

Ohjelmistotuotanto

Luento10

14.4.

Miten laajalti Agilea käytetään

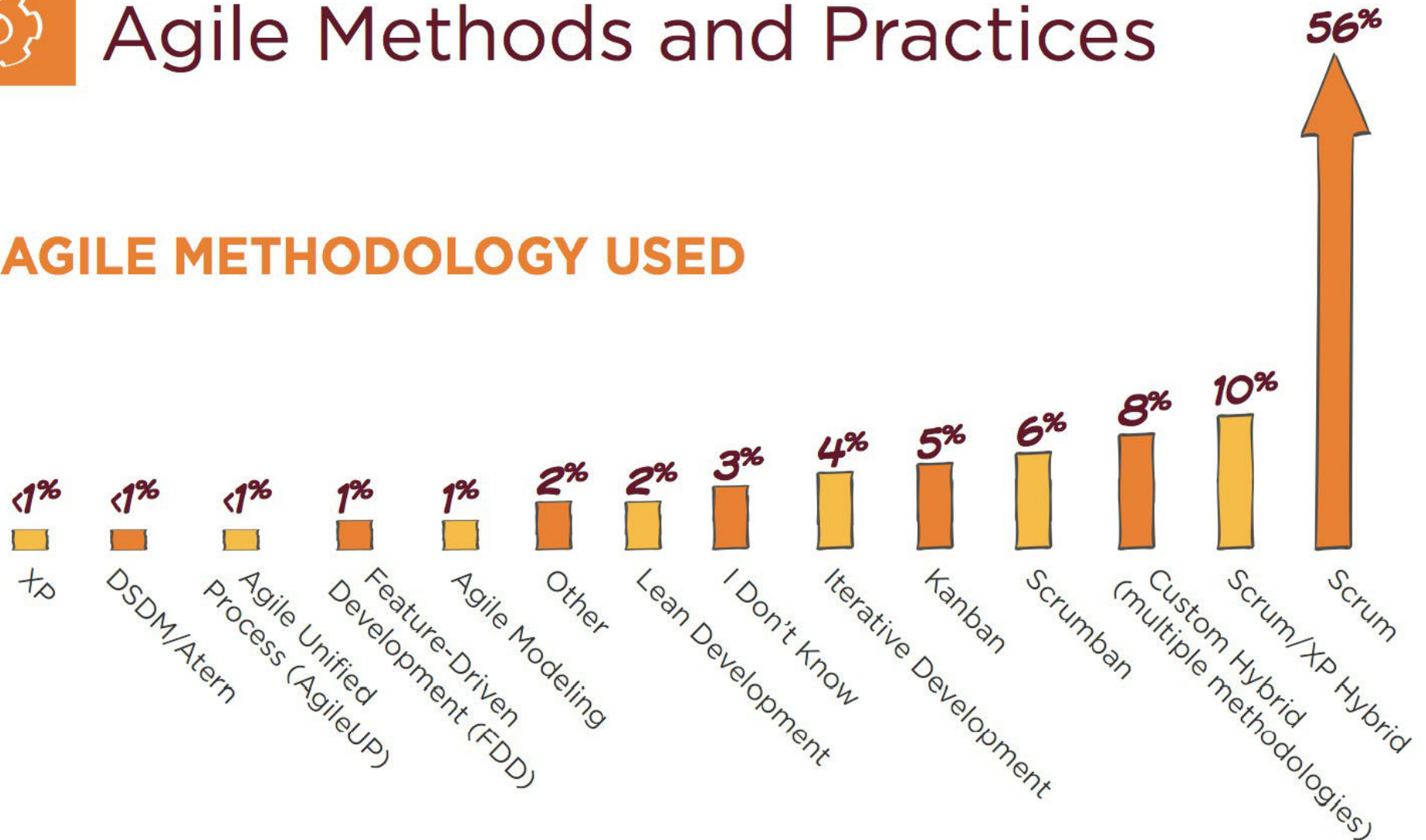
- Forrester surveyed (2009) nearly 1,300 IT professionals and found that **35 percent of respondents stated that agile most closely reflects their development process**
 - <http://www.infoworld.com/d/developer-world/agile-software-development-now-mainstream-190>
- **Agile methodologies are the primary approach for 39 percent** of responding developers, making Agile development the dominant methodology in North America. **Waterfall development, is the primary methodology of 16.5 percent** of respondents (2010)
 - <http://visualstudiomagazine.com/articles/2010/03/01/developers-mix-and-match-agile-approaches.aspx>
- Agile on Suomessa suosittua:
 - The results of the survey reveal that a majority of respondents' **organizational units are using agile and/or lean methods (58%)**
 - Markkula ym.: Survey on Agile and Lean usage in Finnish software industry, ESEM 2012 (ks. ACM digital library)
 - http://esem.cs.lth.se/industry_public/Rodriguezetal_ESEM2012_IndustryTrack_1_0.pdf

