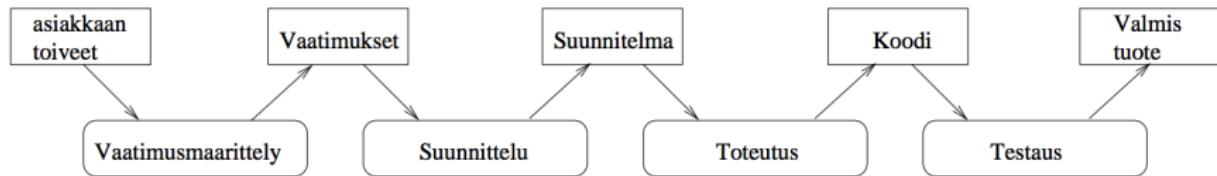


Ohjelmistotuotanto

Matti Luukkainen ja ohjaajat Kalle Ilves, Petri Suhonen, Oskari
Nuottonen, Tuukka Puonti

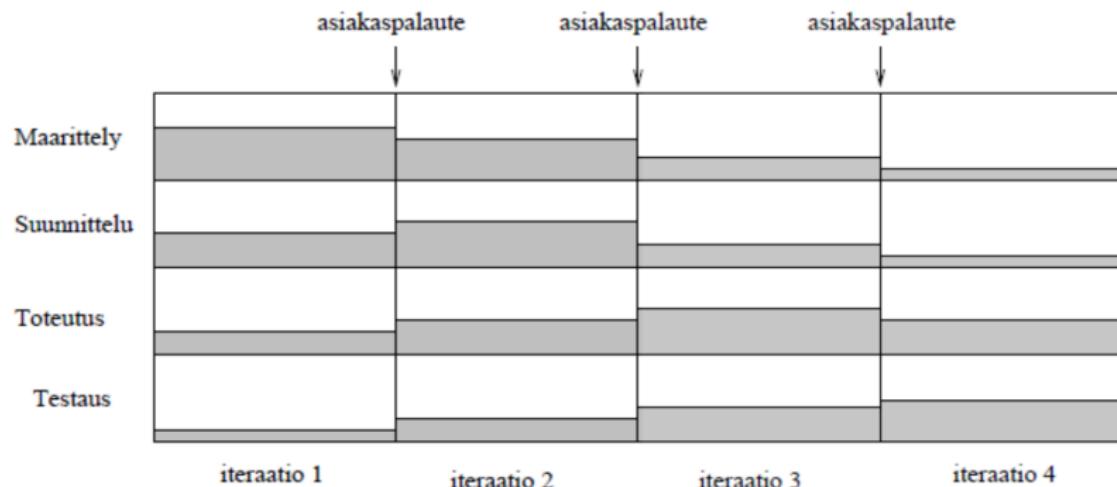
syksy 2022

Vesiputousmalli



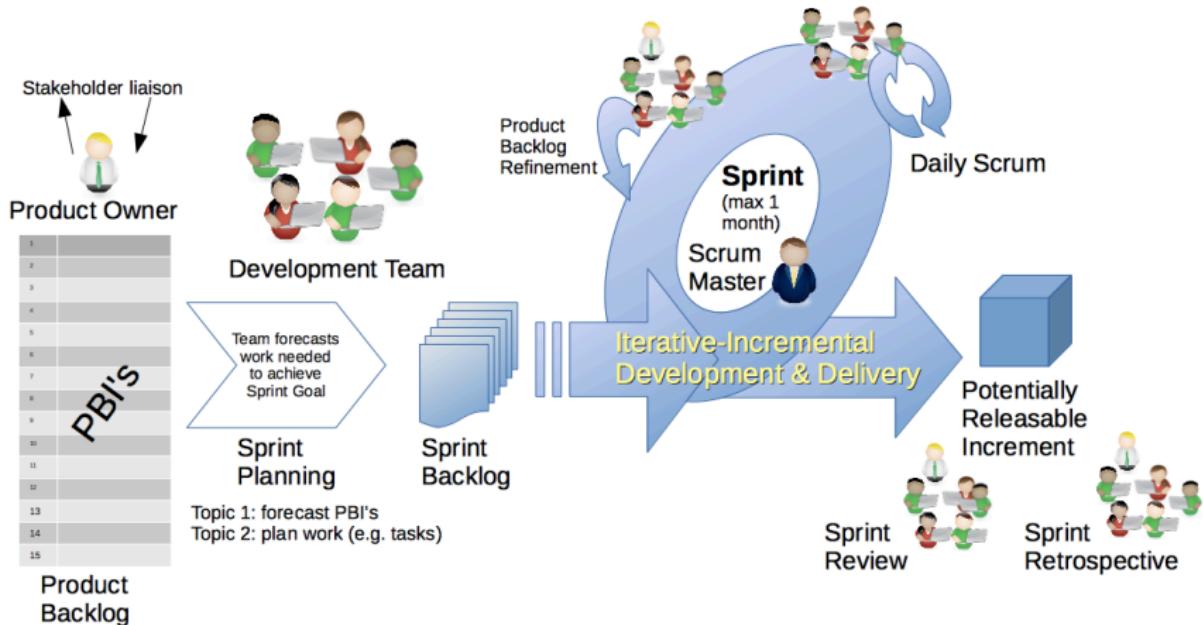
luento 1

Iteratiivinen ohjelmistokehitys



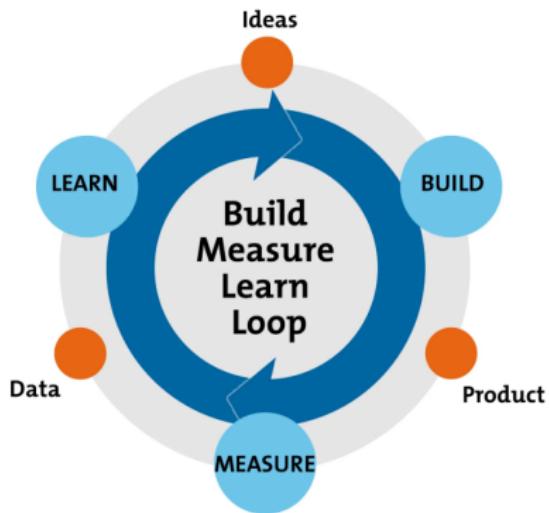
luento 2

Scrum



luento 3

Vaatimusmäärittely 2010-luvulla: Lean startup

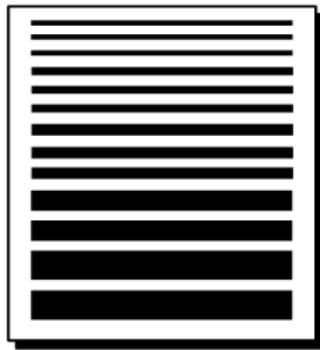


User story

- ▶ Mike Cohn:
 - ▶ *A user story describes functionality that will be valuable to either user or purchaser of software.*
- ▶ User stories are composed of three aspects:
 1. A written description of the story, used for planning and reminder
 2. Conversations about the story to serve to flesh the details of the story
 3. Tests that convey and document details and that will be used to determine that the story is complete

