Ohjelmistotuotanto

Matti Luukkainen ja ohjaajat Jami Kousa, Tero Tapio, Mauri Karlin

syksy 2019

Luento 6

12.11.2019

Ketterien menetelmien testauskäytänteet

- Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallisesta ohjelmistotuotannosta
  - 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 vesiputousmallisesta ohjelmistotuotannosta
  - 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

Ketterien menetelmien testauskäytänteet

- Testauksen rooli ketterissä menetelmissä poikkeaa huomattavasti vesiputousmallisesta ohjelmistotuotannosta
  - 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
- Ideaalitilanteessa testaajia sijoitettu kehittäjätiimiin, ja myös ohjelmoijat kirjoittavat testejä
- *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) -Nimestään huolimatta kyseessä suunnittelu- ja toteutustason tekniikka
  - “sivutuotteena” syntyy kattava joukko automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus

Ketterien menetelmien testauskäytänteet

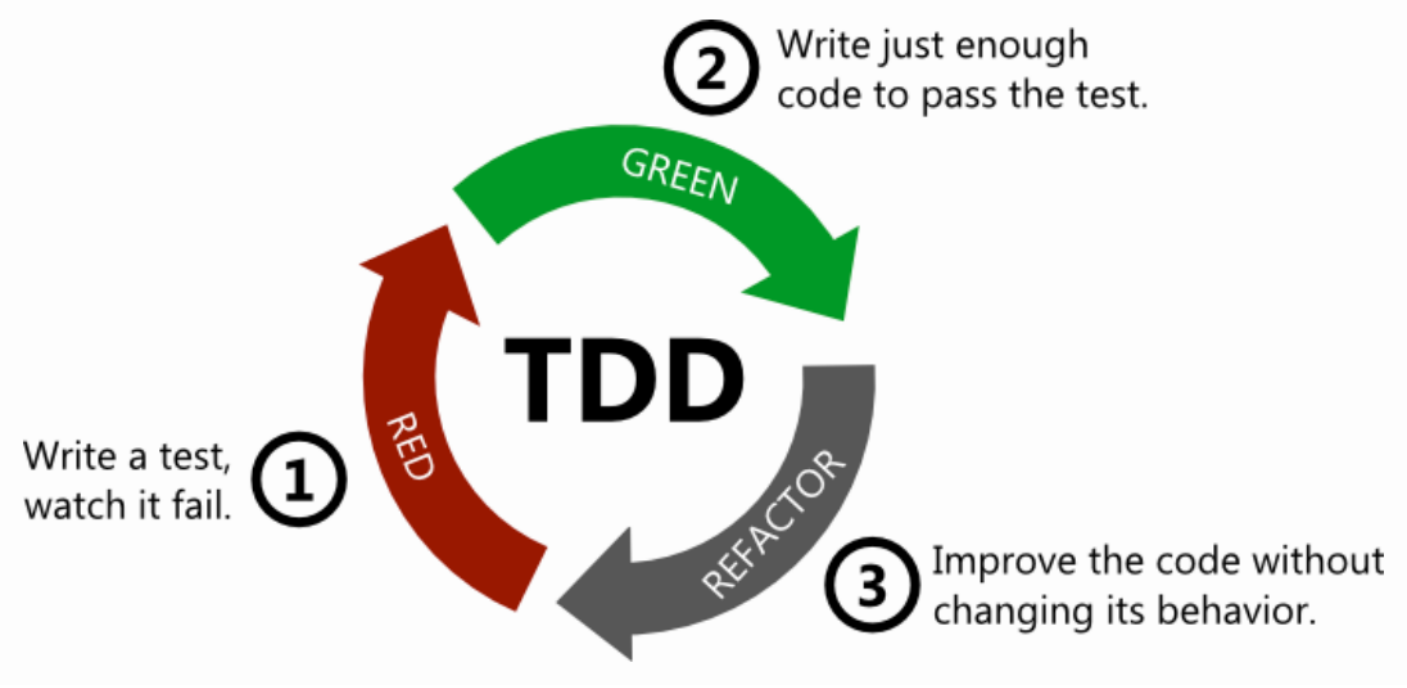
- Test driven development (TDD) -Nimestään huolimatta kyseessä suunnittelu- ja toteutustason tekniikka
  - “sivutuotteena” syntyy kattava joukko automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu testaus
- 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) -Nimestään huolimatta kyseessä suunnittelu- ja toteutustason tekniikka
  - “sivutuotteena” syntyy kattava joukko 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 on suorittaa uusien ominaisuuksien laadunhallintaa siinä vaiheessa kun osa oikeista käyttäjistä on jo 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
- Testit kirjoitetaan ensisijaisesti ajatellen komponentin käyttäjää
  - huomio on komponentin rajapinnassa ja rajapinnan 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 käyttäjää
  - huomio on komponentin rajapinnassa ja rajapinnan helppokäyttöisyydessä, ei niinkään komponentin sisäisessä toteutuksessa
- Komponentin sisäinen rakenne muotoutuu refaktorointien kautta
- Ensin testataan, sitten koodataan, suunnittelu vasta lopussa

