

# Ohjelmistotuotanto

Matti Luukkainen ja ohjaajat Valtteri Kantanen, Hannah  
Leinson, Riku Rauhala, Ville Saastamoinen

syksy 2023

Luento 6

14.11.2022

## Kurssipalaute

- ▶ Kurssipalaute
  - ▶ Kurssilla lopussa kerättävän palautteen lisäksi ns. jatkuva palaute <https://norppa.helsinki.fi>

# 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

## Regressiotestaus

- ▶ Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
  - ▶ Samalla tulee huolehtia, että ei rikota jo toimivia osia

## Regressiotestaus

- ▶ Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
  - ▶ Samalla tulee huolehtia, että ei rikota jo toimivia osia
- ▶ Testit on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- ▶ Tätä käytäntöä sanotaan *regressiotestaukseksi*

# Regressiotestaus

- ▶ Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
  - ▶ Samalla tulee huolehtia, että ei rikota jo toimivia osia
- ▶ Testit on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- ▶ Tätä käytäntöä sanotaan *regressiotestaukseksi*
- ▶ Regressiotestijoukko koostuu kaikista ohjelmistolle tehdyistä testeistä
  - ▶ sisältää yksikkö-, integraatio- ja järjestelmätesteistä

# Regressiotestaus

- ▶ Iteratiivisessa ohjelmistotuotannossa, jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia
  - ▶ Samalla tulee huolehtia, että ei rikota jo toimivia osia
- ▶ Testit on suoritettava uudelleen aina kun ohjelmistoon tehdään muutoksia
- ▶ Tätä käytäntöä sanotaan *regressiotestaukseksi*
- ▶ Regressiotestijoukko koostuu kaikista ohjelmistolle tehdyistä testeistä
  - ▶ sisältää yksikkö-, integraatio- ja järjestelmätesteistä
- ▶ Testaus on työlästä ja regressiotestauksen tarve tekee siitä entistä työläämpää
  - ▶ *Testaus kannattaa automatisoida mahdollisimman suurissa määrin*

# Ketterien menetelmien testauskäytänteet

## Ketterien menetelmien testauskäytänteet

- ▶ Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallista
  - ▶ Sprintin aikana toteutettavat ominaisuudet integroidaan muuhun koodiin sekä testataan

## Ketterien menetelmien testauskäytänteet

- ▶ Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallista
  - ▶ Sprintin aikana toteutettavat ominaisuudet integroidaan muuhun koodiin sekä testataan
- ▶ Sykli ominaisuuden määrittelystä siihen että se on valmis ja testattu on erittäin lyhyt

## Ketterien menetelmien testauskäytänteet

- ▶ Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallista
  - ▶ Sprintin aikana toteutettavat ominaisuudet integroidaan muuhun koodiin sekä testataan
- ▶ Sykli ominaisuuden määrittelystä siihen että se on valmis ja testattu on erittäin lyhyt
- ▶ Testausta tehdään sprintin “ensimmäisestä päivästä” lähtien, testaus integroitu suunnittelun ja toteutukseen

## Ketterien menetelmien testauskäytänteet

- ▶ Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallista
  - ▶ Sprintin aikana toteutettavat ominaisuudet integroidaan muuhun koodiin sekä testataan
- ▶ Sykli ominaisuuden määrittelystä siihen että se on valmis ja testattu on erittäin lyhyt
- ▶ Testausta tehdään sprintin “ensimmäisestä päivästä” lähtien, testaus integroitu suunnittelun ja toteutukseen
- ▶ Automatisointi erittäin tärkeässä roolissa
  - ▶ testejä suoritetaan usein

## Testaajat osana kehitystiimiä

- ▶ Ideaalilanteessa testaajia sijoitettu kehittäjätiimiin, myös ohjelmoijat kirjoittavat testejä
  - ▶ tiimit *cross functional*

## Testaajat osana kehitystiimiä

- ▶ Ideaalilanteessa testaajia sijoitettu kehittäjätiimiin, myös ohjelmoijat kirjoittavat testejä
  - ▶ tiimit *cross functional*
- ▶ Testaajan rooli: *virheiden etsijästä virheiden estääjään*
  - ▶ testaaja auttaa tiimiä kirjoittamaan automatisoituja testejä, jotka pyrkivät estämään bugien pääsyn koodiin
  - ▶ *build quality in*

# Ketterien menetelmien testauskäytänteitä

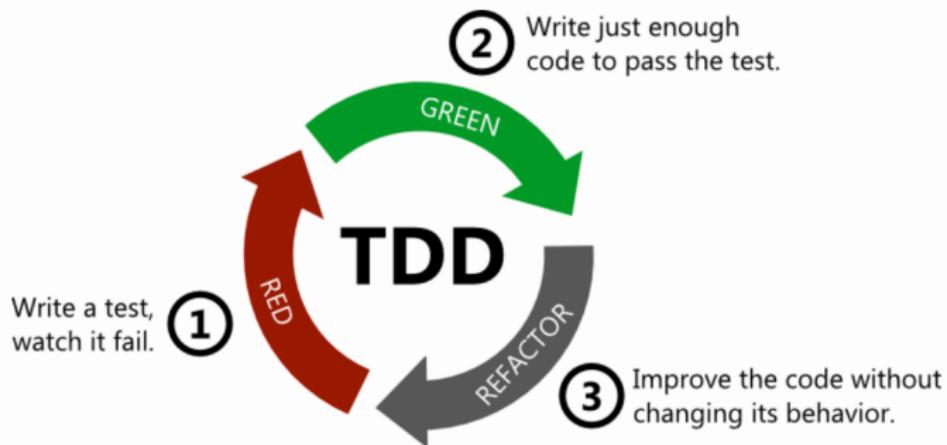
Tänään aiheena

# Ketterien menetelmien testauskäytänteitä

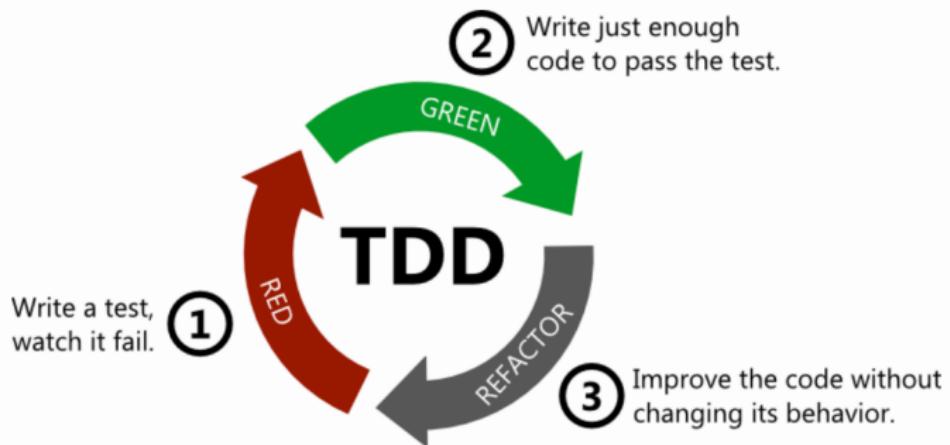
Tänään aiheena

- ▶ Test driven development (TDD)
- ▶ User storyjen tasolla tapahtuva automatisoitu testaus
- ▶ Continuous Integration (CI) eli jatkuva integraatio
- ▶ Exploratory testing, suomeksi tutkiva testaus
- ▶ Tuotannossa tapahtuva testaus

# Test driven development (TDD)

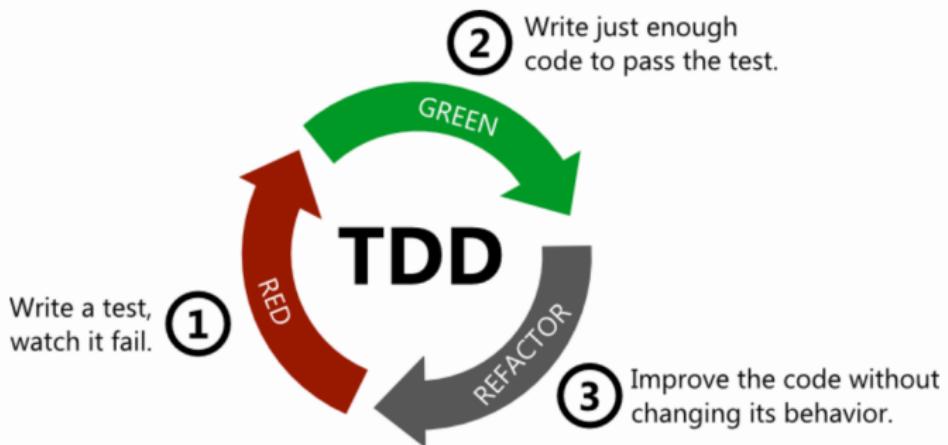


# Test driven development (TDD)



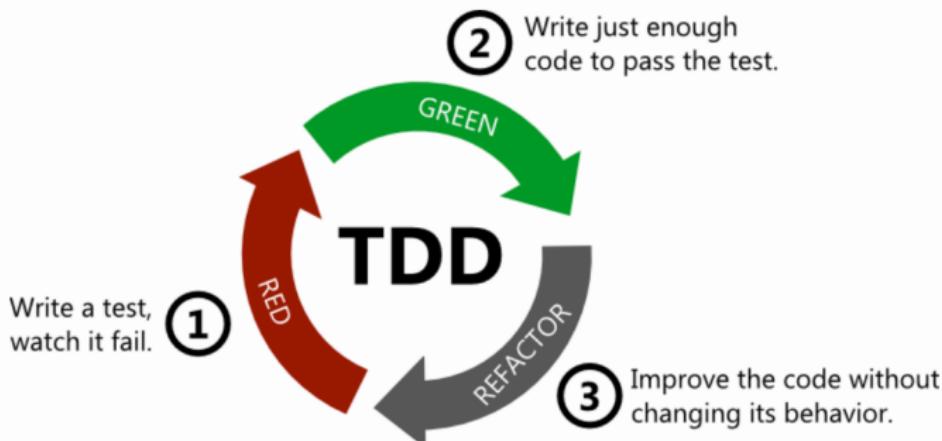
1. Kirjoitetaan sen verran testiä että testi ei mene läpi

