

Ohjelmistotuotanto

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

syksy 2019

Luento 3

4.11.2019

Vaatimusmäärittely engl requirements engineering

- Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset (engl. requirements) rakennettavalle ohjelmistolle

Vaatimusmäärittely engl requirements engineering

- ▶ Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset (engl. requirements) rakennettavalle ohjelmistolle
- ▶ Vaatimukset jakaantuvat kahteen luokkaan
 - ▶ Toiminnalliset vaatimukset (functional requirements)
 - ▶ ohjelman toiminnot
 - ▶ Ei-toiminnalliset vaatimukset (nonfunctional requirements)
 - ▶ koko ohjelmistoa koskevat "laatuvaatimukset" ja
 - ▶ toimintaympäristön asettamat rajoitteet

Vaatimusmäärittely engl requirements engineering

- ▶ Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset (engl. requirements) rakennettavalle ohjelmistolle
- ▶ Vaatimukset jakaantuvat kahteen luokkaan
 - ▶ Toiminnalliset vaatimukset (functional requirements)
 - ▶ ohjelman toiminnot
 - ▶ Ei-toiminnalliset vaatimukset (nonfunctional requirements)
 - ▶ koko ohjelmistoa koskevat "laatuvaatimukset" ja
 - ▶ toimintaympäristön asettamat rajoitteet
- ▶ Vaatimusmäärittelyn tulee ainakin alkaa ennen ohjelmiston suunnittelua ja toteuttamista
 - ▶ vesiputouksessa vaatimukset määritellään heti alussa
 - ▶ iteratiivisessa ja ketterässä kehityksestä projektin kuluessa

Vaatimusmäärittelyn vaiheet

- ▶ Vaatimusmäärittelyn luonne vaihtelee paljon riippuen kehitettävästä ohjelmistosta, kehittäjäorganisaatiosta ja ohjelmistokehitykseen käytettävästä prosessimallista
- ▶ Asiakkaan tai asiakkaan edustajan on oltava prosessissa aktiivisesti mukana

Vaatimusmäärittelyn vaiheet

- ▶ Vaatimusmäärittelyn luonne vaihtelee paljon riippuen kehitettävästä ohjelmistosta, kehittäjäorganisaatiosta ja ohjelmistokehitykseen käytettävästä prosessimallista
- ▶ Asiakkaan tai asiakkaan edustajan on oltava prosessissa aktiivisesti mukana
- ▶ Jaotellaan yleensä muutamaan työvaiheeseen
 - ▶ kartoitus (engl. elicitation)
 - ▶ analyysi
 - ▶ validointi
 - ▶ dokumentointi
 - ▶ hallinnointi
- ▶ Työvaiheet limittyvät ja vaatimusmäärittely etenee spiraalimaisesti tarkentuen

Vaatimusten kartoituksen menetelmiä

- ▶ Selvitetään järjestelmän sidosryhmät (stakeholders) eli tahot, jotka tekemisissä järjestelmän kanssa
- ▶ Käytetään “kaikki mahdolliset keinot” vaatimusten esiin kaivamiseen, esim.:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken

Vaatimusten kartoituksen menetelmiä

- ▶ Selvitetään järjestelmän sidosryhmät (stakeholders) eli tahot, jotka tekemisissä järjestelmän kanssa
- ▶ Käytetään “kaikki mahdolliset keinot” vaatimusten esiin kaivamiseen, esim.:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken
- ▶ Kehittäjätiimi voi strukturoida vaatimusten kartoitusta
 - ▶ Mietitään järjestelmän kuviteltuja käyttäjiä ja keksitään käyttäjille tyypillisiä käyttöskenaarioita
 - ▶ Tehdään paperiprototyyppejä ja käyttöliittymäluonnoksia
- ▶ Skenaarioita ja prototyyppejä läpikäymällä asiakas näkemys tarkentuu

Vaatimusten kartoituksen menetelmiä

- ▶ Selvitetään järjestelmän sidosryhmät (stakeholders) eli tahot, jotka tekemisissä järjestelmän kanssa
- ▶ Käytetään “kaikki mahdolliset keinot” vaatimusten esiin kaivamiseen, esim.:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken
- ▶ Kehittäjätiimi voi strukturoida vaatimusten kartoitusta
 - ▶ Mietitään järjestelmän kuviteltuja käyttäjiä ja keksitään käyttäjille tyypillisiä käyttöskenaarioita
 - ▶ Tehdään paperiprototyyppejä ja käyttöliittymäluonnoksia
- ▶ Skenaarioita ja prototyyppejä läpikäymällä asiakas näkemys tarkentuu
- ▶ Jos ollaan korvaamassa vanhaa järjestelmää, voidaan vaatimuksia selvittää havainnoimalla loppukäyttäjän työskentelyä (etnografia)

- ▶ Kartoitettuja vaatimuksia täytyy **analysoida**, eli ovatko ne
 - ▶ riittävän kattavat
 - ▶ keskenään ristiriidattomia
 - ▶ testattavissa
 - ▶ toteutuminen on mahdollista ja taloudellisesti järkevää