Mitä ketteriä menetelmiä käytetään?



Agile Methods and Practices

AGILE METHODOLOGY USED



- VersionOnen "internetin virallisesta" vuosiraportista
 - <http://info.versionone.com/state-of-agile-development-survey-ninth.html>

Ketterät käytänteet

- VersionOne:

80%	Daily standup	38%	Open work area
79%	Short iterations	36%	Refactoring
79%	Prioritized backlogs	34%	Test-Driven Development (TDD)
71%	Iteration planning	31%	Kanban
69%	Retrospectives	29%	Story mapping
65%	Release planning	27%	Collective code ownership
65%	Unit testing	24%	Automated acceptance testing
56%	Team-based estimation	24%	Continuous deployment
53%	Iteration reviews	21%	Pair programming
53%	Taskboard	13%	Agile games
50%	Continuous integration	9%	Behavior-Driven Development (BDD)
48%	Dedicated product owner		
46%	Single team (integrated dev & testing)		
43%	Coding standards		

- Suomen tilanne:

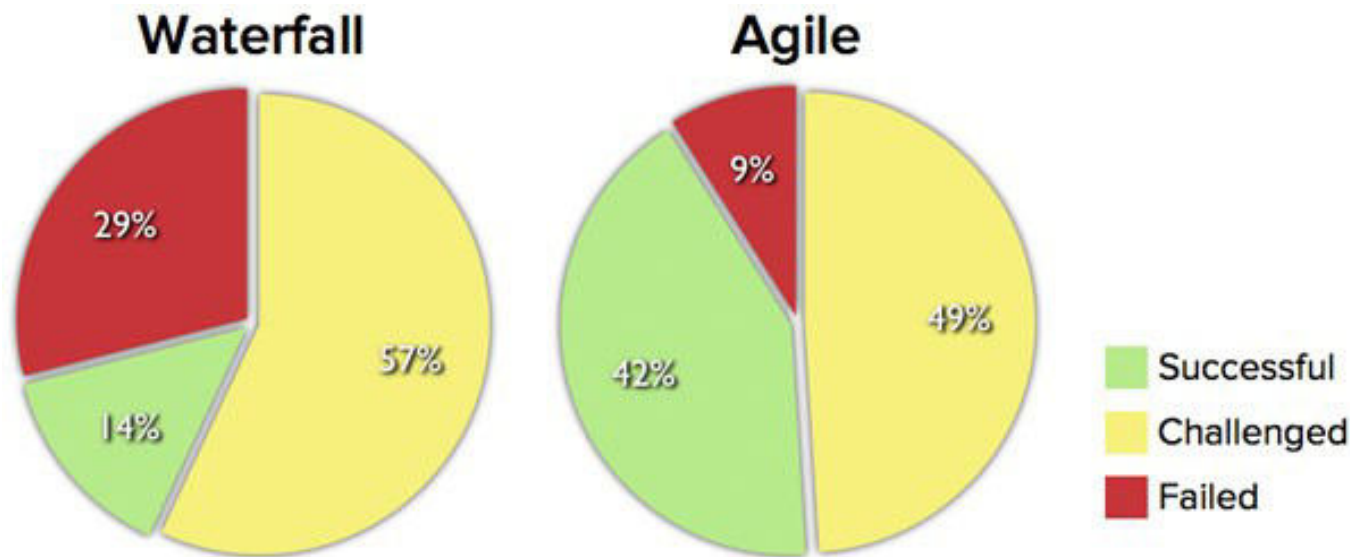
http://esem.cs.lth.se/industry_public/Rodriguezetal_ESEM2012_IndustryTrack_1_0.pdf

