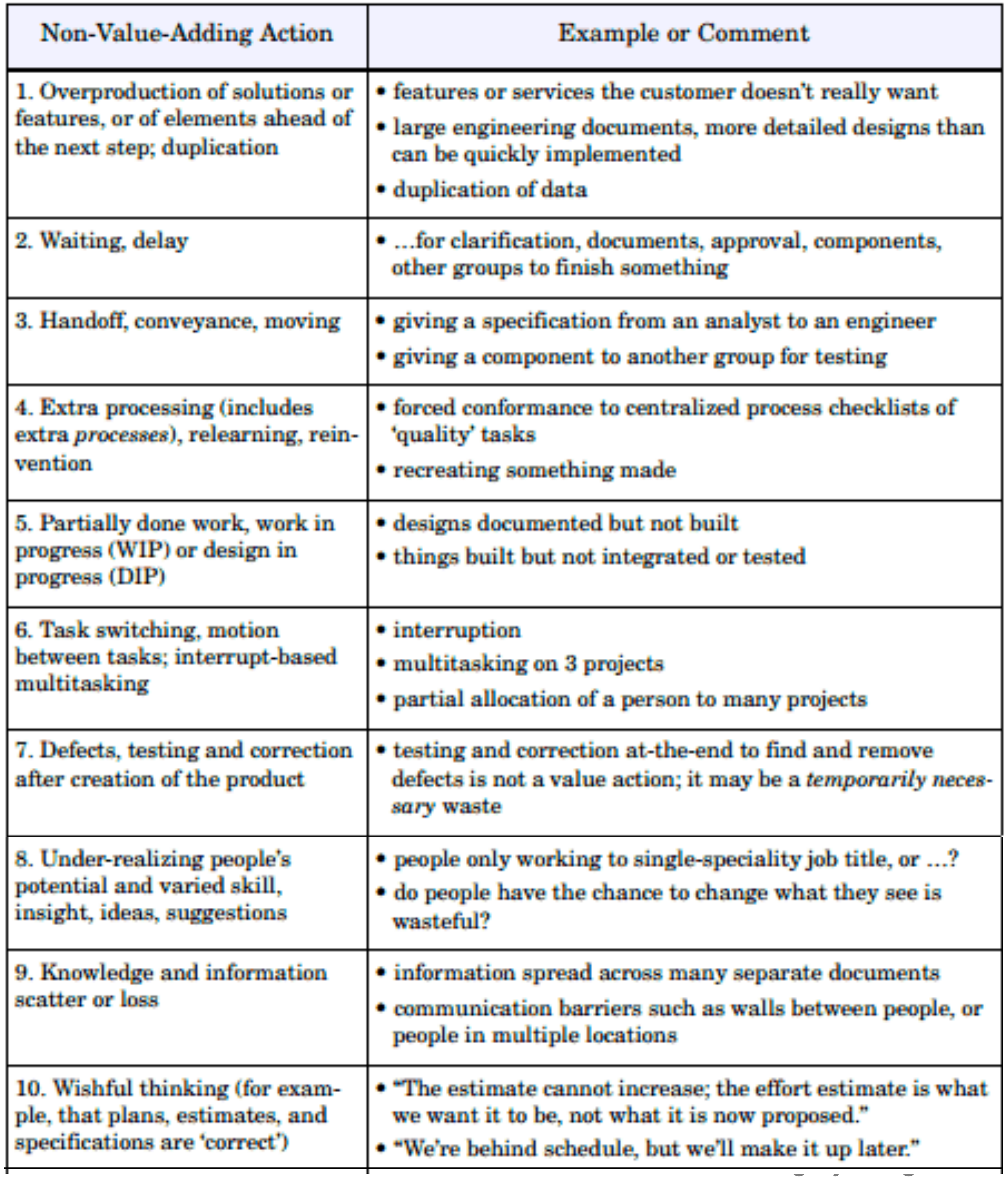
Waste in Lean

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 *processes*), 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 *temporarily necessary* 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

