OTP1

Ohjelmistotuotanto-projekti 1

Ohjelmistotuotantoprojekti 1 (5 op)

- Tavoite
  - Saada kuva ohjelmistotuotantoprosessista kokonaisuutena
  - Ymmärtää ohjelmistotuotannon vaiheet ja erilaiset ohjelmistoprosessit
  - Osata käyttää Scrumia ominaisuus- ja työlistojen suunnittelussa
  - Tuntea ohjelmistotuotannon vaihetuotteiden sisällöt ja osata tuottaa niihin liittyviä dokumentteja
- Samaan aikaan suoritetaan kurssi Kuvaus- ja mallintamismenetelmät
  - Oliopohjaisten järjestelmien UML-mallinnusta
- Samaan aikaan kurssi Käyttäjäkeskeinen suunnittelu
  - Mitä huomioitava käyttöliittymää laadittaessa

Kaikkien ryhmäläisten on oltava paikalla ainakin erikseen merkityillä oppitunneilla. Ryhmätyö sujuu ketterimmin silloin, kun kaikki ryhmän jäsenet työskentelevät yhtä aikaa samassa tilassa ja kommunikoivat kasvokkain.

Ohjelmistotuotanto - Sofware Engineering

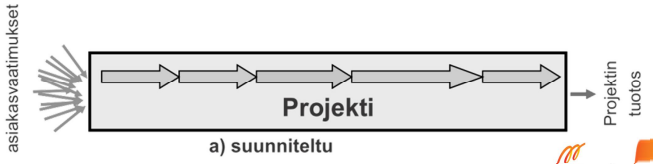
Määritelmiä

- Designing, building and maintaining large software systems
- Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products. *(I.Sommerville)*
- The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them. *(B.W.Boehm)*
- The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines. *(F.L.Bauer)*
- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. *(IEEE Standard 610.12)*
- The technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost constraints. *(R. Fairley)*

OHJELMISTOPROSESSI

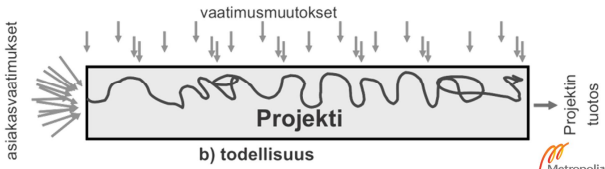
Ohjelmistoprosessi

- Ohjelmiston elinkaaren rakenteen määrittely eli millaisista vaiheista ohjelmiston luonti ja olemassaolo koostuu
  - Suunnittelu, tuotanto, ylläpito
- Joskus puhutaan vaihejakomalleista, nykyään useimmiten prosesseista (ohjelmistotuotanto- tai ohjelmistoprosessi)
- Kun haluatte ohjata prosessia ("tuotantoa"), se täytyy kuvata ensin
  - Pystytte sopimaan työtavoista
  - Ymmärrätte mitä kukin tarkoittaa (yhteinen kieli)
  - Pystytte paremmin valvomaan laatua



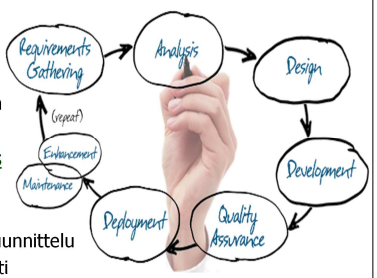
Yleisimmät vaiheet

- Ohjelmistoprosessi on ne työvaiheet, joita tehdään järjestelmän kehittämiseksi
  - Millaisiin järkeviin työvaiheisiin työ on syytä jakaa?
- Prosessimallit ovat abstrakteja esityksiä näistä prosesseista
  - Prosesseilla ja niiden työvaiheilla yritetään kuvata työn eteneminen
- Prosessimallit sovelletaan yritykselle ja sovellusalueelle sopiviksi
  - Käytännön projektit noudattavat kuvauksia enemmän tai vähemmän
- Projektit ovat prosessin ilmentymiä, niissä toteutetaan prosessin työvaiheet



