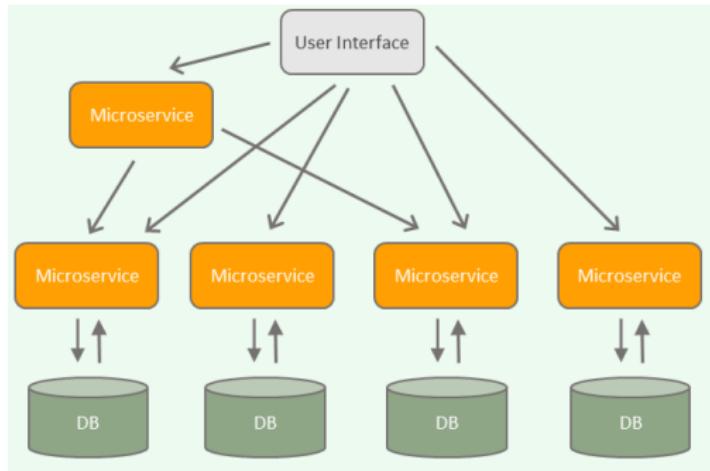


Luento 8

19.11.2024

Mikropalveluarkkitehtuuri



- ▶ sovellus koostataan useista (jopa sadoista) pienistä verkossa toimivista autonomisista palveluista

Laadukkaan koodin tuntomerkejä

- ▶ Laadukkaalla koodilla joukko yhteneviä ominaisuuksia, tai *laatuattribuutteja*, esim. seuraavat:
 - ▶ kapselointi
 - ▶ korkea koheesion aste
 - ▶ riippuvuuksien vähäisyys
 - ▶ toisteettomuus
 - ▶ testattavuus
 - ▶ selkeys

Koheesio Flask-sovelluksessa

```
@app.route("/create_todo", methods=["POST"])
def todo_creation():
    content = request.form.get("content")

    try:
        validate_todo(content)
        create_todo(content)
        return redirect("/")
    except Exception as error:
        flash(str(error))
        return redirect("/new_todo")

# util.py

def validate_todo(content):
    if len(content) < 5:
        raise UserInputError("Todo content length must be greater than 4")

    if len(content) > 100:
        raise UserInputError("Todo content length must be smaller than 100")

# todo_repository.py

def create_todo(content):
    sql = text("INSERT INTO todos (content) VALUES (:content)")
    db.session.execute(sql, { "content": content })
    db.session.commit()
```

Laadukkaan koodin tuntomerkejä

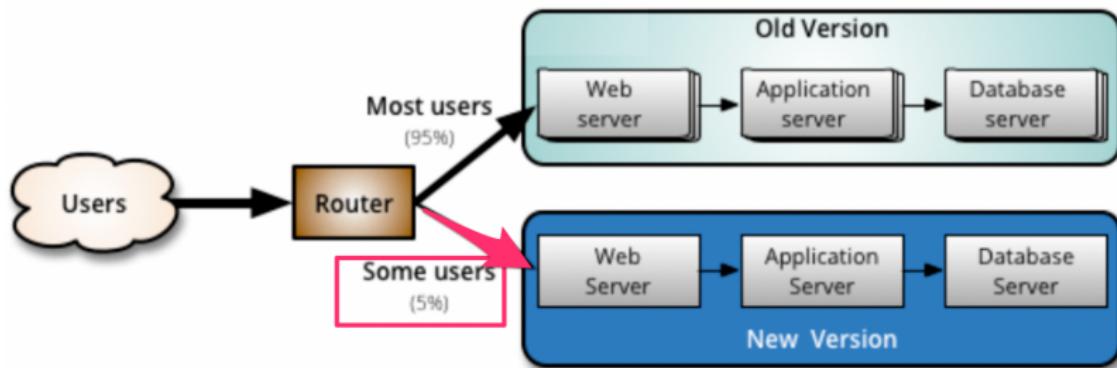
- ▶ Laadukkaalla koodilla joukko yhteneviä ominaisuuksia, tai *laatuatribuutteja*, esim. seuraavat:
 - ▶ kapselointi
 - ▶ korkea koheesion aste
 - ▶ riippuvuuksien vähäisyys
 - ▶ toisteettomuus
 - ▶ testattavuus
 - ▶ selkeys

Luento 7

18.11.2024

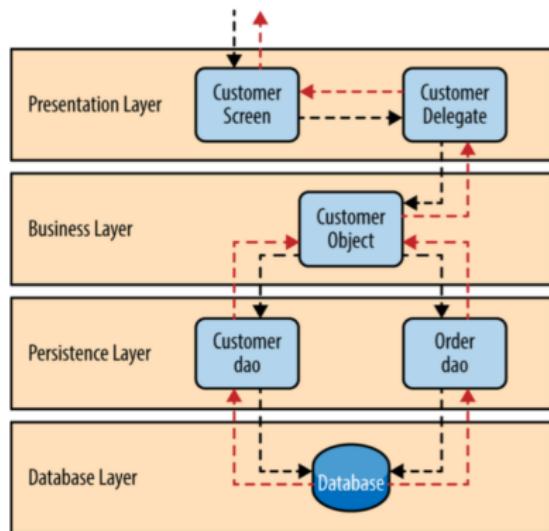
Canary release

- ▶ Kaksi rinnakkaista tuotanto-ympäristöä, joista uudet ominaisuudet viedään toiseen

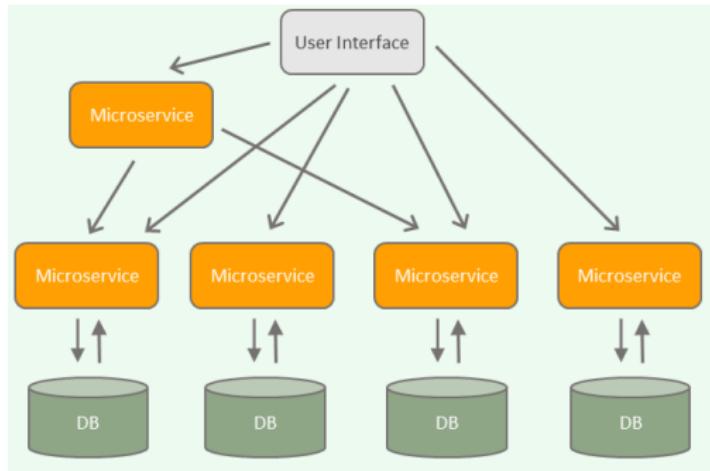


Kerrosarkkitehtuuri

- ▶ Kerros on kokoelma toisiinsa liittyviä olioita, jotka muodostavat toiminnallisuuden suhteen loogisen kokonaisuuden



Mikropalveluarkkitehtuuri



- ▶ sovellus koostataan useista (jopa sadoista) pienistä verkossa toimivista autonomisista palveluista

Luento 1

30.10.2024

Ohjelmiston elinkaari (software lifecycle)

