Ohjelmistotuotanto

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

syksy 2019

Luento 6

12.11.2019

Miniprojektien aloitustilaisuudet

- Aloitustilaisuudet (jokainen osallistuu yhteen tilaisuuteen)
  - maanantai 18.11. klo 14-16
  - tiistai 19.11. klo 14-16
  - keskiviikko 20.11. klo 12-14
  - torstai 21.11. klo 14-16
- Loppudemot (jokainen ryhmä osallistuu toiseen demoista)
  - maanantai 9.12. klo 14-17
  - tiistai 10.12. klo 14-17

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

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*
- *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

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

Ketterien menetelmien testauskäytänteet

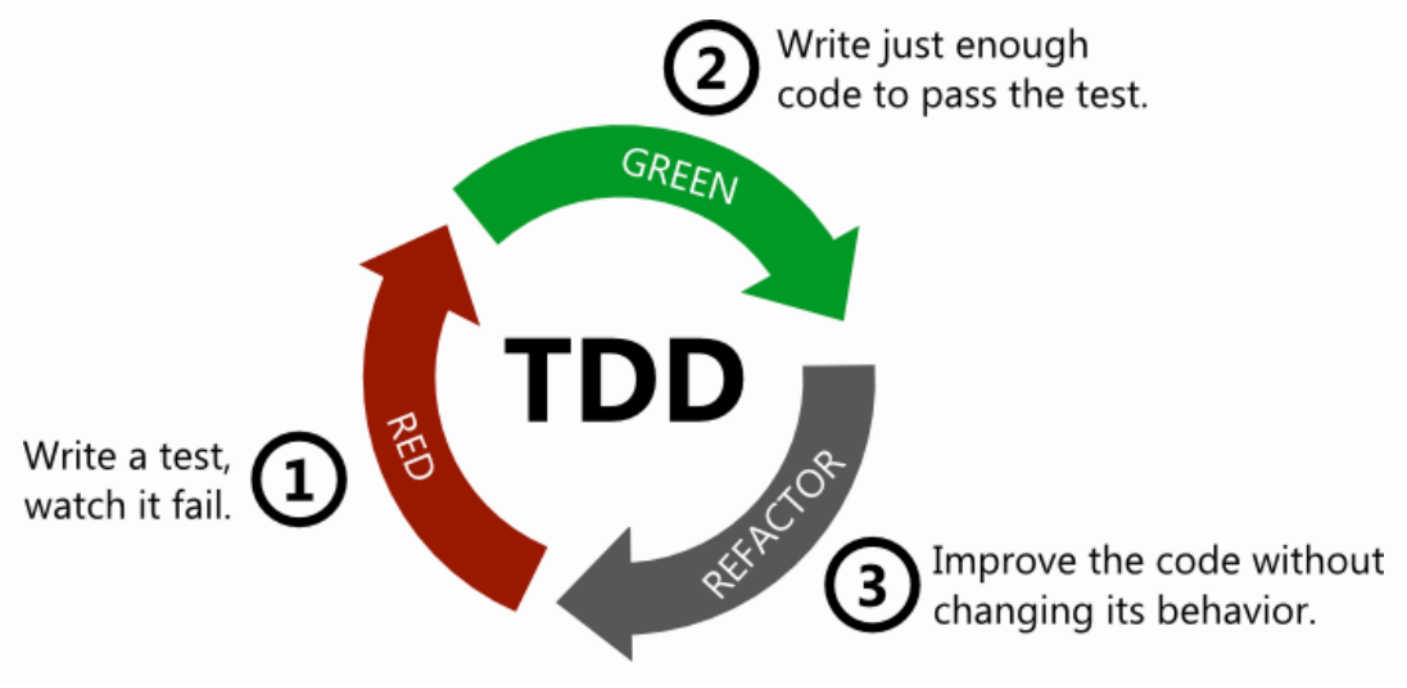
- Test driven development (TDD)
  - Nimestään huolimatta kyseessä suunnittelu- ja toteutustason tekniikka
  - “sivutuotteena” syntyy kattava joukko automaattisesti suoritettavia testejä
- Acceptance Test Driven Development / Behavior Driven Development
  - User storyjen tasolla tapahtuva automatisoitu 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