Ketterät käytänteet Suomesta tehdyssä tutkimuksessa (n=225)

Practices	n	Mean	Median
Prioritized work list	204	4,2	4
Iteration/sprint planning	203	4,1	4
Daily stand-up meetings	209	3,7	4
Unit testing	199	3,7	4
Release planning	196	3,9	4
Active customer participation	196	3,5	4
Self-organizing teams	194	3,5	4
Frequent and incremental delivery of working software	189	4,1	4
Automated builds	185	3,5	4
Continuous integration	182	3,8	4
Test-driven development (TDD)	179	2,7	3
Retrospectives	177	3,6	4
Burn-down charts	174	3,2	3
Pair programming	174	2,4	2
Refactoring	163	3,4	3
Collective code ownership	159	3,3	3

Projektien onnistuminen: ketterä vastaan perinteinen

- Standish CHAOS raport 2012



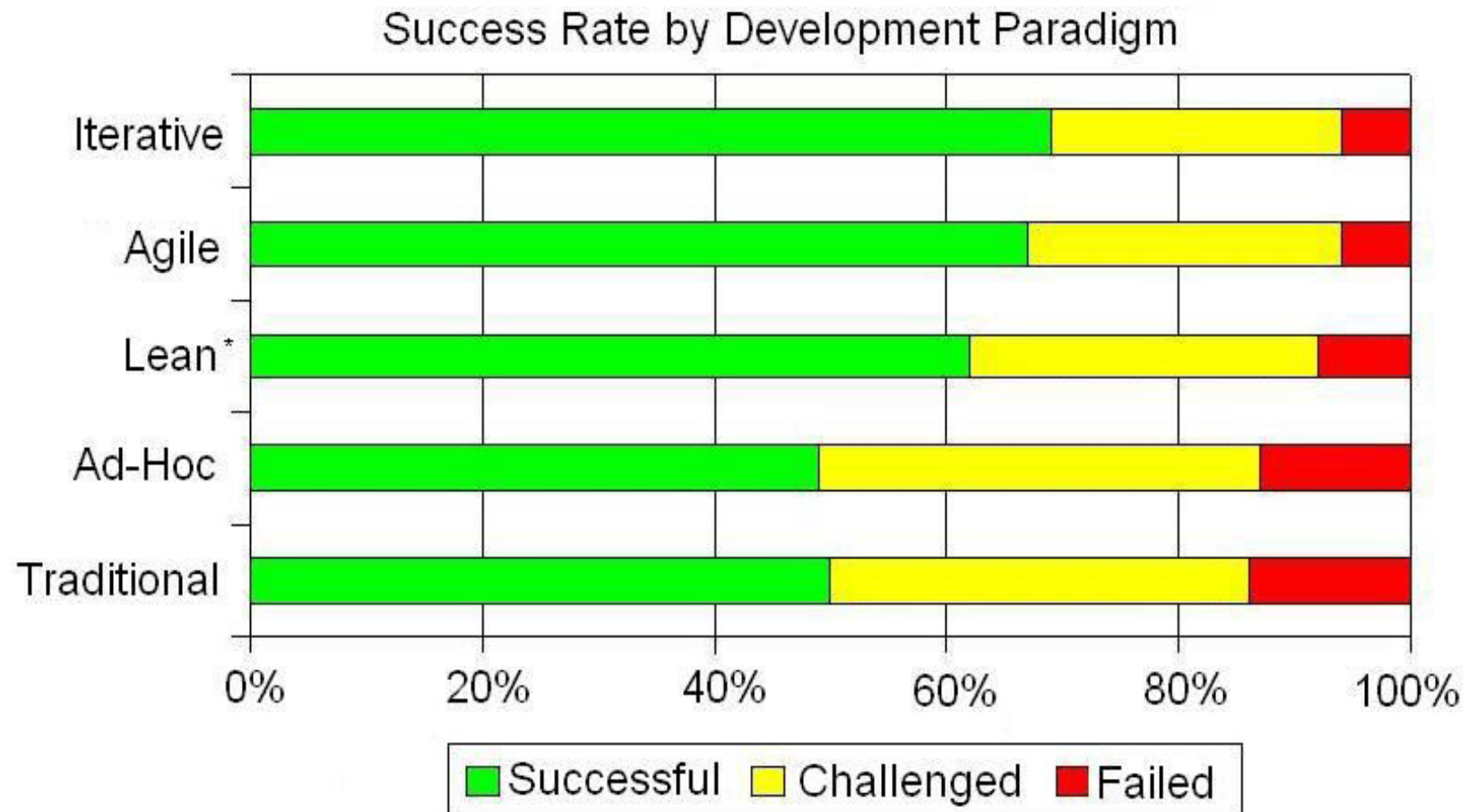
Source: The CHAOS Manifesto, The Standish Group, 2012.