- ▶ Kartoitettuja vaatimuksia täytyy **analysoida**, eli ovatko ne
 - ▶ riittävän kattavat
 - ▶ keskenään ristiriidattomia
 - ▶ testattavissa
 - ▶ toteutuminen on mahdollista ja taloudellisesti järkevää
- ▶ Vaatimukset on myös pakko **dokumentoida** muodossa tai toisessa
 - ▶ Ohjelmistokehittäjiä varten: mitä tehdään
 - ▶ Testaajia varten: toimiiko järjestelmä kuten vaatimukset määrittelevät
- ▶ Joskus vaatimusdokumentti toimii oleellisena osana asiakkaan ja kehittäjien välisessä sopimuksessa

- ▶ Kartoitettuja vaatimuksia täytyy **analysoida**, eli ovatko ne
 - ▶ riittävän kattavat
 - ▶ keskenään ristiriidattomia
 - ▶ testattavissa
 - ▶ toteutuminen on mahdollista ja taloudellisesti järkevää
- ▶ Vaatimukset on myös pakko **dokumentoida** muodossa tai toisessa
 - ▶ Ohjelmistokehittäjiä varten: mitä tehdään
 - ▶ Testaajia varten: toimiiko järjestelmä kuten vaatimukset määrittelevät
- ▶ Joskus vaatimusdokumentti toimii oleellisena osana asiakkaan ja kehittäjien välisessä sopimuksessa

Ja **validoida**:

- ▶ Onko asiakas sitä mieltä että kirjatut vaatimukset edustavat asiakkaan mielipidettä, eli kuvaavat sellaisen järjestelmät mitä asiakas tarvitsee

Toiminnalliset vaatimukset

- ▶ Vaatimukset jakaantuvat toiminnallisiin ja ei-toiminnallisiin vaatimuksiin
- ▶ Toiminnalliset vaatimukset (functional requirements) kuvaavat mitä toimintoja järjestelmällä on
- ▶ Asiakas voi lisätä tuotteen ostoskoriin
- ▶ Onnistuneen luottokorttimaksun yhteydessä asiakkaalle vahvistetaan ostotapahtuman onnistuminen sähköpostitse

Toiminnalliset vaatimukset

- ▶ Vaatimukset jakaantuvat toiminnallisiin ja ei-toiminnallisiin vaatimuksiin
- ▶ Toiminnalliset vaatimukset (functional requirements) kuvaavat mitä toimintoja järjestelmällä on
- ▶ Asiakas voi lisätä tuotteen ostoskoriin
- ▶ Onnistuneen luottokorttimaksun yhteydessä asiakkaalle vahvistetaan ostotapahtuman onnistuminen sähköpostitse
- ▶ Toiminnallisten vaatimusten dokumentointi voi tapahtua esim.
 - ▶ feature-listoina
 - ▶ UML-käyttötapauksina (joita käsiteltiin aiemmin kurssilla Ohjelmistotekniikka)
 - ▶ Ketterissä menetelmissä yleensä *user storyinä*, joihin tutustumme kohta tarkemmin

Ei-toiminnalliset vaatimukset

- ▶ Ei-toiminnalliset vaatimukset (nonfunctional requirements) jakautuvat kahteen luokkaan: *laatuvaatimuksiin* ja *toimintoympäristön* rajoitteisiin
- ▶ Laatuvaatimukset (quality attributes), ovat koko järjestelmän toiminnallisuutta rajoittavia/ohjaavia tekijöitä, esim.
 - ▶ Käytettävyys
 - ▶ Testattavuus
 - ▶ Laajennettavuus
 - ▶ Suorituskyky
 - ▶ Skaalautuvuus
 - ▶ Tietoturva

Ei-toiminnalliset vaatimukset

- ▶ Ei-toiminnalliset vaatimukset (nonfunctional requirements) jakautuvat kahteen luokkaan: *laatuvaatimuksiin* ja *toimintoympäristön* rajoitteisiin
- ▶ Laatuvaatimukset (quality attributes), ovat koko järjestelmän toiminnallisuutta rajoittavia/ohjaavia tekijöitä, esim.
 - ▶ Käytettävyys
 - ▶ Testattavuus
 - ▶ Laajennettavuus
 - ▶ Suorituskyky
 - ▶ Skaalautuvuus
 - ▶ Tietoturva
- ▶ Toimintaympäristön rajoitteita (constraints) ovat esim:
 - ▶ Toteutusteknologia (esim. tulee toteuttaa NodeJS:llä ja Reactilla)
 - ▶ Integroituminen muihin järjestelmiin (esim. kirjautuminen google-tunnuksilla)
 - ▶ Mukautuminen lakeihin ja standardeihin (esim. ei riko GDPR:ää)

Ei-toiminnalliset vaatimukset

- ▶ Ei-toiminnalliset vaatimukset (nonfunctional requirements) jakautuvat kahteen luokkaan: *laatuvaatimuksiin* ja *toimintoympäristön* rajoitteisiin
- ▶ Laatuvaatimukset (quality attributes), ovat koko järjestelmän toiminnallisuutta rajoittavia/ohjaavia tekijöitä, esim.
 - ▶ Käytettävyys
 - ▶ Testattavuus
 - ▶ Laajennettavuus
 - ▶ Suorituskyky
 - ▶ Skaalautuvuus
 - ▶ Tietoturva
- ▶ Toimintaympäristön rajoitteita (constraints) ovat esim:
 - ▶ Toteutusteknologia (esim. tulee toteuttaa NodeJS:llä ja Reactilla)
 - ▶ Integroituminen muihin järjestelmiin (esim. kirjautuminen google-tunnuksilla)
 - ▶ Mukautuminen lakeihin ja standardeihin (esim. ei riko GDPR:ää)

