Ohjelmistutuotanto

Luento 2

14.3.2012

Ohjelmiston tuottaminen ei ole kontrolloitu prosessi

- Vesiputousmallin suurimmat ongelmat ovat seuraavat
  - Yleensä mahdotonta määritellä vaatimukset tyhjentävästi projektin alkuvaiheessa
    - Asiakas heti ei ymmärrä mitä haluaa
    - Bisnesympäristö muuttuu projektin kuluessa
  - Suunnittelu sillä tasolla että ohjelmointi on triviaali ja ennustettava rakennusvaihe, rinnastettavissa esim. talon rakennukseen, on mahdotonta.
    - Ohjelmointi on osa suunnitteluprosessia, ohjelmakoodi on tuotteen lopullinen suunnitelma
    - Suunnittelu on teknisesti haastavaa ja riskejä sisältävää toimintaa
- 90-luvun iteratiiviset prosessimallit korjaavat monia näistä epäkohdista
- Kuitenkin 90-luvun mallit olivat vielä vahvasti ”plan-based” ja olettivat että ohjelmistotuotanto on jossain määrin kontrolloitavissa oleva prosessi
  - Tarkka projektisuunnitelma ja sen noudattaminen
  - Selkeä roolijako: projektipäälliköt, analyytikot, arkkitehdit, ohjelmoijat, testaajat

Ketterä prosessimallit

- Useimmat ohjelmistoprojektit ovat laadultaan uniikkeja
  - Vaatimukset erilaiset kuin millään jo tehdyllä ohjelmistolla
  - Uusi tekijätiimi, omanlaisilla kompetensseilla ja persoonallisuuksilla varustettu
  - Toteutusteknologiat kehittyvät, joten tehdään todennäköisesti tavalla joka ei ole kaikille tunnettu
- Järkevää lähteä oletuksesta että kyseessä ei ole kontrolloitu prosessi jota voidaan tarkkaan etukäteen suunnitella (eli ei ”plan-based”)
- Parempi ajatella tuotekehitysprojektina, näiden kontrollointiin sopii paremmin ”empiirinen prosessi”
  - Inspect-and-adapt
- Tekijät yksilöitä, oletus että yksilöt toimivat paremmin kun heihin luotetaan ja annetaan tiimille vapaus organisoida itse toimintansa
  - ”The whole team”-periaate: tiimi kollektiivina vastuussa aikaansaannoksesta
  - Oletuksena että perinteinen Command-and-control ja jako eri vastuu-alueisiin (suunnittelija, ohjelmoija, testaaja) ei tuota optimaalista tulosta