- Columbus discovering Agile, laaja kyselytutkimus, alustavia tuloksia
 - Early results from the Columbus-area participants show that a typical business system comprising 50,000 lines of code is **completed 31% faster** than the industry average in the QSM industry database of completed projects. Even more remarkable is the **defect rate, which is 75% lower** than the industry norm.
 - <http://www.infoq.com/news/2012/11/success-agile-projects>

Projektien onnistuminen: ketterä vastaan perinteinen

- Scott Ambler, Agile vs perinteinen 2011:

<http://www.drdoobbs.com/architecture-and-design/how-successful-are-it-projects-really/232300110>



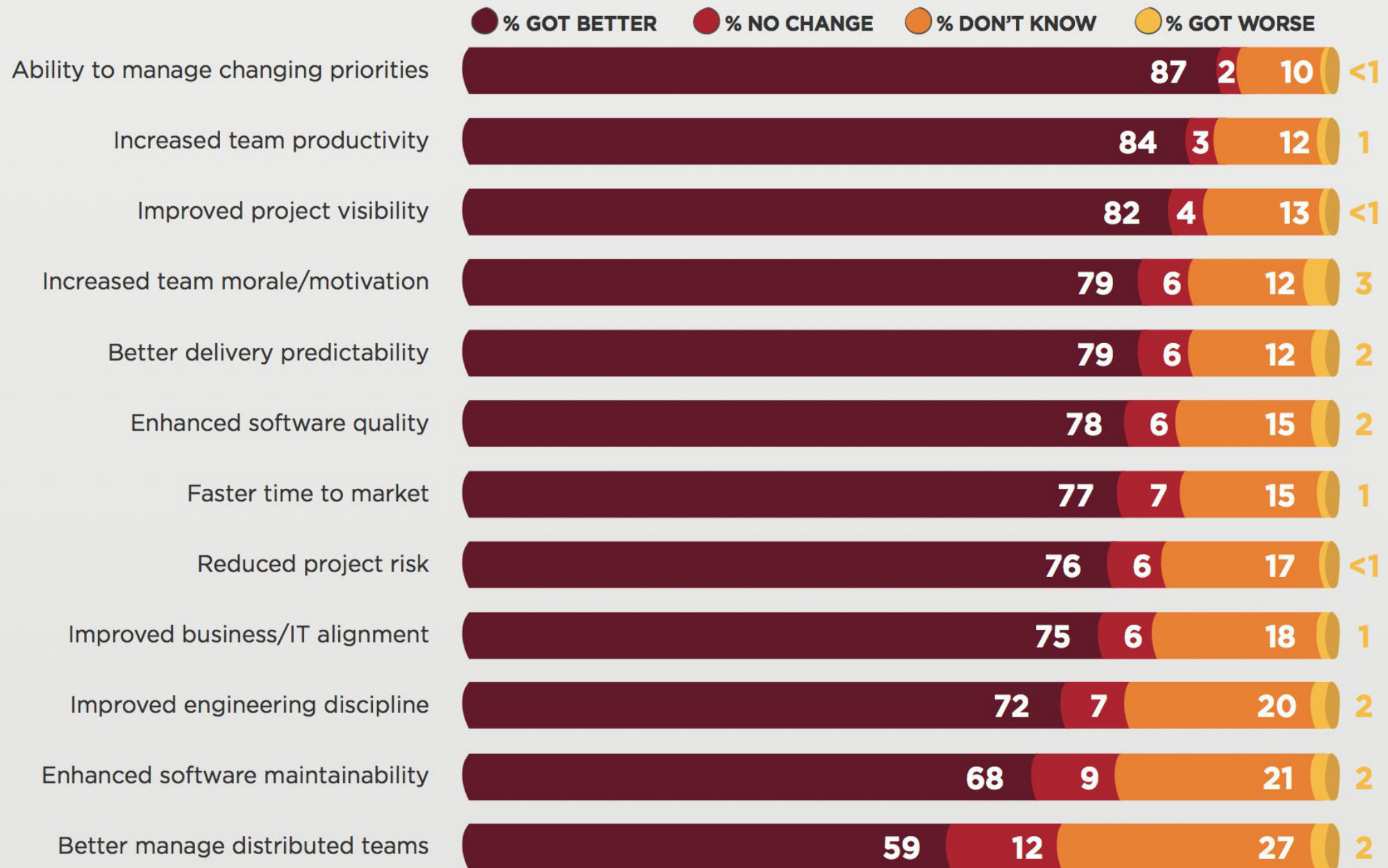
2011 is the first year where we asked about Lean.
We only had 40 respondents for this paradigm.

