

Ohjelmistotuotanto

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

syksy 2023

Luento 3

6.11.2023

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

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

- ▶ Käynnistyvät 13.11 alkavalla viikolla
- ▶ Aloitustilaisuudet
 - ▶ ma 14-16
 - ▶ ti 14-16
 - ▶ to 14-16
 - ▶ to 16-18
 - ▶ pe 12-14
- ▶ Seuraavat viikot: sprinttien katselmus ja suunnittelu samassa aikaikkunassa
- ▶ Loppudemot
 - ▶ ti 12.12 klo 14-16
 - ▶ to 14.12 klo 12-14

- ▶ Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset rakennettavalle ohjelmistolle

Vaatimusmäärittely engl requirements engineering

- ▶ Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset rakennettavalle ohjelmistolle
- ▶ Jakaantuvat kahteen luokkaan
 - ▶ Toiminnalliset vaatimukset
 - ▶ ohjelman toiminnot
 - ▶ Ei-toiminnalliset vaatimukset
 - ▶ koko ohjelmistoa koskevat “laatuvaatimukset” ja
 - ▶ toimintaympäristön asettamat rajoitteet

Vaatimusmäärittely engl requirements engineering

- ▶ Keskeisin ongelma ohjelmistotuotantoprosessissa on määritellä asiakkaan vaatimukset rakennettavalle ohjelmistolle
- ▶ Jakaantuvat kahteen luokkaan
 - ▶ Toiminnalliset vaatimukset
 - ▶ ohjelman toiminnot
 - ▶ Ei-toiminnalliset vaatimukset
 - ▶ 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
 - ▶ ohjelmistokehitykseen käytettävästä prosessimallista

Vaatimusmäärittelyn vaiheet

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

Vaatimusmäärittelyn vaiheet

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

Vaatimusmäärittelyn vaiheet

- ▶ Vaatimusmäärittelyn luonne vaihtelee paljon riippuen
 - ▶ kehitettävästä ohjelmistosta
 - ▶ kehittäjäorganisaatiosta
 - ▶ 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

Vaatimusten kartoituksen menetelmiä

- ▶ Selvitetään järjestelmän sidosryhmät (stakeholders) eli tahot, jotka tekemisissä järjestelmän kanssa
- ▶ Käytetään kaikki mahdolliset keinot:
 - ▶ 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:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken
- ▶ Kehittäjätiimi voi strukturoida vaatimusten kartoitusta
 - ▶ Mietitään *kuviteltuja käyttäjiä* ja keksitään käyttäjille tyypillisiä *käyttöskenaarioita*
 - ▶ Tehdään paperiprototyyppejä ja käyttöliittymäluonnoksia

Vaatimusten kartoituksen menetelmiä

- ▶ Selvitetään järjestelmän sidosryhmät (stakeholders) eli tahot, jotka tekemisissä järjestelmän kanssa
- ▶ Käytetään kaikki mahdolliset keinot:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken
- ▶ Kehittäjätiimi voi strukturoida vaatimusten kartoitusta
 - ▶ Mietitään *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:
 - ▶ Haastatellaan sidosryhmien edustajia
 - ▶ Pidetään brainstormaussessioita asiakkaan ja kehittäjien kesken
- ▶ Kehittäjätiimi voi strukturoida vaatimusten kartoitusta
 - ▶ Mietitään *kuviteltuja käyttäjiä* ja keksitään käyttäjille tyypillisiä *käyttöskenaarioita*
 - ▶ Tehdään paperiprototyyppejä ja käyttöliittymäluonnoksia
- ▶ Skenaarioita ja prototyyppejä läpikäymällä asiakas näkemys tarkentuu
- ▶ Jos ollaan korvaamassa vanhaa järjestelmää, voidaan havainnoida 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

- ▶ 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 vaatimusedokumentti 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 vaatimusedokumentti toimii oleellisena osana asiakkaan ja kehittäjien välisessä sopimuksessa
- ▶ Ja **validoida**:
 - ▶ Onko asiakas sitä mieltä että kirjatut vaatimukset kuvaavat sellaisen järjestelmät mitä asiakas tarvitsee

Toiminnalliset vaatimukset

- ▶ Vaatimukset jakaantuvat toiminnallisiin ja ei-toiminnallisiin vaatimuksiin

Toiminnalliset vaatimukset

- ▶ Vaatimukset jakaantuvat toiminnallisiin ja ei-toiminnallisiin vaatimuksiin
- ▶ *Toiminnalliset vaatimukset* (functional requirements) kuvaavat mitä toimintoja järjestelmällä on
- ▶ Esim:
 - ▶ 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
- ▶ Esim:
 - ▶ 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 kurssilla Ohjelmistotekniikka ennen vuotta 2018)
 - ▶ Ketterissä menetelmissä usein *user storyinä*

