Ohjelmistoprojekti 2

Tervetuloa Ohjelmistoprojekti 2 -kurssille! 👋

Käytänteet

- Kurssilla toteutetaan tiimissiä ohjelmistoprojekti tiimin muodostamasta aiheesta
- Projekti etenee kolmen viikon iteraatioissa kahden opetusperiodin ajan
- Jokaisen iteraation aluksi tiimi suunnittelee iteraation aikana ohjelmistoon toteutettavat toiminallisuudet
- Iteraation aikana tiimi toteuttaa suunnitteltuja toiminallisuuksia itsenäisesti sekä opettajan ohjauksessa
- Iteraation päätteeksi tiimi esittää opettajalle projektin edistymistä ja saavat palautetta
- Opetusta järjestetään viikottain ja **läsnäolossa noudatetaan Haaga-Helian läsnäolokäytänteitä**. Aktiivinen läsnäolo vaikuttaa arviointiin

Arviointi

- Kurssin arviointi perustuu seuraaviin tekijöihin:
  - Tiimityöskentelyn sujuvuuteen ja prosessin noudattamiseen
  - Tiimin lopullisen tuloksen tarkoituksenmukaiseen toimivuuteen
  - Kurssin lopuksi yksilötyönä tehtävään loppuraporttiin, jossa arvioidaan omaa ja tiimin toimintaa
- Arviointi koostuu tiimin tuotoksesta ja tiimin jäsenen omasta panoksesta projektin eteen. Jokainen tiimissä **ei siis välttämättä saa samaa arvosanaa**
- Arvioinnin tukena käytetään kurssin aikana tehtävää itseis- ja vertaisarviointia
- Tarkat arviointikriteerit löytyvät kurssisivulta

Prosessien merkitys ohjelmistotuotannossa

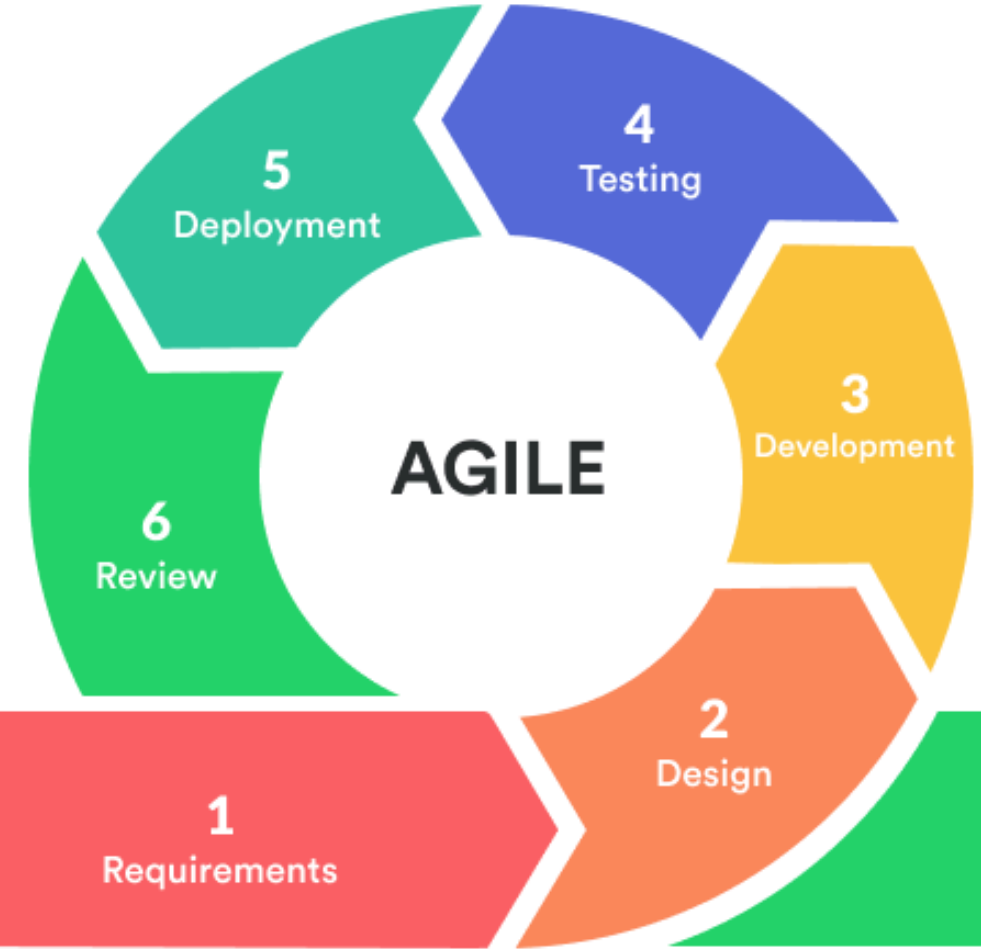
> *"If you go forward 24 months from now, or some amount of time – I can't exactly predict where it is – it's possible that most developers are not coding. It just means that each of us has to get more in tune with what our customers need and what the actual end thing is that we're going to try to go build because that's going to be more and more of what the work is as opposed to sitting down and actually writing code."* —Matt Garman, AWS:n toimitusjohtaja vuonna 2024