# Test driven development (TDD)



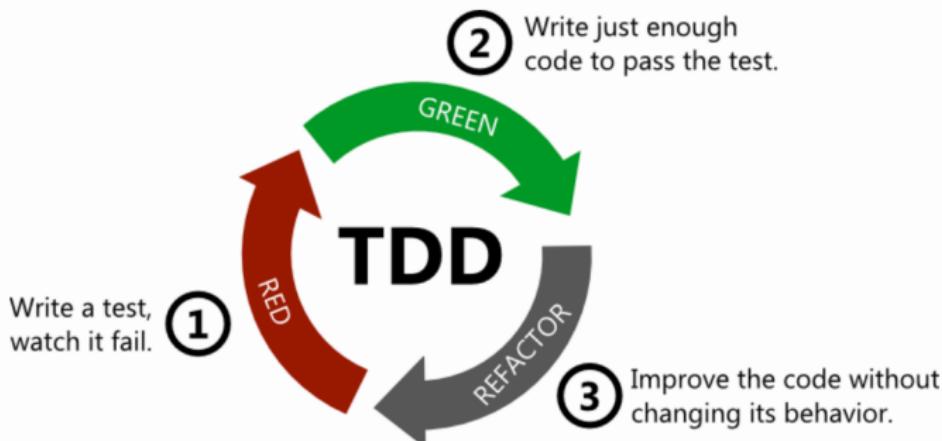
1. Kirjoitetaan sen verran testiä että testi ei mene läpi
2. Kirjoitetaan koodia sen verran, että testi menee läpi

# 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

# 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

# Test driven development (TDD)

- ▶ Yksi XP:n käytänteistä, Kent Beckin kehittämä
  - ▶ Joskus käytössä *tests first development*

## Test driven development (TDD)

- ▶ Yksi XP:n käytänteistä, Kent Beckin kehittämä
  - ▶ Joskus käytössä *tests first development*
- ▶ TDD:llä ohjelmoitaessa toteutettavaa komponenttia ei yleensä ole tapana suunnitella tyhjentävästi etukäteen

# Test driven development (TDD)

- ▶ Yksi XP:n käytänteistä, Kent Beckin kehittämä
  - ▶ Joskus käytössä *tests first development*
- ▶ TDD:llä ohjelmoitaessa toteutettavaa komponenttia ei yleensä ole tapana suunnitella tyhjentävästi etukäteen
- ▶ Testit kirjoitetaan ensisijaisesti ajatellen komponentin käyttöä
  - ▶ huomio komponentin rajapinnassa ja sen helppokäyttöisyydessä
  - ▶ ei niinkään komponentin sisäisessä toteutuksessa

# Test driven development (TDD)

- ▶ Yksi XP:n käytänteistä, Kent Beckin kehittämä
  - ▶ Joskus käytössä *tests first development*
- ▶ TDD:llä ohjelmoitaessa toteutettavaa komponenttia ei yleensä ole tapana suunnitella tyhjentävästi etukäteen
- ▶ Testit kirjoitetaan ensisijaisesti ajatellen komponentin käyttöä
  - ▶ huomio komponentin rajapinnassa ja sen helppokäyttöisyydessä
  - ▶ ei niinkään komponentin sisäisessä toteutuksessa
- ▶ Komponentin sisäinen rakenne muotoutuu refaktorointien kautta

# Test driven development (TDD)

- ▶ Yksi XP:n käytänteistä, Kent Beckin kehittämä
  - ▶ Joskus käytössä *tests first development*
- ▶ TDD:llä ohjelmoitaessa toteutettavaa komponenttia ei yleensä ole tapana suunnitella tyhjentävästi etukäteen
- ▶ Testit kirjoitetaan ensisijaisesti ajatellen komponentin käyttöä
  - ▶ huomio komponentin rajapinnassa ja sen helppokäyttöisyydessä
  - ▶ ei niinkään komponentin sisäisessä toteutuksessa
- ▶ Komponentin sisäinen rakenne muotoutuu refaktorointien kautta
- ▶ “*Ensin testataan, sitten kodataan, suunnitellaan vasta lopussa*”

- ▶ TDD:ssä korostetaan lopputuloksen yksinkertaisuutta

- ▶ TDD:ssä korostetaan lopputuloksen yksinkertaisuutta
- ▶ Toteutetaan toiminnallisuutta vain sen verran, mitä testien läpimeno edellyttää
  - ▶ Ei toteuteta "varalta" ekstratoiminnallisuutta, sillä "You ain't gonna need it" (YAGNI)
  - ▶ *Simplicity – the art of maximizing the amount of work not done – is essential*

- ▶ TDD:ssä korostetaan lopputuloksen yksinkertaisuutta
- ▶ Toteutetaan toiminnallisuutta vain sen verran, mitä testien läpimeno edellyttää
  - ▶ Ei toteuteta "varalta" ekstratoiminnallisuutta, sillä "You ain't gonna need it" (YAGNI)
  - ▶ *Simplicity – the art of maximizing the amount of work not done – is essential*
- ▶ Koodista on vaikea tehdä testattavaa jos se ei ole modulaarista ja löyhästi kytketyistä komponenteista koostuvaan
  - ▶ koodista laadukasta ylläpidettävyyden ja laajennettavuuden kannalta

- ▶ TDD:ssä korostetaan lopputuloksen yksinkertaisuutta
- ▶ Toteutetaan toiminnallisuutta vain sen verran, mitä testien läpimeno edellyttää
  - ▶ Ei toteuteta "varalta" ekstratoiminnallisuutta, sillä "You ain't gonna need it" (YAGNI)
  - ▶ *Simplicity – the art of maximizing the amount of work not done – is essential*
- ▶ Koodista on vaikea tehdä testattavaa jos se ei ole modulaarista ja löyhästi kytketyistä komponenteista koostuvaan
  - ▶ koodista laadukasta ylläpidettävyyden ja laajennettavuuden kannalta
- ▶ Muita TDD:n hyviä puolia:
  - ▶ Rohkaisee ottamaan pieniä askelia kerrallaan ja toimimaan fokusoidusti
  - ▶ Virheet havaitaan nopeasti suuren testijoukon takia

## TDD:llä on myös ikävät puolensa

- ▶ Testikoodia tulee paljon, usein suunnilleen saman verran kuin varsinaista koodia
  - ▶ Toisaalta TDD:llä tehty tuotantokoodi on usein hieman normaalisti tehtyä koodia lyhempi

## TDD:llä on myös ikävät puolensa

- ▶ Testikoodia tulee paljon, usein suunnilleen saman verran kuin varsinaista koodia
  - ▶ Toisaalta TDD:llä tehty tuotantokoodi on usein hieman normaalisti tehtyä koodia lyhempi
- ▶ Jos ja kun koodi muuttuu, tulee testejä ylläpitää

## TDD:llä on myös ikävät puolensa

- ▶ Testikoodia tulee paljon, usein suunnilleen saman verran kuin varsinaista koodia
  - ▶ Toisaalta TDD:llä tehty tuotantokoodi on usein hieman normaalista tehtyä koodia lyhempi
- ▶ Jos ja kun koodi muuttuu, tulee testejä ylläpitää
- ▶ TDD:n käyttö on vaikeampaa mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä

## TDD:llä on myös ikävät puolensa

- ▶ Testikoodia tulee paljon, usein suunnilleen saman verran kuin varsinaista koodia
  - ▶ Toisaalta TDD:llä tehty tuotantokoodi on usein hieman normaalista tehtyä koodia lyhempi
- ▶ Jos ja kun koodi muuttuu, tulee testejä ylläpitää
- ▶ TDD:n käyttö on vaikeampaa mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä
- ▶ Legacy-koodin laajentaminen TDD:llä voi olla haastavaa

## TDD:llä on myös ikävät puolensa

- ▶ Testikoodia tulee paljon, usein suunnilleen saman verran kuin varsinaista koodia
  - ▶ Toisaalta TDD:llä tehty tuotantokoodi on usein hieman normaalista tehtyä koodia lyhempi
- ▶ Jos ja kun koodi muuttuu, tulee testejä ylläpitää
- ▶ TDD:n käyttö on vaikeampaa mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä
- ▶ Legacy-koodin laajentaminen TDD:llä voi olla haastavaa
- ▶ Tutkimusnäytö TDD:n hyödyistä vähäistä

15.1.2024-

15.1.2024-

Open Uni: Test-Driven Development 4 + 1 cr

► Esko Luontola Nitor (Suomen johtava TDD-asiantuntija)

The screenshot shows a web browser window for the course "Test-Driven Development" at [tdd.mooc.fi](https://tdd.mooc.fi). The left sidebar lists course navigation links: "Test-Driven Development" (highlighted in red), "Course overview" (selected), "Practicalities (2021 Pilot)", "Exercises and schedule", "Chapter 1: What is TDD", "Chapter 2: Refactoring and design", "Chapter 3: The Untestables", "Chapter 4: Legacy code", "Chapter 5: Advanced techniques", and "Chapter 6: To infinity and beyond". The main content area features a large graphic with three semi-circular arrows forming a loop: a red arrow pointing clockwise at the top, a green arrow pointing clockwise on the right, and a blue arrow pointing counter-clockwise at the bottom. The text "Test-Driven Development" is written in large red letters across the center of the graphic, with the subtitle "a plunge into the TDD programming technique" in smaller red text below it. The word "Refactor" is written in white on the blue arrow. At the bottom, the word "Course overview" is partially visible in white text.