Ei-toiminnalliset vaatimukset

- ▶ Ei-toiminnalliset vaatimukset jakautuvat kahteen luokkaan
- ▶ *Laatuvaatimukset* (quality attributes), ovat koko järjestelmän toiminnallisuutta rajoittavia/ohjaavia tekijöitä, esim.
 - ▶ Käytettävyys
 - ▶ Saavutettavuus
 - ▶ Tietoturva
 - ▶ Suorituskyky
 - ▶ Skaalautuvuus
 - ▶ Testattavuus
 - ▶ Laajennettavuus

Ei-toiminnalliset vaatimukset

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

Ei-toiminnalliset vaatimukset

- ▶ Ei-toiminnalliset vaatimukset jakautuvat kahteen luokkaan
- ▶ *Laatuvaatimukset* (quality attributes), ovat koko järjestelmän toiminnallisuutta rajoittavia/ohjaavia tekijöitä, esim.
 - ▶ Käytettävyys
 - ▶ Saavutettavuus
 - ▶ Tietoturva
 - ▶ Suorituskyky
 - ▶ Skaalautuvuus
 - ▶ Testattavuus
 - ▶ Laajennettavuus
- ▶ *Toimintaympäristön rajoitteita* (constraints) ovat esim:
 - ▶ Toteutusteknologia (tulee toteuttaa NodeJS:llä ja Reactilla)
 - ▶ Integroituminen muihin järjestelmiin (kirjautuminen HY-tunnuksilla, data SISU:sta)
 - ▶ Mukautuminen lakeihin ja standardeihin (ei riko GDPR:ää)
- ▶ Ei-toiminnalliset vaatimukset vaikuttavat yleensä ohjelman arkkitehtuurin suunnitteluun

Vaatimusmäärittely 1900-luvulla

Vaatimusmäärittely 1900-luvulla

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

Vaatimusmäärittely 1900-luvulla

- ▶ Vesiputousmallissa vaatimusmäärittely erillinen ohjelmistoprosessin vaihe
 - ▶ tehdään kokonaan ennen suunnittelun aloittamista
- ▶ Jos määrittelyssä tehdään virhe, joka huomataan vasta testauksessa, muutoksen tekeminen kallista

Vaatimusmäärittely 1900-luvulla

- ▶ Vesiputousmallissa vaatimusmäärittely erillinen ohjelmistoprosessin vaihe
 - ▶ tehdään kokonaan ennen suunnittelun aloittamista
- ▶ Jos määrittelyssä tehdään virhe, joka huomataan vasta testauksessa, 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 erilliseksi vaiheeksi on osoittautunut utopiaksi

Vaatimusmäärittely 1900-luvulla: ei toimi

- ▶ Ideali jonka mukaan vaatimusmäärittely voidaan irrottaa erilliseksi 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 erilliseksi 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

Vaatimusmäärittely 1900-luvulla: ei toimi

- ▶ Ideali jonka mukaan vaatimusmäärittely voidaan irrottaa erilliseksi 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

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: iteraatioon valitaan toteutettavaksi ne vaatimukset, jotka maksimoivat liiketoiminnallisen arvon

Vaatimusmäärittely 2000-luvulla

- ▶ Jokaisen iteraation aikana tehdään määrittelyä, suunnittelua, ohjelmointia ja testausta

Vaatimusmäärittely 2000-luvulla

- ▶ Jokaisen iteraation aikana tehdään määrittelyä, suunnittelua, ohjelmointia ja testausta
- ▶ Jokainen iteraatio tuottaa valmiin osan järjestelmää

Vaatimusmäärittely 2000-luvulla

- ▶ 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

- ▶ 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 2000-luvulla

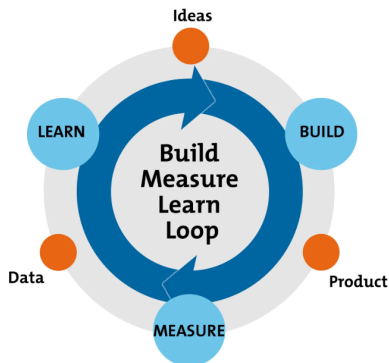
- ▶ Jokaisen iteraation aikana tehdään määrittelyä, suunnittelua, ohjelmointia ja testausta
- ▶ Jokainen iteraatio tuottaa valmiin osan järjestelmää
- ▶ Edellisen iteraation tuotos toimii syötteenä seuraavan iteraation vaatimusten määrittelyyn
- ▶ **Ohjelmisto on mahdollista saada tuotantoon jo ennen kaikkien vaatimusten valmistumista**
- ▶ Kattavana teemana tuottaa asiakkaalle maksimaalisesti arvoa