- Tulevaisuudessa ohjelmistojen monimutkaisuus kasvaa entisestään, mikä lisää ohjelmistoprojektien vaatimustenhallinnan tarvetta sekä kehitystyön koordinoinnin ja hallinnan merkitystä
- Ohjelmistokehittäjän rooli laajenee jatkuvasti ja teknisen toteutuksen ohella työ painottuu yhä enemmän **asiakkaan tarpeiden ymmärtämiseen**, **teknisten ja toiminnallisten vaatimusten määrittelyyn** sekä ohjelmiston **laadunhallintaan**
- Tiimin yhteinen prosessi mahdollistaa monimutkaisten kokonaisuuksien hallinnan tehokkaasti

Ketterä ohjelmistokehitys

> *"When to use iterative development? You should use iterative development only on projects that you want to succeed."* —Martin Fowler

- Ohjelmistoprojekteissa toteutaan ohjelmistoja, jotka perustuvat johonkin tarpeeseen, jonka määrittelee ohjelmistoprojektin asiakas
- Ohjelmistokehittäjien tehtävä on muodostaa asiakkaan tarpeista toteuttamiskelpoisia teknisiä vaatimuksia ja toteuttaa ne sopivilla teknologioilla
- Tarpeiden selvittäminen vaatii jatkuvaa suoraa viestintää asiakkaan kanssa. Tarpeilla on myös tapana muuttua ja tarkentua ohjelmistoprojektin edetessä
- Ohjelmistotuotannossa sovelletaan nykyisin laajalti nk. **ketteriä menetelmiä**, jotka korostavat mm. suoraa viestintää sidosryhmien kanssa ja nopeaa muutoksiin reagointia
- Ketterät menetelmät ovat **iteratiivinen** vaihtoehto perinteisille vaiheellisille ohjelmistotuotantoprosesesseille, kuten vesiputousmalli



Iteratiivisuus ketterässä ohjelmistokehityksessä

- Jotta muutoksiin voitaisiin vastata nopeasti, ketterille prosesseille on tyypillistä, että ohjelmistoa kehitetään lyhyissä iteraatioissa
- Jokaisen iteraation aikana suoritetaan kaikki ohjelmiston elinkaareen vaiheet vaatimusmäärittelystä tuotantoonvientiin