# Storyjen testaaminen

## Storyjen testaaminen

- ▶ User storyn käsite pitää sisällään *hyväksymiskriteerit*
  - ▶ *tests that convey and document details and that will be used to determine that the story is complete*

# Storyjen testaaminen

- ▶ User storyn käsite pitää sisällään *hyväksymiskriteerit*
  - ▶ *tests that convey and document details and that will be used to determine that the story is complete*
- ▶ Storyn *asiakas voi lisätä tuotteen ostoskoriin eräs hyväksymiskriteeri voisi olla*
  - ▶ ollessaan tuotelistaiksessa ja valitessaan tuotteen jota on varastossa, menee tuote ostoskoriin
  - ▶ ja ostoskorin hinta sekä korissa olevien tuotteiden määrä päivittyy oikein

# Storyjen testaaminen

- ▶ User storyn käsite pitää sisällään *hyväksymiskriteerit*
  - ▶ *tests that convey and document details and that will be used to determine that the story is complete*
- ▶ Storyn *asiakas voi lisätä tuotteen ostoskoriin eräs hyväksymiskriteeri voisi olla*
  - ▶ ollessaan tuotelistaiksessa ja valitessaan tuotteen jota on varastossa, menee tuote ostoskoriin
  - ▶ ja ostoskorin hinta sekä korissa olevien tuotteiden määrä päivittyy oikein
- ▶ Hyväksymiskriteereistä saadaan muodostettua suurin osa ohjelmiston järjestelmätason toiminnallisista testeistä

## Järjestelmätestauksen automatisointi, ATDD ja BDD

- ▶ Hyväksymiskriteerit on tarkoituksenmukaista kirjoittaa heti storyn toteuttavan sprintin alussa
  - ▶ yhteistyössä kehitystiimin ja product ownerin kesken
  - ▶ asiakkaan kielellä, käyttämättä teknistä jargonia

## Järjestelmätestauksen automatisointi, ATDD ja BDD

- ▶ Hyväksymiskriteerit on tarkoitukseenmukaista kirjoittaa heti storyn toteuttavan sprintin alussa
  - ▶ yhteistyössä kehitystiimin ja product ownerin kesken
  - ▶ asiakkaan kielellä, käyttämättä teknistä jargonia
- ▶ Ideaalilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia

# Järjestelmätestauksen automatisointi, ATDD ja BDD

- ▶ Hyväksymiskriteerit on tarkoituksenmukaista kirjoittaa heti storyn toteuttavan sprintin alussa
  - ▶ yhteistyössä kehitystiimin ja product ownerin kesken
  - ▶ asiakkaan kielellä, käyttämättä teknistä jargonia
- ▶ Ideaalilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- ▶ Olemassa monia työkaluja
  - ▶ eräs suosituimmista on suomalainen Python-pohjainen *Robot framework* jota kurssin Python-versio käyttää

## Järjestelmätestauksen automatisointi, ATDD ja BDD

- ▶ Hyväksymiskriteerit on tarkoitukseenmukaista kirjoittaa heti storyn toteuttavan sprintin alussa
  - ▶ yhteistyössä kehitystiimin ja product ownerin kesken
  - ▶ asiakkaan kielellä, käyttämättä teknistä jargonia
- ▶ Ideaalilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- ▶ Olemassa monia työkaluja
  - ▶ eräs suosituimmista on suomalainen Python-pohjainen *Robot framework* jota kurssin Python-versio käyttää
- ▶ Käytetään nimitystä *Acceptance test driven development* (ATDD) tai *Behavior driven development* (BDD)
  - ▶ erityisesti jos testit toteutetaan jo iteraation alkupuolella, ennen kun story koodattu

# Käyttäjähallinnan tarjoama palvelu

- ▶ Palvelun vaatimukset määrittelevät user storyt
  - ▶ A new user account can be created if a proper unused username and a proper password are given
  - ▶ User can log in with a valid username/password-combination

# Käyttäjähallinnan tarjoama palvelu

- ▶ Palvelun vaatimukset määrittelevät user storyt
  - ▶ A new user account can be created if a proper unused username and a proper password are given
  - ▶ User can log in with a valid username/password-combination
- ▶ Robot-frameworkia käytettäessä jokaisesta storystä kirjoitetaan .robot- pääteinen tiedosto
  - ▶ sisältää joukon storyyn liittyvä hyväksymistestejä

# Käyttäjähallinnan tarjoama palvelu

- ▶ Palvelun vaatimukset määrittelevät user storyt
  - ▶ A new user account can be created if a proper unused username and a proper password are given
  - ▶ User can log in with a valid username/password-combination
- ▶ Robot-frameworkia käytettäessä jokaisesta storystä kirjoitetaan .robot- pääteinen tiedosto
  - ▶ sisältää joukon storyn liittyvä hyväksymistestejä
- ▶ Storyn hyväksymätestit kirjoitetaan hyödyntäen *avainsanoja*

# 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*

# Avainsanat mäpitäään kooditasolle

- ▶ Avainsanojen määrittely toisten avainsanojen avulla

```
*** Keywords ***
Input Login Command
| Input login

Input Register Command
| Input new

Input Credentials
[Arguments] ${username} ${password}
Input ${username}
Input ${password}
Run Application
```

# Avainsanat mäpitäään kooditasolle

- ▶ Avainsanojen määrittely toisten avainsanojen avulla

```
*** Keywords ***
Input Login Command
| Input login

Input Register Command
| Input new

Input Credentials
[Arguments] ${username} ${password}
Input ${username}
Input ${password}
Run Application
```

- ▶ tai koodina

```
def input(self, value):
    self._io.add_input(value)

def output_should_contain(self, value):
    outputs = self._io.outputs

    if not value in outputs:
        raise AssertionError(
            f"Output \'{value}\' is not in {str(outputs)}"
        )
```

## Käyttöliittymän läpi tapahtuvan testauksen automatisointi

- ▶ Komentoriviä käyttävien sovellusten testaaminen onnistuu helpohkosti, mockaamalla syöte- ja tulostusvirrat

## Käyttöliittymän läpi tapahtuvan testauksen automatisointi

- ▶ Komentoriviä käyttävien sovellusten testaaminen onnistuu helpohkosti, mockaamalla syöte- ja tulostusvirrat
- ▶ Myös Web-sovellusten testauksen automatisointi onnistuu  
▶ eräs ratkaisu *Selenium*, joka mahollistaa selaimen käytön ohjelmointirajapintaa käyttäen

Motivaatio käyttäjän kielellä kirjoitetuille testeille

## Motivaatio käyttäjän kielellä kirjoitetuille testeille

- ▶ Product owner kirjoittaa tiimin kanssa storyyn liittyvät testit
  - ▶ storyn haluttu toiminnallisuus tulee dokumentoitua ja ohjelmoijat ymmärtäävät mistä on kyse

## Motivaatio käyttäjän kielellä kirjoitetuille testeille

- ▶ Product owner kirjoittaa tiimin kanssa storyyn liittyvät testit
  - ▶ storyn haluttu toiminnallisuus tulee dokumentoitua ja ohjelmoijat ymmärtäävät mistä on kyse
- ▶ Koodaajat/testaajat toteuttavat avainsanat siten, että testien automaattinen suoritus onnistuu

## Motivaatio käyttäjän kielellä kirjoitetuille testeille

- ▶ Product owner kirjoittaa tiimin kanssa storyyn liittyvät testit
  - ▶ storyn haluttu toiminnallisuus tulee dokumentoitua ja ohjelmoijat ymmärtäävät mistä on kyse
- ▶ Koodaajat/testaajat toteuttavat avainsanat siten, että testien automaattinen suoritus onnistuu
- ▶ Ei toistaiseksi vielä kovin yleinen tyyli, useimmiten hyväksymätestit kirjoitettu suoraan “normalilla” testikirjastolla
  - ▶ JUnit, Mocha, Jest, RSpec ...