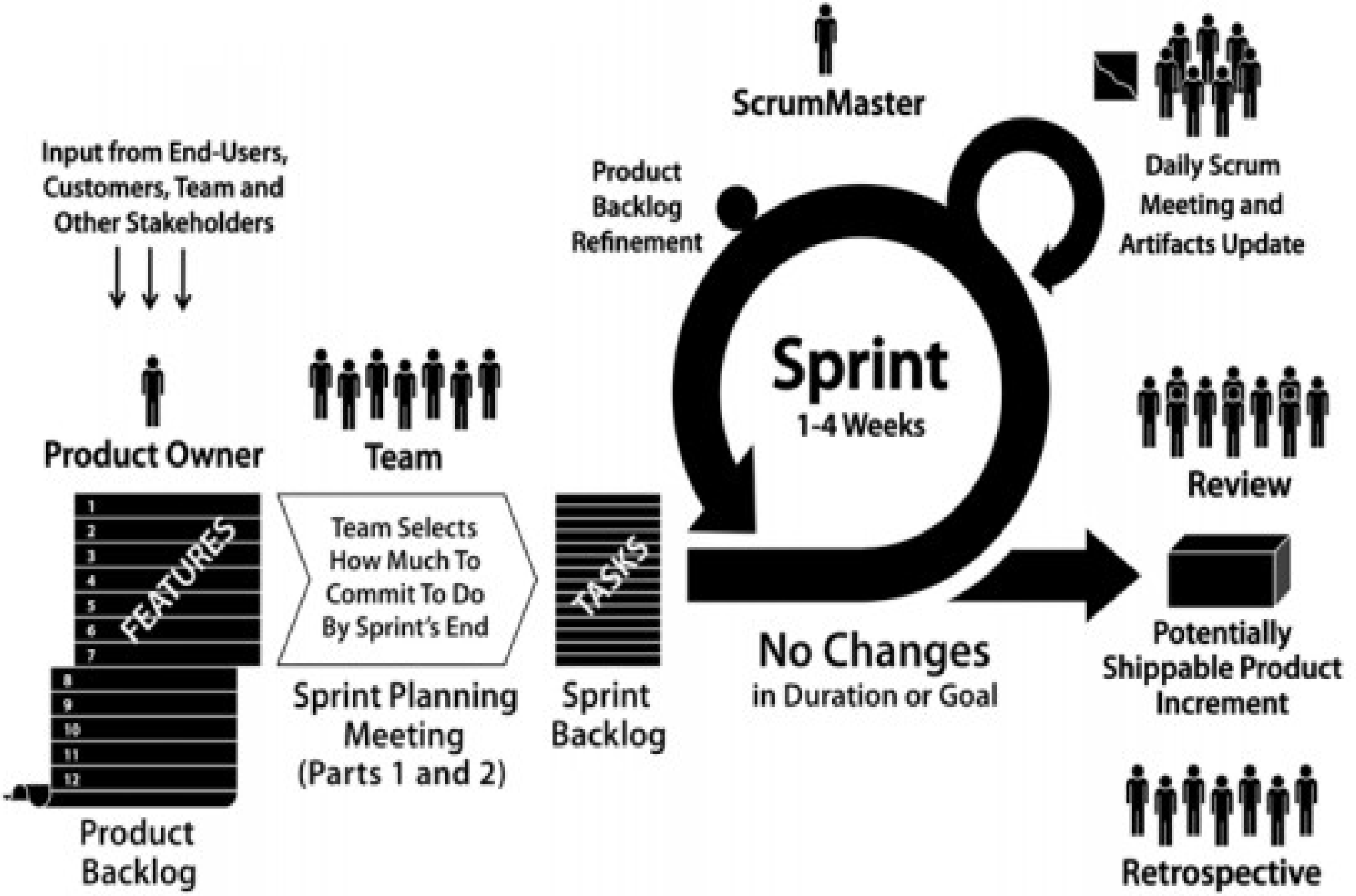
Scrum

- Tutustumme kurssin aikana suhteellisen tarkasti Scrumiin joka on tällä hetkellä selvästi eniten käytetty ketterä menetelmä/prosessimalli
- [Schwaber, Sutherland: The Scrum Guide]
  - Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
  - Scrum is:
    - Lightweight
    - Simple to understand
    - Extremely difficult to master
  - Scrum is a process framework that has been used to manage complex product development since the early 1990s.
  - Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques.
  - Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

Scrum lyhyesti

- Iteratiivinen ja inkrementaalinen menetelmä (tai kehittäjiensä mukaan framework eli menetelmäkehys)
- Kehitys tapahtuu 1-4 viikon iteraatioissa, joita Scrumissa kutsutaan **sprinteiksi**
- **Scrum-tiimi** koostuu 3-10:stä kehittäjästä
- **Scrum master** toimii tiimin apuna ohjaten mm. prosessin noudattamista sekä toimien rajapintana yrityksen hallintoon
- **Product owner** eli tuotteen omistaja hallinnoi projektin backlogia
  - **backlog** sisältää priorisoidussa järjestyksessä projektissa toteutettavan ohjelmiston ominaisuudet/vaatimukset/toiminnot
- Jokaisen sprintin alussa tiimi valitsee projektin backlogista sprintin aikana toteutettavat vaatimukset
- Sprintin aikana scrum-tiimi toteuttaa itseorganisoidusti sprintiin valitut vaatimukset

Scrum: roles, artifacts and evens



Scrum: roles, artifacts and evens

- Käydään vielä läpi hieman seikkaperäisemmin scrumin terminologiaa
- Scrum määrittelee 3 erilaista **roolia**:
  - Kehittäjä
  - Scrum master
  - Product owner
- Scrumiin kuuluvat **artefaktit** eli ”konkreettiset asiat” ovat
  - Product backlog eli projektin kehitysjono
  - Sprint backlog eli sprintin tehtävälista
- Scrumissa tekeminen rytmittyy sprinteihin eli 1-4 viikon mittaisiin iteraatioihin. Sprintteihin kuuluu muutamia **standardipalaverejä** (events):
  - Sprintin kaksiosainen suunnittelupalaveri
  - Daily scrum -palaverit
  - Sprintin katselmus
  - retrospektiivi

