Ohjelmistotuotanto

Matti Luukkainen ja ohjaajat Kalle Ilves, Antti Kantola, Riikka Korolainen, Touko Puro

syksy 2021

Luento 6

16.11.2021

Miniprojektit lähestyvät

- aloitus 22.11. alkavalla viikolla
- ilmoittautuminen alkanut, päättyy perjantaina
- pakollinen, jos et hyväksilue

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 suunnitteluun 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 suunnitteluun ja toteutukseen
- Automatisointi erittäin tärkeässä roolissa, sillä testejä suoritetaan usein

Testaajat osana kehitystiimiä

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

Testaajat osana kehitystiimiä

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

Ketterien menetelmien testauskäytänteet

- Test driven development (TDD)
  - Nimestä huolimatta kyseessä toteutus- ja suunnittelutekniikka
  - sivutuotteena paljon automaattisesti suoritettavia testejä

Ketterien menetelmien testauskäytänteet

- Test driven development (TDD)
  - Nimestä huolimatta kyseessä toteutus- ja suunnittelutekniikka
  - sivutuotteena paljon automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus

Ketterien menetelmien testauskäytänteet

- Test driven development (TDD)
  - Nimestä huolimatta kyseessä toteutus- ja suunnittelutekniikka
  - sivutuotteena paljon automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus
- Continuous Integration (CI) eli jatkuva integraatio
  - Perinteisen integraatio- ja integraatiotestausvaiheen korvaava työskentelytapa

Ketterien menetelmien testauskäytänteet

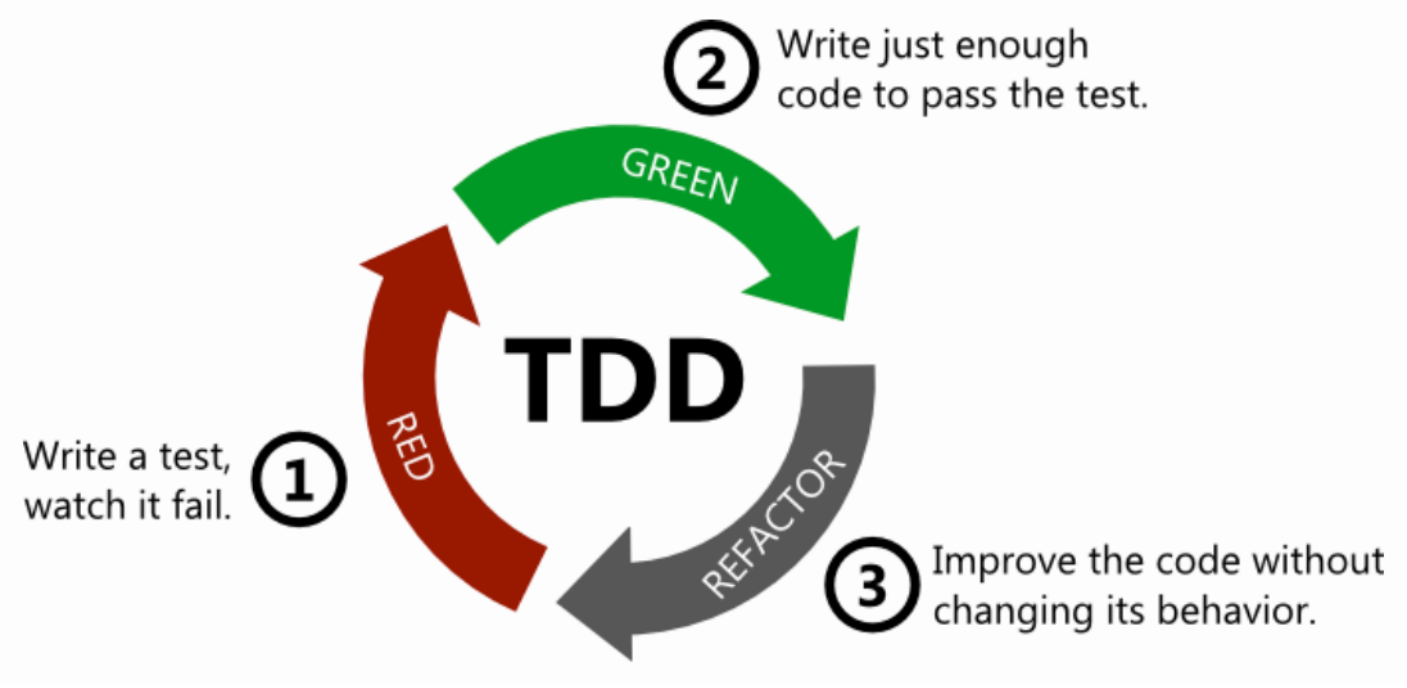
- Test driven development (TDD)
  - Nimestä huolimatta kyseessä toteutus- ja suunnittelutekniikka
  - sivutuotteena paljon automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus
- Continuous Integration (CI) eli jatkuva integraatio
  - Perinteisen integraatio- ja integraatiotestausvaiheen korvaava työskentelytapa
- Exploratory testing, suomeksi tutkiva testaus
  - Järjestelmätestauksen tekniikka, jossa testaaminen tapahtuu ilman formaalia testaussuunnitelmaa