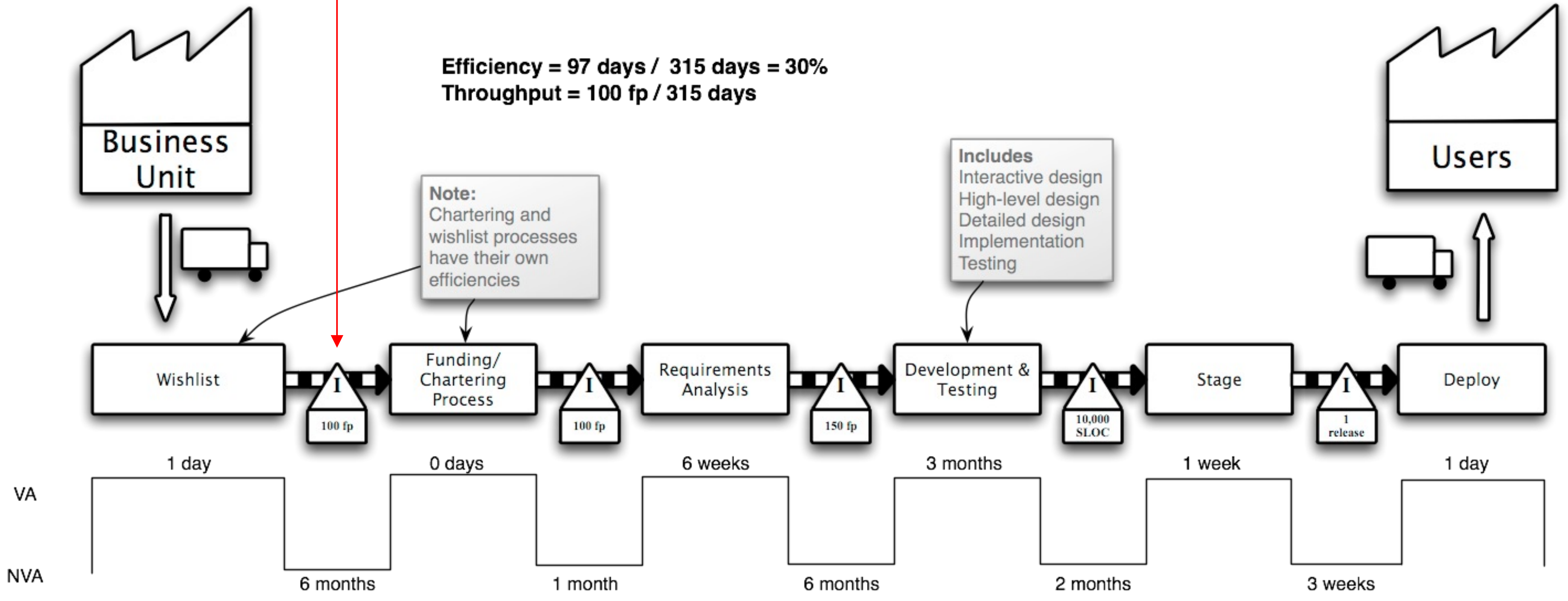
Fundamental in Lean, less focus in Agile

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

FP = function points, a measure of the size (amount) of functionality in a system, not based on code



Efficiency = 97 days / 315 days = 30%
Throughput = 100 fp / 315 days
Business Unit
Users
Note: Chartering and wishlist processes have their own efficiencies
Includes Interactive design High-level design Detailed design Implementation Testing
Wishlist
Funding/ Chartering Process
Requirements Analysis
Development & Testing
Stage
Deploy
I
100 fp
100 fp
150 fp
10,000 SLOC
1 release
1 day
0 days
6 weeks
3 months
1 week
1 day
VA
NVA
6 months
1 month
6 months
2 months
3 weeks

Join at menti.com use code 6602 2891

Mentimeter

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

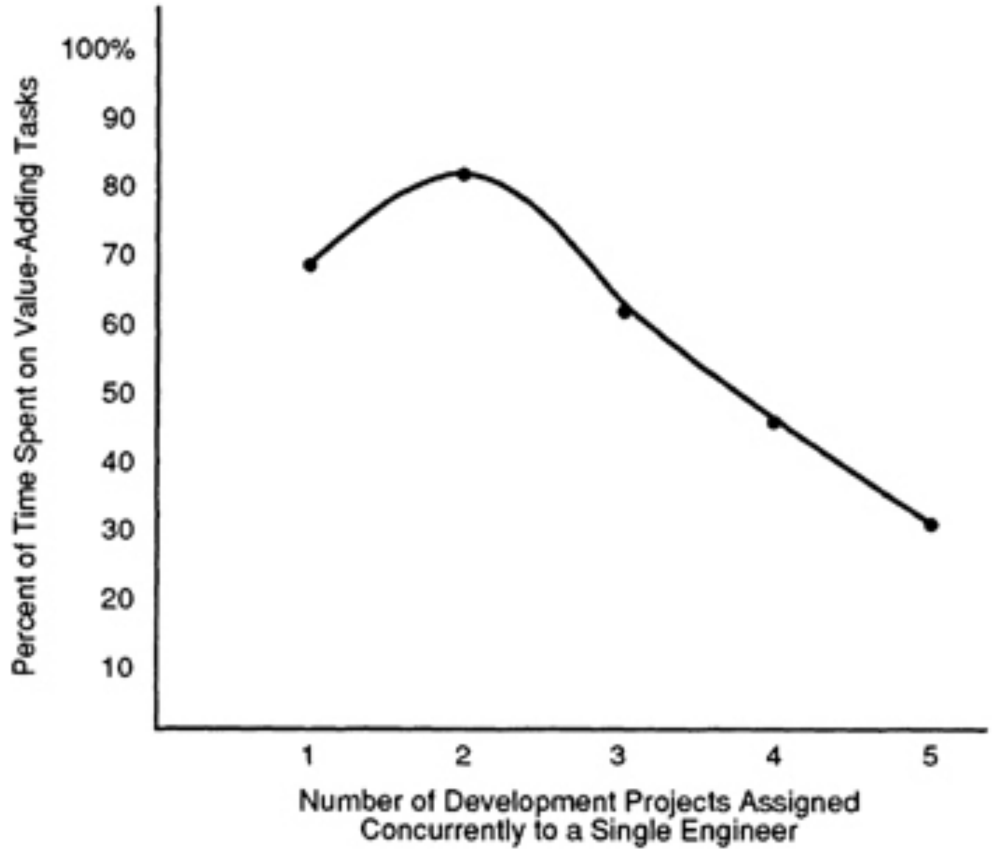
GO TO
menti.com
ENTER THE CODE
6602 2891
0