Product backlog

- Product backlog on siis priorisoitu lista asiakkaan tuotteelle asettamista vaatimuksista eli toivotuista ominaisuuksista ja toiminnoista
  - Voi sisältää myös esim. isompia bugikorjauksia
- Hyvänä käytänteenä pidetään sitä että backlogissa olevat vaatimukset ovat asiakaan tasolla olevia mielekkäitä toiminnallisuuksia
- Backlogin kärjessä olevat vaatimukset valitaan toteutettavaksi seuraavan sprintin aikana
  - Tämän takia kärjessä olevat vaatimukset on yleensä kirjattu tarkemmin kuin backlogin häntäpään vaatimukset
- Usein on tarkoituksena myös estimoida eli arvioida backlogissa olevien vaatimusten toteuttamisen vaatima työmäärä
  - Työmääräarviot tekee kehittäjätiimi
- Scrum ei määrittele missä muodossa backlog ja siinä olevat vaatimukset esitetään
  - Viime vuosina on yleistynyt käytäntö, jossa tehtävät esitetään ns. *User Storyinä*, tutustumme tähän tekniikkaan ensi viikolla

Product owner

- Scrumin mukaan kuka vaan voi milloin tahansa lisätä backlogiin vaatimuksia
- Backlogia priorisoi ainoastaan **Product owner** eli tuotteen omistaja
- Product owner on yksittäinen henkilö
  - Priorisointiin voi toki olla vaikuttamassa useampikin henkilö, mutta Product owner tekee lopulliset päätökset prioriteettien suhteen
- Product owner on vastuussa backlogista
  - Priorisoi vaatimukset maksimoiden asiakkaan tuotteesta saaman hyödyn
  - Varmistaa että kehittäjätiimi ymmärtää toteutettavaksi valitut vaatimukset

Kehittäjätiimi

- Kehittäjätiimi koostuu noin 3-10:stä henkilöstä, kaikista käytetään nimikettä developer, eli periaatteessa tiimin jäsenet eivät ole erikoistuneet pelkästään yhteen rooliin (kuten ohjelmoija, arkkitehti, testaaja)
  - Oletuksena on että tiimin jäsenet työskentelevät 100%:sti tiimissä
  - Koko tiimin tulee oletusarvoisesti työskennellä samassa paikassa, mieluiten yhteisessä tiimille varatussa avokonttorissa
- Tiimi on ”cross-functional”, eli tiimin jäsenten tulisi sisältää kaikki tarvittava kompetenssi järjestelmän suunnitteluun ja toteuttamiseen
- Pääperiaatteena on että kehitystiimiä **ei johdeta** ulkopuolelta
  - Tiimi päättää mihin tavoitteisiin se kussakin sprintissä sitoutuu, eli mitä vaatimuksia backlogista valitaan sprintissä toteutettavaksi
  - päättää (tiettyjen reunaehtojen puitteissa) itse sen miten sprintin tavoite toteutetaan
- Tiimi on siis ***itseorganisoituva*** (self organizing)

Scrum master

- Jokaisella Scrum-tiimillä on **Scrum master**, eli henkilö joka vastaa siitä että Scrumia noudatetaan kehitystyössä
- Ei perinteinen projektipäällikkö vaan ”servant-leader” rooli
  - Rohkaisee tiimiä itseorganisoitumisessa
  - Opastaa hyvien käytänteiden noudattamisessa
  - Järjestää Scrumiin liittyvät palaverit
  - Pyrkii poistamaan kehitystyön esteitä neuvottelemalla tiimiläisten sijaan yrityksen hallinnon kanssa
  - Suojaa tiimiä esim. ulkopuolisten yrityksiltä puuttua sprintin aikaiseen toimintaan
  - Auttaa tuotteenomistajaa product backlogin ylläpitämisessä
- Eli Scrum master tekee kaikkensa jotta tiimillä olisi optimaaliset olosuhteen kehittää tuotetta