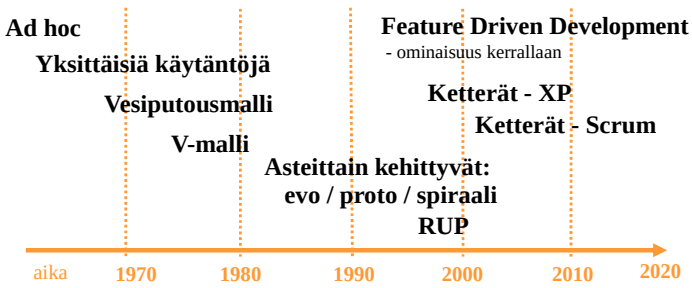
Yleisimmät vaiheet

- Ohjelmiston määrittely
  - Selvitettävä ohjelmistolle asetetut asiakasvaatimukset ja ohjelmistoon liittyvät rajoitukset
- Ohjelmiston suunnittelu ja toteutus
  - Määrittelyn mukainen ohjelmisto pitää saattaa käyttökuntoon
  - Yleisarkkitehtuurin ja moduulien suunnittelu
  - Ohjelmointi ja moduulien integrointi
- Ohjelmiston verifiointi ja validointi
  - Verifiointi tarkastaa, että ohjelmisto toimii virheettömästi
  - Validointi tarkastaa, että ohjelmisto toteuttaa sille määrittelyssä asetetut vaatimukset
- Ohjelmiston ylläpito
  - Ohjelmisto muuttuu elinkaarensa aikana
  - Käyttäjien vaatimukset, käyttöympäristö ja laitteisto muuttuvat



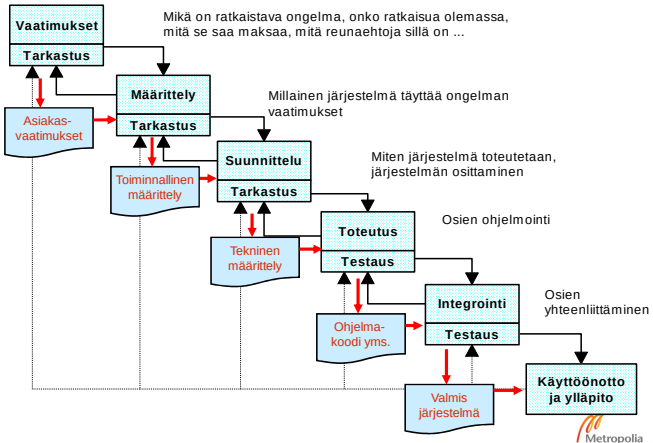
arvio: elinkaarikustannuksista jopa 50-90% ylläpidosta

Prosessimallit - jatkumo perinteitä



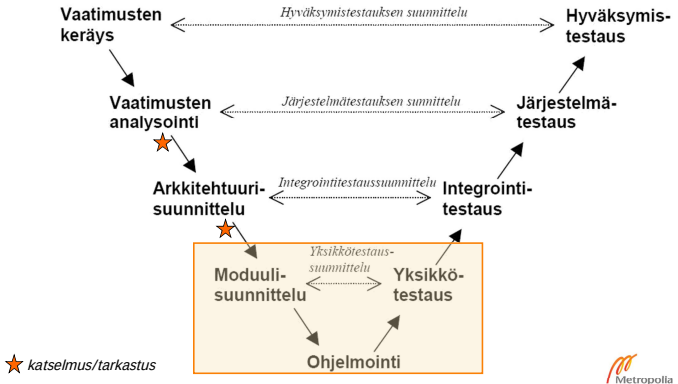
- Se, montako käytäntöä/periaatetta prosessimallissa on, vaihtelee suuresti
- RUP: 120+, XP: 13, Scrum: 10, Kanban: 3