Scrum

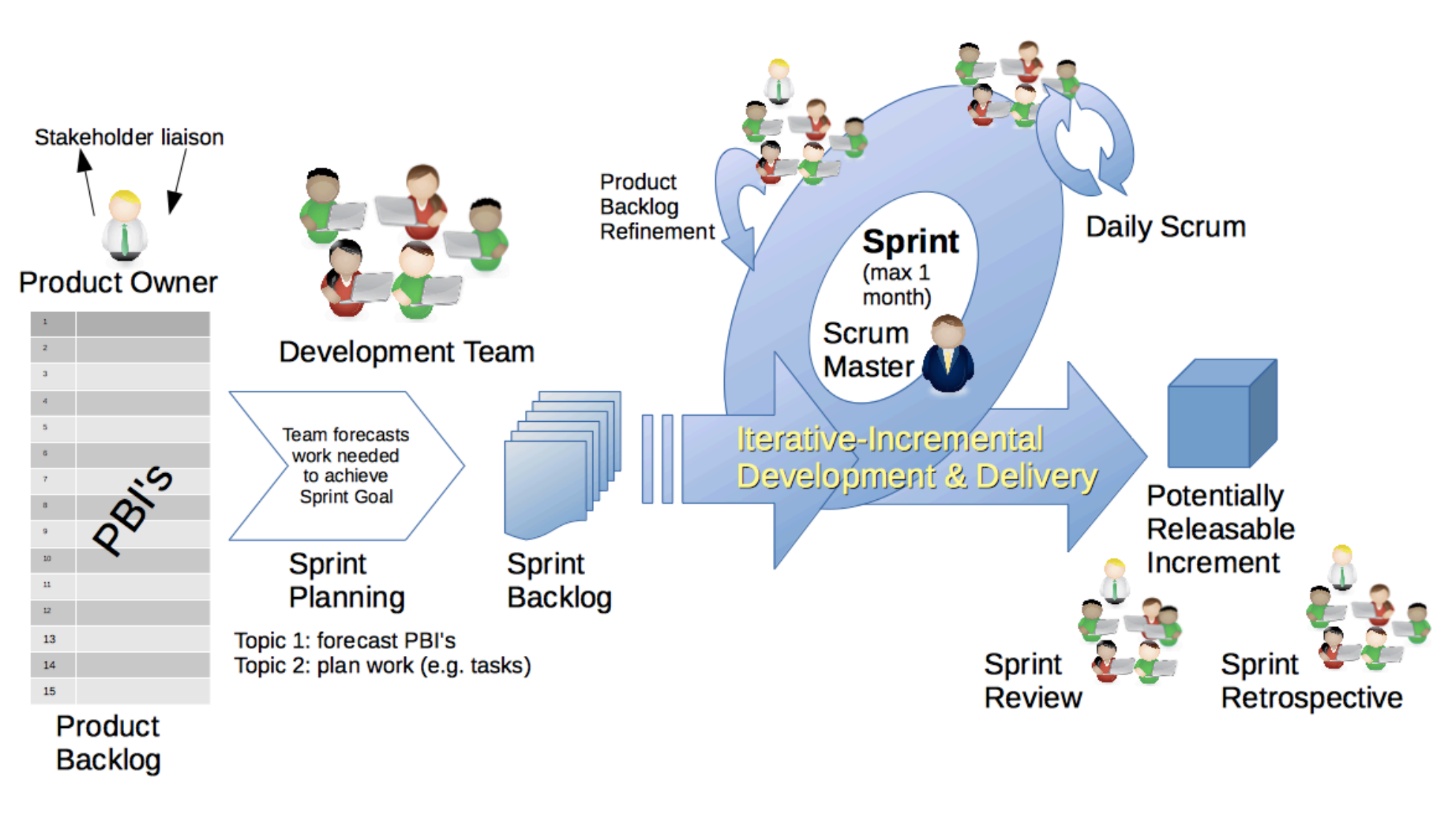
- Kurssin ohjelmistoprojektin projektinhallinnassa noudatetaan suuren suosion saavuttanutta ketteriä menetelmiä soveltavaa **Scrum**-viitekehystä
- Vuonna 2024 julkaistun 17th State of Agile Report -raportin mukaan 63% vastaajista käytti ohjelmistotuotannon prosessinaan Scrumia
- Scrum määrittelee ohjelmistokehitykselle iteratiivisen prosessin, joka etenee tyypillisesti 1-4 viikon iteraatioissa, joita kutsutaan **sprinteiksi**
- Sprintin aikana ohjelmistoon tuotetaan inkrementaalisesti uutta julkaisukelpoista toiminallisuutta
- Kurssilla sprintin pituus on kolme viikkoa