# Ohjelmiston integraatio

## Ohjelmiston integraatio

- ▶ Vesiputousmallissa toteutusvaiheen päättää integrointi
  - ▶ Yksittäin testatut komponentit yhdessä toimivaksi kokonaisuudeksi
  - ▶ Yhteistoiminnallisuus varmistetaan **integraatiotestien** avulla

# Ohjelmiston integraatio

- ▶ Vesiputousmallissa toteutusvaiheen päättää integrointi
  - ▶ Yksittäin testatut komponentit yhdessä toimivaksi kokonaisuudeksi
  - ▶ Yhteistoiminnallisuus varmistetaan **integraatiotestien** avulla
- ▶ Perinteisesti integrointi on tuonut esiin paljon ongelmia
  - ▶ Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensovivia

# Ohjelmiston integraatio

- ▶ Vesiputousmallissa toteutusvaiheen päättää integrointi
  - ▶ Yksittäin testatut komponentit yhdessä toimivaksi kokonaisuudeksi
  - ▶ Yhteistoiminnallisuus varmistetaan **integraatiotestien** avulla
- ▶ Perinteisesti integrointi on tuonut esiin paljon ongelmia
  - ▶ Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensovivia
- ▶ Suurten projektien integrointivaihe on kestnyt ennakoimattoman kauan

# Ohjelmiston integraatio

- ▶ Vesiputousmallissa toteutusvaiheen päättää integrointi
  - ▶ Yksittäin testatut komponentit yhdessä toimivaksi kokonaisuudeksi
  - ▶ Yhteistoiminnallisuus varmistetaan **integraatiotestien** avulla
- ▶ Perinteisesti integrointi on tuonut esiin paljon ongelmia
  - ▶ Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensovivia
- ▶ Suurten projektien integrointivaihe on kestnyt ennakoimattoman kauan
- ▶ Integrointivaiheen ongelmat ovat aiheuttaneet ohjelmaan suunnittelutason muutoksia

# Ohjelmiston integraatio

- ▶ Vesiputousmallissa toteutusvaiheen päätää integrointi
  - ▶ Yksittäin testatut komponentit yhdessä toimivaksi kokonaisuudeksi
  - ▶ Yhteistoiminnallisuus varmistetaan **integraatiotestien** avulla
- ▶ Perinteisesti integrointi on tuonut esiin paljon ongelmia
  - ▶ Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensovivia
- ▶ Suurten projektien integrointivaihe on kestnyt ennakoimattoman kauan
- ▶ Integrointivaiheen ongelmat ovat aiheuttaneet ohjelmaan suunnittelutason muutoksia
- ▶ **Integratiohelvetti**

## Pois integraatiohelvetistä

- ▶ 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa

## Pois integraatiohelvetistä

- ▶ 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- ▶ Muodostui uusi paras käytäne: *daily build* ja *smoke test*
  - ▶ The *smoke test* should exercise the entire system from end to end.
  - ▶ It does not have to be exhaustive,
  - ▶ but it should be capable of exposing major problems

## Pois integraatiohelvetistä

- ▶ 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- ▶ Muodostui uusi paras käytänne: *daily build* ja *smoke test*
  - ▶ The *smoke test* should exercise the entire system from end to end.
  - ▶ It does not have to be exhaustive,
  - ▶ but it should be capable of exposing major problems
- ▶ Daily buildia ja smoke testiä käytettäessä järjestelmän integraatio tehdään (ainakin jollain tarkkuutasolla) joka päivä

## Pois integraatiohelvetistä

- ▶ 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- ▶ Muodostui uusi paras käytäne: *daily build* ja *smoke test*
  - ▶ The *smoke test* should exercise the entire system from end to end.
  - ▶ It does not have to be exhaustive,
  - ▶ but it should be capable of exposing major problems
- ▶ Daily buildia ja smoke testiä käytettäessä järjestelmän integraatio tehdään (ainakin jollain tarkkuutasolla) joka päivä
- ▶ Komponenttien yhtensopivusongelmat huomataan nopeasti ja niiden korjaaminen helpottuu

## Pois integraatiohelvetistä

- ▶ 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- ▶ Muodostui uusi paras käytänne: *daily build* ja *smoke test*
  - ▶ The *smoke test* should exercise the entire system from end to end.
  - ▶ It does not have to be exhaustive,
  - ▶ but it should be capable of exposing major problems
- ▶ Daily buildia ja smoke testiä käytettäessä järjestelmän integraatio tehdään (ainakin jollain tarkkuustasolla) joka päivä
- ▶ Komponenttien yhtensopivuusongelmat huomataan nopeasti ja niiden korjaaminen helpottuu
- ▶ **Tiimin moraali paranee**, kun ohjelmistosta on olemassa päivittäin kasvava toimiva versio

## Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä

## Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä
- ▶ Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositoriossa

## Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä
- ▶ Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositoriossa
- ▶ *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman läheisesti tuotantopalvelinta

## Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä
- ▶ Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositorioissa
- ▶ *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman läheisesti tuotantopalvelinta
- ▶ CI-palvelin tarkkailee repositoriota, muutosten tapahtuessa se hakee koodin, käänää sen ja suorittaa testit

# Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä
- ▶ Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositorioissa
- ▶ *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman läheisesti tuotantopalvelinta
- ▶ CI-palvelin tarkkailee repositoriota, muutosten tapahtuessa se hakee koodin, käänää sen ja suorittaa testit
- ▶ Jos koodi ei käänny tai testit eivät mene läpi, seurauksena poikkeustilanne joka korjattava *välittömästi*

# Päivittäisestä jatkuvaan integraatioon

- ▶ Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - ▶ eräs XP:n käytenäteistä
- ▶ Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositorioissa
- ▶ *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman läheisesti tuotantopalvelinta
- ▶ CI-palvelin tarkkailee repositoria, muutosten tapahtuessa se hakee koodin, käänää sen ja suorittaa testit
- ▶ Jos koodi ei käänny tai testit eivät mene läpi, seurauksena poikkeustilanne joka korjattava *välittömästi*
- ▶ **Integraatiosta vaivaton operaatio**: ohjelmistosta olemassa koko ajan integroitu ja testattu tuore versio

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- ▶ Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- ▶ Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- ▶ Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, vähintään kerran päivässä

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- ▶ Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- ▶ Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, vähintään kerran päivässä
- ▶ CI rohkaisee jakamaan työn pieniin osiin, sellaisiin jotka saadaan testeineen valmiiksi yhden työpäivän aikana
  - ▶ **CI-työprosessin noudattaminen vaatii kurinalaisuutta**

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- ▶ Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- ▶ Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, vähintään kerran päivässä
- ▶ CI rohkaisee jakamaan työn pieniin osiin, sellaisiin jotka saadaan testeineen valmiiksi yhden työpäivän aikana
  - ▶ **CI-työprosessin noudattaminen vaatii kurinalaisuutta**
- ▶ Laskareissa käytetty *GitHub Actions* suosituin SaaS-palveluna toimiva CI-ratkaisu

- ▶ Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- ▶ Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- ▶ Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, vähintään kerran päivässä
- ▶ CI rohkaisee jakamaan työn pieniin osiin, sellaisiin jotka saadaan testeineen valmiiksi yhden työpäivän aikana
  - ▶ **CI-työprosessin noudattaminen vaatii kurinalaisuutta**
- ▶ Laskareissa käytetty *GitHub Actions* suosituin SaaS-palveluna toimiva CI-ratkaisu
- ▶ Vanhempi Jenkins lienee edelleen maailmalla eniten käytetty CI-palvelinohjelmisto
  - ▶ Jenkinsin käyttö edellyttää sen asentamista omalle palvelimelle

Tauko 10 min

# Deployment pipeline

- ▶ Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle

# Deployment pipeline

- ▶ Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- ▶ Integraatioprosessiin lisätään automaattinen deployaus **staging-palvelimelle**
  - ▶ Ympäristö, joka kaikin tavoin mahdollisimman lähellä tuotantoymäristöä

# Deployment pipeline

- ▶ Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- ▶ Integraatioprosessiin lisätään automaattinen deployaus **staging-palvelimelle**
  - ▶ Ympäristö, joka kaikin tavoin mahdollisimman lähellä tuotantoymäristöä
- ▶ Kun uusi versio viety staging-palvelimelle, suoritetaan sille hyväksymistestaus

# Deployment pipeline

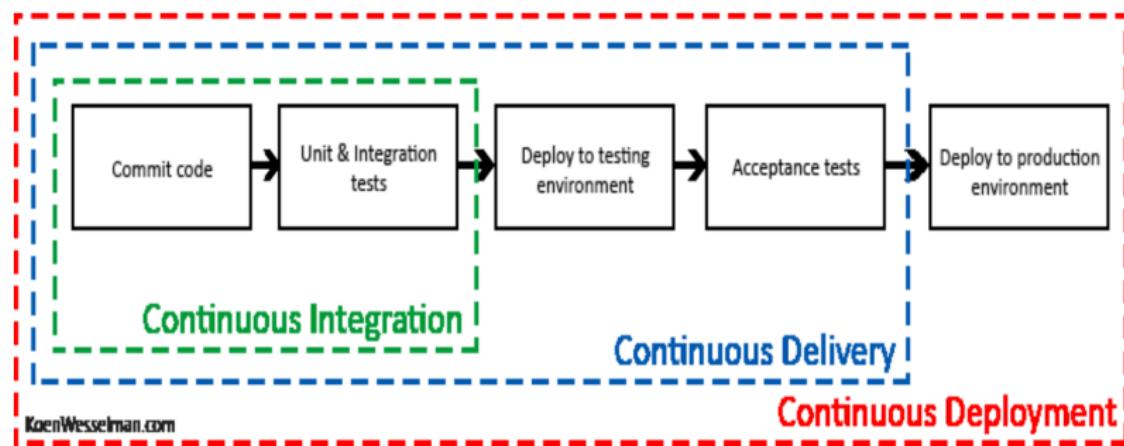
- ▶ Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- ▶ Integraatioprosessiin lisätään automaattinen deployaus **staging-palvelimelle**
  - ▶ Ympäristö, joka kaikin tavoin mahdollisimman lähellä tuotantoymäristöä
- ▶ Kun uusi versio viety staging-palvelimelle, suoritetaan sille hyväksymistestaus
- ▶ ...jonka jälkeen siirto **tuotantopalvelimelle**