- TDD:tä tehtäessä 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:tä tehtäessä 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 tehty koodi on yleensä 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 (mutta ei mahdotonta) mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä
- Legacy-koodin laajentaminen TDD:llä voi olla haastavaa

Riippuvuudet testeissä

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

```
public class LaskinTest {

    @Test
    public void yksiSummaOikein() {
        IOStub io = new IOStub(1, 3, -9999);
        new Laskin(io).suorita();

        assertEquals("summa: 4\n", io.outputs.get(2));
    }
}
```

Itse toteutettu stub/mock

```
class IOStub implements IO {

    int[] inputs;
    int mones;
    ArrayList<String> outputs;

    public IOStub(int... inputs) {
        this.inputs = inputs;
        this.outputs = new ArrayList<String>();
    }

    public int nextInt() {
        return inputs[mones++];
    }

    public void print(String m) {
        outputs.add(m);
    }
}
```

Mockito

- Olemassa useita kirjastoja mock-olioiden luomisen helpottamiseksi, laskareissa *Mockito*
- Kaupan metodin *maksa* pitää tehdä *tilisiirto* kutsumalla *Pankin* metodia

```
Pankki myNetBank = new Pankki();

Viitegeneraattori viitteet = new Viitegeneraattori();

Kauppa kauppa = new Kauppa(myNetBank, viitteet);


kauppa.aloitaOstokset();

kauppa.lisaaOstos(5);

kauppa.lisaaOstos(7);

kauppa.maksa("1111");
```

```
@Test
public void kutsutaanPankkiaOikeallaTilinumerollaJaSummalla() {
    Pankki mockPankki = mock(Pankki.class);
    Viitegeneraattori mockViite = mock(Viitegeneraattori.class);
    kauppa = new Kauppa(mockPankki, mockViite);


    kauppa.aloitaOstokset();
    kauppa.lisaaOstos(5);
    kauppa.lisaaOstos(5);
    kauppa.maksa("1111");


    verify(mockPankki).tilisiirto(eq("1111"), eq(10), anyInt());
}
```

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
- Automaattisen hyväksymistestauksen on olemassa monia työkaluja
  - eräs suosituimmista on suomalainen python-pohjainen *Robot framework*
  - käytämme useita eri kieliä tukevaa *Cucumberia*

Järjestelmätestauksen automatisointi, ATDD ja BDD

- Ideaalitilanteessa storyjen hyväksymiskriteereistä tehdään automaattisesti suoritettavia
- Automaattisen hyväksymistestauksen on olemassa monia työkaluja
  - eräs suosituimmista on suomalainen python-pohjainen *Robot framework*
  - käytämme useita eri kieliä tukevaa *Cucumberia*
- Käytetään nimitystä *Acceptance test driven development* (ATDD) tai *Behavior driven development* (BDD)
  - erityisesti jos testit toteutetaan jo iteraation alkupuolella, ennen kun story koodattu
- Cucumber on BDD-leirin työkalu

Käyttäjähallinnan tarjoama palvelu

- Palvelun vaatimuksen 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 vaatimuksen 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
- Cucumberissa jokaisesta storystä kirjoitetaan *.feature*-päätteinen tiedosto
  - sisältää joukon storyyn liittyvä hyväksymistestejä joita Cucumber kutsuu *skenaarioiksi*
- Storyn hyväksymätestit eli skenaariot kirjoitetaan Gherkin-kielellä, muodossa

```
Given [initial context]
When [event occurs]
Then [ensure some outcomes]
```