Mitä oikeastaan tarkoitetaan projektin onnistumisella?

- Ambler: Here's how respondents, on average, define success:
 - **Time/schedule:** 20% prefer to deliver on time according to the schedule, 26% prefer to deliver when the system is ready to be shipped, and 51% say both are equally important.
 - **Return on investment (ROI):** 15% prefer to deliver within budget, 60% prefer to provide good return on investment (ROI), and 25% say both are equally important.
 - **Stakeholder value:** 4% prefer to build the system to specification, 80% prefer to meet the actual needs of stakeholders, and 16% say both are equally important.
 - **Quality:** 4% prefer to deliver on time and on budget, 57% prefer to deliver high-quality systems that are easy to maintain, and 40% say both are equally important.

Ketteryydellä saavutettuja etuja tarkemmin eriteltynä

- VersionOne 2014



Ketteryydellä saavutettuja etuja Suomessa...

Effect	n	Mean	Median
Improved team communication	204	4,0	4
Enhanced ability to adapt to changes	203	3,9	4
Increased productivity	201	3,8	4
Enhanced process quality	198	3,7	4
Improved learning and knowledge creation	197	3,7	4
Enhanced software quality	196	3,8	4
Accelerated time-to-market/cycle time	192	3,7	4
Reduced waste and excess activities	190	3,5	4
Improved customer collaboration	190	3,7	4
Improved organizational transparency	187	3,5	4
Improved customer understanding	188	3,7	4

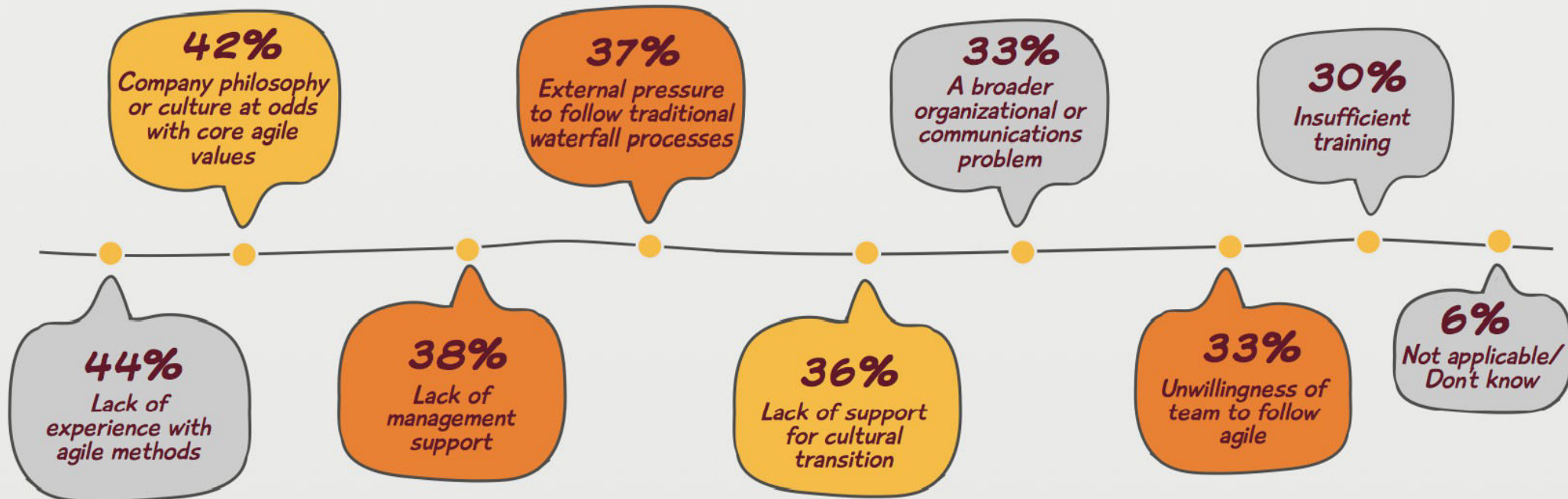
Yleisimpiä syitä ketterien projektien epäonnistumiseen