Ketterien menetelmien testauskäytänteet

- Test driven development (TDD)
  - Nimestä huolimatta kyseessä toteutus- ja suunnittelutekniikka
  - sivutuotteena paljon automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus
- Continuous Integration (CI) eli jatkuva integraatio
  - Perinteisen integraatio- ja integraatiotestausvaiheen korvaava työskentelytapa
- Exploratory testing, suomeksi tutkiva testaus
  - Järjestelmätestauksen tekniikka, jossa testaaminen tapahtuu ilman formaalia testaussuunnitelmaa
- Tuotannossa tapahtuva testaus
  - Nouseva trendi suorittaa uusien ominaisuuksien laadunhallintaa siinä vaiheessa kun osa käyttäjistä ottanut ne käyttöönsä

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 koodataan, 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 koostuvaa
  - TDD:llä koodattu 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 koostuvaa
  - TDD:llä koodattu laadukasta ylläpidettävyyden ja laajennettavuuden kannalta
- Muita TDD:n hyviä puolia:
  - Rohkaisee ottamaan pieniä askelia kerrallaan ja toimimaan fokusoidusti
  - Virheet havaitaan nopeasti suuren testijoukon takia
  - Hyvin kirjoitetut testit toimivat toteutetun komponentin rajapinnan dokumentaationa

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:n käyttö on haastavaa 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 normaalisti tehtyä koodia lyhempi
- Jos ja kun koodi muuttuu, tulee testejä ylläpitää
- TDD:n käyttö on haastavaa mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä
- Legacy-koodin laajentaminen TDD:llä voi olla haastavaa

17.1.2022-

17.1.2022-

Open Uni: Test-Driven Development 4 + 1 cr

Riippuvuudet testeissä

Riippuvuudet testeissä

```
class Laskin:
    def suorita(self):
        while True:
            luku1 = int(input("Luku 1:"))

            if luku1 == -9999:
                return

            luku2 = int(input("Luku 2:"))

            if luku2 == -9999:
                return

            vastaus = self._laske_summa(luku1, luku2)

            print(f"Summa: {vastaus}")

    def _laske_summa(self, luku1, luku2):
        return luku1 + luku2
```

Riippuvuudet testeissä

```
class Laskin:
    def suorita(self):
        while True:
            luku1 = int(input("Luku 1:"))

            if luku1 == -9999:
                return

            luku2 = int(input("Luku 2:"))

            if luku2 == -9999:
                return

            vastaus = self._laske_summa(luku1, luku2)

            print(f"Summa: {vastaus}")

    def _laske_summa(self, luku1, luku2):
        return luku1 + luku2
```

Riippuvuudet testeissä: dependency injection

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

    def suorita(self):
        while True:
            luku1 = int(self._io.lue("Luku 1:"))

            if luku1 == -9999:
                return

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

            if luku2 == -9999:
                return

            vastaus = self._laske_summa(luku1, luku2)