Testit asiakkan kielellä

Feature: User can log in with valid username/password-combination

Scenario: user can login with correct password

Given command login is selected

When username "pekka" and password "akkep" are entered

Then system will respond with "logged in"

Scenario: user can not login with incorrect password

Given command login is selected

When username "pekka" and password "wrong" are entered

Then system will respond with "incorrect username or password"

Mäppäys testeistä suoritettavaan koodiin

```
public class Stepdefs {
    App app;
    StubIO io;
    UserDao userDao;
    AuthenticationService auth;
    List<String> inputLines;

    @Given("^command login is selected$")
    public void commandLoginSelected() throws Throwable {
        inputLines.add("login");
    }

    @When("username {string} and password {string} are entered")
    public void usernameAndPasswordAreEntered(String username, String password) {
       inputLines.add(username);
       inputLines.add(password);

       io = new StubIO(inputLines);
       app = new App(io, auth);
       app.run();
    }

    @Then("system will respond with {string}")
    public void systemWillRespondWith(String expectedOutput) {
        assertTrue(io.getPrints().contains(expectedOutput));
    }
```

Web-sovellusten testaaminen onnistuu Selenium:illa

```
public class Stepdefs {
    WebDriver driver = new ChromeDriver();
    String baseUrl = "http://localhost:4567";

    @Given("login is selected")
    public void loginIsSelected() {
        driver.get(baseUrl);
        WebElement element = driver.findElement(By.linkText("login"));
        element.click();
    }

    @When("correct username {string} and password {string} are given")
    public void correctUsernameAndPasswordAreGiven(String username, String password) {
        assertTrue(driver.getPageSource().contains("Give your credentials to login"));
        WebElement element = driver.findElement(By.name("username"));
        element.sendKeys(username);
        element = driver.findElement(By.name("password"));
        element.sendKeys(password);
        element = driver.findElement(By.name("login"));
        element.submit();
    }

    @Then("user is logged in")
    public void userIsLoggedIn() {
        assertTrue(driver.getPageSource().contains("Ohtu Application main page"));

    }
```

- Ideana on että asiakas tai product owner kirjoittaa tiimissä olevien testaajien tai tiimiläisten kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat toivon mukaan ymmärtävät mistä on kyse

- Ideana on että asiakas tai product owner kirjoittaa tiimissä olevien testaajien tai tiimiläisten kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat toivon mukaan ymmärtävät mistä on kyse
- Koodaajat/testaajat toteuttavat mäppäyksen, joka automatisoi testien suorituksen

- Ideana on että asiakas tai product owner kirjoittaa tiimissä olevien testaajien tai tiimiläisten kanssa yhteistyössä storyyn liittyvät testit
  - Samalla storyn haluttu toiminnallisuus tulee dokumentoitua sillä tarkkuudella, että ohjelmoijat toivon mukaan ymmärtävät mistä on kyse
- Koodaajat/testaajat toteuttavat mäppäyksen, joka automatisoi testien suorituksen
- Ei toistaiseksi vielä kovin yleinen tyyli, useimmiten hyväksymätestit kirjoitettu suoraan “normalilla” testikirjastolla kuten jUnit, Mocha, Jest, ...

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integroini
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - Suoritetaan integraatiotestaus, joka varmistaa yhteistoiminnallisuuden

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integroini
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - Suoritetaan integraatiotestaus, joka varmistaa yhteistoiminnallisuuden
- Perinteisesti juuri integrointivaihe on tuonut esiin suuren joukon ongelmia
- Tarkasta suunnittelusta huolimatta erillisten tiimien toteuttamat komponentit epäyhteensopivia

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integroini
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - Suoritetaan integraatiotestaus, joka varmistaa yhteistoiminnallisuuden
- Perinteisesti juuri 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 integroini
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - Suoritetaan integraatiotestaus, joka varmistaa yhteistoiminnallisuuden
- Perinteisesti juuri 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
- Integraatio on ollut perinteisesti niin hankala vaihe, että sitä kuvaamaan on lanseerattu termi *integratiohelvetti*

Pois integraatiohelvetistä

- 90-luvulla huomattiin, että riskien minimoimiseksi integraatio kannattaa tehdä useammin kuin vain projektin lopussa
- Paras käytänne *daily/nightly 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/nightly 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