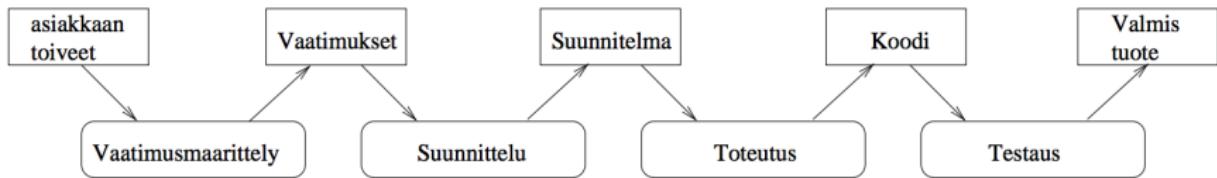
Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

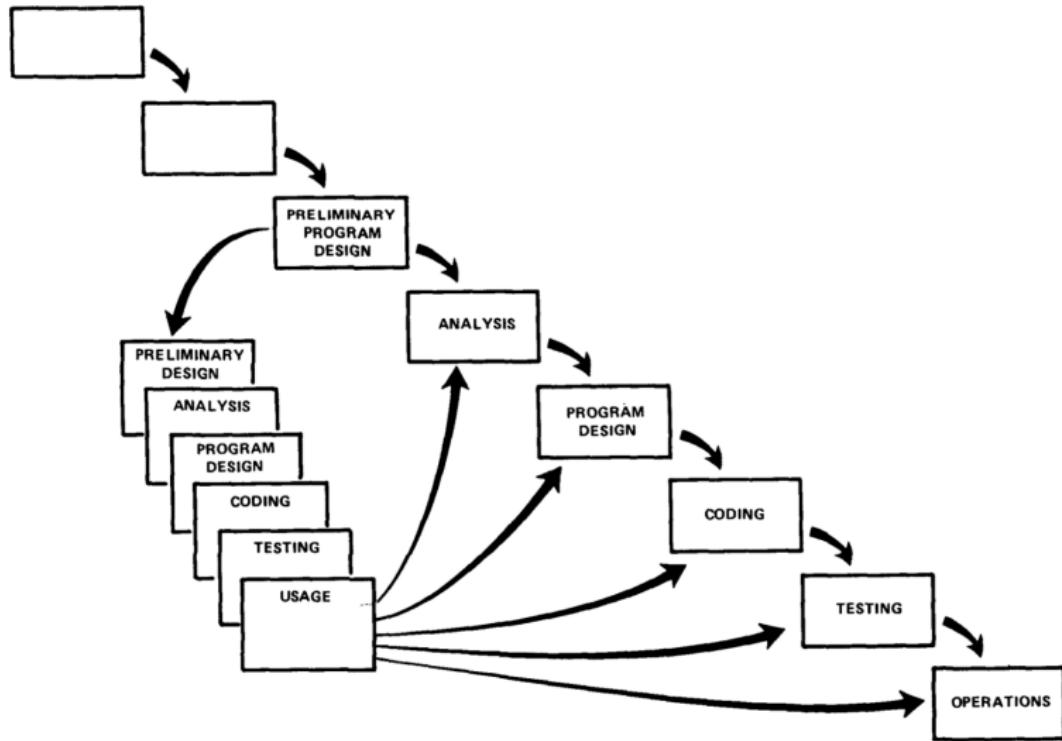
Vaiheista muodostuu ohjelmiston “elinkaari”

Vesiputousmalli

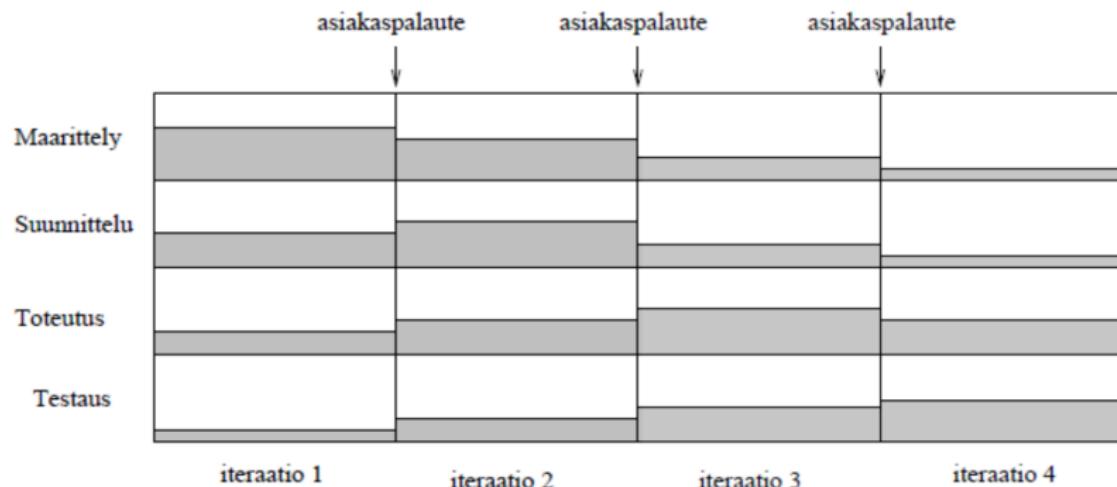
Winston W. Royce: Management of the development of Large Software, 1970



Roycen kahden iteraation malli



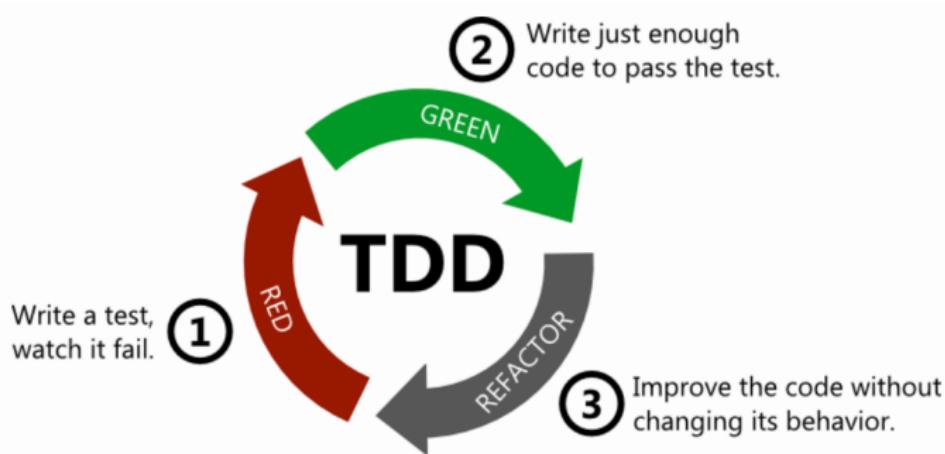
Iteratiivinen ohjelmistokehitys



Luento 6

12.11.2024

Test driven development (TDD)



1. Kirjoitetaan sen verran testiä että testi ei mene läpi
2. Kirjoitetaan koodia sen verran, että testi menee läpi
3. Jos huomataan koodin rakenteen menneen huonoksi refaktoroidaan koodin rakenne paremmaksi
4. Jatketaan askeleesta 1