Vesiputousmalli



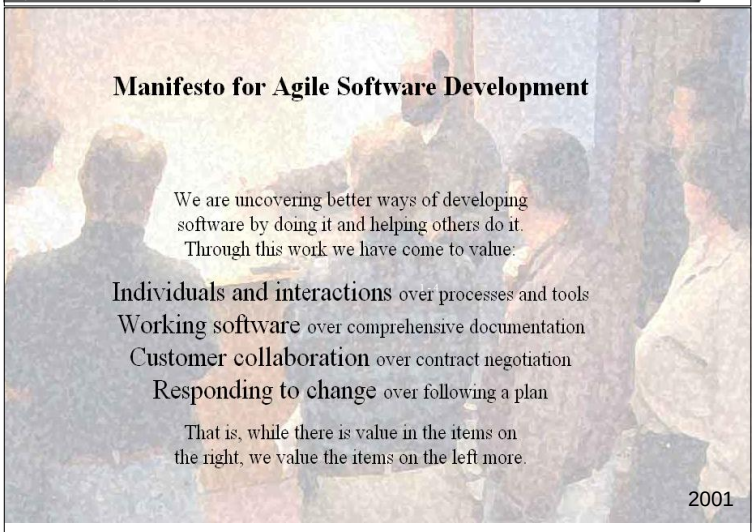
V-malli

- Testaus jaoteltavissa perinteisen vesiputousmallin mukaan



Ketterät menetelmät (Agile SW development)

- Ketterät työtavat yleistyivät voimakkaasti 1990-luvun lopulla
  - muutoksia ei voi eikä haluta välttää - normaalia ohjelmistojen kehityksessä
    - kaiken suunnitteleminen etukäteen vaikeaa
  - vapaamuotoisempi kanssakäyminen: kaikki ei paperilla, osa vain korvassa
  - kommunikointitekniikat kehittyivät ja yleistyivät merkittävästi
    - sähköposti, www, verkkoviestintä, multimedia, ...
- Ketteryys = mahdollisuus reagoida tarpeisiin / muutoksiin nopeammin
  - tuotetaan uusia versioita usein
  - arvioidaan ominaisuuksien tärkeysjärjestys kunkin kierroksen jälkeen
- Väitetään, että
  - lopputuloksena paremmin sitä mitä asiakas haluaa
    - mm. tuotekehityksen viimemuutokset nopeammin mukaan
  - virheettömämpää koodia
  - projektit pysyvät paremmin aikatauluissaan
    - myös asiakas näkee missä milläkin hetkellä mennään
      à parempi mahdollisuus priorisointiin
  - joskus jopa kuluissa saadaan säästöä



agilemanifesto.org www.martinfowler.com/articles/newMethodology.html

Ketterät menetelmät

- Kaikissa suunnilleen samat perusideat, mausteissa eroa
- Asiakaskeskeisyys
  - kehittäminen tapahtuu asiakkaan kanssa jatkuvassa yhteistyössä
  - toteutetaan asiakkaan priorisoimat tärkeät toiminnot ensin
  - säännölliset on-line demot ja katselmoinnit
- Tuotekeskeisyys
  - korostetaan toteutuksen (tuloksen) toimivuutta ja tehtävään sopivuutta
  - toimitetaan tiuhaan uusia, täydennettyjä versioita
- Tiivis yhteistyö ja kommunikointi
  - ohjelmoijien ja kohderyhmien välillä
    - vaatimukset ja aikomukset tulkitaan varmemmin oikein
  - henkilökohtainen kommunikointi ja tiedonvälitys (face-to-face)
    - suullinen sopiminen, suullinen palaute
    - s-posti ja muut verkkoviestinnän keinot vasta toissijainen

vrt. face-to-book J

Ketterät menetelmät

- Tavoite: nopea, mukautuva kehitysprosessi
  - uudet ominaisuudet demonstroidaan asiakkaalle hetimmiten
  - toteutuksessa oltava valmiutta jatkuvaan (tuotteen) muutokseen
  - vaatii kykyä muuttaa toimintaa palautteen perusteella
  - ei saa pantata tietoa / epäilystä / tietämättömyyttä
- Iteratiivisuus
  - suunnittelu, toteutus, testaus ja palaute vuorottelevat vesiputousmallia tiiviimmässä tahdissa
  - "mini V-malliin" perustuvaa iterointia
- Inkrementaalisuus
  - jatkuva integrointi
  - jokaisen iteraation tuotettava lisäarvoa asiakkaalle