0	0	0	0	0	0
1	2	3	4	5	6

Responses are hidden

Cost of context switching / multitaskin

Productivity of Development Engineering Time*



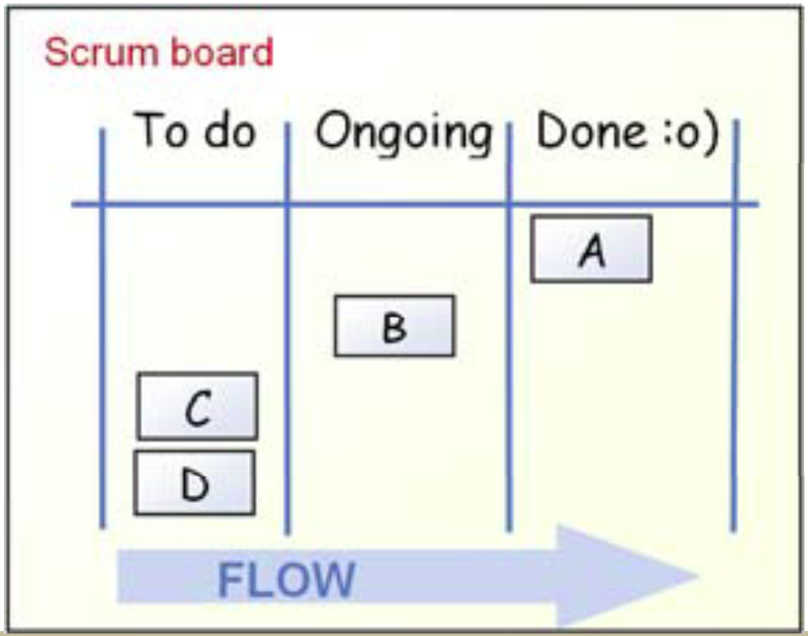
* In the studies underlying this graph, the activities engaged in by development engineers were grouped into two categories—those that added value to a development project, and those that did not. This graph shows the percent of an engineer's activities in value-adding tasks.

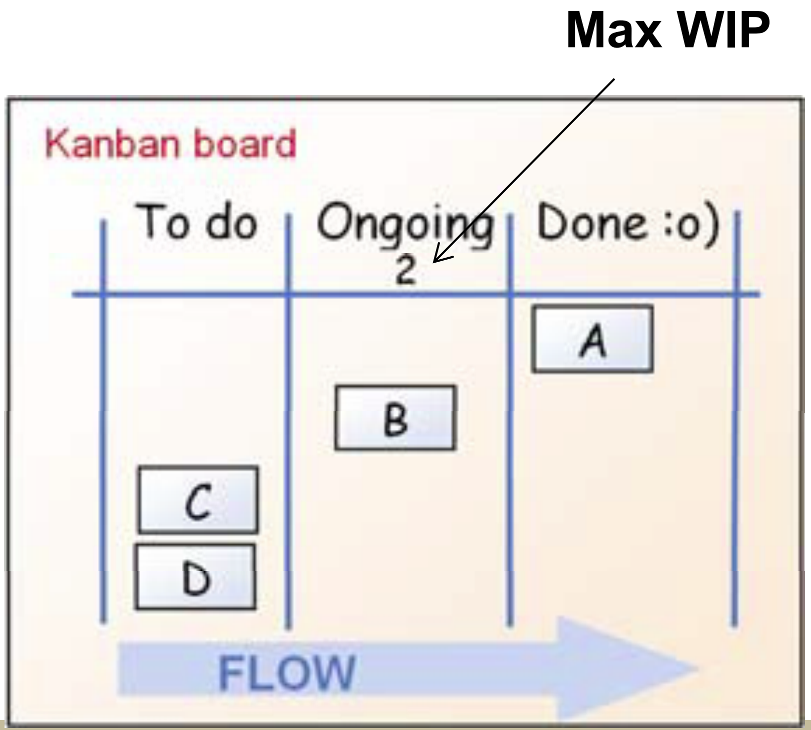
[Clark and Wheelwright 1993]