Vaatimusmäärittely 1900-luvulla

- ▶ Vesiputousmallissa vaatimusmäärittely erillinen ohjelmistoprosessin vaihe, joka on tehdään kokonaan ennen suunnittelun aloittamista

Vaatimusmäärittely 1900-luvulla

- ▶ Vesiputousmallissa vaatimusmäärittely erillinen ohjelmistoprosessin vaihe, joka on tehdään kokonaan ennen suunnittelun aloittamista
- ▶ Tiedetään että jos määrittelyvaiheessa tehdään virhe, joka huomataan vasta testauksessa, on muutoksen tekeminen kallista
- ▶ Tästä loogisena johtopäätöksenä oli tehdä vaatimusmäärittelystä erittäin järeä ja huolella tehty työvaihe

Vaatimusmäärittely 1900-luvulla: ei toimi

- ▶ Ideali jonka mukaan vaatimusmäärittely voidaan irrottaa kokonaan erilliseksi, huolellisesti tehtäväksi vaiheeksi on osoittautunut utopiaksi
- ▶ Vaatimusten muuttumien on väistämätöntä
 - ▶ asiakas ei osaa ilmaista tarpeita, toimintaympäristö muuttuu, vaatimusdokumenttia tulkitaan väärin...

Vaatimusmäärittely 1900-luvulla: ei toimi

- ▶ Ideali jonka mukaan vaatimusmäärittely voidaan irrottaa kokonaan erilliseksi, huolellisesti tehtäväksi vaiheeksi on osoittautunut utopiaksi
- ▶ Vaatimusten muuttumien on väistämätöntä
 - ▶ asiakas ei osaa ilmaista tarpeita, toimintaympäristö muuttuu, vaatimusdokumenttia tulkitaan väärin...
- ▶ Vaatimusmäärittelyä ei ole mahdollista/järkevää irrottaa suunnittelusta ja toteutuksesta
 - ▶ Suunnittelu auttaa ymmärtämään ongelma-aluetta syvällisemmin ja generoi muutoksia vaatimuksiin
 - ▶ Ohjelmia tehdään maksimoiden valmiiden ja muualta, esim. open sourcena saatavien komponenttien käyttö
 - ▶ Jos toteutus otetaan huomioon, on helpompi arvioida vaatimusten toteuttamisen hintaa

Vaatimusmäärittely 2000-luvulla

- ▶ Iteratiivisen ja ketterän ohjelmistotuotannon tapa on integroida kaikki ohjelmistotuotannon vaiheet yhteen
- ▶ Projektin alussa määritellään vaatimuksia tarkemmalla tasolla ainakin yhden iteraation tarpeiden verran

Vaatimusmäärittely 2000-luvulla

- ▶ Iteratiivisen ja ketterän ohjelmistotuotannon tapa on integroida kaikki ohjelmistotuotannon vaiheet yhteen
- ▶ Projektin alussa määritellään vaatimuksia tarkemmalla tasolla ainakin yhden iteraation tarpeiden verran
- ▶ Ohjelmistokehittäjät arvioivat vaatimusten toteuttamisen hintaa
- ▶ Asiakas priorisoi vaatimukset siten, että iteraatioon valitaan toteutettavaksi vaatimukset, jotka tuovat mahdollisimman paljon liiketoiminnallista arvoa

Vaatimusmäärittely 2000-luvulla

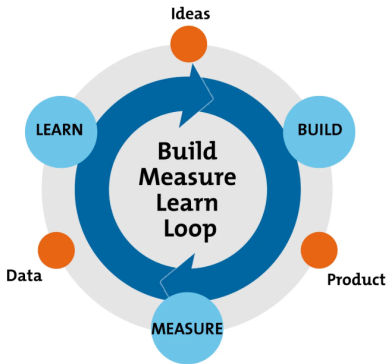
- ▶ Iteratiivisen ja ketterän ohjelmistotuotannon tapa on integroida kaikki ohjelmistotuotannon vaiheet yhteen
- ▶ Projektin alussa määritellään vaatimuksia tarkemmalla tasolla ainakin yhden iteraation tarpeiden verran
- ▶ Ohjelmistokehittäjät arvioivat vaatimusten toteuttamisen hintaa
- ▶ Asiakas priorisoi vaatimukset siten, että iteraatioon valitaan toteutettavaksi vaatimukset, jotka tuovat mahdollisimman paljon liiketoiminnallista arvoa
- ▶ Jokaisen iteraation aikana tehdään määrittelyä, suunnittelua, ohjelmointia ja testausta
- ▶ Jokainen iteraatio tuottaa valmiin osan järjestelmää
- ▶ Edellisen iteraation tuotos toimii syötteenä seuraavan iteraation vaatimusten määrittelyyn