Sprintti

- Scrumissa kehitystyö siis jakautuu 1-4 viikon mittaisiin iteraatioihin eli sprintteihin
  - Sprintin kesto on projektissa tyypillisesti aina sama, nykyään suositus lienee 2 viikkoa
  - Sprintti on ”time-boxed”, eli sprinttiä ei missään olosuhteissa pidennetä
- Jokaisen sprintin alussa tiimi valitsee projektin backlogista sprintin aikana toteutettavat vaatimukset
  - Backlog on priorisoitu ja vaatimukset valitaan aina priorisoidun listan kärjestä
- Tiimi valitsee sprinttiin ainoastaan sen verran toteutettavaa minkä valmistumiseen se kykenee **sitoutumaan** (commit)
- Scrumissa periaatteena on, että jokaisen sprintin lopuksi tuotteesta on oltava olemassa **toimiva versio**
- Sprintin aikana scrum-tiimi toteuttaa itseorganisoidusti sprintiin valitut ohjelmiston ominaisuudet
- Sprintin aikana tiimille ei esitetä uusia vaatimuksia

Definition of done

- Scrum kuten kaikki muutkin ketterät menetelmät asettavat suuren painoarvon tuotetun ohjelmiston laadulle
- Jokaisessa sprintissä siis tulee lopputuloksena olla toimiva, valmiiksi tehty osa ohjelmistoa
- Scrumissa on määriteltävä projektitasolla mitä takoitetaan että jokin vaatimus on toteutettu valmiiksi, eli **definition of done**
- Valmiiksi tehty määritellään yleensä tarkoittamaan sitä, että vaatimus on
  - analysoitu, suunniteltu, ohjelmoitu, testattu, testaus automatisoitu, dokumentoitu ja integroitu muuhun ohjelmistoon
- Eli kun sprintin lopussa tavoitteena on olla toimiva ohjelma, tarkoitetaan sillä nimenomaan definition of done:n tasolla toimivia ja valmiiksitehtyjä vaatimuksia
  - Jos joitain ohjelman osia on tehty huolimattomasti, Scrum master hylkää ne ja siirtää toteutettavaksi seuraavaan sprinttiin
- Jos sprintin aikana osoittautuu että tiimi ei ehdi toteuttamaan kaikea joihin se sitoutui, **ei ole hyväksyttävää tinkiä laadusta**, vaan osa vaatimuksista jätetään seuraavaan sprinttiin

Sprint planning

- Ennen jokaista sprinttiä järjestetään kaksiosainen sprintin suunnittelukokous
- Ensimmäisen osan tarkoitus on selvittää **mitä sprintin aikana tehdään**
  - Tuotteeomistaja esittelee product backlogin kärjessä olevat vaatimukset
  - Tiimin on tarkoitus olla riittävällä tasolla selvillä mitä vaatimuksilla tarkoitetaan
  - Tiimi sitoutuu niin moneen tehtävälistan vaatimukseen kuin se arvioi kykenevänsä sprintin aikana toteuttamaan
- Sprintin aikana toteutettavien vaatimusten lisäksi kokouksen aikana asetetaan *sprintin tavoite* (sprint goal)
  - Tavoite on yksittäisiä vaatimuksia geneerisempi ilmaus siitä mitä tulevassa sprintissä on tarkoitus tehdä
- Suunnittelukokouksen ensimmäisen osan maksimikesto on 4 tuntia jos sprinttien pituus on 4 tuntia ja muuten 2 tuntia

Sprint planning

- Suunnittelukokouksen toisen osan aikana selvitetään **miten sprintin tavoitteet saavutetaan**
  - Product owner ei ole välttämättä enää paikalla, mutta on tavoitettavissa esim puhelimitse
- Toisen osan aikana tiimi suunnittelee toteutettavaksi valitut vaatimukset tarvittavalla tasolla
- Aikaansaannoksena on usein lista tehtävistä (**task**) jotka sprintin aikana on toteutettava jotta sprinttiin valitut vaatimukset saadaan toteutettua
- Suunnittelun aikana identifioidut tehtävät kirjataan **sprintin backlogiin** eli tehtävälistaan
- Myös toisen osan maksimikesto on 4 tuntia jos sprinttien pituus on 4 tuntia ja muuten 2 tuntia

- Palaamme sprintin suunnitteluun tarkemmin ja konkreettisten esimerkkien kanssa ensi viikolla