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

    def _laske_summa(self, luku1, luku2):
        return luku1 + luku2
```

Riippuvuudet testeissä: valeriippuvuudet

- Dependency Injection -suunnittelumalli mahdollistaa stubien eli “vale”-riippuvuuksien asettamisen luokille testistä käsin

Riippuvuudet testeissä: valeriippuvuudet

- Dependency Injection -suunnittelumalli mahdollistaa stubien eli “vale”-riippuvuuksien asettamisen luokille testistä käsin
- Stubeihin voidaan kovakoodata metodikutsujen tulokset valmiiksi
- Testi voi kysellä stubilta millä arvoilla sitä kutsuttiin
  - tällaisia stubeja kutsutaan *mock*-olioiksi

Riippuvuudet testeissä: valeriippuvuudet

- Dependency Injection -suunnittelumalli mahdollistaa stubien eli “vale”-riippuvuuksien asettamisen luokille testistä käsin
- Stubeihin voidaan kovakoodata metodikutsujen tulokset valmiiksi
- Testi voi kysellä stubilta millä arvoilla sitä kutsuttiin
  - tällaisia stubeja kutsutaan *mock*-olioiksi

```
class TestLaskin(unittest.TestCase):
    def test_yksi_summa_oikein(self):
        io = StubIO(["1", "3", "-9999"])
        laskin = Laskin(io)
        laskin.suorita()

        self.assertEqual(io.outputs[0], "Summa: 4")
```

Itse toteutettu stub/mock

```
class StubIO:
    def __init__(self, inputs):
        self.inputs = inputs
        self.outputs = []

    def lue(self, teksti):
        return self.inputs.pop(0)

    def kirjoita(self, teksti):
        self.outputs.append(teksti)
```

Mockito

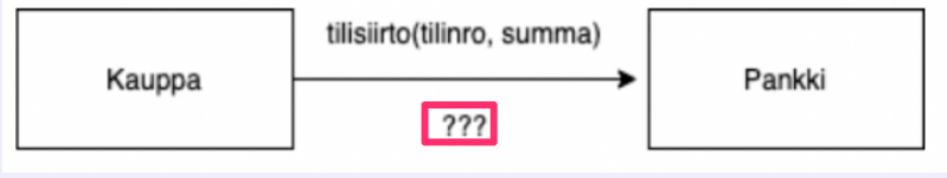
- Olemassa useita kirjastoja mock-olioiden luomisen helpottamiseksi, laskareissa *mock* (Python) *Mockito* (Java)

Mockito

- Olemassa useita kirjastoja mock-olioiden luomisen helpottamiseksi, laskareissa *mock* (Python) *Mockito* (Java)
- 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")
```



```
def test_kutsutaan_pankkia_oikealla_tilinumerolla_ja_summalla(self):
    pankki_mock = Mock()
    viitegeneraattori_mock = Mock(wraps=Viitegeneraattori())

    kauppa = Kauppa(pankki_mock, viitegeneraattori_mock)

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

    # katsotaan, että ensimmäisen ja toisen parametrin arvo on oikea
    pankki_mock.maksa.assert_called_with("1111", 10, ANY)
```

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 tuotelistauksessa 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 tuotelistauksessa 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

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 tuotelistauksessa 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
- 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

- Ideaalitilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- Olemassa monia työkaluja
  - eräs suosituimmista on suomalainen python-pohjainen *Robot framework* jota kurssin Python-versio käyttää
  - Kurssin Java-versiossa käytössä *Cucumber*

Järjestelmätestauksen automatisointi, ATDD ja BDD

- Ideaalitilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- Olemassa monia työkaluja
  - eräs suosituimmista on suomalainen python-pohjainen *Robot framework* jota kurssin Python-versio käyttää
  - Kurssin Java-versiossa käytössä *Cucumber*
- Käytetään nimitystä *Acceptance test driven development* (ATDD) tai *Behavior driven development* (BDD)
  - erityisesti jos testit toteutetaan jo iteraation alkupuolella, ennen kun story koodattu

Järjestelmätestauksen automatisointi, ATDD ja BDD

- Ideaalitilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- Olemassa monia työkaluja
  - eräs suosituimmista on suomalainen python-pohjainen *Robot framework* jota kurssin Python-versio käyttää
  - Kurssin Java-versiossa käytössä *Cucumber*
- 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-* päätteinen 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-* päätteinen tiedosto
  - sisältää joukon storyyn 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 mäpätään 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 mäpätään 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
  - *headless-selaimet* toinen vaihtoehto