Vaatimusmäärittely 2000-luvulla

- ▶ Iteratiivisen ja ketterän ohjelmistotuotannon tapa on integroida kaikki ohjelmistotuotannon vaiheet yhteen
- ▶ Projektin alussa määritellään vaatimuksia tarkemmalla tasolla ainakin yhden iteraation tarpeiden verran
- ▶ Ohjelmistokehittäjät arvioivat vaatimusten toteuttamisen hintaa
- ▶ Asiakas priorisoi vaatimukset siten, että iteraatioon valitaan toteutettavaksi vaatimukset, jotka tuovat mahdollisimman paljon liiketoiminnallista arvoa
- ▶ Jokaisen iteraation aikana tehdään määrittelyä, suunnittelua, ohjelmointia ja testausta
- ▶ Jokainen iteraatio tuottaa valmiin osan järjestelmää
- ▶ Edellisen iteraation tuotos toimii syötteenä seuraavan iteraation vaatimusten määrittelyyn
- ▶ Ohjelmisto on mahdollista saada tuotantoon jo ennen kaikkien vaatimusten valmistumista

Vaatimusmäärittely 2010-luvulla: Lean startup

- ▶ Eric Riesin 2011 julkaisema kirja Lean startup kuvaa systemaattisen tavan kartoittaa vaatimuksia erityisen epävarmoissa konteksteissa
- ▶ Malli perustuu kolmiosaisen build-measure-learn-syklin toistamiseen



- ▶ Esim. internetpalveluja tai mobiilisovelluksia rakennettaessa käyttäjien tarpeista ei ole minkäänlaista varmuutta, voidaan vain olettaa mitä ihmiset tulisivat käyttämään
- ▶ Alkuvaiheessa ei edes ole vielä käyttäjiä, joilta voitaisiin kysyä

- ▶ Esim. internetpalveluja tai mobiilisovelluksia rakennettaessa käyttäjien tarpeista ei ole minkäänlaista varmuutta, voidaan vain olettaa mitä ihmiset tulisivat käyttämään
- ▶ Alkuvaiheessa ei edes ole vielä käyttäjiä, joilta voitaisiin kysyä
- ▶ Otetaan lähtökohdaksi jokin idea siitä, mitä käyttäjät haluavat
- ▶ Tehdään hypoteesi miten asiakkaat käyttäytyisivät, jos kyseinen järjestelmä/toiminnallisuus/ominaisuusjoukko olisi toteutettu

- ▶ Esim. internetpalveluja tai mobiilisovelluksia rakennettaessa käyttäjien tarpeista ei ole minkäänlaista varmuutta, voidaan vain olettaa mitä ihmiset tulisivat käyttämään
- ▶ Alkuvaiheessa ei edes ole vielä käyttäjiä, joilta voitaisiin kysyä
- ▶ Otetaan lähtökohdaksi jokin idea siitä, mitä käyttäjät haluavat
- ▶ Tehdään hypoteesi miten asiakkaat käyttäytyisivät, jos kyseinen järjestelmä/toiminnallisuus/ominaisuusjoukko olisi toteutettu
- ▶ Rakennetaan nopeasti **minimal viable product (MVP)** joka toteuttaa ominaisuuden
- ▶ MVP laitetaan tuotantoon ja mitataan miten asiakkaat käyttäytyvät uuden ominaisuusjoukon suhteen

- ▶ Jos MVP jonkin ominaisuuden uusi versio, käytetään **A/B-testausta**
 - ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä
- ▶ Järjestelmästä mitattua käyttäytymistä verrataan alussa asetettuun hypoteesiin ja katsotaan olivatko toteutetut vaatimukset käyttäjien mieleen

- ▶ Jos MVP jonkin ominaisuuden uusi versio, käytetään **A/B-testausta**
 - ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä
- ▶ Järjestelmästä mitattua käyttäytymistä verrataan alussa asetettuun hypoteesiin ja katsotaan olivatko toteutetut vaatimukset käyttäjien mieleen
- ▶ Jos toteutettu idea ei osoittautunut hyväksi, voidaan palata järjestelmän edelliseen versioon ja jatkaa build-measure-learn-sykliä
- ▶ Lean startup -menetelmällä on siis tarkoitus oppia systemaattisesti ja mahdollisimman nopeasti mitä asiakkaat haluavat

Vaatimusmäärittely ja projektisuunnittelu ketterässä prosessimallissa

User story

-Ketterän vaatimusmäärittelyn tärkein työväline on user story

- ▶ Alan suurimman auktoriteetin Mike Cohnin mukaan: *A user story describes functionality that will be valuable to either user or purchaser of software.*

User story

-Ketterän vaatimusmäärittelyn tärkein työväline on user story

- ▶ Alan suurimman auktoriteetin Mike Cohnin mukaan: *A user story describes functionality that will be valuable to either user or purchaser of software.*
- ▶ User stories are composed of three aspects:
 1. A written description of the story, used for planning and reminder
 2. Conversations about the story to serve to flesh the details of the story
 3. Tests that convey and document details and that will be used to determine that the story is complete

User story

- ▶ User storyt kuvaavat loppukäyttäjän kannalta arvoa tuottavia toiminnallisuuksia

User story

- ▶ User storyt kuvaavat loppukäyttäjän kannalta arvoa tuottavia toiminnallisuuksia
- ▶ User story on karkean tason tekstuaalinen kuvaus
- ▶ ja lupaus/muistutus siitä, että toiminnallisuuden vaatimukset on selvitettävä asiakkaan kanssa