Vaatimusmäärittely 2010-luvulla: Lean startup

- ▶ Eric Ries (2011): *Lean startup*
 - ▶ kuvaa systemaattisen tavan kartoittaa vaatimuksia erityisen epävarmoissa konteksteissa

Vaatimusmäärittely 2010-luvulla: Lean startup

- ▶ Eric Ries (2011): *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 minkäänlaista varmuutta
 - ▶ Alkuvaiheessa ei edes ole vielä käyttäjiä, joilta voitaisiin kysyä
 - ▶ voidaan vain olettaa mitä ihmiset tulisivat käyttämään

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

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

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

- ▶ Jos MVP jonkin toiminnallisuuden uusi versio, käytetään

A/B-testausta

- ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä

- ▶ Jos MVP jonkin toiminnallisuuden uusi versio, käytetään

A/B-testausta

- ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä
- ▶ Mitattua käyttäytymistä verrataan alussa asetettuun hypoteesiin
 - ▶ olivatko toteutetut toiminnallisuuden käyttäjien mieleen

- ▶ Jos MVP jonkin toiminnallisuuden uusi versio, käytetään **A/B-testausta**
 - ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä
- ▶ Mitattua käyttäytymistä verrataan alussa asetettuun hypoteesiin
 - ▶ olivatko toteutetut toiminnallisuuden käyttäjien mieleen
- ▶ Jos toteutettu idea ei osoittautunut hyväksi, voidaan palata järjestelmän edelliseen versioon
 - ▶ Jos idea on hyvä, toteutetaan sen toiminnallisuus robustilla tavalla

- ▶ Jos MVP jonkin toiminnallisuuden uusi versio, käytetään **A/B-testausta**
 - ▶ uusi ominaisuus julkaistaan osalle käyttäjistä, loput jatkavat vanhan ominaisuuden käyttöä
- ▶ Mitattua käyttäytymistä verrataan alussa asetettuun hypoteesiin
 - ▶ olivatko toteutetut toiminnallisuuden käyttäjien mieleen
- ▶ Jos toteutettu idea ei osoittautunut hyväksi, voidaan palata järjestelmän edelliseen versioon
 - ▶ Jos idea on hyvä, toteutetaan sen toiminnallisuus robustilla tavalla
- ▶ Menetelmällä on siis tarkoitus oppia systemaattisesti ja mahdollisimman nopeasti mitä asiakkaat haluavat

TAUKO 10 minuuttia

Vaatimusmäärittely ja projektisuunnittelu ketterässä prosessimallissa

User story

User story

- ▶ Ketterän vaatimusmäärittelyn tärkein työväline on user story
- ▶ Mike Cohn:
 - ▶ *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
- ▶ Mike Cohn:
 - ▶ *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*

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 toimittava, jotta storyn voidaan todeta 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 toimittava, jotta storyn voidaan todeta 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: ~~Must~~ 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 payment isn't sufficient
- The person buying the pass must be a currently enrolled student.
- The student may only buy one pass per month.

Hyvän storyn kriteerejä

Hyvän storyn kriteerejä

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

Hyvän storyn kriteerejä

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

Hyvän storyn kriteerejä

- ▶ User storyn tulee kuvata sovelluksen käyttäjälle arvoa tuottavia toimintoja
 - ▶ Käytettävä asiakkaan kieltä, ei teknistä jargonia
- ▶ User story tulisi kuvata “end to end”-toiminnallisuutta (kattaen käyttöliittymän, bisneslogiikan, ja tietokannan)
 - ▶ 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

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

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 yhtäaikaisia käyttäjiä sivulla on maksimissaan 1000*

Ketterää vaatimusten hallintaa...

Alustava backlog

- ▶ Projektin alussa etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog

Alustava backlog

- ▶ Projektin alussa etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat:
 - ▶ haastattelut, brainstormaus, paperiprototyypit, käyttöliittymäluonnokset...