Web-sovellusten testaaminen onnistuu Selenium:illa

```
*** Variables ***
${SERVER}  localhost:5000
${BROWSER}  chrome
${HOME URL}  http://${SERVER}
${LOGIN URL}  http://${SERVER}/login

*** Keywords ***
Open And Configure Browser
    Open Browser  browser=${BROWSER}
    Maximize Browser Window
    Set Selenium Speed  ${DELAY}

Login Page Should Be Open
    Title Should Be  Login

Login Should Fail With Message
    [Arguments]  ${message}
    Login Page Should Be Open
    Page Should Contain  ${message}

Go To Login Page
    Go To  ${LOGIN URL}
```

- Robot sisältää runsaasti valmiiksi määriteltyjä avainsanoja

Motivaatio käyttäjän kielellä kirjoitetuille testeille

- Product owner kirjoittaa tiimin kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat ymmärtävät mistä on kyse

Motivaatio käyttäjän kielellä kirjoitetuille testeille

- Product owner kirjoittaa tiimin kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat ymmärtävät mistä on kyse
- Koodaajat/testaajat toteuttavat mäppäyksen, joka automatisoi testien suorituksen

Motivaatio käyttäjän kielellä kirjoitetuille testeille

- Product owner kirjoittaa tiimin kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat ymmärtävät mistä on kyse
- Koodaajat/testaajat toteuttavat mäppäyksen, joka automatisoi testien suorituksen
- Ei toistaiseksi vielä kovin yleinen tyyli, useimmiten hyväksymätestit kirjoitettu suoraan “normalilla” testikirjastolla
  - jUnit, Mocha, Jest, rspec ...

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integrointi
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - yhteistoiminnallisuus varmistetaan integraatiotestein

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integrointi
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - yhteistoiminnallisuus varmistetaan integraatiotestein
- Perinteisesti integrointivaihe on tuonut esiin suuren joukon ongelmia
- Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensopivia

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integrointi
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - yhteistoiminnallisuus varmistetaan integraatiotestein
- Perinteisesti integrointivaihe on tuonut esiin suuren joukon ongelmia
- Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensopivia
- Suurten projektien integrointivaihe on kestänyt ennakoimattoman kauan
- Integrointivaiheen ongelmat ovat aiheuttaneet ohjelmaan suunnittelutason muutoksia

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integrointi
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - yhteistoiminnallisuus varmistetaan integraatiotestein
- Perinteisesti integrointivaihe on tuonut esiin suuren joukon ongelmia
- Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensopivia
- Suurten projektien integrointivaihe on kestänyt 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
- 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

Pois integraatiohelvetistä

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

Pois integraatiohelvetistä

- 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- 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 yhteensopivuusongelmat huomataan nopeasti ja niiden korjaaminen helpottuu

Pois integraatiohelvetistä

- 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- 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 yhteensopivuusongelmat 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
- *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman läheisesti tuotantopalvelinta