Scrumin roolit

- Ohjelmistoprojektista vastaa **Scrum-tiimi**, jossa on kolme erilaista roolia
- Ohjelmiston toteutuksesta vastaa **kehittäjätiimi**, joka koostuu tyypillisesti 3-9 ohjelmistokehittäjästä
- Sprintin aikana kehittäjätiimi toteuttaa itseorganisoidusti sprinttiin valitut ohjelmiston toiminnallisuudet
- **Scrum master** on usein kehittäjätiimin jäsen, joka toimii sen apuna ohjaten mm. prosessin noudattamisessa ja parantamisessa
- **Tuoteomistaja** (product owner) määrittelee ja priorisoi kehittäjätiimin työtä hallinnoimalla projektin **product backlogia**, joka sisältää priorisoidussa järjestyksessä projektissa toteutettavalle ohjelmistolle asetetut vaatimukset

Scrumin tapahtumat

- Scrumissa käytetään ennaltasovittuja tapahtumia, jotka toistuvat jokaisessa sprintissä
- Sprintti alkaa **sprintin suunnittelulla** (sprint planning), jonka aikana päätetään mitä toiminallisuutta sprintin aikana toteutetaan
- Sprintin aikana järjestetään lyhyitä kehittäjätiimin sisäisiä **päiväpalavereja** (daily scrum), joissa jokainen tiimin jäsen kertoo vuorallaan, miten kehitystyö etenee
- Sprintti päättyy **sprinttikatselmukseen** (sprint review), jonka aikana kehittäjätiimi esittelee sprintin aikana toteutetut toiminallisuudet kaikille kehitettävästä tuotteesta kiinnostuneille sidosryhmille
- Sprintin päätteeksi järjestetään **retrospektiivi**, jossa kehittäjätiimi tarkastelee Scrum masterin johdolla omaa työskentelyprosessiaan ja pyrkivät kehittämään sitä


Stakeholder liaison
Product Owner
Development Team
Product Backlog Refinement
Sprint
(max 1 month)
Scrum Master
Daily Scrum
PBI's
Team forecasts work needed to achieve Sprint Goal
Sprint Planning
Sprint Backlog
Iterative-Incremental Development & Delivery
Potentially Releasable Increment
Topic 1: forecast PBI's
Topic 2: plan work (e.g. tasks)
Sprint Review
Sprint Retrospective
Product Backlog

Tiimien muodostaminen ja aiheen ideointi

1. Jakaudutaan noin 4-5 hengen tiimeihin
2. Tehkää tiimin sisällä lyhyt esittelykierros. Jokainen voi kertoa vuorollaan esimerkiksi:
   - Minkälainen tausta on opinnoissa tai mahdollisesti työelämässä?
   - Mitkä ovat omat kiinnostuksen kohteet ohjelmistokehityksessä?
   - Mitkä ovat omat vahvuusalueet ohjelmointikielissä, tai muissa toteutusteknologioissa?
   - Mitä odotuksia on kurssin suhteen?
3. Valitkaa tiimillenne nimi
4. Alkakaa ideoimaan yhdessä ohjelmistoprojektin aihetta

Aiheen valinta ja rajaus

- Projektissa ei välttämättä ole ulkoista tuoteomistajaa, vaan **tiimi toimii itse tuoteomistajan roolissa**
- Tiimi joutuu tällöin itse määrittelemään projektin vaatimuksia ja priorisoimaan niitä
- Aiheen ideoinnissa voi lähteä liikkeelle laajasta ideasta, kuten "kurssiarvostelu-sovellus", tai "sanaston harjoittelu -peli"
- Tämän jälkeen laajasta ideasta voi muodostaa konkreettisia toiminallisuuksia priorisoiden käyttäjän kannalta tärkeimpiä toiminallisuuksia
- Aluksi kannattaa tähdätä **Minimum Viable Product** (MVP) -toteutukseen, josta toiminallisuutta voi lähteä laajentamaan
- Näin vältetään nk. *"Scope creep"*-ilmiötä, jossa projektin laajuus leviää hallitsemattomasti, eikä projektin ydintoiminnallisuus valmistu järkevässä aikataulussa

Aiheen valinta ja rajaus

> *"Keep it simple, stupid!"*
> —KISS-periaate

- Yksinkertainen ja toimiva sovellus on käyttäjän kannalta mielekkäämpi kuin suuret ja yksityiskohtaiset suunnitelmat hienosta sovelluksesta, jota ei koskaan ehditty toteuttaa