Alustava backlog

- ▶ Projektin alussa etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat:
 - ▶ haastattelut, brainstormaus, paperiprototyypit, käyttöliittymäluonnokset...
- ▶ Alustavan storyjen keräämisvaiheen ei ole tarkoituksenmukaista kestää kovin kauaa, maksimissaan muutaman päivän

Alustava backlog

- ▶ Projektin alussa etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat:
 - ▶ haastattelut, brainstormaus, paperiprototyypit, käyttöliittymäluonnokset...
- ▶ Alustavan storyjen keräämisvaiheen ei ole tarkoituksenmukaista kestää kovin kauaa, maksimissaan muutaman päivän
- ▶ User story on muistilappu ja lupaus tarkennuksesta:
 - ▶ Turhiin detaljeihin ei puututa
 - ▶ Ei edes tavoitella täydellistä ja kattavaa listaa vaatimuksista, asioita tarkennetaan myöhemmin

Alustava backlog

- ▶ Projektin alussa etsimään ja määrittelemään user storyja ja muodostaa näistä alustava product backlog
- ▶ Käytettävissä ovat kaikki yleiset vaatimusten kartoitustekniikat:
 - ▶ haastattelut, brainstormaus, paperiprototyypit, käyttöliittymäluonnokset...
- ▶ Alustavan storyjen keräämisvaiheen ei ole tarkoituksenmukaista kestää kovin kauaa, maksimissaan muutaman päivän
- ▶ User story on muistilappu ja lupaus tarkennuksesta:
 - ▶ Turhiin detaljeihin ei puututa
 - ▶ Ei edes tavoitella täydellistä ja kattavaa listaa vaatimuksista, asioita tarkennetaan myöhemmin
- ▶ Kun alustavat storyt identifioitu, ne priorisoidaan ja työmäärä arvioidaan karkealla tasolla

Backlogin priorisointi

Backlogin priorisointi

- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia
- ▶ **Priorisoinnin hoitaa product owner**

Backlogin priorisointi

- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia
- ▶ **Priorisoinnin hoitaa product owner**
- ▶ Motivaationa on pyrkiä maksimoimaan asiakkaan kehitettävästä ohjelmistosta saama hyöty/arvo

Backlogin priorisointi

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

Backlogin priorisointi

- ▶ Prioriteetti määrää järjestyksen, missä ohjelmistokehittäjät toteuttavat ohjelmiston ominaisuuksia
- ▶ **Priorisoinnin hoitaa product owner**
- ▶ Motivaationa on pyrkiä maksimoimaan asiakkaan kehitettävästä ohjelmistosta saama hyöty/arvo
- ▶ Tärkeimmät asiat halutaan toteuttaa nopeasti
 - ▶ saada tuotteen alustava versio nopeasti julkaistua
- ▶ Arvon lisäksi priorisoinnissa kannattaa huomioida
 - ▶ Storyn toteuttamiseen kuluva työmäärä
 - ▶ Storyn kuvaamaan ominaisuuteen sisältyvä tekninen riski

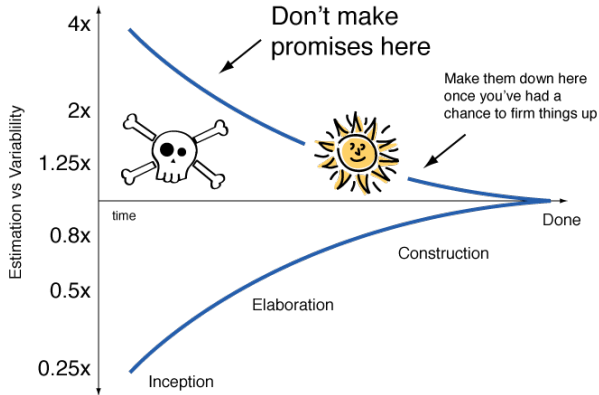
Estimointi

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

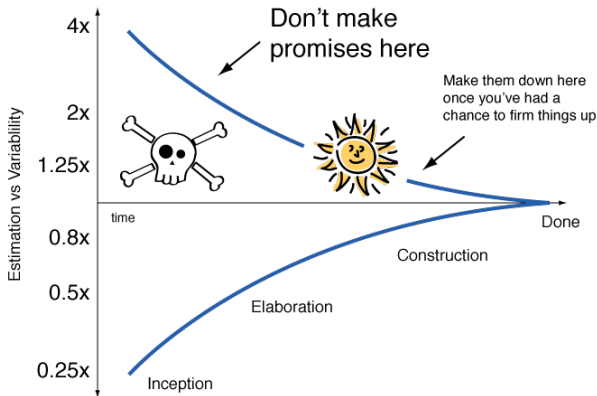
- ▶ User storyjen estimointiin eli niiden viemän työmäärän arvioimiseen on kaksi motivaatiota
 - ▶ Auttaa asiakasta priorisoinnissa
 - ▶ Mahdollistaa koko projektin viemän ajan arvioinnin
- ▶ Työmäärän arvioimiseen on kehitetty vuosien varrella useita erilaisia menetelmiä
 - ▶ Kaikille yhteistä on se,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