User story

- ▶ User storyt kuvaavat loppukäyttäjän kannalta arvoa tuottavia toiminnallisuuksia
- ▶ User story on karkean tason tekstuaalinen kuvaus
- ▶ ja lupaus/muistutus siitä, että toiminnallisuuden vaatimukset on selvitettävä asiakkaan kanssa
- ▶ Seuraavat voisivat olla verkkokaupan user storyjen tekstuaalisia kuvauksia:
 - ▶ Asiakas voi lisätä tuotteen ostoskoriin
 - ▶ Asiakas voi poistaa ostoskorissa olevan tuotteen
 - ▶ Asiakas voi maksaa luottokortilla ostoskorissa olevat tuotteet

User story

- ▶ User storyt kuvaavat loppukäyttäjän kannalta arvoa tuottavia toiminnallisuuksia
- ▶ User story on karkean tason tekstuaalinen kuvaus
- ▶ ja lupaus/muistutus siitä, että toiminnallisuuden vaatimukset on selvitettävä asiakkaan kanssa
- ▶ Seuraavat voisivat olla verkkokaupan user storyjen tekstuaalisia kuvauksia:
 - ▶ Asiakas voi lisätä tuotteen ostoskoriin
 - ▶ Asiakas voi poistaa ostoskorissa olevan tuotteen
 - ▶ Asiakas voi maksaa luottokortilla ostoskorissa olevat tuotteet
- ▶ User story ei ole perinteinen vaatimusmääritelmä, joka ilmaisee tyhjentävästi miten joku toiminnallisuus tulee toteuttaa

User story

- ▶ Kun user story päätetään toteuttaa, on sen tarkat vaatimukset pakko selvittää
- ▶ Story on lupaus kommunikoinnista asiakkaan kanssa
conversations about the story to serve to flesh the details of the story

User story

- ▶ Kun user story päätetään toteuttaa, on sen tarkat vaatimukset pakko selvittää
- ▶ Story on lupaus kommunikoinnista asiakkaan kanssa *conversations about the story to serve to flesh the details of the story*
- ▶ Määritelmän kolmas alikohta sanoo että storyyn kuuluu *Tests that convey and document details and that will be used to determine that the story is complete*
- ▶ Storyyn testejä kutsutaan *hyväksymätesteiksi* (acceptance test) tai *hyväksymäkriteereiksi* (acceptance criteria)

User story

- ▶ Kun user story päätetään toteuttaa, on sen tarkat vaatimukset pakko selvittää
- ▶ Story on lupaus kommunikoinnista asiakkaan kanssa *conversations about the story to serve to flesh the details of the story*
- ▶ Määritelmän kolmas alikohta sanoo että storyyn kuuluu *Tests that convey and document details and that will be used to determine that the story is complete*
- ▶ Storyyn testejä kutsutaan *hyväksymätesteiksi* (acceptance test) tai *hyväksymäkriteereiksi* (acceptance criteria)
- ▶ Yleensä joukko konkreettisia testiskenaarioita joiden on toimittava, jotta storyn kuvaaman toiminnallisuuden katsotaan olevan valmis

User story

- ▶ Kun user story päätetään toteuttaa, on sen tarkat vaatimukset pakko selvittää
- ▶ Story on lupaus kommunikoinnista asiakkaan kanssa *conversations about the story to serve to flesh the details of the story*
- ▶ Määritelmän kolmas alikohta sanoo että storyyn kuuluu *Tests that convey and document details and that will be used to determine that the story is complete*
- ▶ Storyyn testejä kutsutaan *hyväksymätesteiksi* (acceptance test) tai *hyväksymäkriteereiksi* (acceptance criteria)
- ▶ Yleensä joukko konkreettisia testiskenaarioita joiden on toimittava, jotta storyn kuvaaman toiminnallisuuden katsotaan olevan valmis
- ▶ Luonne vaihtelee projekteittain
 - ▶ Tekstinä dokumentoituja skenaarioita
 - ▶ parhaassa tapauksessa automaattisesti suoritettavia testejä

Front of Card

173

As a student I want to purchase
a parking pass so that I can
drive to school

Priority: ~~High~~ Should
Estimate: 4

Back of Card

Confirmations:

- ~~The student must pay the correct amount.~~
- One pass for one month is issued at a time.
- The student will not receive a pass if the amount isn't sufficient.
- The person buying the pass must be an enrolled student.
- The student may only buy one pass per month.

Hyvän storyn kriteerit

- ▶ User storyn tulee kuvata sovelluksen käyttäjälle arvoa tuottavia toimintoja
- ▶ Käytettävä asiakkaan kieltä, ei teknistä jargonia

Hyvän storyn kriteerit

- ▶ User storyn tulee kuvata sovelluksen käyttäjälle arvoa tuottavia toimintoja
- ▶ Käytettävä asiakkaan kieltä, ei teknistä jargonia
- ▶ User story tulisi kuvata “end to end”-toiminnallisuutta (käyttöliittymä, bisneslogiikka, tietokanta)
 - ▶ Esimerkki huonosta storystä *lisää jokaisesta asiakkaasta rivi tietokantatauluun customers*