Testit asiakkan kielellä

*** Test Cases ***

Run | Debug | Run in Interactive Console

Login With Correct Credentials

```
Set Username  kalle
Set Password  kalle123
Submit Credentials
Login Should Succeed
```

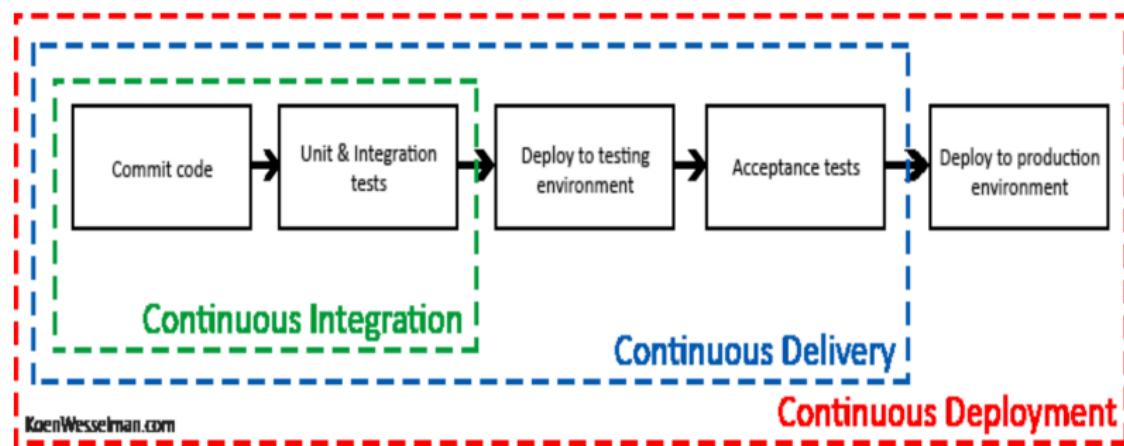
Run | Debug | Run in Interactive Console

Login With Incorrect Password

```
Set Username  kalle
Set Password  wrong
Submit Credentials
Login Should Fail With Message  Invalid username or password
```

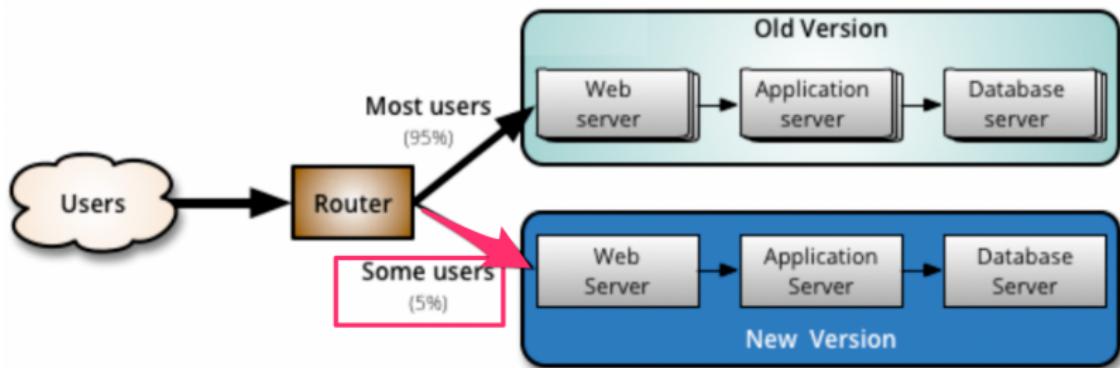
Deployment pipeline

- Vaiheet, joiden suorittaminen edellytetään, että commitattu koodi saadaan siirrettyä staging/tuotantojärjestöön



Canary release

- ▶ Kaksi rinnakkaista tuotanto-ympäristöä, joista uudet ominaisuudet viedään toiseen



Luento 5

11.11.2024

Testauksen tasot

- ▶ *Yksikkötestaus* (unit testing)
 - ▶ Yksittäisten luokkien, metodien ja moduulien testaus erillään muusta kokonaisuudesta
- ▶ *Integraatiotestaus* (integration testing)
 - ▶ Yksittäin testattujen komponenttien liittäminen yhteen eli integrointi ja kokonaisuuden testaus
- ▶ *Järjestelmätestaus* (system testing)
 - ▶ Toimiiko ohjelmisto vaatimuksiin kirjatulla tavalla?
 - ▶ Tutkii järjestelmää kokonaisuudessaan: *end to end -testaus*
 - ▶ Jakautuu useisiin alalajeihin
- ▶ *Käyttäjän hyväksymistestaus* (user acceptance testing)
 - ▶ Loppukäyttäjän tuotteelle suorittama testaus

Testisyötteiden valinta: palautussovellus

- Mitä testitapauksia kannattaisi valita palautussovelluksen testaamiseen?

Create a submission for part2

Mark all exercises you have done (check the box if the exercise is done)

1 2 3 4 5 6 7 8 9 10 11 12

Mark all

Clear all

Used hours (reading the material and completing exercises)

GitHub repository

https://github.com/mluukkai/put_your_repository_name_here

Comments

Pressing send will submit this whole part. Any exercises you have not marked done above for this part can not be marked done later. If you by accident submit the wrong number of exercises contact the course teacher or Discord admins.

Send

Cancel

Ohtuvarasto: tyhjä, puolitäysi, täysi

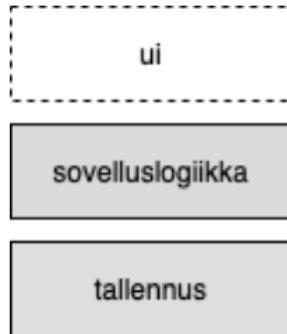
```
class Varasto:
    def __init__(self, tilavuus, alkusalto = 0):
        self.tilavuus = tilavuus
        self.saldo = alkusalto

    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
```