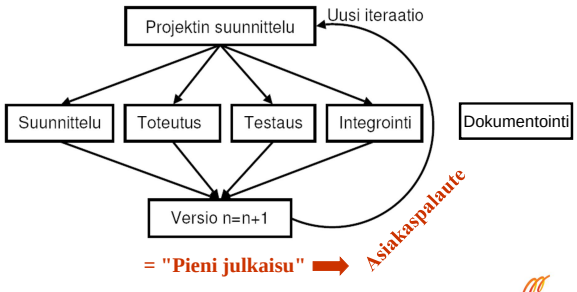
Ketterät menetelmät

- Suositaan pienehköjä tiimejä
  - tiivis toimintayhteisö, rehellinen kommunikaatio
  - kysyy jäseneltä monipuolisia atk-alan taitoja
  - sama tiimi tekee kaiken alusta loppuun
  - kaikki osaavat ronkkia suunnitelmaa ja kaikkea koodia
- Tiimin itseorganisoituvuus ja toiminnan "säätäminen"
  - kokoonpano projektin osaamistarpeiden mukaan
  - kukin tiimi taaplaa omalla tyylillään ja menetelmillään
  - mitä teimme hyvin?
  - mitä tällä kertaa opimme?
  - miten voimme parantaa toimintaamme?
- Yksinkertaisesti, ja yksinkertaista vielä lisää
  - maksimoi taito jättää tekemättä se mitä ei tarvitse tehdä (vielä)
  - "ei me tulla tarvitsemaan sitäkään, eipä etkoilla"
  - "älä puserra puppua", joskus jopa 75% turhia vaatimuksia

Suunnittelu
Ohjelmointi
Testaus
Dokumentointi
Asiakasdemo
Ylläpito
yMs.

Ketterät menetelmät

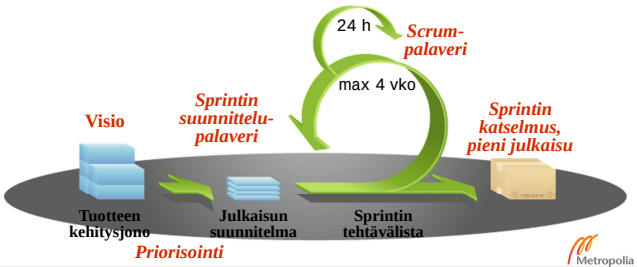
- Iteratiivisuus, inkrementaalisuus, jatkuva integrointi
- Jatkuva testaus + laadunvarmistus + tuloksen arviointi



SCRUM

Scrum = Projektinäkökulma

- Miten projektia edistetään
- Hanke jaetaan 2-4 viikkoa kestäviin pyrähdyksiin, sprintteihin
  - tiimi toteuttaa sprintin tehtävälistalla olevat tehtävät
  - vaiheen aikana tehtäviin ei tehdä muutoksia
- Ei kiinnitä suunnittelu- / toteutusmenetelmiä tai työkaluja
  - tiimi tekee valitsemallaan tavalla



scrum = aloitus, ryhmitys, tungos, lauma

Scrum - Roolit

1. (Tuote)Omistaja, tuotteen tilannut asiakas (product owner, PO)
   - visio tuotteesta: määrittää tavoitteet, user storyt
   - ylläpitää (omistaa) tuotteen kehitysjonoa / ominaisuuslistaa
   - käytettävissä jatkuvasti yksityiskohtien selvittämisessä
   - priorisointi, rahoitus
2. Kehitystiimi, alle kymmenen henkilöä
   - tekninen osaaminen
   - itseorganisoituva, itsenäinen, ei roolijakoa etukäteen
   - kehittää omat työtapansa ja menetelmänsä
   - ottaa vastuun: mitä tekee seuraavassa sprintissä
   - auttaa asiakasta
3. Scrum-mestari (scrum-master, SM)
   - eräänlainen tiimin projektipäällikkö / -sihteeri, ”yleisjantunen”
   - huolehtii, että työskentely sujuu häiriöittä
   - pitää tuotteen omistajan ajan tasalla: varmistaa asiakasarvon toteutumisen
   - ei kuitenkaan erityistä määräysvaltaa tai esimiesasemaa

Scrum - Listat (backlogs)