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

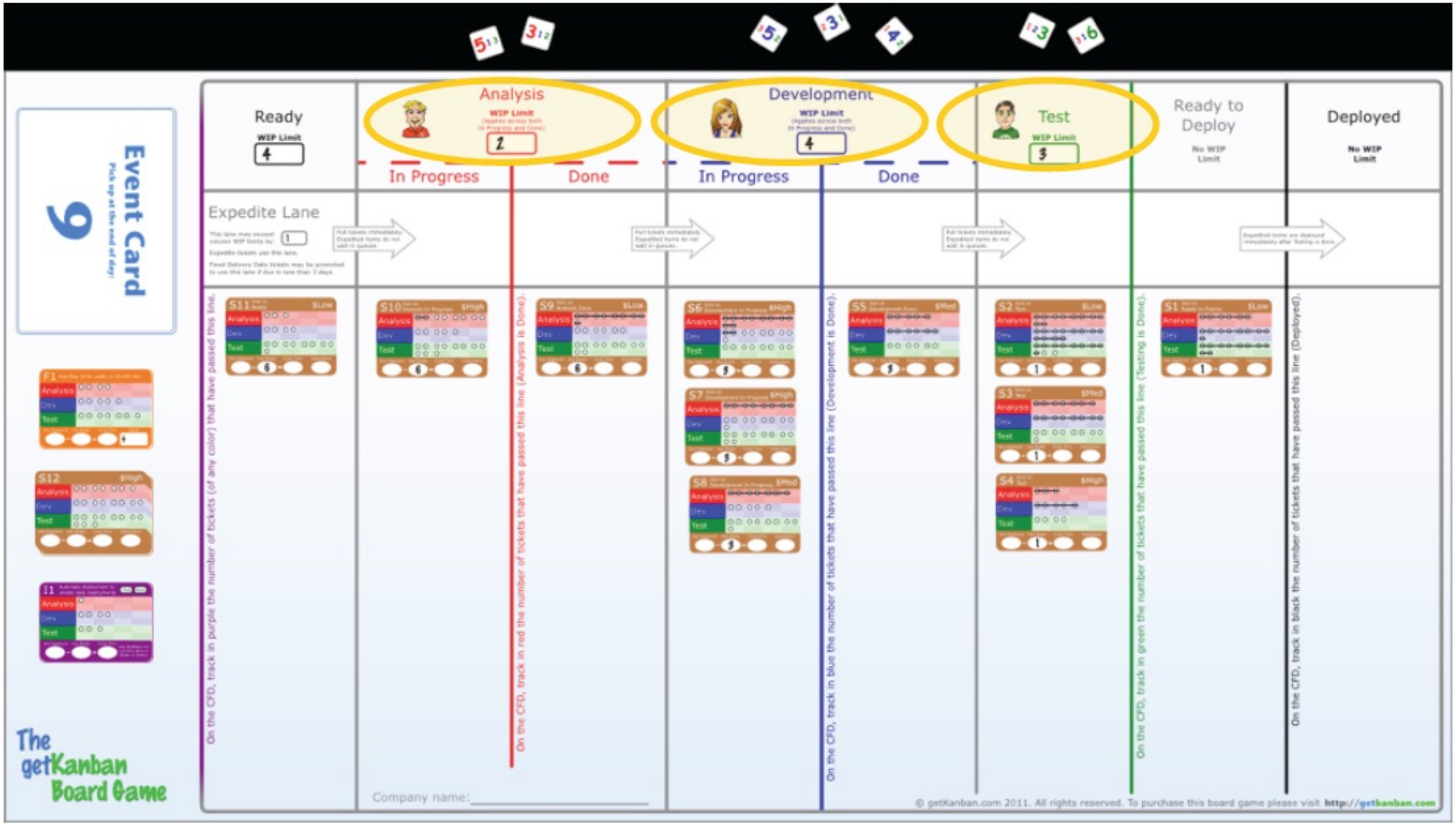




From: Kanban and Scrum - making the most of both by Henrik Kniberg and Mattias Skarin on Dec 21, 2009

[redacted] [redacted]