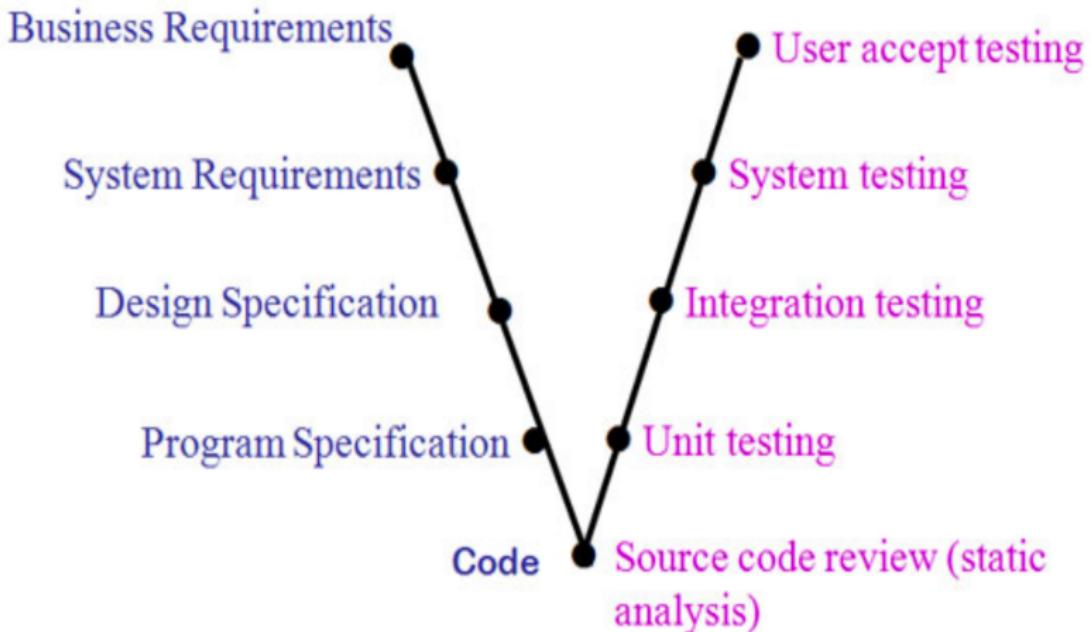
► Rakenteeseen perustuva integraatio



► Ominaisuksiin perustuva integraatio



“V-malli”

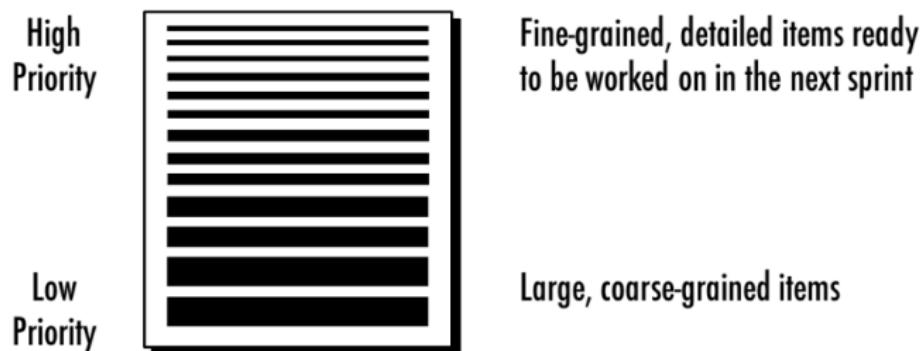


Luento 4

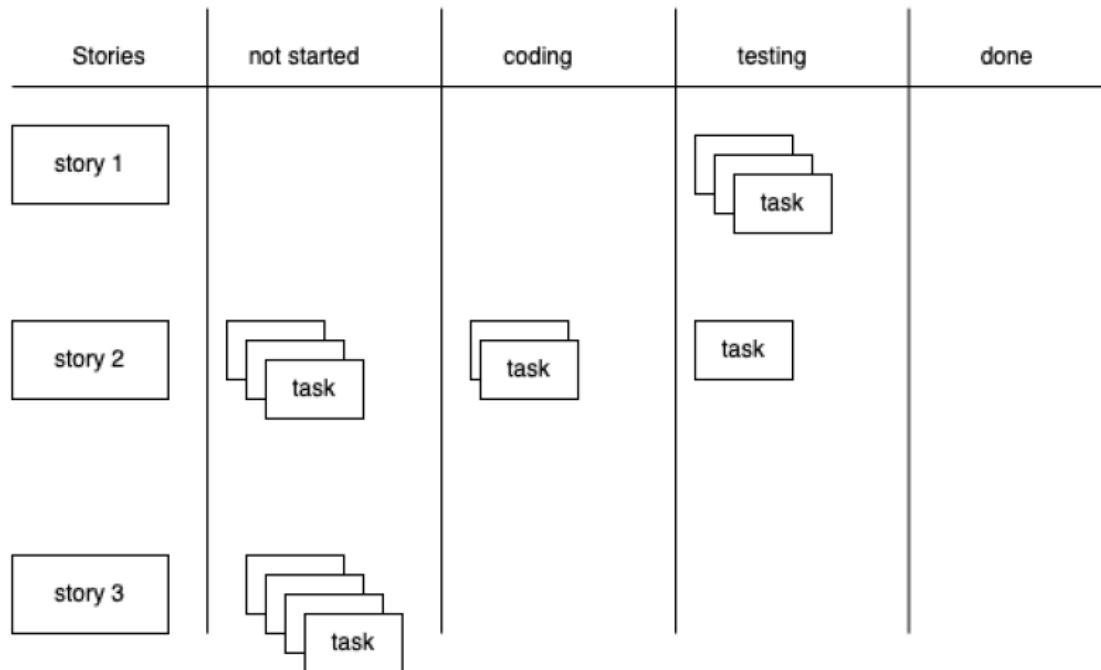
7.11.2024

Hyvä product backlog on DEEP

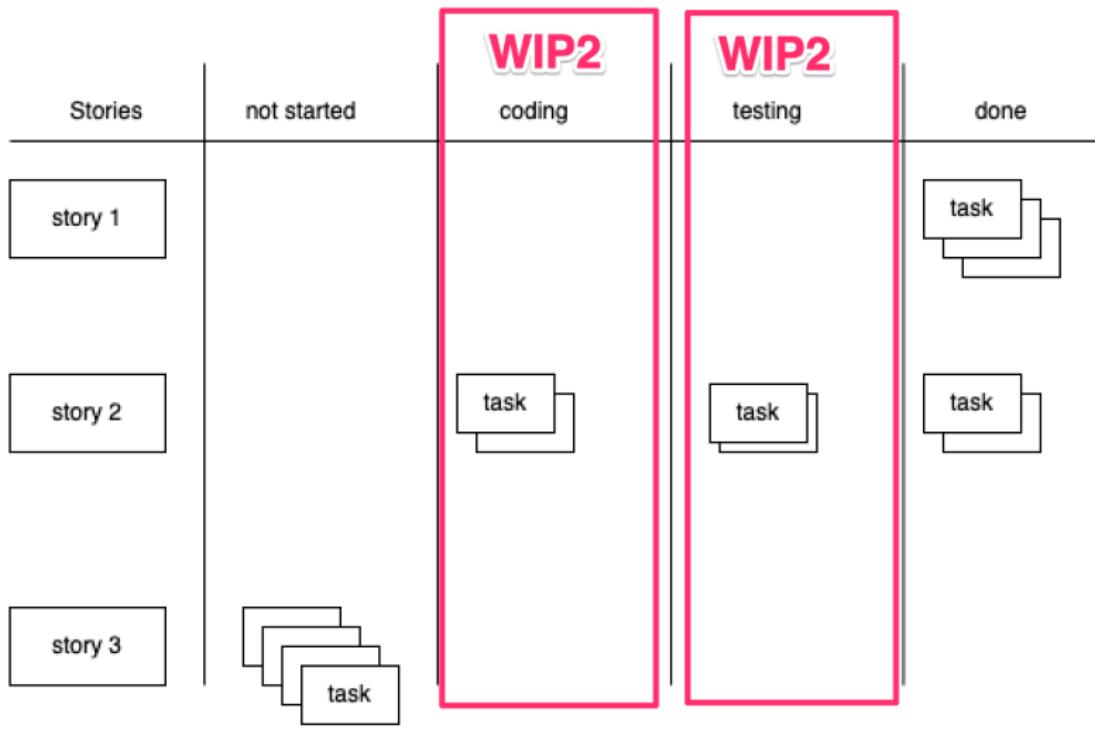
- ▶ Detailed appropriately
- ▶ Estimated
- ▶ Emergent
- ▶ Prioritized