1. Koko tuotteen kehitysjono / ominaisuuslista (product backlog)
   - lista, joka sisältää tuotteen omistajan kannalta arvokkaita asioita
   - tuotteelle toteutettavaksi aiotut ominaisuudet/piirteet ~ (asiakas)vaatimukset
   - priorisointi ~ tiekartta
   - työmääräarviot, aikataulu ~ julkaisusuunnitelma
   - alkupään ominaisuudet pilkottu riittävän pieniksi, jotta työmäärä pystytään arvioimaan (10 työpäivää yksi henkilö)
2. Julkaisusuunnitelma / julkaisun ominaisuuslista (release backlog)
   - tuotteen kehitysjonosta seuraavaan julkaisuun valitut vaatimukset
3. Sprintin tehtävälista (sprint backlog)
   - tiimin todo-päiväkirja
   - julkaisuun valitut vaatimukset pilkottava pienehköihin osatehtäviin (tasks)
   - usein puhtaasti teknisiä asioita joita pitää tehdä, jotta haluttu ominaisuus saadaan lisättyä tuotteeseen
   - suunnittelu, toteutus, testaus, dokumentointi

backlog = suma, kasauma, ruuhka

Scrum - Palaverit

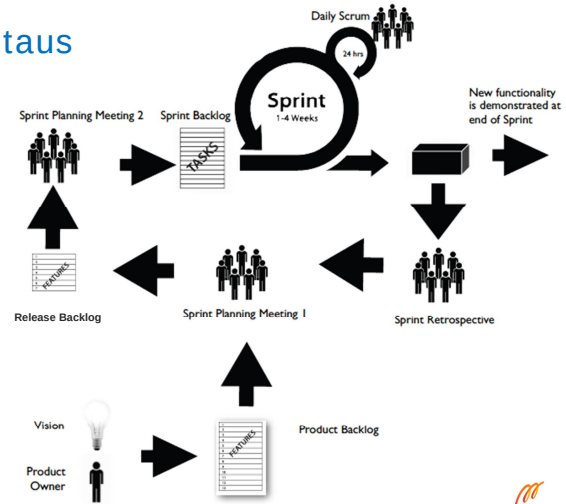
1. Sprintin suunnittelupalaveri (sprint planning)
   - tiimi poimii yhdessä asiakkaan kanssa tuotteen kehitysjonosta prioriteetti-järjestyksessä ne, jotka se pystyy yhdessä sprintissä tekemään
   - tiimi pilkkoo ominaisuudet (osa)tehtäviksi, a’ 1/2 pv – kork. 2 pv
2. Päivittäinen Scrum-palaveri (daily scrum)
   - kuka tahansa voi osallistua, mutta vain tiimin jäsenillä puheoikeus
   - kestää korkeintaan vartin
   - jokainen tiimiläinen vastaa 3 kysymykseen
     - Mitä olet tehnyt edellisen palaverin jälkeen?
     - Mitä aiot tehdä seuraavaan palaveriin mennessä?
     - Mikä on hankaloittanut työtäsi?
   - ei muuta keskustelua
   - Scrum-mestari pyrkii heti poistamaan esille nousseita työn haittatekijöitä
3. Sprintin asiakasdemo (sprint review)
   - jokaisen sprintin lopuksi "julkaisutilaisuus"
   - tuotteen omistaja päättää, mitkä vaatimukset on täytetty (hyväksyy)
4. Sprintin arviointipalaveri (sprint retrospective)
   - silloin tällöin
   - tiimi keskustelee onnistumisestaan
   - sprinttiä ei pidennetä, tiimin opittava arvioimaan työmäärä paremmin

Milloin valmis on valmis? (definition of done, DoD)

"Tuote on parasta tuoda markkinoille, kun se on valmis."

- Hyväksymiskriteerit kiinnitettävä etukäteen
  - heti, kun ominaisuus otettu mukaan sprinttiin
- Sprintti / julkaisu ei ole valmis, ellei myös testaus ja dokumentointi kunnossa
  - myös testaus- ja dokumentointitehtävät kirjattava sprintin tehtävälistoihin ja otettava huomioon aikatauluissa
- Dokumentointi kaikkien jäsenten tehtävä
  - dokumentointia tehtävä jatkuvasti, ei vain lopuksi
- Testaaminen kaikkien jäsenten tehtävä
  - testausta tehtävä jatkuvasti, ei vain lopuksi
  - Automatisointi
- Laatutyö kuuluu kaikille

Kertaus



XP

PERIAATTEITA JA KÄYTÄNTÖJÄ

XP - eXtreme Programming

- Määrittelee joukon hyväksi havaittuja ohjelmistokehityksen periaatteita ja käytäntöjä
- Näistä poimitaan yleensä ketterästi "rusinat pullasta":
  noudatetaan sellaisia käytänteitä, jotka koetaan järkeviksi