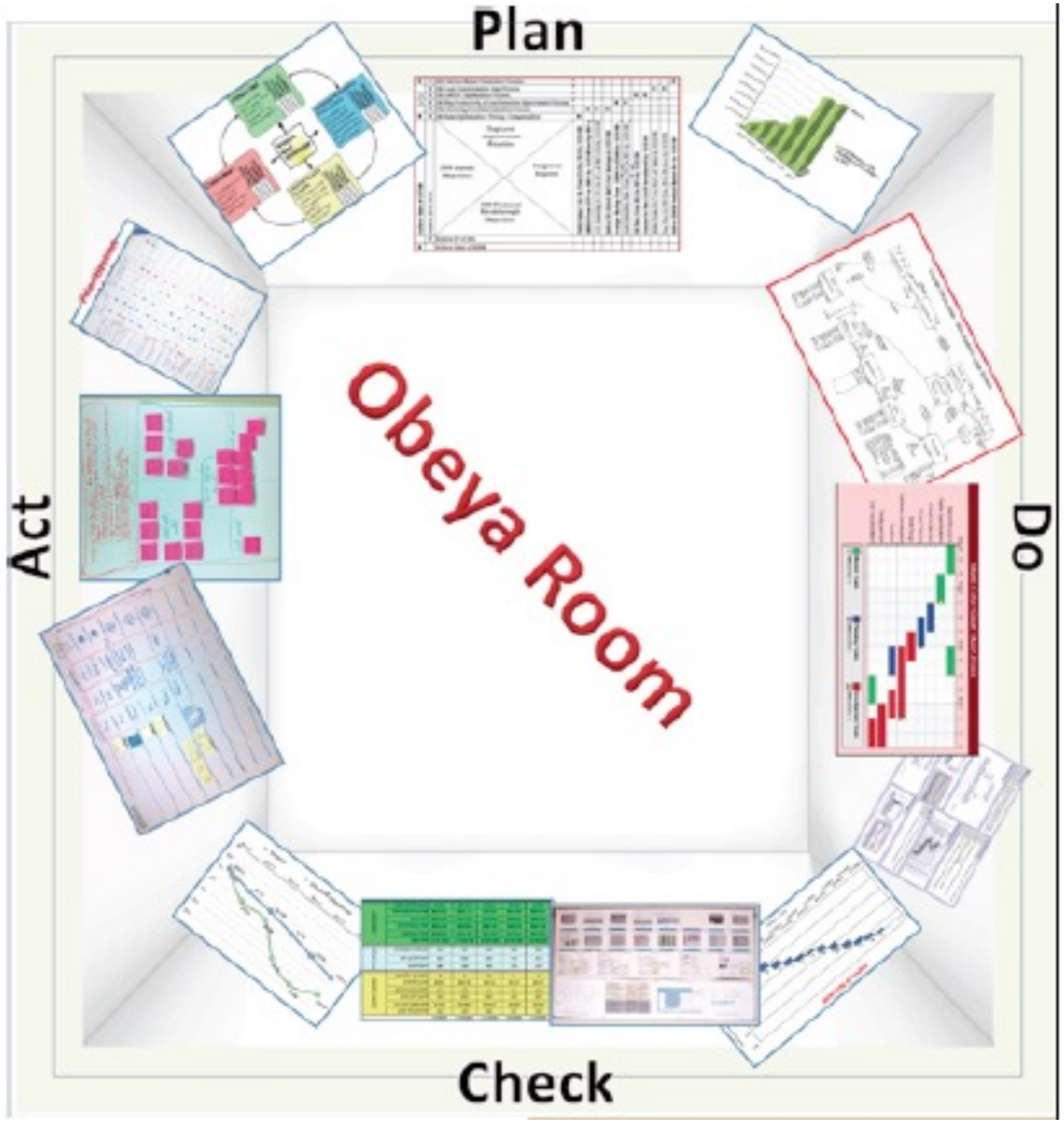
Visualization is important in Lean

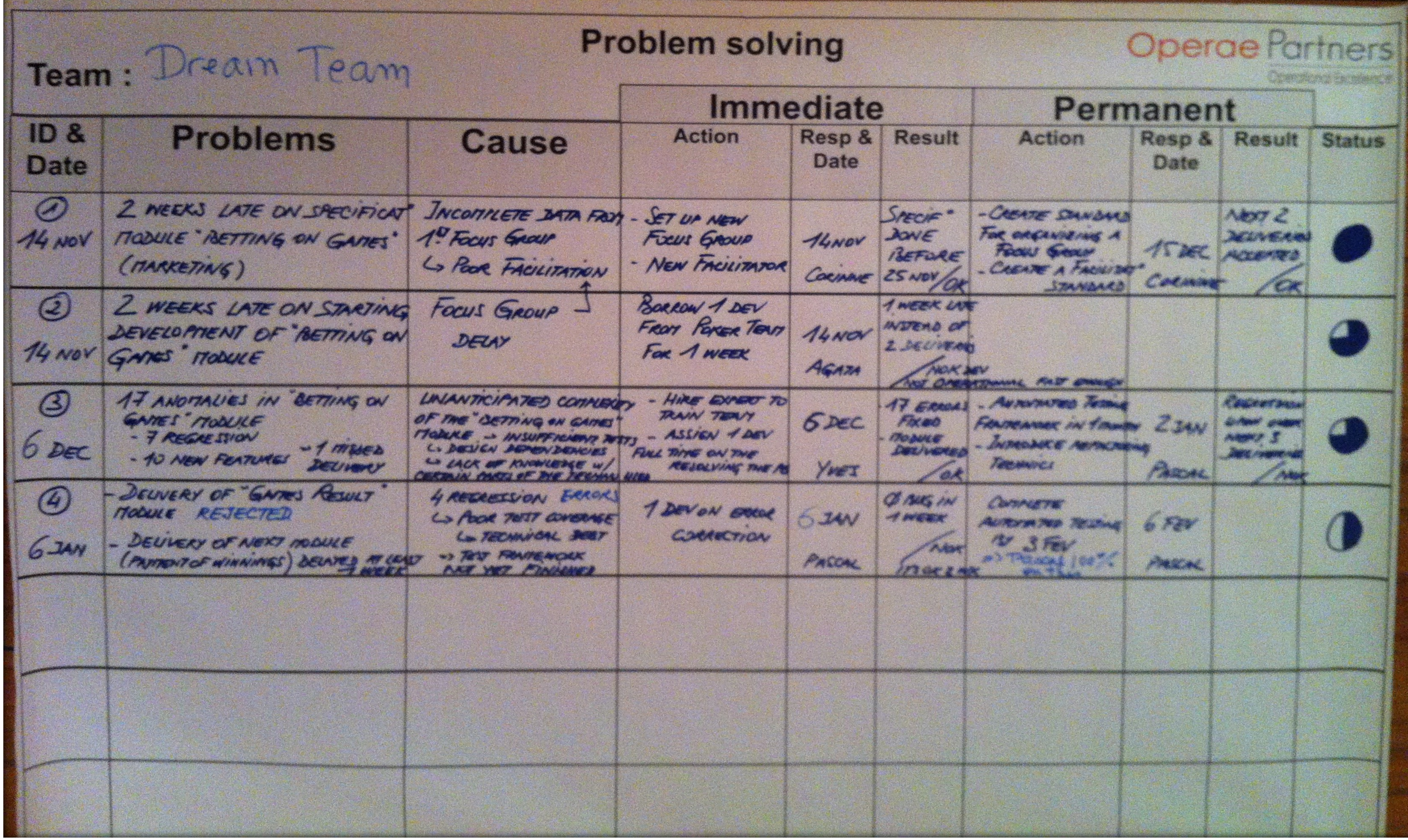


Event Card
9
Ready
WIP Limit
4
Analysis
WIP Limit
2
In Progress
Done
Development
WIP Limit
4
In Progress
Done
Test
WIP Limit
3
Ready to Deploy
No WIP Limit
Deployed
No WIP Limit
Expedite Lane
S11
S10
S9
S6
S7
S8
S5
S2
S3
S4
S1
F1
S12
I1
Analysis
Dev
Test
On the CFD, track in purple the number of tickets (of any color) that have passed this line.
On the CFD, track in red the number of tickets that have passed this line (Analysis is Done).
On the CFD, track in blue the number of tickets that have passed this line (Development is Done).
On the CFD, track in green the number of tickets that have passed this line (Testing is Done).
On the CFD, track in black the number of tickets that have passed this line (Deployed).
Company name:
The getKanban Board Game
© getKanban.com 2011. All rights reserved. To purchase this board game please visit http://getkanban.com

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





Problem solving

Operae Partners
Operational Excellence

Team : Dream Team

ID & Date	Problems	Cause	Immediate Action	Immediate Resp & Date	Immediate Result	Permanent Action	Permanent Resp & Date	Permanent Result	Status
① 14 NOV	2 WEEKS LATE ON SPECIFICAT° MODULE "BETTING ON GAMES" (MARKETING)	INCOMPLETE DATA FROM 1ST FOCUS GROUP ↳ POOR FACILITATION	- SET UP NEW FOCUS GROUP - NEW FACILITATOR	14 NOV CORINNE	SPECIF° DONE BEFORE 25 NOV / OK	- CREATE STANDARD FOR ORGANIZING A FOCUS GROUP - CREATE A FACILITAT° STANDARD	15 DEC CORINNE	NEXT 2 DELIVERIES ACCEPTED / OK	●