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 käyttöä
  - 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 käyttöä
  - huomio on komponentin rajapinnassa ja rajapinnan 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 on komponentin rajapinnassa ja rajapinnan helppokäyttöisyydessä
  - ei niinkään komponentin sisäisessä toteutuksessa
- Komponentin sisäinen rakenne muotoutuu refaktorointien kautta
- *"Ensin testataan, sitten koodataan, suunnitellaan vasta lopussa"*

- 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 tehty koodi on yleensä 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 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 mm. käyttöliittymä-, tietokanta- ja verkkoyhteyksistä huolehtivan koodin yhteydessä
- Legacy-koodin laajentaminen TDD:llä voi olla haastavaa

Riippuvuudet testeissä

Riippuvuudet testeissä

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

Riippuvuudet testeissä

- 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

- 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

```
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*

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
- 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
- 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 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
- Cucumberissa jokaisesta storystä kirjoitetaan *.feature*-päätteinen tiedosto
  - sisältää joukon storyyn liittyvä hyväksymistestejä joita Cucumber kutsuu *skenaarioiksi*

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
- 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));
    }
```

Käyttöliittymän läpi tapahtuvan testauksen automatisointi

- Komentoriviä käyttäjien sovellusten testaaminen onnistuu helpohkosti, mockaamalla syöte- ja tulostusvirrat