Sprint backlog

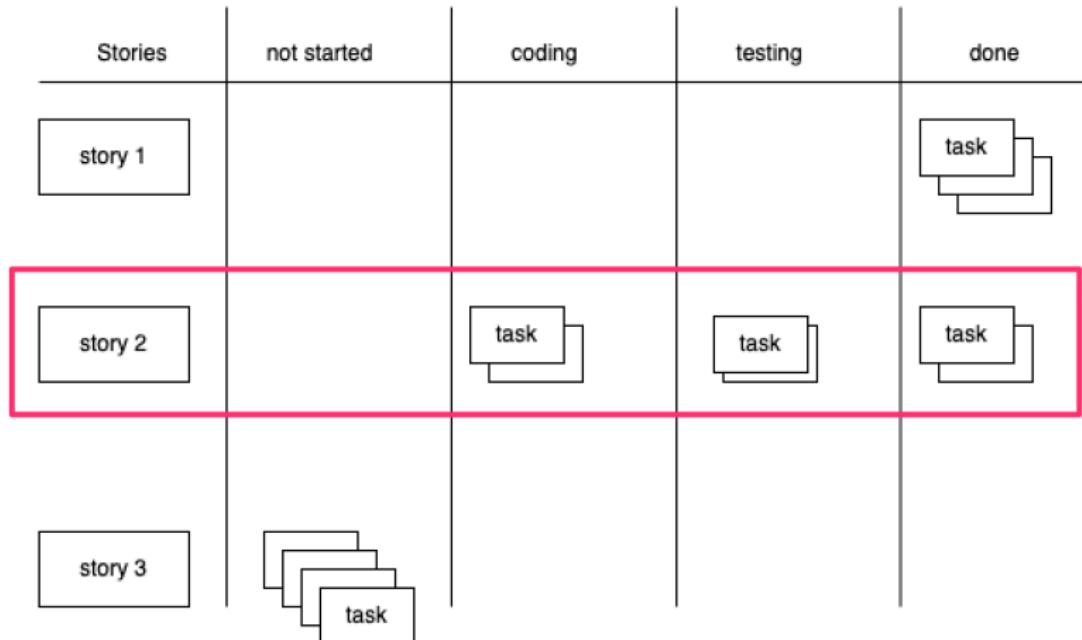


WIP-rajoitteet



WIP-rajoitteet

WIP vain yksi story työn alla

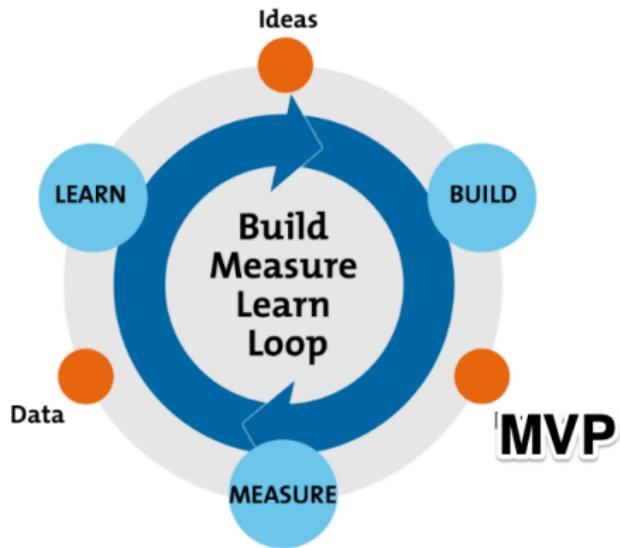


Luento 3

Ohjelmiston elinkaari (software lifecycle)

- ▶ **Vaatimusten analysointi ja määrittely**
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

Vaatimusmäärittely 2010-luvulla: Lean startup



User story

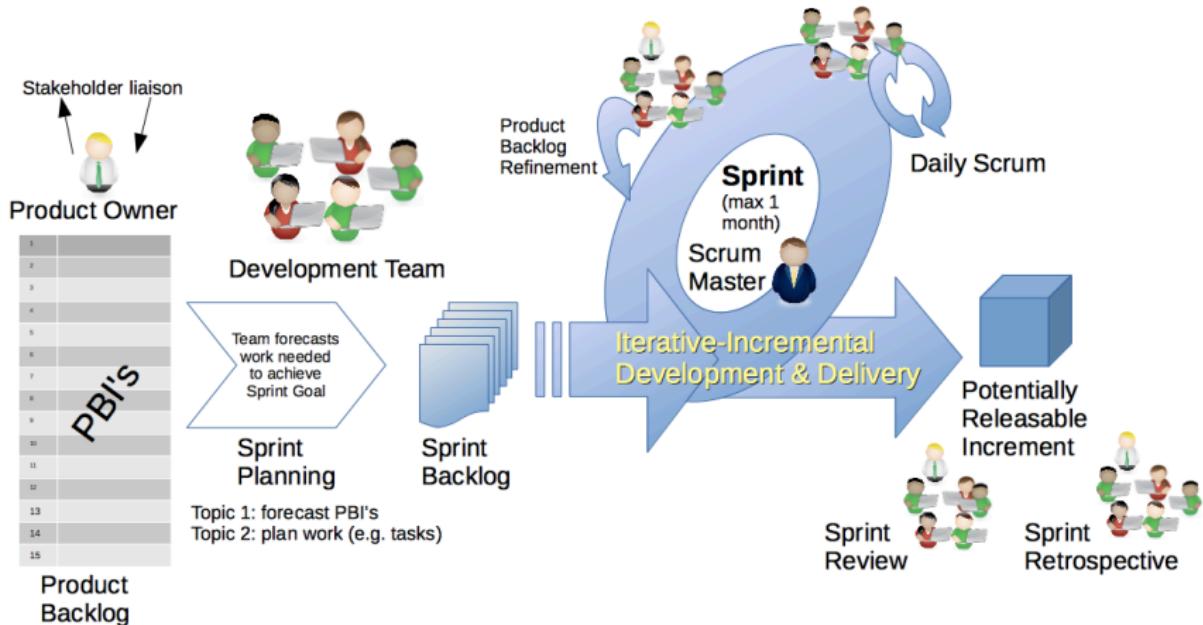
- ▶ Mike Cohn:
 - ▶ *A user story describes **functionality that will be valuable** to either user or purchaser of software.*
- ▶ User stories are composed of three aspects:
 1. **A written description**
 2. **Conversations**
 3. **Tests**