# Deployment pipeline

- ▶ Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- ▶ Integraatioprosessiin lisätään automaattinen deployaus **staging-palvelimelle**
  - ▶ Ympäristö, joka kaikin tavoin mahdollisimman lähellä tuotantoymäristöä
- ▶ Kun uusi versio viety staging-palvelimelle, suoritetaan sille hyväksymistestaus
- ▶ ...jonka jälkeen siirto **tuotantopalvelimelle**
- ▶ Parhaassa tapauksessa staging-ympäristössä tehtävien hyväksymätestien suoritus on automatisoitu
  - ▶ Ohjelmisto kulkee koko *deployment pipeline* läpi automaattisesti

# Deployment pipeline

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



## Termejä: jatkuva toimittaminen ja toimitusvalmius

- ▶ **Jatkuvasta toimittamisen engl. continuous deployment**
  - ▶ Jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon

## Termejä: jatkuva toimittaminen ja toimitusvalmius

- ▶ **Jatkuvasta toimittaminen engl. continuous deployment**
  - ▶ Jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon
- ▶ **Jatkuva toimitusvalmius engl. continuous delivery:**
  - ▶ deployment-päätös tehdään ihmisen toimesta

## Termejä: jatkuva toimittaminen ja toimitusvalmius

- ▶ **Jatkuvasta toimittaminen engl. continuous deployment**
  - ▶ Jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon
- ▶ **Jatkuvaa toimitusvalmius engl. continuous delivery:**
  - ▶ deployment-päätös tehdään ihmisen toimesta
- ▶ Viime aikojen trendi julkaista web-palvelusta jopa kymmeniä uusia versiota päivästä
  - ▶ Amazon, Netflix, Facebook, Smartly...

## Tutkiva testaaminen

- ▶ Jotta järjestelmä saadaan riittävän virheettömäksi, on testaus suoritettava erittäin perusteellisesti

## Tutkiva testaaminen

- ▶ Jotta järjestelmä saadaan riittävän virheettömäksi, on testaus suoritettava erittäin perusteellisesti
- ▶ Perinteinen tapa järjestelmätestauksen on ollut laatia ennen testausta hyvin perinpohjainen suunnitelma
  - ▶ Jokaisesta testistä on kirjattu testisyötteet ja odotettu tulos

## Tutkiva testaaminen

- ▶ Jotta järjestelmä saadaan riittävän virheettömäksi, on testaus suoritettava erittäin perusteellisesti
- ▶ Perinteinen tapa järjestelmätestauksen on ollut laatia ennen testausta hyvin perinpohjainen suunnitelma
  - ▶ Jokaisesta testistä on kirjattu testisyötteet ja odotettu tulos
- ▶ Tuloksen tarkastaminen
  - ▶ Verrataan ohjelmiston toimintaa testitapaukseen kirjattuun odotettuun tulokseen

<b>Test Scenario ID</b>	Login-1	<b>Test Case ID</b>	Login-1B
<b>Test Case Description</b>	Login – Negative test case	<b>Test Priority</b>	High
<b>Pre-Requisite</b>	NA	<b>Post-Requisite</b>	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/201 7 11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: *****	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/201 7 11:45 AM]: Invalid login attempt stopped

<b>Test Scenario ID</b>	Login-1	<b>Test Case ID</b>	Login-1B	
<b>Test Case Description</b>	Login – Negative test case	<b>Test Priority</b>	High	
<b>Pre-Requisite</b>	NA	<b>Post-Requisite</b>	NA	

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/201 7 11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: *****	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/201 7 11:45 AM]: Invalid login attempt stopped

<b>Test Scenario ID</b>	Login-1	<b>Test Case ID</b>	Login-1B	
<b>Test Case Description</b>	Login – Negative test case	<b>Test Priority</b>	High	
<b>Pre-Requisite</b>	NA	<b>Post-Requisite</b>	NA	

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/201 7 11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: *****	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/201 7 11:45 AM]: Invalid login attempt stopped

- ▶ Automatisoitujen hyväksymistestien luonne sama, syöte on tarkkaan kiinnitetty samoin kuin odotettu tuloskin

- ▶ Automatisoitujen hyväksymistestien luonne sama, syöte on tarkkaan kiinnitetty samoin kuin odotettu tuloskin
- ▶ Jos testataan vain etukäteen mietittyjen testien avulla, ei kaikkia yllättäviä tilanteita osata ennakoida

- ▶ Automatisoitujen hyväksymistestien luonne sama, syöte on tarkkaan kiinnitetty samoin kuin odotettu tuloskin
- ▶ Jos testataan vain etukäteen mietittyjen testien avulla, ei kaikkia yllättäviä tilanteita osata ennakoida
- ▶ Hyvät testaajat ovat aina tehneet “virallisen” dokumentoidun testauksen lisäksi epävirallista “ad hoc”-testausta

- ▶ Automatisoitujen hyväksymistestien luonne sama, syöte on tarkkaan kiinnitetty samoin kuin odotettu tuloskin
- ▶ Jos testataan vain etukäteen mietittyjen testien avulla, ei kaikkia yllättäviä tilanteita osata ennakoida
- ▶ Hyvät testaajat ovat aina tehneet “virallisen” dokumentoidun testauksen lisäksi epävirallista “ad hoc”-testausta
- ▶ Tästä tullut virallisesti hyväksytty testauksen muoto, kulkee nimellä *tutkiva testaaminen* (engl. exploratory testing)

## Tutkiva testaaminen

- ▶ *Exploratory testing is simultaneous learning, test design and test execution*

## Tutkiva testaaminen

- ▶ *Exploratory testing is simultaneous learning, test design and test execution*
- ▶ Testitapauksia ei suunnitella kattavasti etukäteen
  - ▶ Testaaja pyrkii kokemuksensa avulla löytämään järjestelmästä virheitä
  - ▶ Testaaja ohjaa toimintaansa suorittamiensa testien tuloksen perusteella

## Tutkiva testaaminen

- ▶ *Exploratory testing is simultaneous learning, test design and test execution*
- ▶ Testitapauksia ei suunnitella kattavasti etukäteen
  - ▶ Testaaja pyrkii kokemuksensa avulla löytämään järjestelmästä virheitä
  - ▶ Testaaja ohjaa toimintaansa suorittamiensa testien tuloksen perusteella
- ▶ Tutkiva testaaminen ei kuitenkaan etene täysin sattumanvaraisesti
  - ▶ Testaussessiolle asetetaan tavoite: mitä tutkitaan ja minkälaisia virheitä etsitään

# Tutkiva testaaminen

- ▶ *Exploratory testing is simultaneous learning, test design and test execution*
- ▶ Testitapauksia ei suunnitella kattavasti etukäteen
  - ▶ Testaaja pyrkii kokemuksensa avulla löytämään järjestelmästä virheitä
  - ▶ Testaaja ohjaa toimintaansa suorittamiensa testien tuloksen perusteella
- ▶ Tutkiva testaaminen ei kuitenkaan etene täysin sattumanvaraisesti
  - ▶ Testaussessiolle asetetaan tavoite: mitä tutkitaan ja minkälaisia virheitä etsitään
- ▶ Tavoite voi liittyä esim. muutaman user storyn toiminnallisuuteen
  - ▶ *testataan ostosten lisäystä ja poistoa ostoskorista*

## Tutkiva testaaminen

- ▶ Keskeistä on kaikkien ohjelmiston tapahtuvien asioiden havainnointi
  - ▶ Etukäteen määritellyissä testeissä havainnoidaan ainoastaan reagoiko järjestelmä odotetulla tavalla

## Tutkiva testaaminen

- ▶ Keskeistä on kaikkien ohjelmiston tapahtuvien asioiden havainnointi
  - ▶ Etukäteen määritellyissä testeissä havainnoidaan ainoastaan reagoiko järjestelmä odotetulla tavalla
- ▶ Kiinnitetään huomio myös varsinaisen testauksen koteen ulkopuoleisiin asioihin
  - ▶ Klikkaillaan käyttöliittymän nappuloita epäloogisissa tilanteissa
  - ▶ Jos huomataan selaimen osoiterivillä URL  
<https://www.webshopshop.com/ostoskorri?id=10> katsotaan mitä tapahtuu jo id muutetaan käsin
  - ▶ ...