luento 4

High
Priority

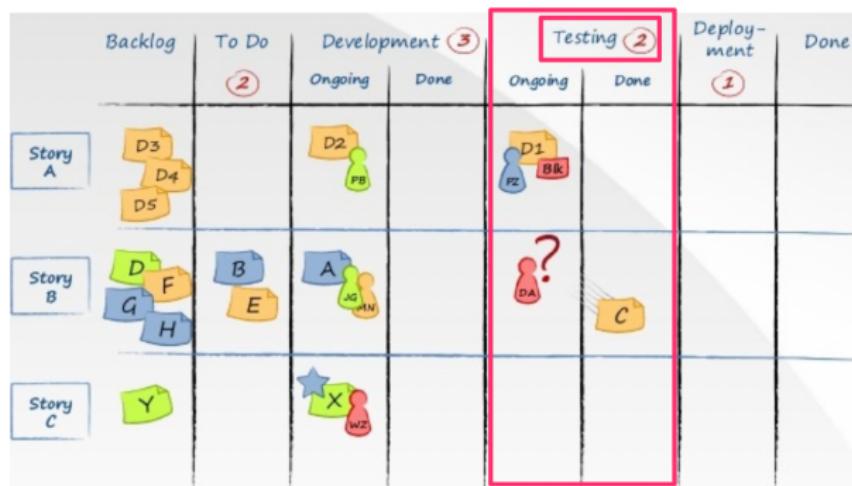


Fine-grained, detailed items ready
to be worked on in the next sprint

Low
Priority

Large, coarse-grained items

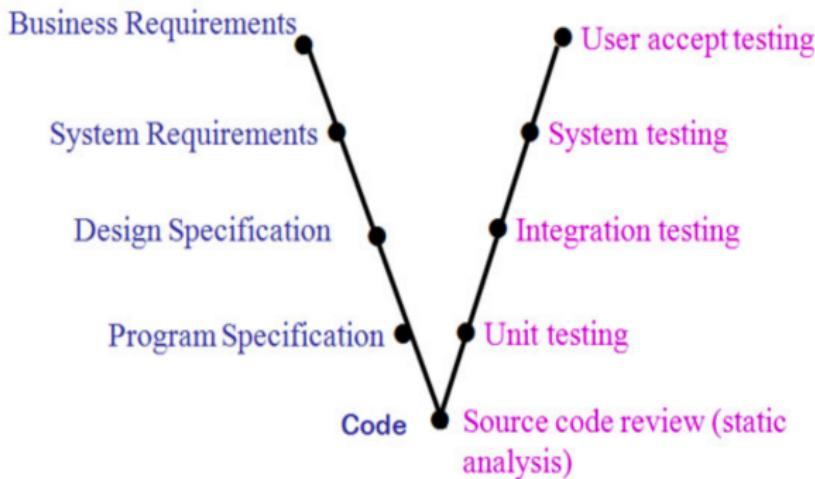
WIP-rajoitteet



PBI	TASKS	IN PROGRESS	BLOCKED	DONE
1. Add PBI to scope				✓
2. PRD and SPRINT				✓
3. Add PBI descriptions & complexity				✓
4. PRD				✓
5. Define PBI priority	Sub: Create 2018 PRD (PPT slides and notes)	Sub: Summarize PRD (PRD notes)	Sub: Create PRD checklists	Sub: Create PRD checklist
6. Planning PBI backlog	Sub: Create PRD backlog	Sub: Refactor PRD backlog	Sub: Refactor PRD backlog	Sub: Refactor PRD backlog
7. Create PRD backlog	Sub: Create PRD backlog	Sub: Refactor PRD backlog	Sub: Refactor PRD backlog	Sub: Refactor PRD backlog
8. PRD				✓

luento 5

- ▶ *Yksikkötestaus* (unit testing)
- ▶ *Integraatiotestaus* (integration testing)
- ▶ *Järjestelmätestaus* (system testing)
- ▶ *Käyttäjän hyväksymistestaus* (user acceptance testing)



Ohtuvarasto: tyhjä, puolitäysi, täysi

```
class Varasto:
    def __init__(self, tilavuus, alkusalto = 0):
        self.tilavuus = tilavuus
        self.saldo = alkusalto

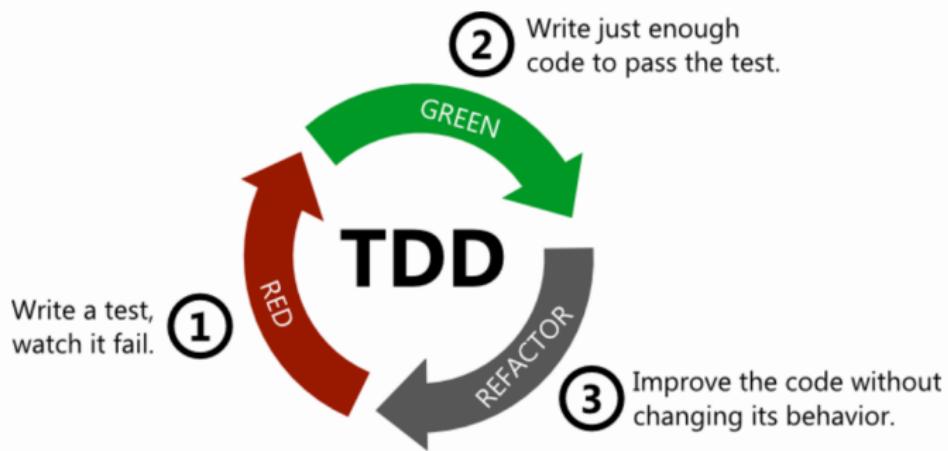
    def ota_varastosta(self, maara):
        if maara < 0:
            return 0.0

        if maara > self.saldo:
            kaikki_mita_voidaan = self.saldo
            self.saldo = 0.0
            return kaikki_mita_voidaan

        self.saldo = self.saldo - maara
        return maara
```

luento 6

Test driven development (TDD)



Riippuvuudet testeissä: dependency injection