Software Engineering Practice	*XP Principles*
Code Reviews are good	Review code all the time
Testing is good	Everybody tests all the time
Design is good	Part of daily business
Simplicity is good	Always have the simplest design that does the job
Architecture is important	Everybody works in refining architecture
Integration Testing is important	Continuously integrate and test
Short Iterations are good	Make iterations really short

XP

Yhteiset käytännöt	Iteraatiot Yhteinen sanasto (metafora) Avoin työtila Aiemmista kokemuksista oppiminen
Ohjelmoijien käytännöt	Testilähtöinen ohjelmointi (testaus) Pariohjelmointi Uudelleenrakentaminen Yhteisomistajuus Jatkuva integrointi Et tule tarvitsemaan sitä (yksinkertainen rakenne)
Hallinnon käytännöt	Hyväksytty vastuu Taustatuki Neljännesvuosikatsaus Peili Tasainen työtahti (40-tuntinen työviikko)
Asiakkaan käytännöt	Tarinoiden kerronta Julkaisujen suunnittelu (suunnittelupeli) Hyväksyntätestaus Lyhyin väliajoin tuotettavat julkaisut (pienet julkaisut)

Collins & Miller

Ohjelmoijien käytännöt

- Koodin "yhteishuoltajuus" (collective ownership)
  - kuka tahansa voi (osaa) ronkkia mitä tahansa koodia
- Noudata vakiintuneita koodauskäytäntöjä (code conventions)
  - tee koodisi lukeminen helpoksi toisille
  - noudata esim. tunnusten nimeämisessä johdonmukaisuutta
  - kirjoita kommentit
- Uudelleenrakentaminen (refactoring)
  - vaikka sait sen toimimaan, niin viilaa se vieläkin paremmaksi
  - paranna / yksinkertaista moduulien rakennetta
  - paranna toteutuksen tehokkuutta
- Et tule tarvitsemaan sitä (simple design)
  - muista, että ohjelmisto tekee vain sen mikä tarpeen
  - älä puuhastele turhia va-utsi-raitoja tai tarpeettomia tör-ä-yttimiä

Ohjelmoijien käytännöt

- Pariohjelmointi (pair programming)
  - kehitä koodia kaverin kanssa saman tietokoneen ääressä
  - vaihda rooleja (koodaaja, katselmoija)
- Testilähtöinen kehitys (test-driven programming, TDD)
  - kirjoita testit ennen koodaamista
  - automatisoi ja aja testit säännöllisesti
- Jatkuva integrointi (continuous integration)
  - integroi koodi muuhun kokonaisuuteen vähintään joka päivä
  - mutta aja ensin testejä kunnes ne menevät läpi
- Käytä versionhallintaa
  - vie testaamasi uutuudet viimeisimmän julkaisun versioon
  - yhteishuoltajuus sekä jatkuva integrointi vaatii tätä

Asiakkaan käytännöt

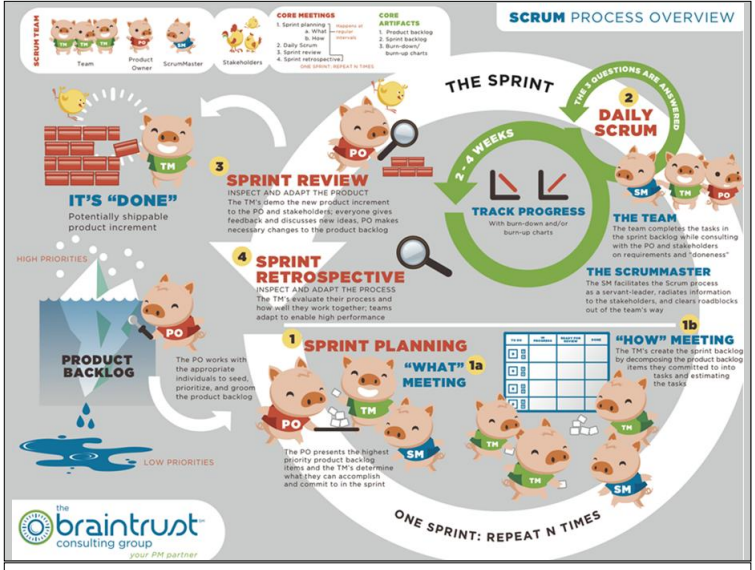
- Läsnäoleva asiakas (on-site customer)
  - pidä asiakas koko ajan mukana tiimissä
- Suunnittelupeli (the planning game)
  - valitse iteraatiossa toteutettavat käyttäjätarinat
  - arvioi toteuttamisen aikataulu
  - voit korvata käyttäjätarinoilla määrittelydokumenttia
- Tarinoiden kerronta, vertauskuvat (metaphor)
  - anna vertauskuvallinen selitys järjestelmästä
  - voit korvata teknistä arkkitehtuurikuvausta (osittain)
- Hyväksymiskriteerit (acceptance tests, ATTD)
  - kiinnitä iteraation hyväksymiskriteerit etukäteen
- Pienet inkrementit, pienet julkaisut
  - pidä iteraatio lyhyenä, 1-4 viikkoa
- max 40 tuntinen työviikko