# Tutkiva testaaminen

- ▶ Keskeistä on kaikkien ohjelmiston tapahtuvien asioiden havainnointi
  - ▶ Etukäteen määritellyissä testeissä havainnoidaan ainoastaan reagoiko järjestelmä odotetulla tavalla
- ▶ Kiinnitetään huomio myös varsinaisen testauksen koteen ulkopuoleisiin asioihin
  - ▶ Klikkaillaan käyttöliittymän nappuloita epäloogisissa tilanteissa
  - ▶ Jos huomataan selaimen osoiterivillä URL  
`https://www.webshopshop.com/ostoskorpi?id=10` katsotaan mitä tapahtuu jo id muutetaan käsin
  - ▶ ...
- ▶ Tietoturvan testaamisessa on monia tyypillisiä skenaariota, joita testataan tutkivan testaamisen menetelmin
  - ▶ Esim. SQL- ja JavaScript-injektiot

## Tutkiva testaaminen

- ▶ Löydettyjen virheiden toistuminen jatkossa kannattaa eliminoida tekemällä automatisoituja regressiotestejä

## Tutkiva testaaminen

- ▶ Löydettyjen virheiden toistuminen jatkossa kannattaa eliminoida tekemällä automatisoituja regressiotestejä
- ▶ Tutkivaa testaamista ei kannata käyttää regressiotestaamisen menetelmänä
  - ▶ Testataan sprintin yhteydessä toteutettuja uusia ominaisuuksia

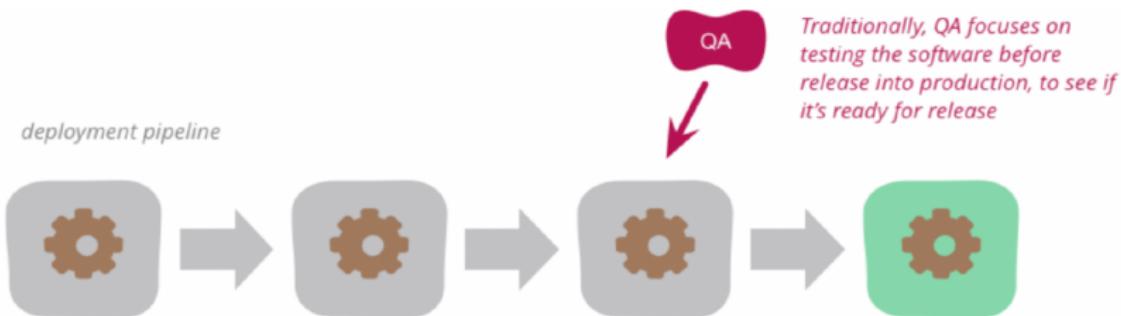
## Tutkiva testaaminen

- ▶ Löydettyjen virheiden toistuminen jatkossa kannattaa eliminoida tekemällä automatisoituja regressiotestejä
- ▶ Tutkivaa testaamista ei kannata käyttää regressiotestaamisen menetelmänä
  - ▶ Testataan sprintin yhteydessä toteutettuja uusia ominaisuuksia
- ▶ Ei vaihtoehto normaaleille tarkkaan etukäteen määritellyille testeille vaan niitä täydentävä testauksen muoto

## Tuotannossa tapahtuva testaaminen ja laadunhallinta

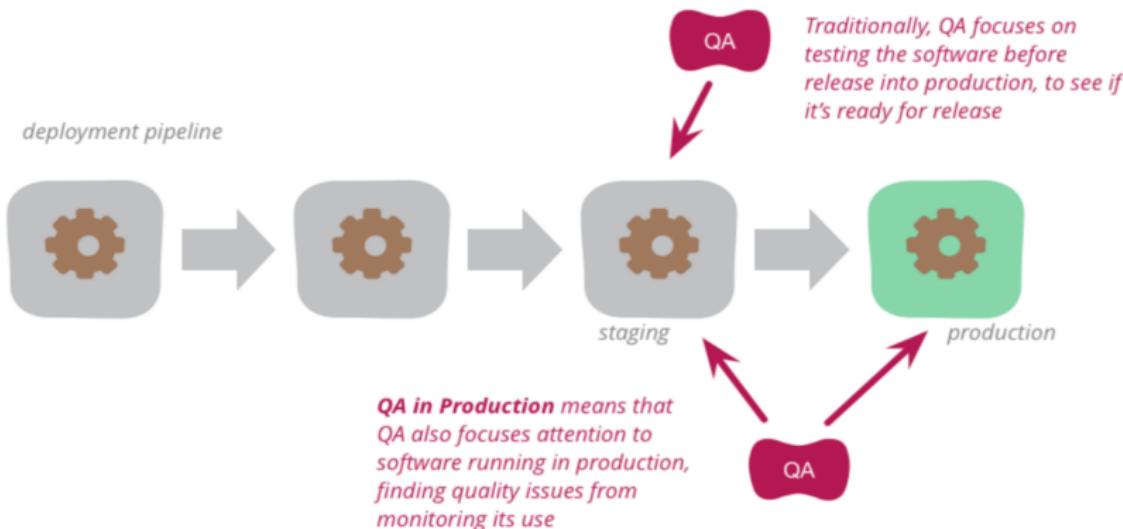
# Tuotannossa tapahtuva testaaminen ja laadunhallinta

- ▶ Perinteisesti ajateltu: kaikki laadunhallintaan tehdään ennen kuin uudet toiminnallisuudet otetaan käyttöön



# Tuotannossa tapahtuva testaaminen ja laadunhallinta

- ▶ Perinteisesti ajateltu: kaikki laadunhallintaan tehdään ennen kuin uudet toiminnallisuudet otetaan käyttöön



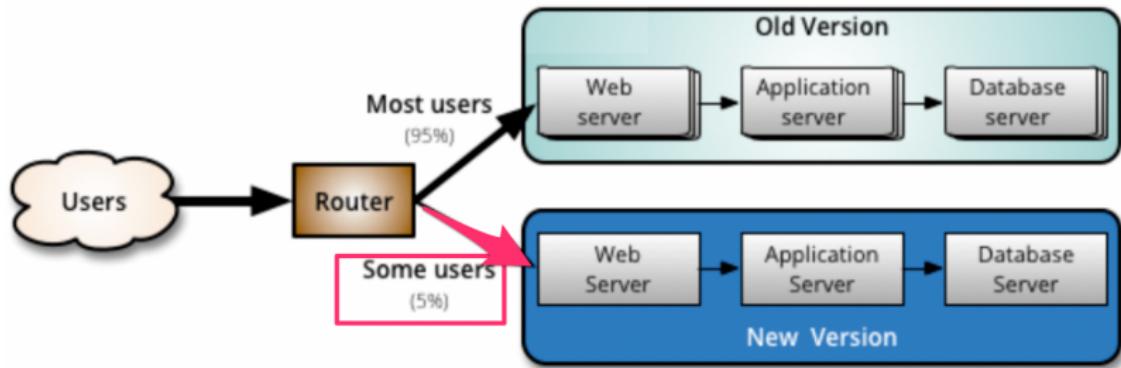
- ▶ Viime aikainen trendi on tehdä osa laadunhallinnasta *monitoroimalla* tuotannossa olevaa ohjelmistoa

## Canary release

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

# Canary release

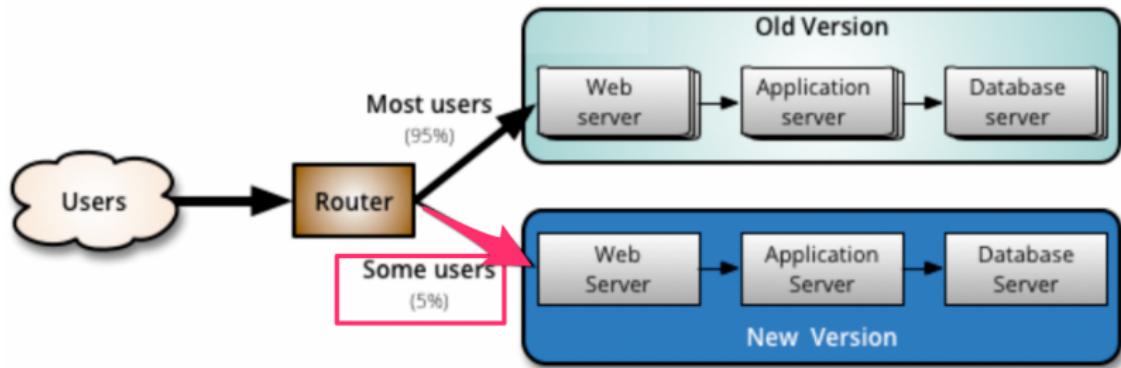
- ▶ Kaksi rinnakkaista tuotanto-ympäristöä, joista uudet ominaisuudet viedään toiseen



- ▶ Uuden ominaisuuden sisältävään ympäristöön ohjataan osa järjestelmän käyttäjistä

# Canary release

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



- ▶ Uuden ominaisuuden sisältävään ympäristöön ohjataan osa järjestelmän käyttäjistä
- ▶ Uuden ominaisuuden sisältämää versiota *monitoroidaan*
  - ▶ jos ei ongelmia ohjataan kaikki liikenne uuteen versioon

- ▶ Uuden version toimivaksi varmistaminen perustuu järjestelmän *monitorointiin*

- ▶ Uuden version toimivaksi varmistaminen perustuu järjestelmän *monitorointiin*
- ▶ Esim. sosiaalisen median palvelussa
  - ▶ palvelun muistin ja prosessoriajan kulutusta
  - ▶ verkkoliikenteen määrää
  - ▶ sovelluksen eri sivujen vasteaikoja
  - ▶ kirjautuneiden käyttäjien määrää
  - ▶ luettujen ja lähetettyjen viestien märiä per käyttäjä
  - ▶ kirjautuneen käyttäjän sovelluksessa viettämää aikaa