Käyttöliittymän läpi tapahtuvan testauksen automatisointi

- Komentoriviä käyttäjien 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

```
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 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 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 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 integrointi
  - Yksittäin testatut komponentit integroidaan yhdessä toimivaksi kokonaisuudeksi
  - Suoritetaan integraatiotestaus, joka varmistaa yhteistoiminnallisuuden

Ohjelmiston integraatio

- Vesiputousmallissa toteutusvaiheen päättää integrointi
  - 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 integrointi
  - 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 integrointi
  - 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 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
- 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

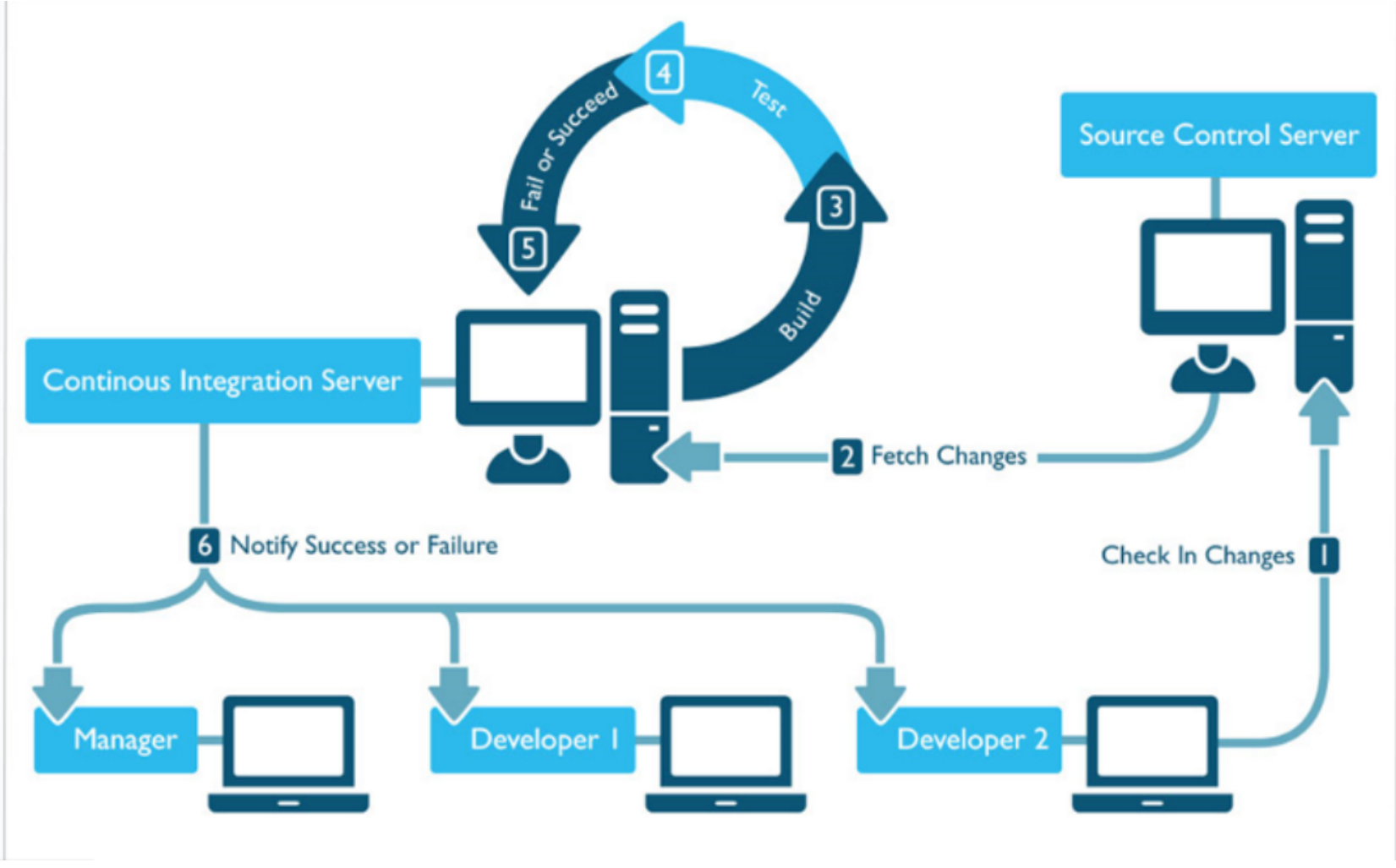
- Syntyi idea toistaa integraatiota vielä päivittäistä sykliäkin useammin: *jatkuva integraatio eli continuous integration*
  - eräs XP:n käytenäteistä
- Koodi, automatisoidut testi, konfiguraatiot ja build-skriptit pidetään keskitetyssä repositoriossa
- Yksittäinen palvelin, jonka konfiguraatio vastaa 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 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
- Tarkoituksena on, että jokainen kehittäjä integroi tekemänsä työn 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

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

- Tarkoituksena on, että jokainen kehittäjä integroi tekemänsä työn 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

- 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 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 joustavasti, 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 *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 useammassa) 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