Hyvän storyn kriteerit

- ▶ Bill Wake *INVEST in good User Stories*, kuusi toivottavaa ominaisuutta
 - ▶ Independent
 - ▶ Negotiable
 - ▶ Valuable to user or customer
 - ▶ Estimable
 - ▶ Small
 - ▶ Testable

Luento 2

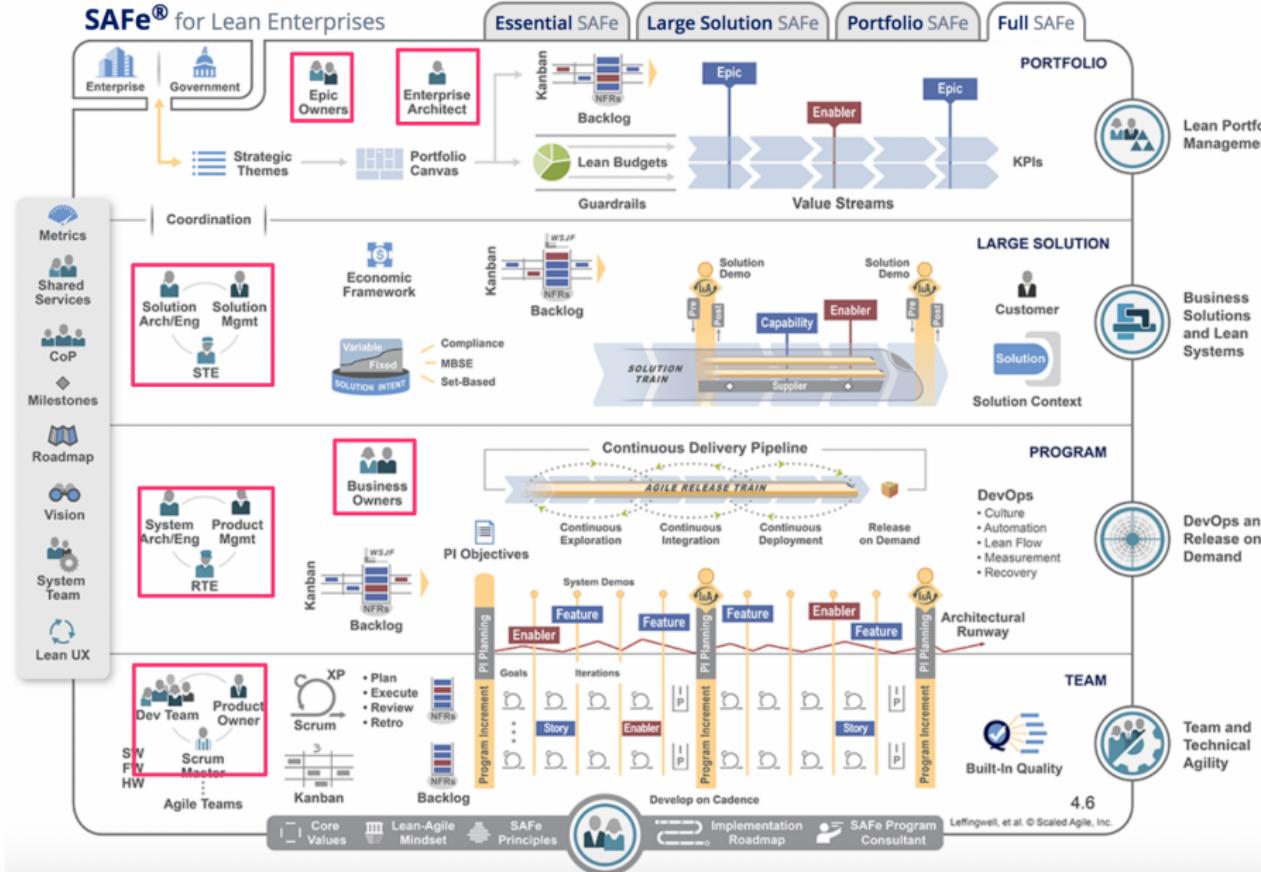
Scrum kuvana



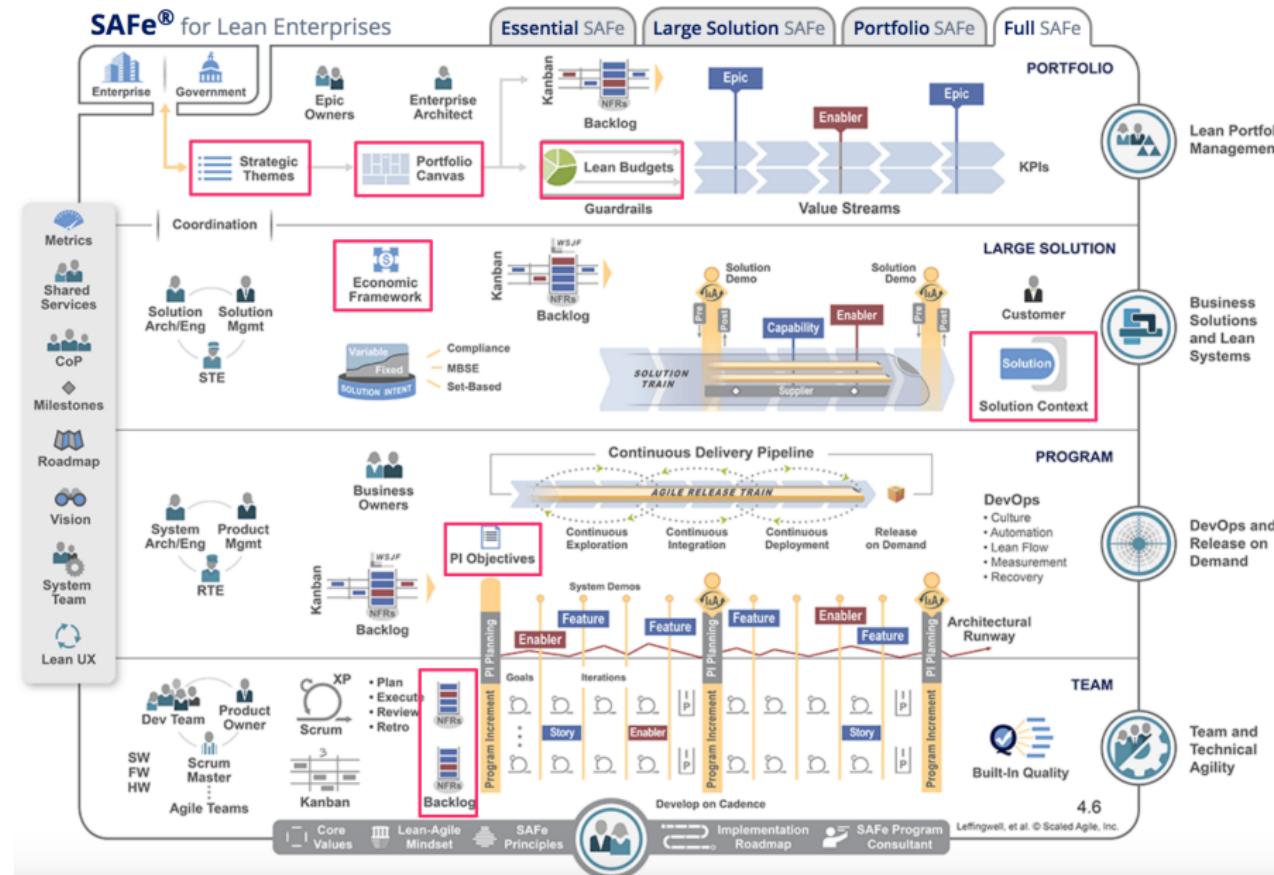
Luento 10

28.11.2023

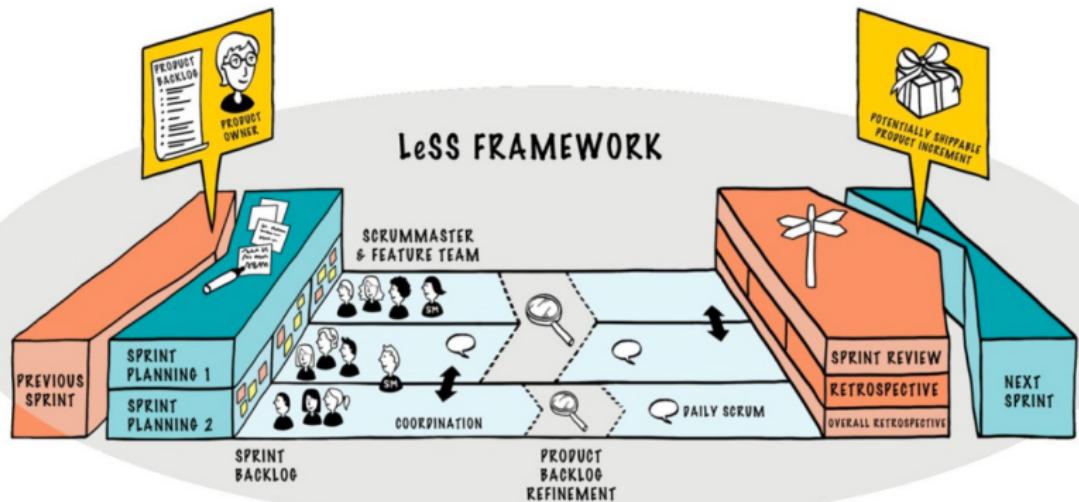
SAFe: paljon rooleja



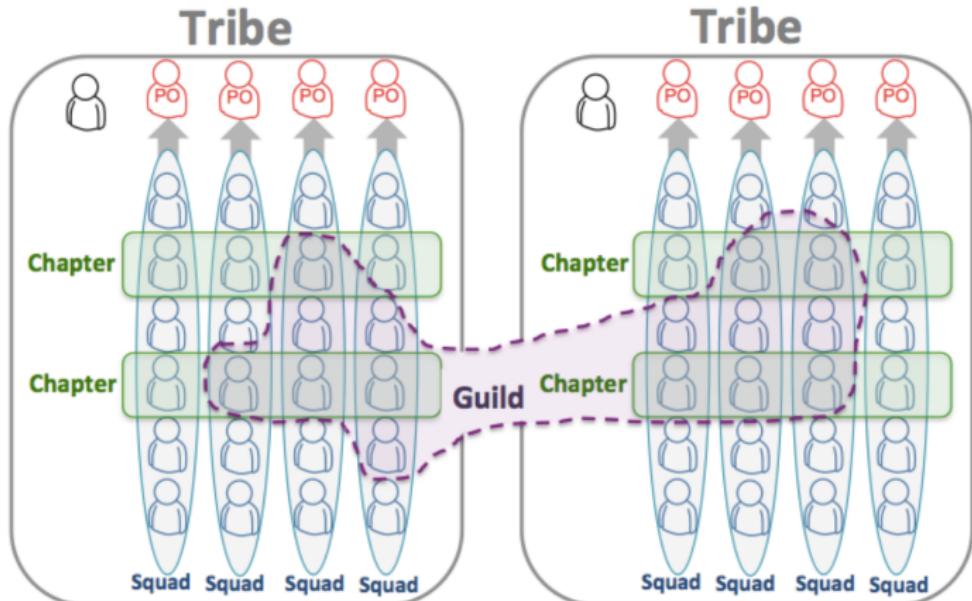
SAFe: paljon artefakteja



LeSS



Spotifyn malli



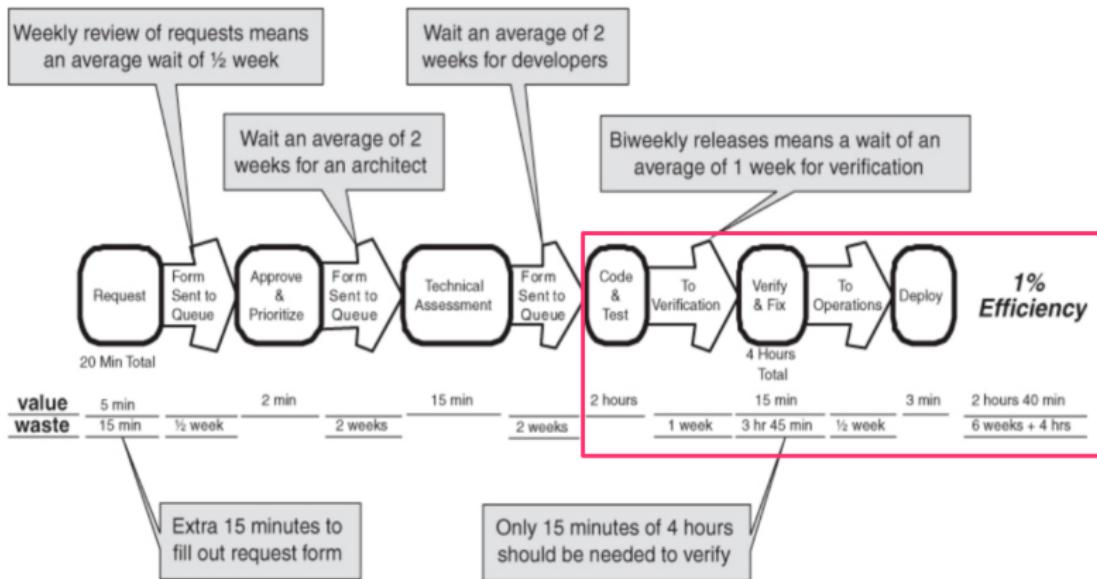
Luento 9

29.11.2023

Lean thinking house

Sustainable shortest lead time, best quality and value (to people and society), most customer delight, lowest cost, high morale, safety		
<p>Respect for People</p> <ul style="list-style-type: none">- don't trouble your 'customer'- "develop people, then build products"- no wasteful work- teams & individuals evolve their own practices and improvements- build partners with stable relationships, trust, and coaching in lean thinking- develop teams	<p>Product Development</p> <ul style="list-style-type: none">- long-term great engineers- mentoring from manager-engineer-teacher- cadence- cross-functional- team room + visual mgmt- entrepreneurial chief engineer/product mgr- set-based concurrent dev- create more knowledge <p>14 Principles</p> <ul style="list-style-type: none">long-term, flow, pull, less variability & overburden, Stop & Fix, master norms, simple visual mgmt, good tech, leader-teachers from within, develop exceptional people, help partners be lean, Go See, consensus, reflection & kaizen	<p>Continuous Improvement</p> <ul style="list-style-type: none">- Go See- kaizen- spread knowledge- small, relentless- retrospectives- 5 Whys- eyes for waste<ul style="list-style-type: none">* variability, overburden, NVA ... (handoff, WIP, info scatter, delay, multi-tasking, defects, wishful thinking..)- perfection challenge- work toward flow (lower batch size, Q size, cycle time)
Management applies and teaches lean thinking, and bases decisions on this long-term philosophy		