- ▶ Uuden version toimivaksi varmistaminen perustuu järjestelmän *monitorointiin*
- ▶ Esim. sosiaalisen median palvelussa
  - ▶ palvelun muistin ja prosessoriajan kulutusta
  - ▶ verkkoliikenteen määrää
  - ▶ sovelluksen eri sivujen vasteaikoja
  - ▶ kirjautuneiden käyttäjien määrää
  - ▶ luettujen ja lähetettyjen viestien määrä per käyttäjä
  - ▶ kirjautuneen käyttäjän sovelluksessa viettämää aikaa
- ▶ Monitoroidaan palvelimen yleisen toimivuuden lisäksi *käyttäjätason metriikoita* (engl. business level metrics)

- ▶ Uuden version toimivaksi varmistaminen perustuu järjestelmän *monitorointiin*
- ▶ Esim. sosiaalisen median palvelussa
  - ▶ palvelun muistin ja prosessoriajan kulutusta
  - ▶ verkkoliikenteen määrää
  - ▶ sovelluksen eri sivujen vasteaikojen
  - ▶ kirjautuneiden käyttäjien määrää
  - ▶ luettujen ja lähetetyjen viestien määrä per käyttäjä
  - ▶ kirjautuneen käyttäjän sovelluksessa viettämää aikaa
- ▶ Monitoroidaan palvelimen yleisen toimivuuden lisäksi *käyttäjätason metriikoita* (engl. business level metrics)
- ▶ Jos suuria eroja aiempaan, tehdään **rollback edelliseen versioon**
  - ▶ esim. kirjautuneet käyttäjät eivät lähetä viestejä samaa määrää kuin keskimäärin normaalista

- ▶ Uuden version toimivaksi varmistaminen perustuu järjestelmän *monitorointiin*
- ▶ Esim. sosiaalisen median palvelussa
  - ▶ palvelun muistin ja prosessoriajan kulutusta
  - ▶ verkkoliikenteen määrää
  - ▶ sovelluksen eri sivujen vasteaikoja
  - ▶ kirjautuneiden käyttäjien määrää
  - ▶ luettujen ja lähetetyjen viestien märiä per käyttäjä
  - ▶ kirjautuneen käyttäjän sovelluksessa viettämää aikaa
- ▶ Monitoroidaan palvelimen yleisen toimivuuden lisäksi *käyttäjätason metriikoita* (engl. business level metrics)
- ▶ Jos suuria eroja eroja aiempaan, tehdään **rollback edelliseen versioon**
  - ▶ esim. kirjautuneet käyttäjät eivät lähetä viestejä samaa määrää kuin keskimäärin normaalista
- ▶ Testauksen ja kaikkien tuotantoon vientiin liittyvän on syytä tapahtua automatisoidusti

## Feature toggle

- ▶ *Feature togglejen* avulla voidaan canary releaset toteuttaa käyttämällä yhtä tuotantopalvelinta

## Feature toggle

- ▶ *Feature togglejen* avulla voidaan canary releaset toteuttaa käyttämällä yhtä tuotantopalvelinta
- ▶ Koodiin *ehtolauseita*: osa liikenteestä ohjataan vanhan toteutuksen sijaan testauksen alla olevaan toteutukseen

## Feature toggle

- ▶ *Feature togglejen* avulla voidaan canary releaset toteuttaa käyttämällä yhtä tuotantopalvelinta
- ▶ Koodiin *ehtolauseita*: osa liikenteestä ohjataan vanhan toteutuksen sijaan testauksen alla olevaan toteutukseen
- ▶ Esim. some-palvelussa feature toggle: *osalle käytetään näytetään uuden algoritmin perusteella generoitu lista uutisia*

```
def recommended_news_generator(user):  
    if is_in_canary_release(user):  
        return experimental_recommendation_algorithm(user)  
    else:  
        return recommendation_algorithm(user)
```

## Feature togglejen soveltaminen

- ▶ Aluksi piilotetaan uusi ominaisuus käyttäjiltä feature toggleilla
  - ▶ eli toggle palauttaa vanhan version normaaleille käyttäjille

## Feature togglejen soveltaminen

- ▶ Aluksi piilotetaan uusi ominaisuus käyttäjiltä feature toggleilla
  - ▶ eli toggle palauttaa vanhan version normaaleille käyttäjille
- ▶ Sovellus kehittäjien mahdollista valita kumman version toggle palauttaa

## Feature togglejen soveltaminen

- ▶ Aluksi piilotetaan uusi ominaisuus käyttäjiltä feature toggleilla
  - ▶ eli toggle palauttaa vanhan version normaaleille käyttäjille
- ▶ Sovellus kehittäjien mahdollista valita kumman version toggle palauttaa
- ▶ Kun valmiina laajempaan testiin, julkaistaan esim.
  - ▶ ensin kehittäjäyrityksen omaan käyttöön
  - ▶ sitten osalle käyttäjistä canary releasena

## Feature togglejen soveltaminen

- ▶ Aluksi piilotetaan uusi ominaisuus käyttäjiltä feature toggleilla
  - ▶ eli toggle palauttaa vanhan version normaaleille käyttäjille
- ▶ Sovellus kehittäjien mahdollista valita kumman version toggle palauttaa
- ▶ Kun valmiina laajempaan testiin, julkaistaan esim.
  - ▶ ensin kehittäjäyrityksen omaan käyttöön
  - ▶ sitten osalle käyttäjistä canary releasena
- ▶ Lopulta feature toggle ja vanha toteutus voidaan poistaa

- ▶ Suuret internetpalvelut soveltavat laajalti canary releaseihin ja feature flageihin perustuvaa kehitysmallia
  - ▶ Facebook, Netflix, Google, Flickr, ...
  - ▶ Suomessa esim. Veikkaus

The screenshot shows the top navigation bar of the Netflix mobile website, which includes the Netflix logo, search, language selection (LAPSET), and account settings. Below the header, the main content area has a light gray background. A dark gray callout box contains the text "Osallistun testeihin ja esikatseluihin" and "Poista valinta siirtyäksesi normaalialiin käyttäjäkokemuksen." To the right of this box is a blue toggle switch labeled "KÄYTÖSSÄ". Below the callout box, a message states: "Osallistu testeihin, joilla kehitetään Netflix-käyttäjäkokemusta. Saat nähdäksesi mahdollisia uutuuksia ennen muita." At the bottom left of the content area is a blue button labeled "Valmis".

NETFLIX

Etusivu TV-ohjelmat Elokuvat Hiljattain lisätty Oma lista

LAPSET

Osallistuminen testeihin

Osallistun testeihin ja esikatseluihin  
Poista valinta siirtyäksesi normaalialiin käyttäjäkokemuksen.

KÄYTÖSSÄ

Osallistu testeihin, joilla kehitetään Netflix-käyttäjäkokemusta. Saat nähdäksesi mahdollisia uutuuksia ennen muita.

Valmis

- ▶ Suuret internetpalvelut soveltavat laajalti canary releaseihin ja feature flageihin perustuvaa kehitysmallia
  - ▶ Facebook, Netflix, Google, Flickr, ...
  - ▶ Suomessa esim. Veikkaus

The screenshot shows a mobile version of the Netflix website. At the top, there is a navigation bar with links for 'Etusivu', 'TV-ohjelmat', 'Elokuvat', 'Hiljattain lisätty', and 'Oma lista'. On the right side of the bar are search, user profile ('LAPSET'), and settings icons. Below the navigation bar, the main content area has a title 'Osallistuminen testeihin' (Participate in tests). A grey callout box contains the text 'Osallistun testeihin ja esikatseluihin' (Participate in tests and preview) and 'Poista valinta siirtyäksesi normaalaihin käyttäjäkokemuukseen.' (Remove selection to switch to normal user experience). To the right of this box is a blue toggle switch labeled 'KÄYTÖSSÄ' (On). Below the callout box, a note states 'Osallistu testeihin, joilla kehitetään Netflix-käyttäjäkokemusta. Saat nähdäksesi mahdollisia uutuuksia ennen muita.' (Participate in tests where we develop the Netflix user experience. You will see new features before others). At the bottom left of the content area is a blue button labeled 'Valmis' (Ready).

- ▶ A/B-testaus: arvioidaan onko uusi toteutus parempi kuin vanha

- ▶ Suuret internetpalvelut soveltavat laajalti canary releaseihin ja feature flageihin perustuvaa kehitysmallia
  - ▶ Facebook, Netflix, Google, Flickr, ...
  - ▶ Suomessa esim. Veikkaus

The screenshot shows the top navigation bar of the Netflix mobile website, featuring the Netflix logo, search icon, 'LAPSET' (Children), and a user profile icon. Below the navigation is a dark header bar with links for 'Etusivu', 'TV-ohjelmat', 'Elokuvat', 'Hiljattain lisätty', and 'Oma lista'. The main content area has a light gray background. A section titled 'Osallistuminen testeihin' (Participate in tests) is displayed. It contains a message: 'Osallistun testeihin ja esikatseluihin' (Participate in tests and preview sessions) and 'Poista valinta siirtyäksesi normaalaihin käyttäjäkokemuukseen.' (Remove selection to switch to normal user experience). To the right is a blue toggle switch labeled 'KÄYTÖSSÄ' (On). Below this section, a note reads: 'Osallistu testeihin, joilla kehitetään Netflix-käyttäjäkokemusta. Saat nähdäksesi mahdollisia uutuuksia ennen muita.' (Participate in tests where we develop the Netflix user experience. You will see new features before others). At the bottom left is a blue button labeled 'Valmis' (Ready).