Hyvän storyn kriteerit

- ▶ User storyn tulee kuvata sovelluksen käyttäjälle arvoa tuottavia toimintoja
- ▶ Käytettävä asiakkaan kieltä, ei teknistä jargonia
- ▶ User story tulisi kuvata “end to end”-toiminnallisuutta (käyttöliittymä, bisneslogiikka, tietokanta)
 - ▶ Esimerkki huonosta storystä *lisää jokaisesta asiakkaasta rivi tietokantatauluun customers*
- ▶ Edellinen sivu erään muodin mukaisessa muodossa
 - ▶ *As a type of user, I want functionality so that business value*
 - ▶ *As a student I want to purchase a parking pass so that I can drive to school*

Hyvän storyn kriteerit

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

Hyvän storyn kriteerit

- ▶ Bill Wake *INVEST in good User Stories*, kuusi toivottavaa ominaisuutta
 - ▶ Independent
 - ▶ Negotiable
 - ▶ Valuable to user or customer
 - ▶ Estimable
 - ▶ Small
 - ▶ Testable
- ▶ **Independent:** storyjen pitäisi olla toteutusjärjestykseltään mahdollisimman riippumattomia
 - ▶ antaa asiakkaalle enemmän vapauksia

Hyvän storyn kriteerit

- ▶ Bill Wake *INVEST in good User Stories*, kuusi toivottavaa ominaisuutta
 - ▶ Independent
 - ▶ Negotiable
 - ▶ Valuable to user or customer
 - ▶ Estimable
 - ▶ Small
 - ▶ Testable
- ▶ **Independent:** storyjen pitäisi olla toteutusjärjestykseltään mahdollisimman riippumattomia
 - ▶ antaa asiakkaalle enemmän vapauksia
- ▶ **Negotiable:** storyn luonne “muistilappuna” ja keskusteluna
- ▶ **Valuable**

Hyvän storyn kriteerit

- ▶ **Estimatable:** storyn toteuttamisen vaatima työmäärä pitää olla arvioitavissa kohtuullisella tasolla
- ▶ **Small** storyt on oltava riittävän pieniä, yhden sprintin aikana toteutettavissa olevia

Hyvän storyn kriteerit

- ▶ **Estimatable:** storyn toteuttamisen vaatima työmäärä pitää olla arvioitavissa kohtuullisella tasolla
- ▶ **Small** storyt on oltava riittävän pieniä, yhden sprintin aikana toteutettavissa olevia
- ▶ **Testability:** storyille pitää pystyä laatimaan kriteerit, joiden avulla voi yksikäsitteisesti todeta onko story toteutettu hyväksyttävästi
 - ▶ Ei-toiminnalliset vaatimukset (esim. suorituskyky, käytettävyys) aiheuttavat usein haasteita testattavuudelle
 - ▶ Esim. story *verkkokaupan tulee toimia tarpeeksi nopeasti kovassakin kuormituksessa*
 - ▶ voidaan muotoilla testattavaksi seuraavasti: *käyttäjän vasteaika saa olla korkeintaan 0.5 sekuntia 99% tapauksissa jos yhtäaikaaisia käyttäjiä sivulla on maksimissaan 1000*

Alustava backlog

- ▶ Projektin aluksi kannattaa heti ruveta etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat: haastattelut, brainstormaus...

Alustava backlog

- ▶ Projektin aluksi kannattaa heti ruveta etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat: haastattelut, brainstormaus...
- ▶ Alustavan storyjen keräämisvaiheen ei ole tarkoituksenmukaista kestää kovin kauaa, maksimissaan muutaman päivän
- ▶ User storyjen luonne (muistilappu ja lupaus, että vaatimus tarkennetaan ennen toteutusta) tekee niistä hyvän työkalun projektin aloitukseen
 - ▶ Turhiin detaljeihin ei puututa
 - ▶ Ei edes tavoitella täydellistä ja kattavaa listaa vaatimuksista, asioita tarkennetaan myöhemmin

Alustava backlog

- ▶ Projektin aluksi kannattaa heti ruveta etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat: haastattelut, brainstormaus...
- ▶ Alustavan storyjen keräämisvaiheen ei ole tarkoituksenmukaista kestää kovin kauaa, maksimissaan muutaman päivän
- ▶ User storyjen luonne (muistilappu ja lupaus, että vaatimus tarkennetaan ennen toteutusta) tekee niistä hyvän työkalun projektin aloitukseen
 - ▶ Turhiin detaljeihin ei puututa
 - ▶ Ei edes tavoitella täydellistä ja kattavaa listaa vaatimuksista, asioita tarkennetaan myöhemmin
- ▶ Kun alustava user storyista on kerätty, ne priorisoidaan ja niiden vaatima työ määrä arvioidaan karkealla tasolla

Backlogin priorisointi

- ▶ Kuten muistamme priorisoinnin hoitaa product owner
- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia

Backlogin priorisointi

- ▶ Kuten muistamme priorisoinnin hoitaa product owner
- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia
- ▶ Priorisoinnin motivaationa on pyrkiä maksimoimaan asiakkaan kehitettävästä ohjelmistosta saama hyöty/arvo
- ▶ Tärkeimmät asiat halutaan toteuttaa mahdollisimman nopeasti: saadaan tuotteen alustava versio nopeasti julkaistua