- 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 useammassa) 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
  - Ympäristö, joka on konfiguraatioiden ja sovelluksen käsittelemän datan osalta mahdollisimman lähellä tuotantoympäristöä

Deployment stagingiin

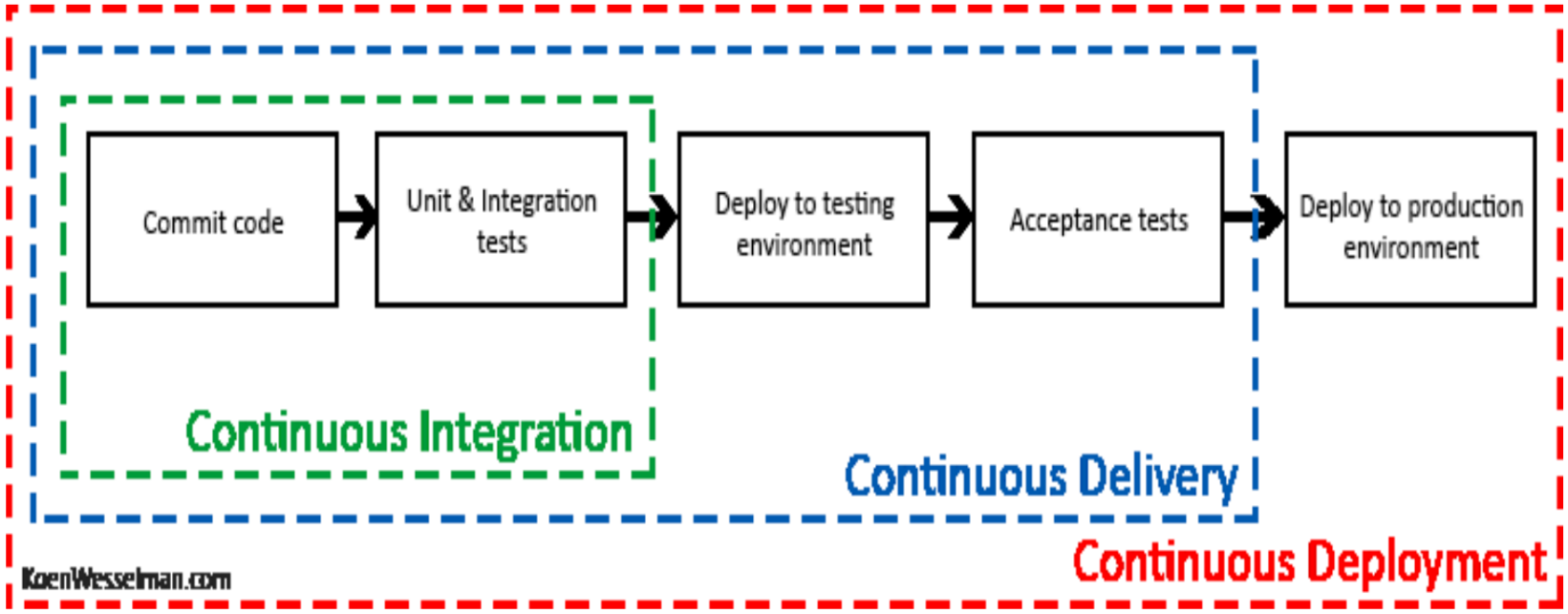
- Viimeaikaisen trendin mukaan CI:tä viedään vielä askel pidemmälle
- Integraatioprosessiin lisätään automaattinen *deployaus* staging-palvelimelle
  - Ympäristö, joka on konfiguraatioiden ja sovelluksen käsittelemän datan osalta mahdollisimman lähellä tuotantoympäristöä
- Kun uusi versio deployattu 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 on konfiguraatioiden ja sovelluksen käsittelemän datan osalta 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

- Niitä käännöksen ja testauksen 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 aikoina on ruvettu suosimaan tyyliä, jossa web-palvelusta julkaistaan uusia versioita jopa kymmeniä kertoja päivästä

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
  - 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

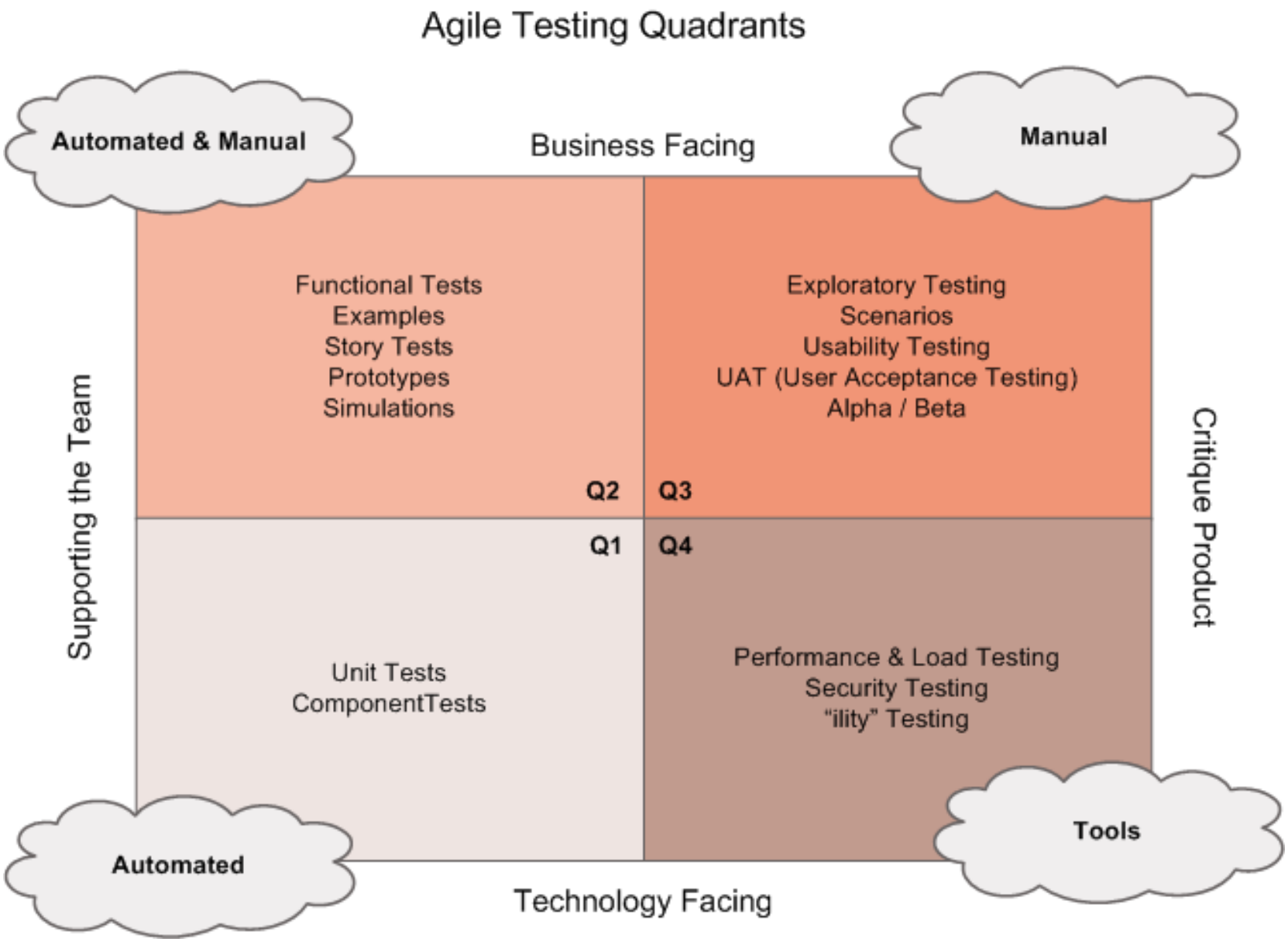
Tutkiva testaaminen

- 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

Yhteenveto - ketterän testauksen nelikettä



Yhteenveto - ketterän testauksen nelikettä

- business facing vs. technology facing
  - testataanko käyttäjän kokemaa toiminnallisuutta vai ohjelmiston teknisiä ominaisuuksia
- supporting team vs. critique to the product
  - onko testaus sovelluskehittäjien tukena vai ulkoista laatua varmistamassa

Yhteenveto - ketterän testauksen nelikettä

- business facing vs. technology facing
  - testataanko käyttäjän kokemaa toiminnallisuutta vai ohjelmiston teknisiä ominaisuuksia