- VersionOne 2014

LEADING CAUSES OF FAILED AGILE PROJECTS

In cases where agile projects were unsuccessful, most respondents pointed to lack of experience with agile methods (**44%**). Of note, two of the top five causes of failure were related to company culture – company philosophy or culture at odds with core agile values at **42%** and lack of support for cultural transition at **36%**.

*Respondents were able to make multiple selections.



Evidenssiä on, mutta...

- Oikeastaan kaikki edelliset olivat kyselytutkimuksia
 - käsitteitä ei ole kunnolla määritelty (esim. mitä ketteryydellä tai projektin onnistumisella tarkoitetaan)
 - Kyselyyn osallistuneet eivät välttämättä edusta tasaisesti koko populaatiota
 - Kaikkien kyselyjen tekijät eivät puolueettomia menetelmien suhteen (esim. Ambler ja VersionOne)
- Eli tutkimusten validitetti on kyseenalainen
- Toisaalta kukaan ei ole edes yrittänyt esittää evidenssiä, jonka mukaan vesiputousmalli toisi systemaattisia etuja ketteriin menetelmiin verrattuna
- Myös akateemista tutkimusta on todella paljon (mm. Markkulan ym. kyselytutkimus) ja eri asioihin kohdistuvaa. Akateemisenkin tutkimuksen systemaattisuus, laatu ja tulosten yleistettävyys vaihtelee
 - Ohjelmistotuotannossa on liian paljon muuttujia, jotta jonkin yksittäisen tekijän vaikutusta voitaisiin täysin vakuuttavasti mitata empiirisesti
 - Menetelmiä soveltavat kuitenkin aina ihmiset, ja mittaustulos yhdellä ohjelmistotiimillä ei välttämättä yleisty mihinkään muihin olosuhteisiin
- Olemassa olevan evidenssin nojalla kuitenkin näyttää siltä, että ongelmistaan huolimatta ketterät menetelmät ovat ainakin joissakin tapauksissa järkevä tapa ohjelmistokehitykseen

Koe

Koe

- Tiistaina 5.5 klo 16:00 – salissa A111
- Kurssin pisteytys
 - Koe 20p
 - Laskarit 10p
 - Miniprojekti 10p
- Kurssin läpikäyty edellyttää
 - 50% pisteistä
 - 50% kokeen pisteistä
 - Hyväksyttyä miniprojektia
- Kokeessa on sallittu yhden A4:n kokoinen käsin, itse kynällä kirjoitettu lunttilappu