Päivittäisestä jatkuvaan integraatioon

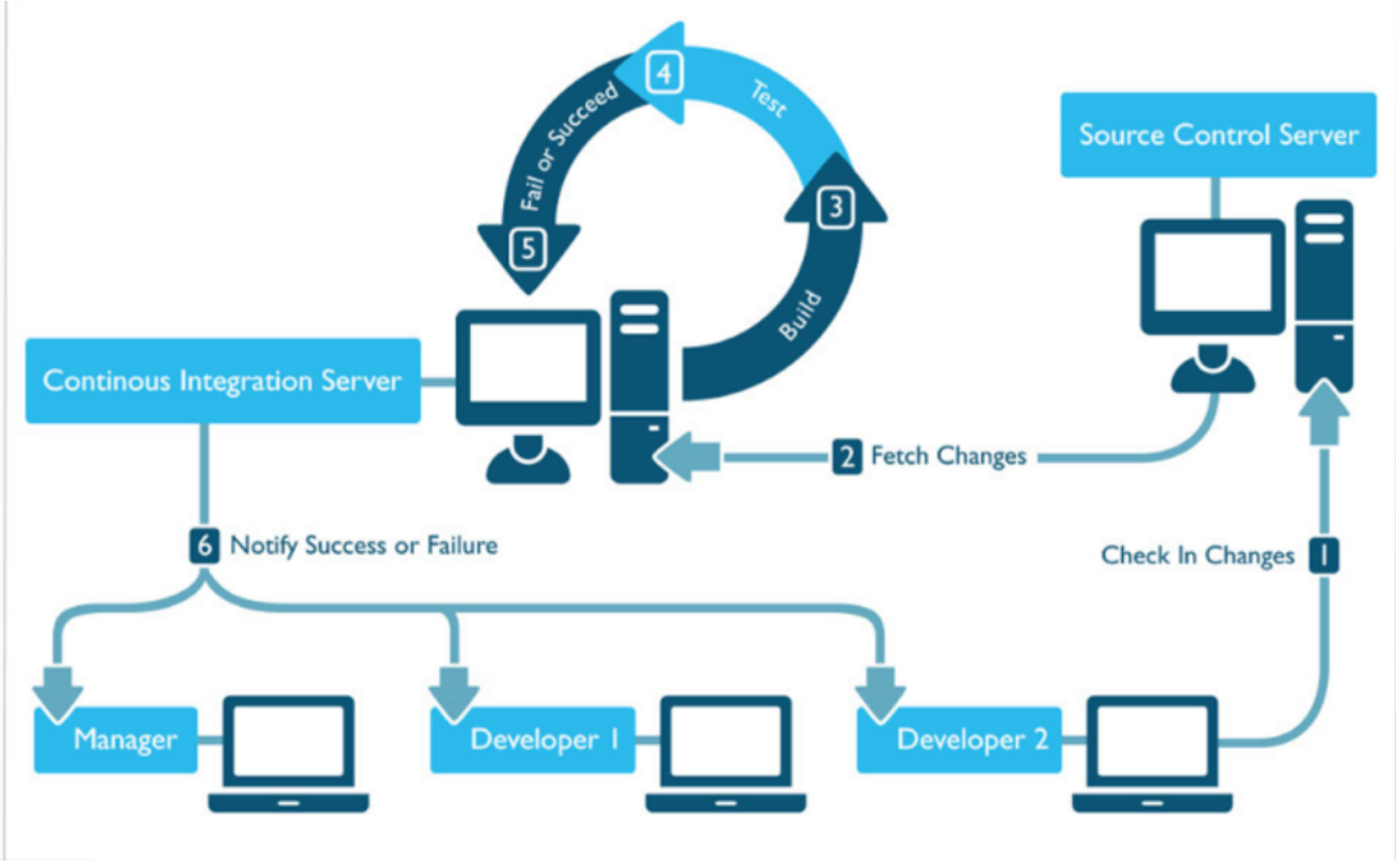
- Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - eräs XP:n käytenäteistä
- Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositoriossa
- *CI-palvelin*: vastaa konfiguraatioilta mahdollisimman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


Source Control Server
Continous Integration Server
Fail or Succeed
4
Test
3
Build
5
2 Fetch Changes
6 Notify Success or Failure
Check In Changes 1
Manager
Developer 1
Developer 2

- 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
- Suorittaa testit paikallisesti, ja jos ne menevät läpi, pushaa koodin CI-palvelimelle
  - Näin minimoituu mahdollisuus, että lisätty koodi toimii ainoastaan kehittäjän koneella

- Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- Suorittaa testit paikallisesti, ja jos ne menevät läpi, pushaa koodin CI-palvelimelle
  - Näin minimoituu mahdollisuus, että lisätty koodi toimii ainoastaan kehittäjän koneella
- Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, min kerran päivässä

- Sovelluskehittäjä aloittaa työskentelyn hakemalla koodin uusimman version versionhallinnasta
- Kehittäjä integroi koodinsa heti muuhun koodiin ja tekee riittävän määrän automatisoituja testejä
- Suorittaa testit paikallisesti, ja jos ne menevät läpi, pushaa koodin CI-palvelimelle
  - Näin minimoituu mahdollisuus, että lisätty koodi toimii ainoastaan kehittäjän koneella
- Tarkoitus että jokainen kehittäjä integroi koodinsa muuhun koodiin mahdollisimman usein, min kerran päivässä
- CI rohkaisee jakamaan työn pieniin osiin, sellaisiin jotka saadaan testeineen valmiiksi yhden työpäivän aikana
- CI-työprosessin noudattaminen vaatii kurinalaisuutta

- Jotta CI-prosessi toimisi jouhevasti, tulee testien ajamisen tapahtua nopeasti, yli 10 min alkaa olla liikaa
- Jos osa testeistä on hitaita, voidaan testit konfiguroida ajettavaksi kahdessa (tai useammassa) vaiheessa
  - *commit build*:in läpimeno antaa kehittäjälle oikeuden pushata koodi repositorioon
  - CI-palvelimella suoritetaan myös hitaammat testit sisältävä *secondary build*