Daily scrum – päiväpalaveri

- Jokainen päivä sprintin aikana aloitetaan **daily scrumilla** eli korkeintaan 15 minuutin mittaisella palaverilla
- Aina samaan aikaan, samassa paikassa, kaikkien kehittäjien oltava paikalla
- Jokainen tiimin jäsen vastaa vuorollaan kolmeen kysymykseen
  - Mitä sain aikaan edellisen tapaamisen jälkeen?
  - Mitä aion saada aikaan ennen seuraavaa tapaamista?
  - Mitä esteitä etenemiselläni on?
- Kuka tahansa saa olla seuraamassa daily scrumia mutta vain tiimin jäsenillä on puheoikeus
- Palaverin on tarkoitus olla lyhyt ja muu keskustelu ei ole sallittua
  - Jos jollakin on ongelmia, Scrum master keskustelee asianomaisen kanssa daily scrumin jälkeen
- Jos muuhun palaverointiin, esim. suunnitteluun tai vaatimusten tarkentamiseen on tarvetta, tulee palaverit järjestää daily scrumista erillään
  - Scrum ei ota kantaa muihin palavereihin

Sprintin katselmointi

- Sprintin päätteeksi järjestetään sprint review eli katselmointi
- Katselmointiin voi osallistua kuka tahansa
- Informaali tilaisuus jonka aikana tiimi esittelee sprintin aikaansannoksia
  - Katselmoinnissa tarkastellaan/demotaan toteutettua, toimivaa ohjelmistoa, powerpoint-kalvojen näyttäminen katselmoinnissa on kielletty!
- Scrum master huolehtii että ainoastaan niitä ominaisuuksia demonstroidaan jotka on toteutettu ”kokonaan” eli definition of donen mukaisesti
- Product owner varmistaa mitkä vaatimuksista toteutettiin ja mitkä jäivät toteuttamatta ja siirretään takaisin product backlogiin
- Katselmoinnin aikana kuka tahansa saa antaa palautetta tuotteesta ja esim. ehdottaa uusia vaatimuksia lisättäväksi product backlogiin
- Katselmointi aiheuttaa usein myös tarpeen product backlogin osittaiseen uudelleenpriorisointiin
- Myös katselmoinnin kesto on rajoitettu (4h tai 2h riippuen sprintin kestosta)

Retrospektiivi – inspect and adapt

- Sprintin katselmoinnin ja seuraavan sprintin alun välissä pidettävä palaveri jonka aikana tiimi tarkastelee omaa työprosessiaan
  - Identifioidaan mikä meni hyvin ja missä asioissa on parantamisen varaa
  - Mietitään ratkaisuja joihinkin ongelmakohtiin pyritään toimimaan ratkaisujen mukaan seuraavan sprintin aikana
- Retrospektiivien ja koko scrumin tärkein ohjenuora onkin **inspect and adapt:**
  - Lyhyt kehityssykli mahdollistaa vaatimusten uudelleenpriorisoinnin ja muuttamisen ymmärryksen kasvaessa tai bisnesympäristön muuttuessa
  - Retrospektiivi kannustaa tiimiä jatkuvasti parantamaan työprosessiaan
  - Daily scrumit tuovat esiin projektin tilanteen päivittäisellä tasolla kaikille tiimin jäsenille
  - Jokaisen sprintin yhteydessä järjestetään uusi sprintin suunnittelu joka mahdollistaa kehitystyön aikana opitun huomioimisen priorisoinnissa ja uusien ominaisuuksien suunnittelussa

No silver bullet...

- Scrum on osoittautunut monin paikoin paremmaksi tavaksi ohjelmistojen tuottamiseen kuin vesiputousmalli tai muut suunnitelmavetoiset mallit
- Scrum ei kuitenkaan ole mikään ”silver bullet” ja Scrumin käytön yleistyessä myös epäonnistuneiden Scrum-projektien määrä kasvaa
- Yksi ongelmista on ns. **scrumbut**
  - ”we are doing scrum but ...”, ks, http://antoine.vernois.net/scrumbut/
- Toisin kuin esim. eXtreme Programming, Scrum ei määrittele mitään teknisiä käytänteitä vaan luottaa itseorganisoidun tiimin kykyyn tuottaa laadukasta jälkeä. Läheskään aina tämä ei toteudu ja lupaavan alun jälkeen tiimi saattaa joutua vaikueuksiin, ks:
  - ks. http://www.martinfowler.com/bliki/FlaccidScrum.html
- Hajautettu ohjelmistotuotanto, alihankkijoiden käyttö ja massiivista kokoluokkaa olevat projektit aiheuttavat edelleen haasteita Scrumille ja muillekin ketterille menetelmille vaikkakin asiaan on viime vuosina kiinnitetty huomiota
- Robert Marinin Scrum kritiikkiä (Martin on yksi agile manifestin allekirjoittajista)
  - http://tech.groups.yahoo.com/group/scrumdevelopment/message/44851
- Päätetään alustava Scrumiin tutustumisemme menetelmän kehittäjien sanoihin *”Scrum is easy to undestand but extremely difficult to master”*