② 14 NOV	2 WEEKS LATE ON STARTING DEVELOPMENT OF "BETTING ON GAMES" MODULE	FOCUS GROUP DELAY	BORROW 1 DEV FROM POKER TEAM FOR 1 WEEK	14 NOV AGATA	1 WEEK LATE INSTEAD OF 2 DELIVERED / NOK DEV NOT OPERATIONAL FAST ENOUGH				◕
③ 6 DEC	17 ANOMALIES IN "BETTING ON GAMES" MODULE - 7 REGRESSION - 10 NEW FEATURES → 1 MISSED DELIVERY	UNANTICIPATED COMPLEXITY OF THE "BETTING ON GAMES" MODULE → INSUFFICIENT TESTS ↳ DESIGN DEPENDENCIES ↳ LACK OF KNOWLEDGE W/ CERTAIN PARTS OF THE [illegible]	- HIRE EXPERT TO TRAIN TEAM - ASSIGN 1 DEV FULL TIME ON THE RESOLVING THE [illegible]	6 DEC YVES	- 17 ERRORS FIXED - MODULE DELIVERED / OK	- AUTOMATED TESTING FRAMEWORK IN [illegible] - INTRODUCE REFACTORING TECHNICS	2 JAN PASCAL	REGRESSION [illegible] NEXT 3 DELIVERIES / NOK	◕
④ 6 JAN	- DELIVERY OF "GAMES RESULT" MODULE REJECTED - DELIVERY OF NEXT MODULE (PAYMENT OF WINNINGS) DELAYED AT LEAST 1 WEEK	4 REGRESSION ERRORS ↳ POOR TEST COVERAGE ↳ TECHNICAL DEBT → TEST FRAMEWORK NOT YET FINISHED	1 DEV ON ERROR CORRECTION	6 JAN PASCAL	Ø BUG IN 1 WEEK / NOK [illegible]	COMPLETE AUTOMATED TESTING BY 3 FEV => [illegible] 100% [illegible]	6 FEV PASCAL		◑

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 that can test that the code does what it is supposed to do. By being in the same team, means testing can be done continually, minimizing the amount of code waiting to be released.
- Requirements of self-organized team
  - Everybody should be heard