LOPUKSI
OTP1 VS. OTP2
SCRUM-KUVA
DEVOPS-KUVA

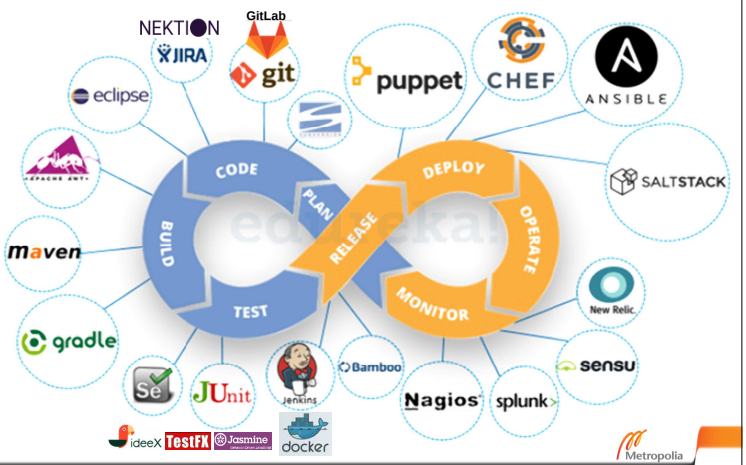
Lopuksi

- OTP ei ole pelkkä koodausprojekti
  - Toteutus ei välttämättä edes tule valmiiksi näiden projektien aikana
- Tarkoitus harjoitella ketteriä menetelmiä ja käytäntöjä
  - Miten käyttää projektissa versionhallintaa
  - Miten tehdä TDD-tyyliin työtä
  - Miten virittää jatkuva integrointi
- Tarkoitus harjoitella käytännössä, miten Scrum-projekti viedään läpi
  - Miten kirjoittaa käyttäjätarinoita
  - Miten arvioida mitä sprinttiin uskaltaa ottaa
  - Miten nähdä joka päivä, kuinka paljon työtä tehty ja paljonko jäljellä
  - Miten päättää sprintti

Katso Software Engineering -kirjan kirjoittajan Ian Sommervillen video "10 kysymystä ohjelmistotuotannosta"
- https://www.youtube.com/watch?v=gi5kxGslkNc



DevOps · 24/7-järjestelmä: kehitä, käytä, kehitä, käytä …



Onnistuminen

CHAOS MANIFESTO
Think Big, Act Small

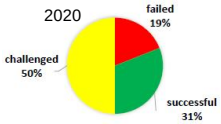
	1994	1996	1998	2000	2002	2004	2006	2008	2010	2012	2013	2014	2015
succeeded	16 %	27 %	26 %	28 %	34 %	29 %	35 %	32 %	37 %	39 %	41%	36%	36%
failed	31 %	40 %	28 %	23 %	15 %	18 %	19 %	24 %	21 %	18 %	19%	17%	19%
challenged	53 %	33 %	46 %	49 %	51 %	53 %	46 %	44 %	42 %	43 %	40%	47%	45%

2015

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

2013-2017

2020



Ongelmakohtia

34%	**Failure to integrate people**
28%	**Failure to teach team-based culture**
21%	**Communication between Developers / Product-Owner**
9%	**Communication between Developers / Quality Assurance**
8%	**Scrum Master problem**

VersionOne: State of Agile Development Survey for 2012