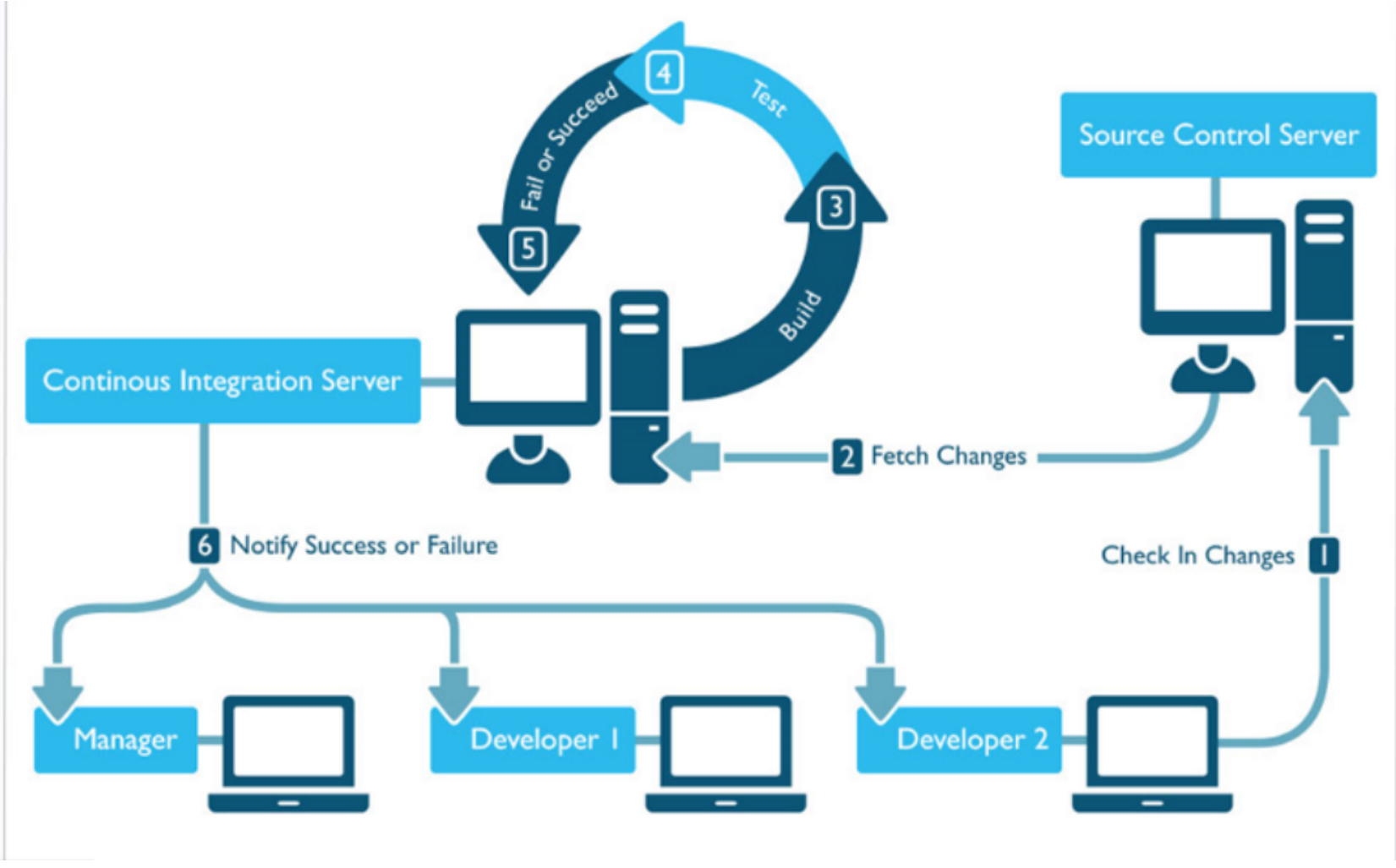
- Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - eräs XP:n käytenäteistä
- Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositoriossa
- Yksittäinen palvelin, jonka konfiguraatio vastaa mahdollisimman läheisesti tuotantopalvelimen konfiguraatiota, toimii CI-palvelimena

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
- Yksittäinen palvelin, jonka konfiguraatio vastaa mahdollisimman läheisesti tuotantopalvelimen konfiguraatiota, toimii CI-palvelimena
- 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
- Yksittäinen palvelin, jonka konfiguraatio vastaa mahdollisimman läheisesti tuotantopalvelimen konfiguraatiota, toimii CI-palvelimena
- 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 tarkoitus tehdä 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 esim. konfiguraatioerojen takia ainoastaan kehittäjän paikallisella työasemalla