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

- Related to unrealistic productivity gains by the Scrum guru’s in Software Innovation of Scrum vs. Kanban
- 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

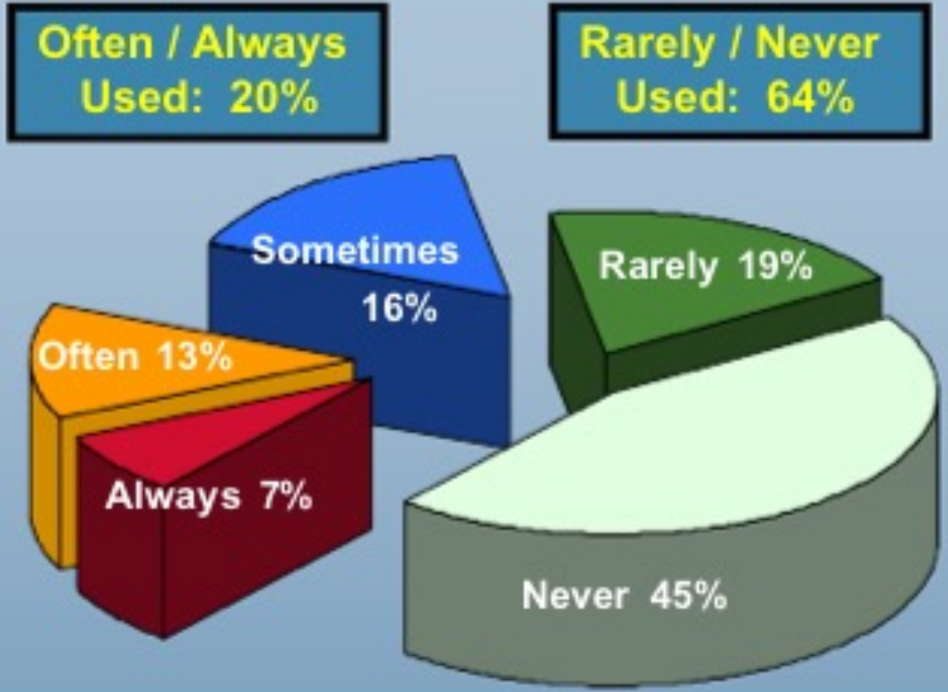
Simpler formulation:
Minimize the amount of necessary work done

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, see lecture 13. September

Waste 1: Extra Features

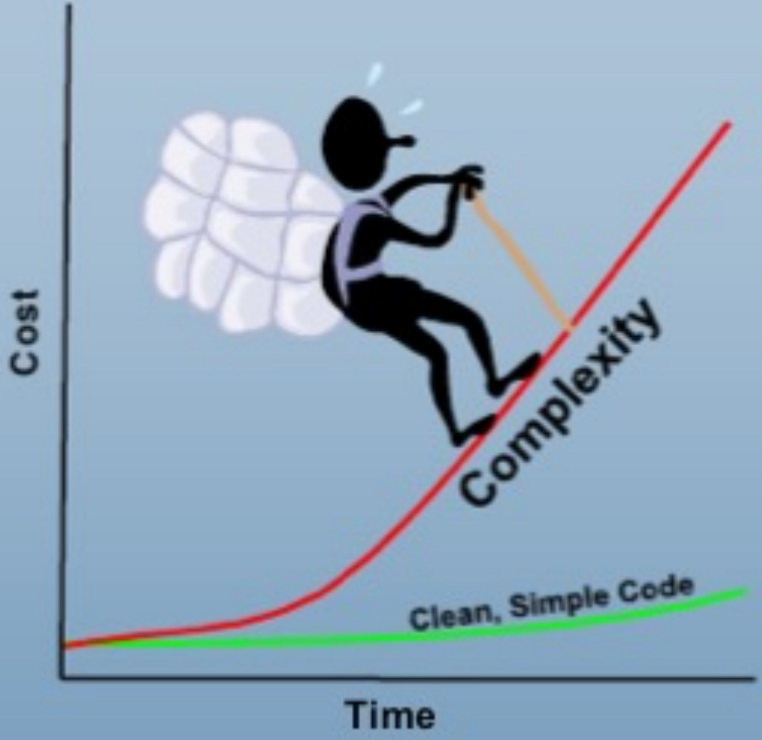
“Chaos reports”: Be a bit sceptic, cf.“catastrophism”, textbook p. 26