Mitä kokeessa **ei** tarvitse osata

- Git
- Maven
- Jenkins
- JUnit
- Mockito
- EasyB
- Selenium
- Ebean
- Java 8

Reading list – eli lue nämä

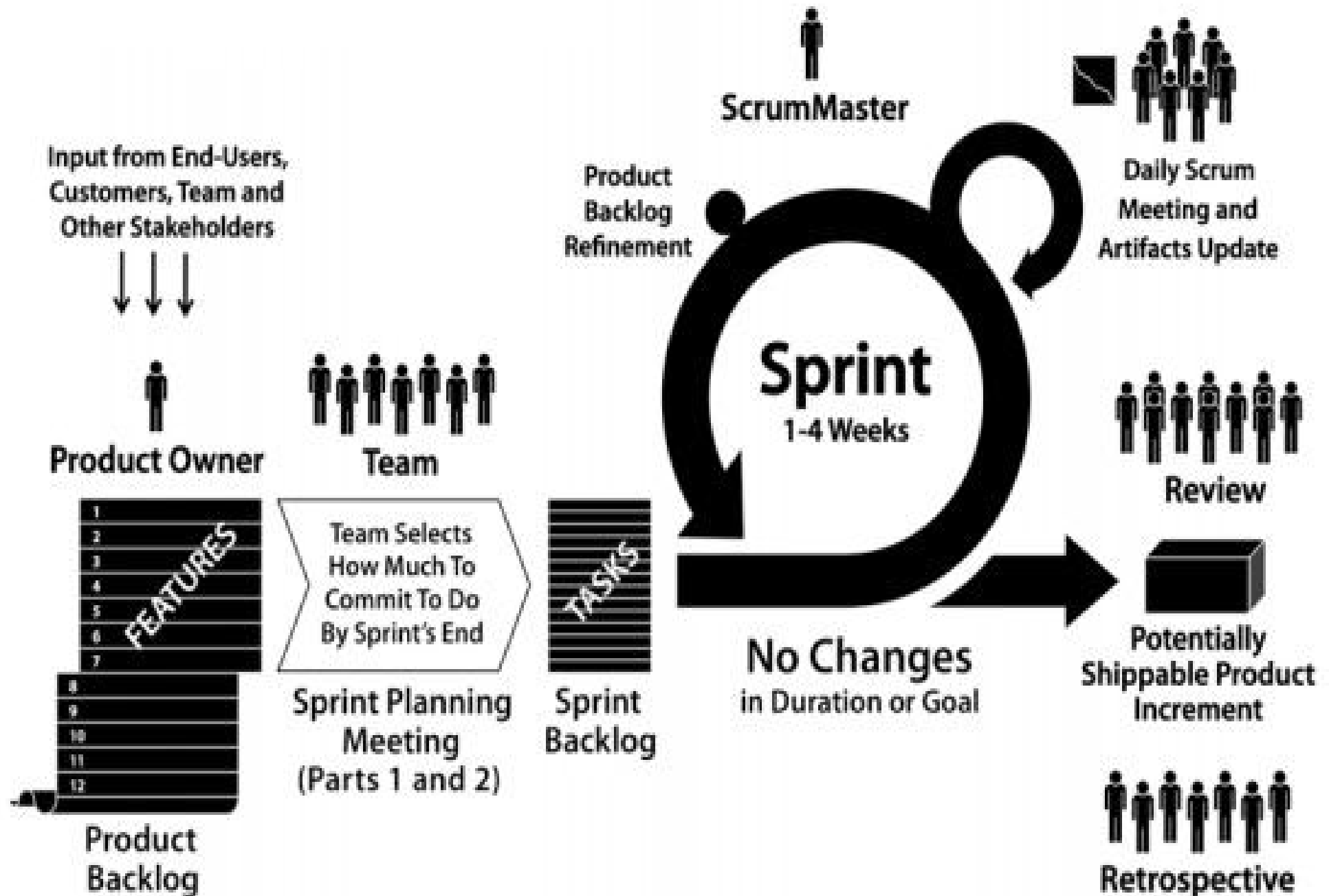
- Luentomonisteet, luentoihin 8 ja 9 liittyvät koodiesimerkit ja laskarit (paitsi edellisellä sivulla mainittujen osalta)
- <http://martinfowler.com/articles/newMethodology.html>
- http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
- <http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>
 - Sivut 1-86
- <http://martinfowler.com/articles/continuousIntegration.html>
- <http://martinfowler.com/articles/designDead.html>
- http://sourcemaking.com/design_patterns
 - Tarpeellisissa määrin