- ▶ User storyjen estimointiin eli niiden 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



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

Suhteelliseen kokoon perustuva estimointi

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

Suhteelliseen kokoon perustuva estimointi

- ▶ Ominaisuuksien toteuttamiseen menevän tarkan ajan arvioiminen on vaikeaa
- ▶ Ohjelmistokehittäjät pystyvät jossain määrin arvioida *eri ominaisuuksien vaatimaa työmäärää suhteessa toisiinsa*
- ▶ Esim.
 - ▶ *Tuotteen lisääminen ostoskoriin toteuttaminen vie yhtä kauan kuin 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 ominaisuuksien vaatimaa työmäärää suhteessa toisiinsa*
- ▶ Esim.
 - ▶ *Tuotteen lisääminen ostoskoriin toteuttaminen vie yhtä kauan kuin 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 estimoii

- ▶ Estimointi tapahtuu **aina** ohjelmistokehitystiimin toimesta
 - ▶ Product owner tarkentaa estimoitaviin storyihin liittyviä vaatimuksia
- ▶ Estimointia auttaa user storyn pilkkominen teknisiin työvaiheisiin

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
 - ▶ oliot/tietorakenteet ostoskorin ja ostoksen esittämiseen
 - ▶ html-näkymää päivitettävä tarvittavilla painikkeilla
 - ▶ Kontrolleri painikkeiden käsittelyyn
 - ▶ yksikkötestit kontrollerille ja tietorakenteille
 - ▶ 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

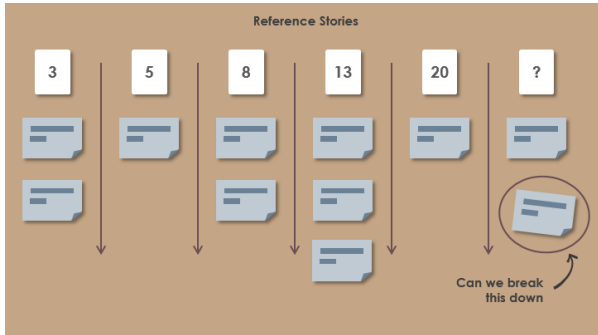
- ▶ Estimoinnissa tulee arvioida storyn viemä aika *definition of donen* tarkkuudella
- ▶ Tämä sisältää yleensä kaiken storyn toteuttamiseen liittyvän
 - ▶ määrittely, suunnittelu, toteutus, automatisoitujen tekstien tekeminen, testaus, integrointi ja dokumentointi
- ▶ Estimointi on joka tapauksessa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

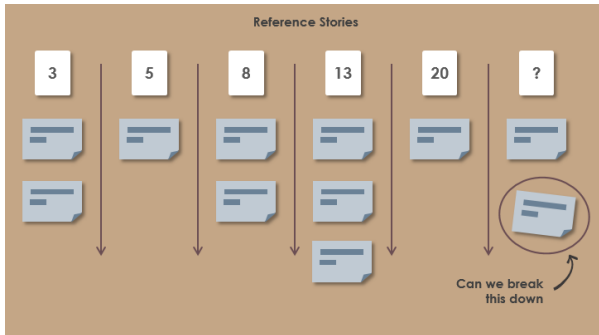
Estimoinnin menetelmiä

- Kiinnitetään muutama erikokoinen story *referenssiksi* ja verrata muiden storyjen työmäärää näihin



Estimoinnin menetelmiä

- Kiinnitetään muutama erikokoinen story *referenssiksi* ja verrata muiden storyjen työmäärää 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.



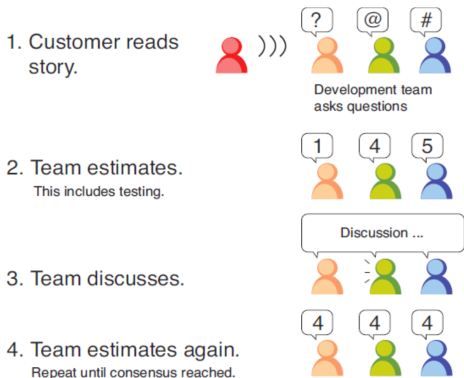
3. Team discusses.



4. Team estimates again.
Repeat until consensus reached.



Planning poker: osallistetaan koko tiimi



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