- supporting team vs. critique to the product
  - onko testaus sovelluskehittäjien tukena vai ulkoista laatua varmistamassa
- Eri tyyppiset testit suurelta osin automatisoitavissa
  - poikkeuksena *tutkiva testaaminen* ja *käyttäjän hyväksymätestaus* edellyttävät maniaalista työtä

Yhteenveto - ketterän testauksen nelikettä

- business facing vs. technology facing
  - testataanko käyttäjän kokemaa toiminnallisuutta vai ohjelmiston teknisiä ominaisuuksia
- supporting team vs. critique to the product
  - onko testaus sovelluskehittäjien tukena vai ulkoista laatua varmistamassa
- Eri tyyppiset testit suurelta osin automatisoitavissa
  - poikkeuksena *tutkiva testaaminen* ja *käyttäjän hyväksymätestaus* edellyttävät maniaalista työtä
- Kaikilla neljänneksillä on oma roolinsa ketterässäkehityksessä
  - Tilanteesta riippuu missä suhteessa laadunhallintaan käytettävissä olevat resurssit kannattaa kuhunkin neljännekseen kohdentaa

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 on komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoimatta
  - esim. jos kyseessä minimal viable product

- Jos ohjelmistossa on komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoimatta
  - esim. jos kyseessä minimal viable product
- Tätä ei tiedetä yleensä ennalta ja väliaikaiseksi tarkoitettu komponentti voi jäädä järjestelmään vuosiksi

- Jos ohjelmistossa on komponentteja, jotka tullaan ehkä poistamaan tai korvaamaan pian, ei niiden testejä kannata automatisoimatta
  - esim. jos kyseessä minimal viable product
- Tätä ei tiedetä yleensä ennalta ja 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

- Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- Mielummin integraatiotason testejä ohjelman isompien komponenttien rajapintoja vasten
  - Pysyvät todennäköisemmin valideina komponenttien sisäisen rakenteen muuttuessa

- Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- Mielummin integraatiotason testejä ohjelman isompien komponenttien rajapintoja vasten
  - Pysyvät todennäköisemmin valideina komponenttien sisäisen rakenteen muuttuessa
- Yksikkötestaus hyödyllisimmillään kompleksia logiikkaa sisältäviä luokkia testattaessa

- Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- Mielummin integraatiotason testejä ohjelman isompien komponenttien rajapintoja vasten
  - Pysyvät todennäköisemmin valideina komponenttien sisäisen rakenteen muuttuessa
- Yksikkötestaus hyödyllisimmillään kompleksia logiikkaa sisältäviä luokkia testattaessa
- Käyttöliittymän läpi suoritettavat, käyttäjän interaktiota simuloivat testit usein hyödyllisimpiä
  - Liian aikaisin tehtynä ne tosin saattavat aiheuttaa kohtuuttoman paljon ylläpitovaivaa

- Kattavien yksikkötestien tekeminen ei yleensä ole mielekästä ohjelman kaikille luokille
- Mielummin integraatiotason testejä ohjelman isompien komponenttien rajapintoja vasten
  - Pysyvät todennäköisemmin valideina komponenttien sisäisen rakenteen muuttuessa
- Yksikkötestaus hyödyllisimmillään kompleksia logiikkaa sisältäviä luokkia testattaessa
- Käyttöliittymän läpi suoritettavat, käyttäjän interaktiota simuloivat testit usein hyödyllisimpiä
  - Liian aikaisin tehtynä ne tosin saattavat aiheuttaa kohtuuttoman paljon ylläpitovaivaa
- Automaattisia testejä kannattaa tehdä etenkin niiden komponenttien rajapintoihin, joita muokataan usein

- 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 tuotantoympäristössä

- 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 tuotantoympäristössä
- Oppikirjamääritelmän mukaista TDD:tä sovelletaan harvoin
  - Välillä TDD on hyödyllinen väline, esim. testattaessa rajapintoja, joita käyttäviä komponentteja ei ole vielä olemassa
  - Testit tekee samalla vaivalla kuin koodia käyttävän "pääohjelman"