- 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 esim. konfiguraatioerojen takia ainoastaan kehittäjän paikallisella työasemalla

- Tarkoituksena on, että jokainen kehittäjä integroi tekemänsä työn muuhun koodiin mahdollisimman usein, vähintään kerran päivässä, mielellään useammin

- 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 esim. konfiguraatioerojen takia ainoastaan kehittäjän paikallisella työasemalla

- Tarkoituksena on, että jokainen kehittäjä integroi tekemänsä työn muuhun koodiin mahdollisimman usein, vähintään kerran päivässä, mielellään useammin

- CI rohkaisee jakamaan työn pieniin osiin, sellaisiin jotka saadaan testeineen valmiiksi yhden työpäivän aikana
- CI-työprosessin noudattaminen vaatii kurinalaisuutta

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

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

- Jotta CI-prosessi toimisi joustavasti, tulee testien ajamisen tapahtua nopeasti, yli 10 min alkaa olla liikaa
- Jos osa testeistä on hitaita, voidaan testit konfiguroida ajettavaksi kahdessa (tai tarvittaessa useammassakin) vaiheessa
  - *commit build*:in läpimeno antaa kehittäjälle oikeuden pushata koodi repositorioon
  - CI-palvelimella suoritetaan myös hitaammat testit sisältävä *secondary build*
- Laskareissa käytetty *CircleCI* on yksi monista SaaS-palveluna toimivista CI-ratkaisuista, monien mielestä tämän hetken paras
  - Toinen suosittu on Travis
- Näitä paljon vanhempi Jenkins lienee edelleen maailmalla eniten käytetty CI-palvelinohjelmisto
  - Jenkinsin käyttö edellyttää sen asentamista omalle palvelimelle
- GitHub *actions* 15.11.2020

Deployment stagingiin

- Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- Integraatioprosessiin lisätään automaattinen *deployaus* staging-palvelimelle
  - Staging-palvelin on ympäristö, joka on konfiguraatioidensa ja myös sovelluksen käsittelemän datan osalta mahdollisimman lähellä varsinaista tuotantoympäristöä