Features / Functions Used in a Typical System



Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Cost of Complexity

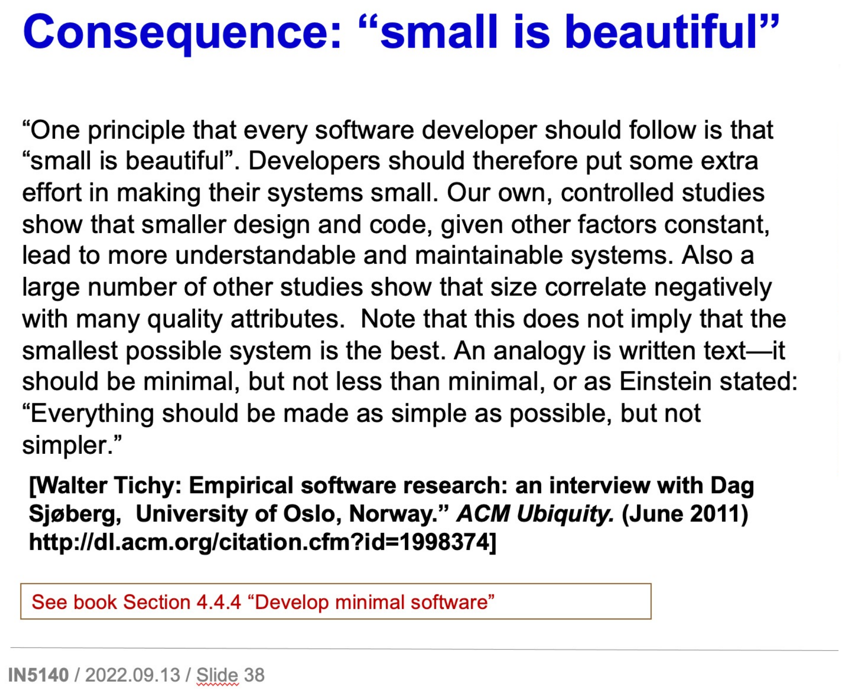


The Biggest opportunity for increasing Software Development Productivity: Write Less Code!

Lean principle: Just-in-Time

- By not developing functionality before it’s needed, the need may disappear or changed
- 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

Lean Cause of waste: Complexity



Consequence: "small is beautiful"

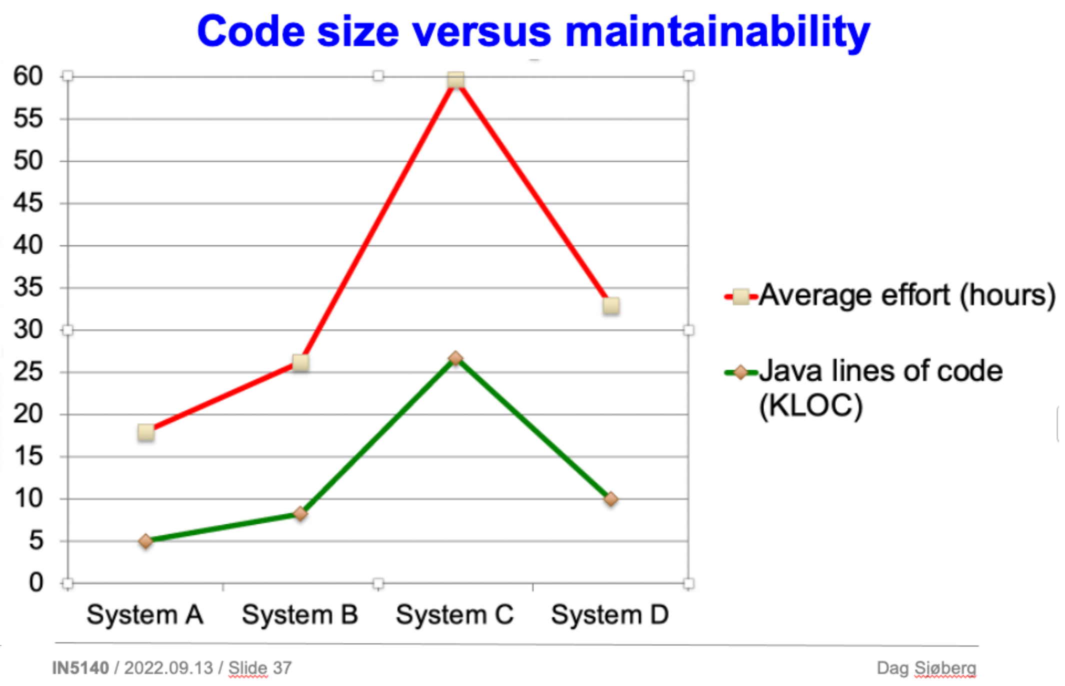
"One principle that every software developer should follow is that "small is beautiful". Developers should therefore put some extra effort in making their systems small. Our own, controlled studies show that smaller design and code, given other factors constant, lead to more understandable and maintainable 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"

IN5140 / 2022.09.13 / Slide 38

Code size versus maintainability



IN5140 / 2022.09.13 / Slide 37

Dag Sjøberg

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 (JIT)

Make decisions as late as possible, that is, not before they are demanded

- 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 is 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”

*Not included as an organization principle in the book.
Agile manifesto principle 12 is considered a practice by Meyer.

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

- Catastrophes often a result of sequences of small errors
- Improving processes requires improvements of many (small) elements
- Consider problems as challenges – opportunities for learning, don’t blame individuals
- Solutions require time but is an investment (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 “platitutes” by Meyer

Summary

- Close relationship between Lean and Agile
- Principles like avoiding waste and ensuring flow are more prevalent in Lean
- The technical principles in Agile are more closely related to software development