- Jotta CI-prosessi toimisi jouhevasti, tulee testien ajamisen tapahtua nopeasti, yli 10 min alkaa olla liikaa
- Jos osa testeistä on hitaita, voidaan testit konfiguroida ajettavaksi kahdessa (tai useammassa) vaiheessa
  - *commit build*:in läpimeno antaa kehittäjälle oikeuden pushata koodi repositorioon
  - CI-palvelimella suoritetaan myös hitaammat testit sisältävä *secondary build*
- Laskareissa käytetty *GitHub Actions* on yksi monista SaaS-palveluna toimivista CI-ratkaisuista, monien mielestä tämän hetken paras
  - CircleCI ja Travis muita yleisesti käytettyjä

- Jotta CI-prosessi toimisi jouhevasti, tulee testien ajamisen tapahtua nopeasti, yli 10 min alkaa olla liikaa
- Jos osa testeistä on hitaita, voidaan testit konfiguroida ajettavaksi kahdessa (tai useammassa) vaiheessa
  - *commit build*:in läpimeno antaa kehittäjälle oikeuden pushata koodi repositorioon
  - CI-palvelimella suoritetaan myös hitaammat testit sisältävä *secondary build*
- Laskareissa käytetty *GitHub Actions* on yksi monista SaaS-palveluna toimivista CI-ratkaisuista, monien mielestä tämän hetken paras
  - CircleCI ja Travis muita yleisesti käytettyjä
- Näitä paljon vanhempi Jenkins lienee edelleen maailmalla eniten käytetty CI-palvelinohjelmisto
  - Jenkinsin käyttö edellyttää sen asentamista omalle palvelimelle

Deployment stagingiin

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

Deployment stagingiin

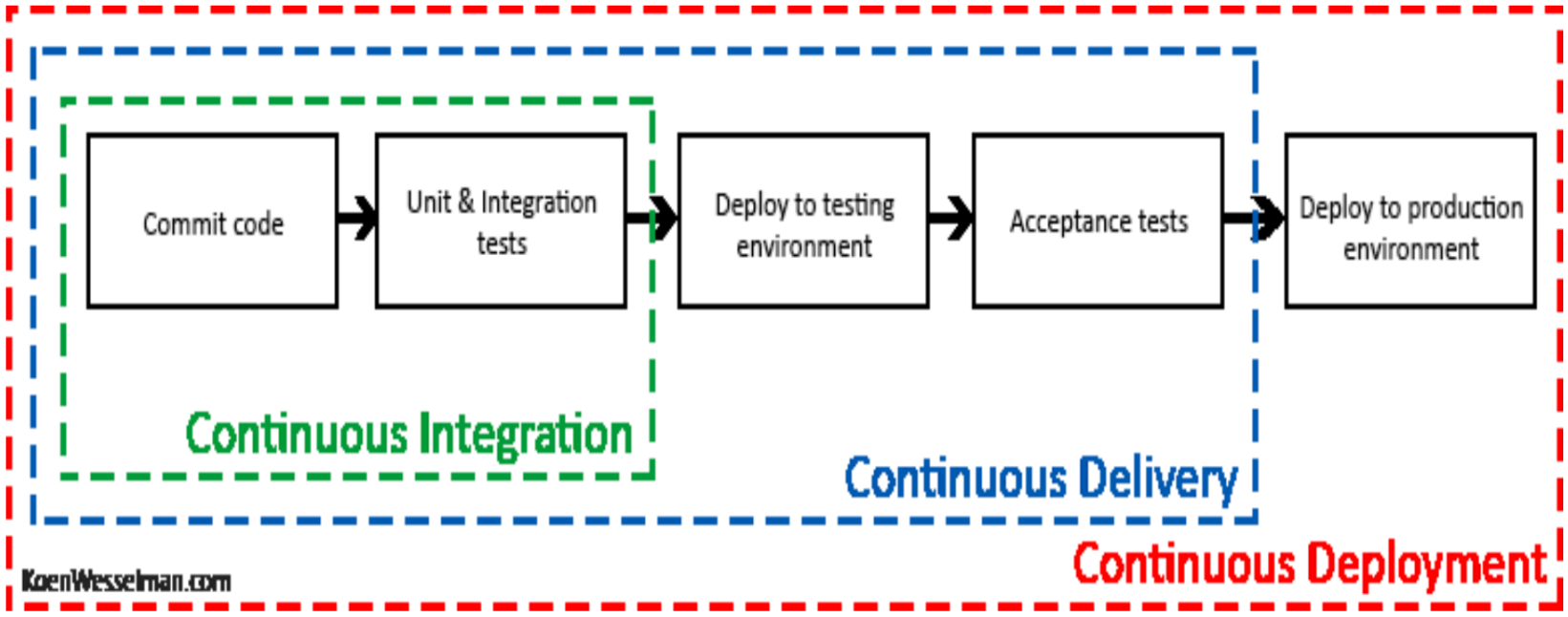
- Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- Integraatioprosessiin lisätään automaattinen *deployaus* staging-palvelimelle
  - Ympäristö, joka kaikin tavoin mahdollisimman lähellä tuotantoympäristöä
- Kun uusi versio deployattu staging-palvelimelle, suoritetaan sille hyväksymistestaus
- jonka jälkeen uusi versio voidaan siirtää *tuotantopalvelimelle*