Backlogin priorisointi

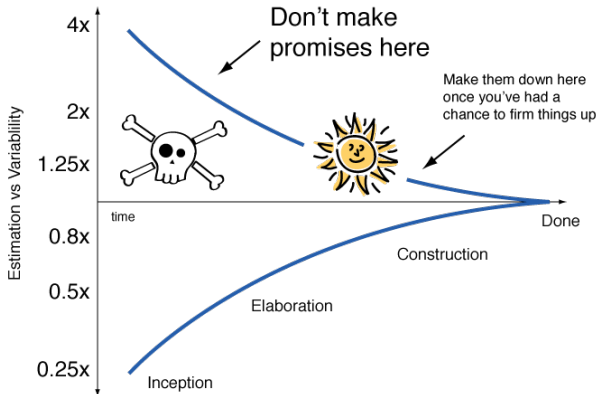
- ▶ Kuten muistamme priorisoinnin hoitaa product owner
- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia
- ▶ Priorisoinnin motivaationa on pyrkiä maksimoimaan asiakkaan kehitettävästä ohjelmistosta saama hyöty/arvo
- ▶ Tärkeimmät asiat halutaan toteuttaa mahdollisimman nopeasti: saadaan tuotteen alustava versio nopeasti julkaistua
- ▶ Storyn tuoman arvon lisäksi priorisoinnissa kannattaa huomioida
 - ▶ Storyn toteuttamiseen kuluva työmäärä
 - ▶ Storyn kuvaamaan ominaisuuteen sisältyvä tekninen riski

- ▶ User storyjen viemän työmäärän arvioimiseen on kaksi motivaatiota
 - ▶ Auttaa asiakasta priorisoinnissa
 - ▶ Mahdollistaa koko projektin viemän ajan arvioinnin

- ▶ User storyjen viemän työmäärän arvioimiseen on kaksi motivaatiota
 - ▶ Auttaa asiakasta priorisoinnissa
 - ▶ Mahdollistaa koko projektin viemän ajan arvioinnin
- ▶ Työmäärän arvioimiseen on kehitetty vuosien varrella useita erilaisia menetelmiä
- ▶ Kaikille yhteistä on se, että ne eivät toimi kunnolla, **tarkkoja työmääräarvioita on mahdoton antaa**

- ▶ User storyjen viemän työmäärän arvioimiseen on kaksi motivaatiota
 - ▶ Auttaa asiakasta priorisoinnissa
 - ▶ Mahdollistaa koko projektin viemän ajan arvioinnin
- ▶ Työmäärän arvioimiseen on kehitetty vuosien varrella useita erilaisia menetelmiä
- ▶ Kaikille yhteistä on se, että ne eivät toimi kunnolla, **tarkkoja työmääräarvioita on mahdoton antaa**
- ▶ Mitä kauempana tuotteen/ominaisuuden valmistuminen on, sitä epätarkempia työmääräarviot ovat

cone of uncertainty



- Ketterän kehityksen *lähtökohta* on että estimointi on epävarmaa ja tarkentuu vasta projektin kuluessa
- ei tehdä sitovia estimointiin perustuvia lupauksia

Suhteelliseen kokoon perustuva estimointi

- ▶ Ominaisuuksien toteuttamiseen menevän tarkan ajan arvioiminen on vaikeaa
- ▶ Ohjelmistokehittäjät pystyvät jossain määrin arvioida eri tehtävien vaatimaa työmäärää suhteessa toisiinsa

Suhteelliseen kokoon perustuva estimointi

- ▶ Ominaisuuksien toteuttamiseen menevän tarkan ajan arvioiminen on vaikeaa
- ▶ Ohjelmistokehittäjät pystyvät jossain määrin arvioida eri tehtävien vaatimaa työmäärää suhteessa toisiinsa
- ▶ *Tuotteen lisääminen ostoskoriin toteuttaminen* vie yhtä kauan kuin *User storyn Tuotteen poistaminen ostoskorista*
- ▶ *Ostoskorissa olevien tuotteiden maksaminen luottokortilla* taas vie noin kolme kertaa kauemmin kun edelliset

Suhteelliseen kokoon perustuva estimointi

- ▶ Ominaisuuksien toteuttamiseen menevän tarkan ajan arvioiminen on vaikeaa
- ▶ Ohjelmistokehittäjät pystyvät jossain määrin arvioida eri tehtävien vaatimaa työmäärää suhteessa toisiinsa
- ▶ *Tuotteen lisääminen ostoskoriin toteuttaminen* vie yhtä kauan kuin *User storyn Tuotteen poistaminen ostoskorista*
- ▶ *Ostoskorissa olevien tuotteiden maksaminen luottokortilla* taas vie noin kolme kertaa kauemmin kun edelliset
- ▶ Ketterissä menetelmissä käytetäänkin yleisesti *suhteelliseen kokoon perustuvaa estimointia*
 - ▶ yksikkönä arvioinnissa on yleensä **story point**
 - ▶ Ei yleensä vastaa mitään todellista tuntimäärää

Kehittäjätiimi estimoi

- ▶ Estimointi tapahtuu **aina** ohjelmistokehitystiimin toimesta
- ▶ Product owner tarkentaa estimoitaviin storyihin liittyviä vaatimuksia