- ▶ A/B-testaus: arvioidaan onko uusi toteutus parempi kuin vanha
- ▶ Kerrallaan voi olla menossa useita kymmeniä A/B-testattavia eksperimenttejä

# Loppupäätelmiä testauksesta

## Loppupäätelmiä testauksesta

- ▶ Ketterissä menetelmissä kantavana teemana on *arvon tuottaminen asiakkaalle*
  - ▶ Sopii ohjeeksi myös arvioitaessa testauksen laajuutta
  - ▶ Testauksella ei ole itseisarvoista merkitystä
  - ▶ Testaamattomuus alkaa pian heikentää tuotteen laatua liikaa

## Loppupäätelmiä testauksesta

- ▶ Ketterissä menetelmissä kantavana teemana on *arvon tuottaminen asiakkaalle*
  - ▶ Sopii ohjeeksi myös arvioitaessa testauksen laajuutta
  - ▶ Testauksella ei ole itseisarvoista merkitystä
  - ▶ Testaamattomuus alkaa pian heikentää tuotteen laatua liikaa
- ▶ Testausta ja laadunhallintaa on tehtävä paljon ja toistuvasti eli automatisointi on yleensä pidemmällä tähtäimellä kannattavaa

## Loppupäätelmiä testauksesta

- ▶ Ketterissä menetelmissä kantavana teemana on *arvon tuottaminen asiakkaalle*
  - ▶ Sopii ohjeeksi myös arvioitaessa testauksen laajuutta
  - ▶ Testauksella ei ole itseisarvoista merkitystä
  - ▶ Testaamattomuus alkaa pian heikentää tuotteen laatua liikaa
- ▶ Testausta ja laadunhallintaa on tehtävä paljon ja toistuvasti eli automatisointi on yleensä pidemmällä tähtäimellä kannattavaa
- ▶ Automatisointi ei ole halpaa eikä helppoa
  - ▶ Väärin, väärään aikaan tai väärälle tasolle tehdyt automatisoidut testit voivat tuottaa enemmän harmia ja kustannuksia kuin hyötyä

- ▶ Jos ohjelmistossa komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoida
  - ▶ esim. jos kyseessä *minimal viable product*

- ▶ Jos ohjelmistossa komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoida
  - ▶ esim. jos kyseessä *minimal viable product*
- ▶ Väliaikaiseksi tarkoitettu komponentti voi jäädä järjestelmään vuosiksi...

- ▶ Jos ohjelmistossa komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoida
  - ▶ esim. jos kyseessä *minimal viable product*
- ▶ Väliaikaiseksi tarkoitettu komponentti voi jäädä järjestelmään vuosiksi...
- ▶ Kokonaan uutta ohjelmistoa tai komponenttia tehtäessä kannattaa ohjelman rakenteen ensin antaa stabiloitua, kattavammat testit vasta myöhemmin

- ▶ Jos ohjelmistossa komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoida
  - ▶ esim. jos kyseessä *minimal viable product*
- ▶ Väliaikaiseksi tarkoitettu komponentti voi jäädä järjestelmään vuosiksi...
- ▶ Kokonaan uutta ohjelmistoa tai komponenttia tehtäessä kannattaa ohjelman rakenteen ensin antaa stabiloitua, kattavammat testit vasta myöhemmin
- ▶ *Testattavuus* tulee pitää koko ajan mielessä

- ▶ Oppikirjamääritelmän mukaista TDD:tä sovelletaan harvoin
  - ▶ Välillä TDD hyödyllinen, esim. testattaessa rajapintoja, joita käyttäviä komponentteja ei ole vielä olemassa
  - ▶ Testit tekee samalla vaivalla kuin “pääohjelman”

- ▶ Oppikirjamääritelmän mukaista TDD:tä sovelletaan harvoin
  - ▶ Välillä TDD hyödyllinen, esim. testattaessa rajapintoja, joita käyttäviä komponentteja ei ole vielä olemassa
  - ▶ Testit tekee samalla vaivalla kuin “pääohjelman”
- ▶ Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille

- ▶ Oppikirjamääritelmän mukaista TDD:tä sovelletaan harvoin
  - ▶ Välillä TDD hyödyllinen, esim. testattaessa rajapintoja, joita käyttäviä komponentteja ei ole vielä olemassa
  - ▶ Testit tekee samalla vaivalla kuin “pääohjelman”
- ▶ Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- ▶ Yksikkötestaus hyödyllisimmillään kompleksia logiikkaa sisältäviä luokkia testattaessa

- ▶ Oppikirjamääritelmän mukaista TDD:tä sovelletaan harvoin
  - ▶ Välillä TDD hyödyllinen, esim. testattaessa rajapintoja, joita käyttäviä komponentteja ei ole vielä olemassa
  - ▶ Testit tekee samalla vaivalla kuin “pääohjelman”
- ▶ Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- ▶ Yksikkötestaus hyödyllisimmillään kompleksia logiikkaa sisältäviä luokkia testattaessa
- ▶ Mielummin integraatiotason testejä ohjelman isompien komponenttien rajapintoja vasten
  - ▶ Pysyvät todennäköisemmin valideina komponenttien sisäisen rakenteen muuttuessa

- ▶ Automaattisia testejä kannattaa tehdä etenkin niiden komponenttien rajapintoihin, joita muokataan usein

- ▶ Automaattisia testejä kannattaa tehdä etenkin niiden komponenttien rajapointoihin, joita muokataan usein
- ▶ Käyttöliittymän läpi suoritettavat, käyttäjän interaktioita simuloivat testit usein hyödyllisimpää
  - ▶ Liian aikaisin tehtynä ne saattavat aiheuttaa kohtuuttoman paljon ylläpitovaivaa

- ▶ Testitapauksista kannattaa aina tehdä todellisia käyttöskenaarioita vastaavia
  - ▶ Pelkkiä testauskattavuutta kasvattavia testejä on turha tehdä

- ▶ Testitapauksista kannattaa aina tehdä todellisia käyttöskenaarioita vastaavia
  - ▶ Pelkkiä testauskattavuutta kasvattavia testejä on turha tehdä
- ▶ Erityisesti järjestelmätason testeissä kannattaa käyttää mahdollisimman oikeanlaista dataa
  - ▶ Koodissa hajoaa aina jotain kun käytetään oikeaa dataa riippumatta siitä miten hyvin testaus on suoritettu

- ▶ Testitapauksista kannattaa aina tehdä todellisia käyttöskenaarioita vastaavia
  - ▶ Pelkkiä testauskattavuutta kasvattavia testejä on turha tehdä
- ▶ Erityisesti järjestelmätason testeissä kannattaa käyttää mahdollisimman oikeanlaista dataa
  - ▶ Koodissa hajoaa aina jotain kun käytetään oikeaa dataa riippumatta siitä miten hyvin testaus on suoritettu
- ▶ Parasta on jos staging-ympäristössä on käytössä sama *data kuin* tuotantoymäristössä

- ▶ Ehdottomasti kaikkein tärkein laadunhallinnan kannalta on **mahdollisimman usein tapahtuva käyttöönotto**
  - ▶ edellyttää hyvin rakennettua deployment pipelineä, kohtuullista testauksen automatisointia

- ▶ Ehdottomasti kaikkein tärkein laadunhallinnan kannalta on **mahdollisimman usein tapahtuva käyttöönotto**
  - ▶ edellyttää hyvin rakennettua deployment pipelineä, kohtuullista testauksen automatisointia
- ▶ Trunk based development auttaa nopeaa käyttöönottoa feature brancheihin verrattuna

- ▶ Ehdottomasti kaikkein tärkein laadunhallinnan kannalta on **mahdollisimman usein tapahtuva käyttöönotto**
  - ▶ edellyttää hyvin rakennettua deployment pipelineä, kohtuullista testauksen automatisointia
- ▶ Trunk based development auttaa nopeaa käyttöönottoa feature brancheihin verrattuna
- ▶ Suosittelen että käyttöönotto tapahtuu niin usein kuin mahdollista, jopa useita kertoja päivässä
  - ▶ takaa sen, että pahoja integrointiongelmia ei synny
  - ▶ sovellukseen syntyvät regressiot havaitaan ja pystytään korjaamaan mahdollisimman nopeasti

- ▶ Ehdottomasti kaikkein tärkein laadunhallinnan kannalta on **mahdollisimman usein tapahtuva käyttöönotto**
  - ▶ edellyttää hyvin rakennettua deployment pipelineä, kohtuullista testauksen automatisointia
- ▶ Trunk based development auttaa nopeaa käyttöönottoa feature brancheihin verrattuna
- ▶ Suosittelen että käyttöönotto tapahtuu niin usein kuin mahdollista, jopa useita kertoja päivässä
  - ▶ takaa sen, että pahoja integrointiongelmia ei synny
  - ▶ sovellukseen syntyvät regressiot havaitaan ja pystytään korjaamaan mahdollisimman nopeasti
- ▶ Nopea käyttöönotto **pakottaa** laatuun