Deployment stagingiin

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

Deployment pipeline

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



Jatkuva toimitusvalmius ja käyttöönotto

- Jos jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon, puhutaan *jatkuvasta käyttöönotosta* engl. *continuous deployment*

Jatkuva toimitusvalmius ja käyttöönotto

- Jos jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon, puhutaan *jatkuvasta käyttöönotosta* engl. *continuous deployment*
- Jos deployment-päätös tehdään ihmisen toimesta, sovelletaan *jatkuvaa toimitusvalmiuutta* engl. *continuous delivery*
  - on olemassa hyviä syitä miksi ihan kaikkea ei haluta heti julkaista loppukäyttäjille asti

Jatkuva toimitusvalmius ja käyttöönotto

- Jos jokainen testit läpäisevä commit päätyy automaattisesti tuotantoon, puhutaan *jatkuvasta käyttöönotosta* engl. *continuous deployment*
- Jos deployment-päätös tehdään ihmisen toimesta, sovelletaan *jatkuvaa toimitusvalmiuutta* engl. *continuous delivery*
  - on olemassa hyviä syitä miksi ihan kaikkea ei haluta heti julkaista loppukäyttäjille asti
- Viime aikojen trendi julkaista web-palvelusta jopa kymmeniä uusia versiota päivästä
  - Amazon, Netflix, Facebook, Smartly...

Tutkiva testaaminen

- Jotta järjestelmä saadaan riittävän virheettömäksi, on testauksen suoritettava erittäin perusteellisesti

Tutkiva testaaminen

- Jotta järjestelmä saadaan riittävän virheettömäksi, on testauksen 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 testauksen 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

Test Scenario ID	Login-1	Test Case ID	Login-1B
Test Case Description	Login – Negative test case	Test Priority	High
Pre-Requisite	NA	Post-Requisite	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 10/17/2017 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 10/17/2017 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
- Ketterässä ohjelmistotuotannossa tavoite voi jäsentyä muutaman user storyn toiminnallisuuden ympärille
  - *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 kohteen ulkopuoleisiin asioihin
  - Klikkaillaan käyttöliittymän nappuloita epäloogisissa tilanteissa
  - Jos huomataan selaimen osoiterivillä URL https://www.webshopshop.com/ostoskori?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 kohteen ulkopuoleisiin asioihin
  - Klikkaillaan käyttöliittymän nappuloita epäloogisissa tilanteissa
  - Jos huomataan selaimen osoiterivillä URL https://www.webshopshop.com/ostoskori?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
- Sen avulla kannattaa ensisijaisesti testata sprintin yhteydessä toteutettuja uusia ominaisuuksia

Tutkiva testaaminen

- Löydettyjen virheiden toistuminen jatkossa kannattaa eliminoida tekemällä automatisoituja regressiotestejä
- Tutkivaa testaamista ei kannata käyttää regressiotestaamisen menetelmänä
- Sen avulla kannattaa ensisijaisesti testata sprintin yhteydessä toteutettuja uusia ominaisuuksia
- Tutkiva testaaminen siis ei ole vaihtoehto normaaleille tarkkaan etukäteen määritellyille testeille vaan niitä täydentävä testauksen muoto