Tärkeitä teemat vielä pikakelauksella

Luento 1

- Termi software engineering
 - Mitä pitää sisällään
- Prosessimallit
 - Vaiheet
 - Vaatimusmäärittely
 - Suunnittelu
 - Toteutus
 - Testaus
 - Ylläpito
 - vesiputous/lineaarinen/BUFD
 - Iteratiivinen
 - Ketterä
- Motivaatio prosessimallien kehittymiselle

Luento 2: Scrum



Luento 3: vaatimusmäärittely

- Vaatimukset jakautuvat
 - Toiminnallisiin
 - Ei-toiminnallisiin (rajoitteet ja laatuvaatimukset)
- Vaatimusmäärittelyn luonne ja vaiheet
 - oldschool vs. moderni
- Ketterä vaatimustenhallinta
 - User story
 - Arvoa tuottava toiminnallisuus
 - "Card, conversation, confirmation"
 - INVEST
 - Estimointi

Luento 4

- Ketterä vaatimustenhallinta
 - Product backlog
 - DEEP
 - Julkaisun suunnittelu
 - Velositeetti
- Sprintin suunnittelu
 - Storyjen valinta / planning game
 - Storyistä taskeihin
- Sprint backlog
 - Taskboard
 - burndown

Luento 5

- Validointi "are we building the right product"
 - Katselmointi ja tarkastukset
 - Vaatimusten validointi (ketterä vs. trad)
 - Koodin katselmointi
- Verifiointi "are we building the product right"
 - Vastaako järjestelmä vaatimusmäärittelyä
- Verifiointi tapahtuu yleensä testauksen avulla
 - Testauksen tasot:
 - Yksikkö-, Integraatio-, Järjestelmä-, Hyväksymätestaus
 - Käsitteitä:
 - black box, white box, ekvivalenssiluokka, raja-arvo, testauskattavuus
 - regressiotestaus
 - Ohjelman ulkoinen laatu vs. Sisäinen laatu

Luento 6

- Testaus ketterissä menetelmissä
 - Automaattiset regressiotestit tärkeät
- TDD
 - Red – green – refactor
 - Enemmän suunnittelua kun testausta, testit sivutuotteena
- Storytason testaus / ATDD / BDD
- Jatkuva integraatio
 - ”integraatiohelvetti” → Daily build / smoke test → jatkuva integraatio → continuous delivery
 - Workflow jatkuvassa integraatiossa
 - Continuous delivery ja Continuous deployment
- Seuraava käsiteltiin oikeastaan vasta luennolla 7
- Tutkiva testaus
 - ”Exploratory testing is simultaneous learning, test design and test execution”

Luento 7

- Ohjelmiston arkkitehtuurin määritelmiä
- Arkkitehtuurimallit: kerrosarkkitehtuuri
- Arkkitehtuurin kuvaaminen
 - Monia näkökulmia, erilaisia kaavioita
- Arkkitehtuuri ketterissä menetelmissä
 - Ristiriita arkkitehtuurivetoisuuden ja ketterien menetelmien välillä
 - Inkrementaalinen arkkitehtuuri
 - Edut ja haitat

Luento 8 – oliosuunnittelu

- Helposti ylläpidettävän eli sisäiseltä laadultaan hyvän koodin tunnusmerkit ja laatuattribuutit
 - kapselointi, koheesio, riippuvuuksien vähäisyys, toisteettomuus, selkeys, testattavuus
- Oliosuunnittelun periaatteita
 - Single responsibility principle
 - Program to an interface not to an implementation
 - Favour composition over inheritance
 - DRY eli Don't repeat yourself
- Suunnittelumalleja
 - Composed method
 - Static factory
 - Strategy
 - Command
 - Template method

Luento 9

- Suunnittelumalleja
 - Dekoraattori
 - Rakentaja (builder)
 - Adapteri
 - Komposiitti
 - Proxy
 - Mvc
 - Observer
- Aiemmin kurssilla kolme suunnittelumallia
 - Riippuvuuksien injektointi (dependency injection)
 - Singleton
 - DAO, Data access object
- Domain driven design ja kerrosarkkitehtuuri
- Käsitteet tekninen velka technical/design debt, koodihaju ja refaktorointi