Kehittäjätiimi estimoit

- ▶ Estimointi tapahtuu **aina** ohjelmistokehitystiimin toimesta
- ▶ Product owner tarkentaa estimoitaviin storyihin liittyviä vaatimuksia
- ▶ Estimointia auttaa User storyn pilkkominen teknisiin työvaiheisiin
- ▶ *Tuotteen lisääminen ostoskoriin*, voisi sisältää toteutuksen kannalta seuraavat tekniset tehtävät:
 - ▶ tarvitaan sessio, joka muistaa asiakkaan domain-olio ostoskorin ja ostoksen esittämiseen
 - ▶ html-näkymää päivitettävä tarvittavilla painikkeilla
 - ▶ Kontrolleri painikkeiden käsittelyyn
 - ▶ yksikkötestit kontrollerille ja domain-olioille
 - ▶ hyväksymätestien automatisointi
- ▶ Jos kyseessä on samantapainen toiminnallisuus kuin joku aiemmin toteutettu, ei pilkkomista välttämättä tarvita

Estimointi definition of donen tarkkuudella

- ▶ Estimoinnissa tulee arvioida storyn viemä aika *definition of donen* tarkkuudella
- ▶ tämä sisältää yleensä kaiken storyn toteuttamiseen liittyvän
 - ▶ määrittely, suunnittelu, toteutus, automatisoitujen tekstien tekeminen, testaus, integrointi ja dokumentointi

Estimointi definition of donen tarkkuudella

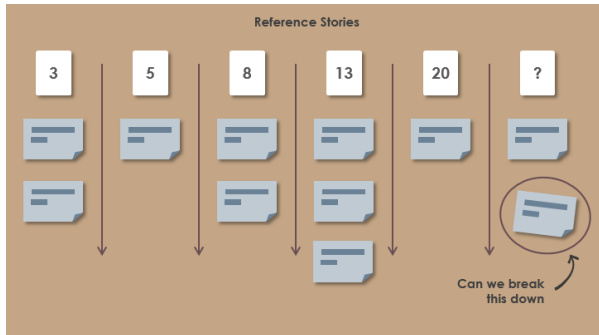
- ▶ Estimoinnissa tulee arvioida storyn viemä aika *definition of donen* tarkkuudella
- ▶ tämä sisältää yleensä kaiken storyn toteuttamiseen liittyvän
 - ▶ määrittely, suunnittelu, toteutus, automatisoitujen tekstien tekeminen, testaus, integrointi ja dokumentointi
- ▶ Estimointi on joka tapauksessa suhteellisen epätarkkaa, joten estimoinnin on tarkoitus tapahtua nopeasti
- ▶ Storyn estimointiin kannattaa käyttää aikaa max 15 minuuttia

Estimointi definition of donen tarkkuudella

- ▶ Estimoinnissa tulee arvioida storyn viemä aika *definition of donen* tarkkuudella
- ▶ tämä sisältää yleensä kaiken storyn toteuttamiseen liittyvän
 - ▶ määrittely, suunnittelu, toteutus, automatisoitujen tekstien tekeminen, testaus, integrointi ja dokumentointi
- ▶ Estimointi on joka tapauksessa suhteellisen epätarkkaa, joten estimoinnin on tarkoitus tapahtua nopeasti
- ▶ Storyn estimointiin kannattaa käyttää aikaa max 15 minuuttia
- ▶ Jos se ei riitä, storya ei tunneta niin hyvin että se kannattaisi estimoida
 - ▶ story kannattaanee pilkkoa

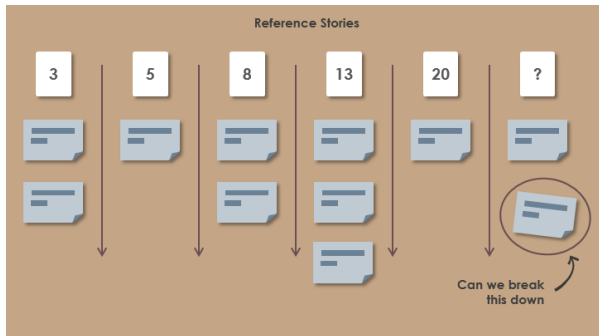
Estimoinnin menetelmiä

- kiinnittää muutama erikokoinen story *referenssiksi* ja verrata sitten muiden storyjen vaativuutta näihin



Estimoinnin menetelmiä

- kiinnittää muutama erikokoinen story *referenssiksi* ja verrata sitten muiden storyjen vaativuutta näihin



- Käytetään yläpäästä harvenevaa skaalaa esim. 1, 2, 3, 5, 10, 20, 40, 100
- Koska isojen storyjen estimointiin liittyy suuri epävarmuus, ei teeskennellä että skaala olisi yläpäästä tarkka

Planning poker: osallistetaan koko tiimi

1. Customer reads story.



2. Team estimates.
This includes testing.



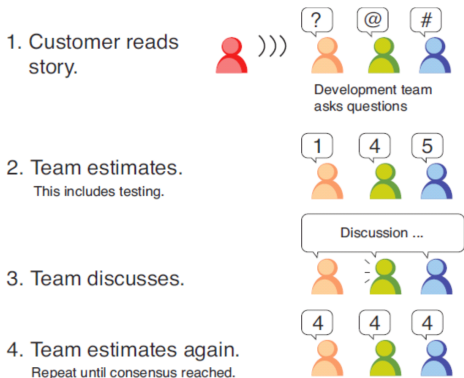
3. Team discusses.



4. Team estimates again.
Repeat until consensus reached.



Planning poker: osallistetaan koko tiimi



- Kaikille yhtenäinen näkemys sisällöstä ja tieto leviämään kaikille (transparency)