```
class Laskin:
    def __init__(self, io):
        self._io = io

    def suorita(self):
        while True:
            lukul = int(self._io.lue("Luku 1:"))

            if lukul == -9999:
                return

            luku2 = int(self._io.lue("Luku 2:"))

            if luku2 == -9999:
                return

            vastaus = self._laske_summa(lukul, luku2)

            self._io.kirjoita(f"Summa: {vastaus}")

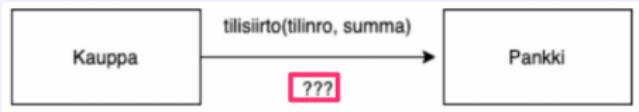
    def _laske_summa(self, lukul, luku2):
        return lukul + luku2
```

Mock-kirjastot

- ▶ Kaupan metodin *maksa* pitää tehdä *tilisiirto* kutsumalla *Pankin* metodia

```
my_net_bank = Pankki()
viitteet = Viitegeneraattori()
kauppa = Kauppa(my_net_nank, viitteet)

kauppa.aloita_ostokset()
kauppa.lisaa_ostos(5)
kauppa.lisaa_ostos(7)
kauppa.maksa("1111")
```



Testit asiakkan kielellä

```
*** Test Cases ***
Register With Valid Username And Password
    Input Credentials  kalle  kalle123
    Output Should Contain  New user registered

Register With Already Taken Username And Valid Password
    Input Credentials  kalle  kalle123
    Input Register Command
    Input Credentials  kalle  foobarfoo1
    Output Should Contain  User with username kalle already exists

Register With Too Short Username And Valid Password
    Input Credentials  k  kalle123
    Output Should Contain  Username too short

Register With Valid Username And Too Short Password
    Input Credentials  kalle  k
    Output Should Contain  Password too short
```

- ▶ *Input Credentials, Output should contain ym avainsanoja*