Perimmäisen syyn analyysi: five whys

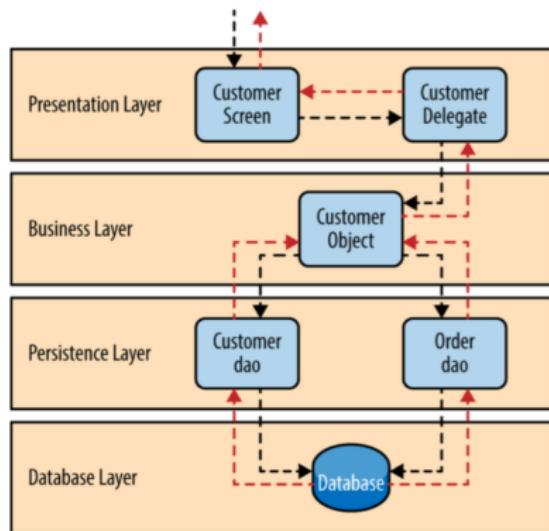


Luento 7

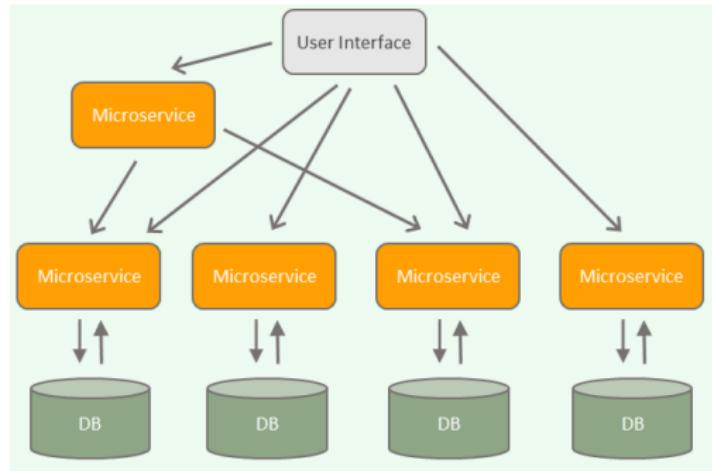
20.11.2023

Kerrosarkkitehtuuri

- ▶ Kerros on kokoelma toisiinsa liittyviä olioita, jotka muodostavat toiminnallisuuden suhteen loogisen kokonaisuuden



Mikropalveluarkkitehtuuri



- ▶ sovellus koostataan useista (jopa sadoista) pienistä verkossa toimivista autonomisista palveluista

Luento 1

30.10.2024

Ohjelmiston elinkaari (software lifecycle)

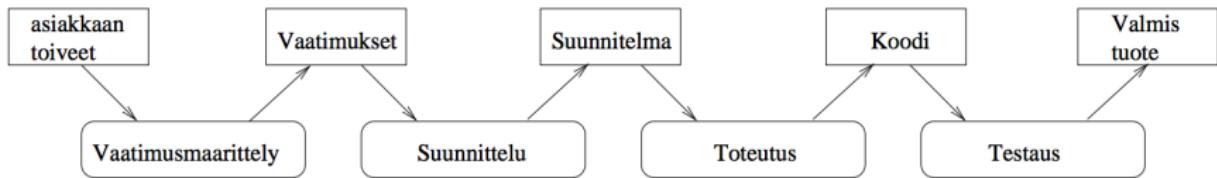
Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

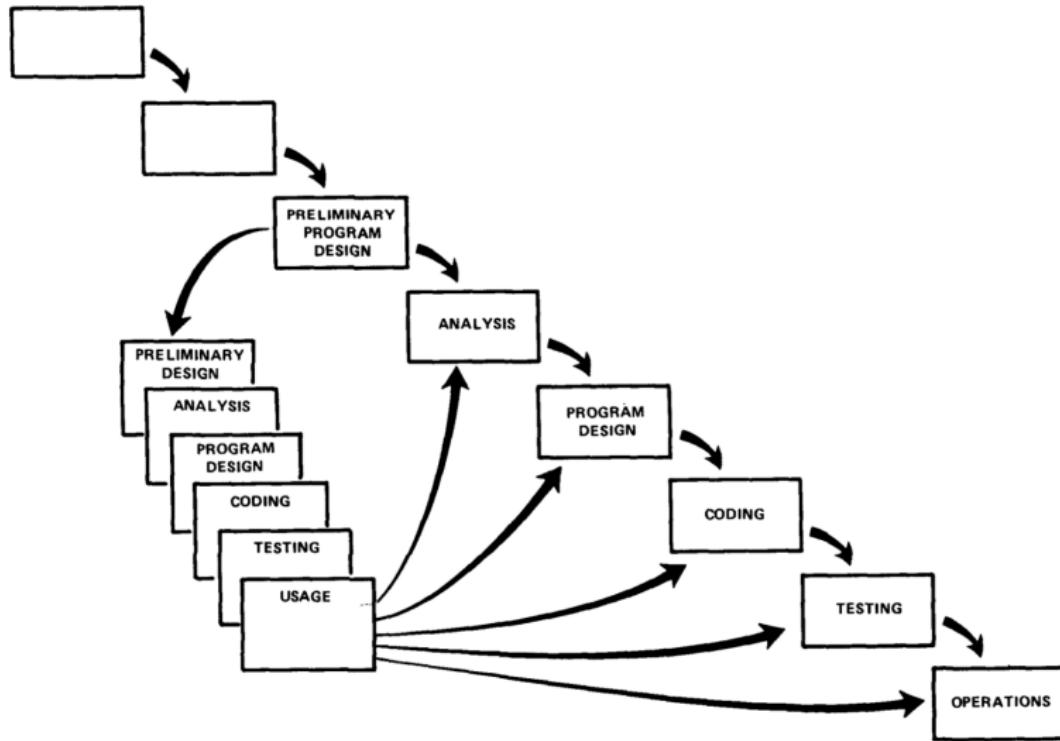
Vaiheista muodostuu ohjelmiston “elinkaari”

Vesiputousmalli

Winston W. Royce: Management of the development of Large Software, 1970



Roycen kahden iteraation malli



Iteratiivinen ohjelmistokehitys