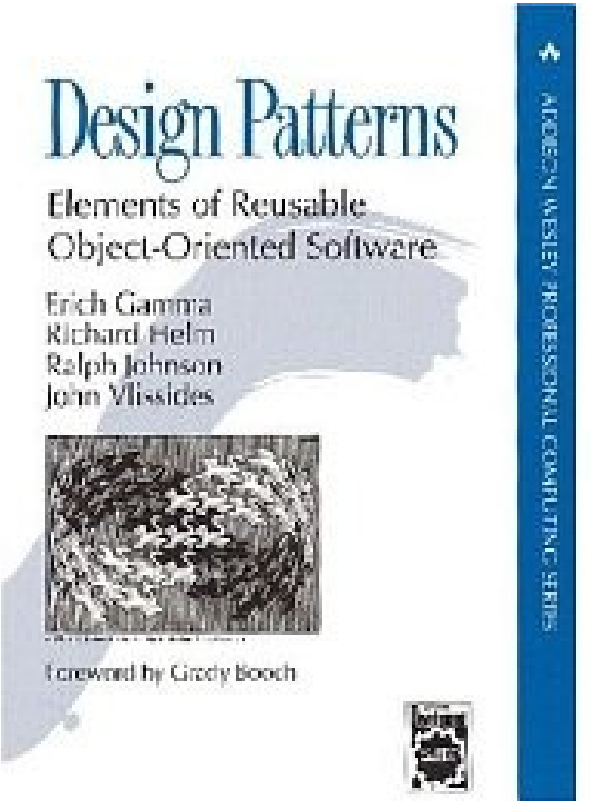
Design patterns – suunnittelumallit

- Ohjelmistotuotannon yksi paha ongelma on se, että pyörä keksitään jatkuvasti uudelleen
- Design patternit eli suunnittelumallit tarjavat pienen lääkkeen tähän ongelmaan
- Mistä on kysymys? Annetaan ”Gang of four:in” kertoa:

A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems.

It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples. The solution is a general arrangement of objects and classes that solve the problem.

The solution is customized and implemented to solve the problem in a particular context



Design patters – suunnittelumallit

- Suunnittelumallit siis dokumentoivat hyväksi tunnettuja suunnitteluratkaisuja
- Gang of Fourin kirja 1994 lanseerasi suunnittelumallien käsittelyn tietotekniikkaan
  - Siitä lähtien on ilmestynyt lukematon määrä julkaisuja aiheeseen liittyen
- Suunnittelumallit jakautuvat useisiin ”aliluokkiin”
  - Olioiden luomista helpottavat mallit
  - Ohjelman oliorakennetta ohjaavat mallit
  - Ohjelman suoritusta ja laskentaa ohjaavat mallit
- On olemassa myös muita ”patterneja”, mm:
  - Architectural patterns: arkkitehtuurimallit
  - Project management patterns: esim. Scrumin voidaan ajatella olevan tälläinen
- Tutustumme kurssin aikana muutamiin suunnittelumalleihin

Design pattern of the day – dependency injection

- Kurssin ensimmäinen suunnittelumalli **dependency injection**, eli riippuvuuksien injektointi ei löydy juuri miltään perinteiseltä suunnittelumallilistalta
- Kyseessä on perimmiltään hyvin yksinkertainen menetelmä jonka avulla luokkien riippuvuuksia toisista luokista pyritään minimoimaan
  - Yksi DI:n suurista motivoivista tekijöistä on yksikkötestauksen helpottaminen

- Lyhyt ja simppeli selitys asiasta: http://jamesshore.com/Blog/Dependency-Injection-Demystified.html
- Luennolla esitetyt koodit löytyvät viikon 2 laskareiden tehtävänannosta
- Riippuvuuksien injektointi yhdessä ns. Inversion of Control -periaatteen kanssa on noussut viime vuosina suosituksi rakenneperiaatteeksi sovelluskehyksissä, esim. Spring:issä
  - ks. http://martinfowler.com/articles/injection.html