Deployment stagingiin

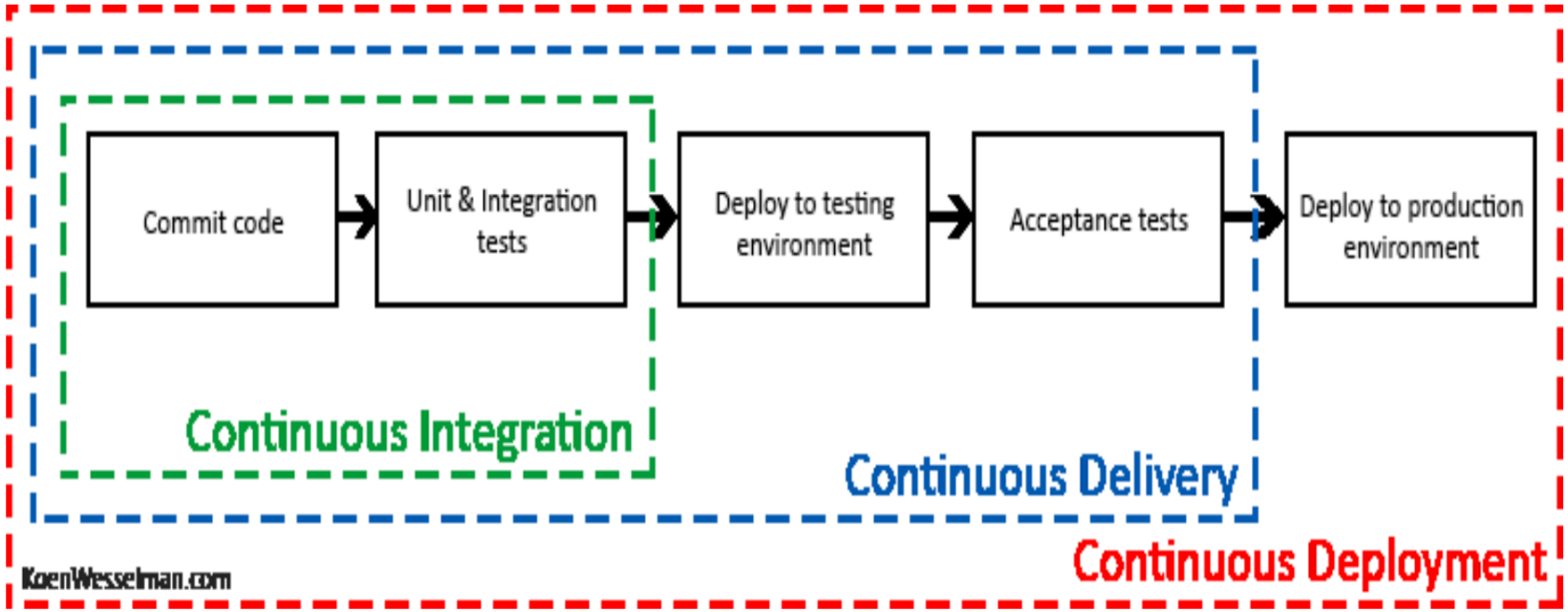
- Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- Integraatioprosessiin lisätään automaattinen *deployaus* staging-palvelimelle
  - Staging-palvelin on ympäristö, joka on konfiguraatioidensa ja myös sovelluksen käsittelemän datan osalta mahdollisimman lähellä varsinaista tuotantoympäristöä
- Kun uusi versio deployattu staging-palvelimelle, suoritetaan 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
  - Staging-palvelin on ympäristö, joka on konfiguraatioidensa ja myös sovelluksen käsittelemän datan osalta mahdollisimman lähellä varsinaista tuotantoympäristöä
- Kun uusi versio deployattu staging-palvelimelle, suoritetaan hyväksymistestaus
- jonka jälkeen uusi versio voidaan siirtää *tuotantopalvelimelle*
- Parhaassa tapauksessa myös staging-ympäristössä tehtävien hyväksymätestien suoritus on automatisoitu
  - Ohjelmisto kulkee koko *deployment pipelinen läpi* automaattisesti

Deployment pipeline

- Deployment pipeline tarkoittaa niitä käännöksen ja testauksen vaiheita, joiden suorittamista edellytetään, että commitattu koodi saadaan siirrettyä tuotantoympäristöön



Jatkuva toimitusvalmius ja käyttöönotto

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

Jatkuva toimitusvalmius ja käyttöönotto

- Jos jokainen commit joka läpäisee testit päätyy automaattisesti tuotantoon, puhutaan *jatkuvasta käyttöönotosta* engl. *continuous deployment*
- Jos taas tuotantoonvientipäätös tehdään ihmisen toimesta, sovelletaan *jatkuvaa toimitusvalmiuutta* engl. *continuous delivery*
  - on olemassa paljon hyviä syitä miksi ihan kaikkea ei haluta julkaista loppukäyttäjille asti
- Viime aikoina on ruvettu suosimaan tyyliä, jossa web-palvelusta julkaistaan tuotantoon uusia jopa kymmeniä kertoja päivästä

Tutkiva testaaminen

- Jotta järjestelmä saadaan niin virheettömäksi, että se voidaan laittaa tuotantoon, on testauksen oltava erittäin perusteellinen

Tutkiva testaaminen

- Jotta järjestelmä saadaan niin virheettömäksi, että se voidaan laittaa tuotantoon, on testauksen oltava erittäin perusteellinen
- 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

Tutkiva testaaminen

- Jotta järjestelmä saadaan niin virheettömäksi, että se voidaan laittaa tuotantoon, on testauksen oltava erittäin perusteellinen
- 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
- Automatisoitujen hyväksymistestien luonne sama, syöte on tarkkaan kiinnitetty samoin kuin odotettu tuloskin

Tutkiva testaaminen

- Jotta järjestelmä saadaan niin virheettömäksi, että se voidaan laittaa tuotantoon, on testauksen oltava erittäin perusteellinen
- 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
- 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

Tutkiva testaaminen

- Jotta järjestelmä saadaan niin virheettömäksi, että se voidaan laittaa tuotantoon, on testauksen oltava erittäin perusteellinen
- 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
- 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 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 yritetään muuttaa käsin id:tä

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 yritetään muuttaa käsin id:tä
- 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