

Ohjelmistotuotanto

Syksy 2024

Luento 1

28.10.2024



David Gilmour

OCTOBER Sold Out, Sold Out, 31

HOLLYWOOD BOWL

PERFORMING SONGS FROM THE NEW ALBUM

'LUCK AND STRANGE' AND PINK FLOYD CLASSICS

Ohjelmistotuotanto

Luenntot:

- ▶ Matti Luukkainen
- ▶ Joulukuun vierailijat

Ohjaajat:

- ▶ Sini Arkko
- ▶ Taneli Härkönen
- ▶ Riku Rauhala
- ▶ Pooki Vehviläinen
- ▶ Antti Vuorenmaa

Team Ohtu



Ohjelmistotuotanto

?

Ohjelmistotuotanto

Johdanto *ohjelmistotuotantoon* (engl. software engineering), eli systemaattiseen tapaan tehdä hieman laajempia ohjelmistoja useamman hengen tiimissä ulkoiselle asiakkaalle

Ohjelmistotuotanto

Johdanto *ohjelmistotuotantoon* (engl. software engineering), eli systemaattiseen tapaan tehdä hieman laajempia ohjelmistoja useamman hengen tiimissä ulkoiselle asiakkaalle

Erityinen paino ns. ketterissä (engl. agile) ohjelmistotuotantomenetelmissä

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyössä, esim. *ohjelmistoprojektissa TKT20007*

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyössä, esim. *ohjelmistoprojektissa TKT20007*

Suoritettuaan kurssin opiskelija

- ▶ tuntee ohjelmistoprosessin vaiheet (vaatimusmäärittely, suunnittelu, toteutus ja laadunhallinta)

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyömissä, esim. *ohjelmistoprojektissa TKT20007*

Suoritettuaan kurssin opiskelija

- ▶ tuntee ohjelmistoprosessin vaiheet (vaatimusmäärittely, suunnittelu, toteutus ja laadunhallinta)
- ▶ tietää miten vaatimuksia hallitaan ketterässä ohjelmistotuotantoprosessissa

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyömissä, esim. *ohjelmistoprojektissa TKT20007*

Suoritettuaan kurssin opiskelija

- ▶ tuntee ohjelmistoprosessin vaiheet (vaatimusmäärittely, suunnittelu, toteutus ja laadunhallinta)
- ▶ tietää miten vaatimuksia hallitaan ketterässä ohjelmistotuotantoprosessissa
- ▶ ymmärtää suunnittelun, toteutuksen ja testauksen vastuut ja luonteen ketterässä ohjelmistotuotannossa

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyömissä, esim. *ohjelmistoprojektissa TKT20007*

Suoritettuaan kurssin opiskelija

- ▶ tuntee ohjelmistoprosessin vaiheet (vaatimusmäärittely, suunnittelu, toteutus ja laadunhallinta)
- ▶ tietää miten vaatimuksia hallitaan ketterässä ohjelmistotuotantoprosessissa
- ▶ ymmärtää suunnittelun, toteutuksen ja testauksen vastuut ja luonteen ketterässä ohjelmistotuotannossa
- ▶ ymmärtää laadunhallinnan perusteet

Kurssin oppimistavoitteet

Tiedolliset ja tekniset valmiudet toimia *juniorikehittäjän roolissa* pienessä ohjelmistotyömissä, esim. *ohjelmistoprojektissa TKT20007*

Suoritettuaan kurssin opiskelija

- ▶ tuntee ohjelmistoprosessin vaiheet (vaatimusmäärittely, suunnittelu, toteutus ja laadunhallinta)
- ▶ tietää miten vaatimuksia hallitaan ketterässä ohjelmistotuotantoprosessissa
- ▶ ymmärtää suunnittelun, toteutuksen ja testauksen vastuut ja luonteen ketterässä ohjelmistotuotannossa
- ▶ ymmärtää laadunhallinnan perusteet
- ▶ osaa toimia ympäristössä, jossa ohjelmistokehitys tapahtuu hallitusti ja toistettavalla tavalla

Esitietovaatimukset

- ▶ Tietokannat ja Internet tai vastaavat tiedot

Esitietovaatimukset

- ▶ Tietokannat ja Internet tai vastaavat tiedot
- ▶ Vanhalta nimeltään *Aineopintojen harjoitustyö: tietokantasovellus*

Esitietovaatimukset

- ▶ Tietokannat ja Internet tai vastaavat tiedot
- ▶ Vanhalta nimeltään *Aineopintojen harjoitustyö: tietokantasovellus*
- ▶ ja näiden esitiedot: Ohpe, Ohja, Tikape
- ▶ hyötyä jos Ohte ja Lapio

Kurssin rakenne

► Luennot

- ma ja ti 12-14, paitsi
 - viikko 1: ei luentoaa
 - viikko 2: ti 12, to 12, pe 12
- Ohjelmistotuotantoon liittyvää käsitteistöä ja teoriaa
- Vierailuluennot (2 viimeistä viikkoa)

Kurssin rakenne

- ▶ Luennot
 - ▶ ma ja ti 12-14, paitsi
 - ▶ viikko 1: ei luentoaa
 - ▶ viikko 2: ti 12, to 12, pe 12
 - ▶ Ohjelmistotuotantoon liittyvää käsitteistöä ja teoriaa
 - ▶ Vierailuluennot (2 viimeistä viikkoa)
- ▶ Laskarit
 - ▶ teoriaa kertaavat *monivalintatehtävät*, deadline su 23.59
 - ▶ viikon 1 deadline poikkeaa
 - ▶ versionhallintaa, testaamista ja ohjelmistojen konfigurointia käsittelevät, deadline *maanantaina klo 23:59*

Kurssin rakenne

- ▶ Luennot
 - ▶ ma ja ti 12-14, paitsi
 - ▶ viikko 1: ei luentoaa
 - ▶ viikko 2: ti 12, to 12, pe 12
 - ▶ Ohjelmistotuotantoon liittyvää käsitteistöä ja teoriaa
 - ▶ Vierailuluennot (2 viimeistä viikkoa)
- ▶ Laskarit
 - ▶ teoriaa kertaavat *monivalintatehtävät*, deadline su 23.59
 - ▶ viikon 1 deadline poikkeaa
 - ▶ versionhallintaa, testaamista ja ohjelmistojen konfigurointia käsitlevät, deadline *maanantaina klo 23:59*
 - ▶ oletettu kuormittavuus on noin 8 tuntia ensimmäisen kahden viikon aikana, viikolla 3 noin 6 tuntia ja 4 tuntia sen jälkeen
 - ▶ monivalintoihin vastaaminen nopeaa, jos...

Kurssin rakenne

- ▶ Luennot
 - ▶ ma ja ti 12-14, paitsi
 - ▶ viikko 1: ei luentoaa
 - ▶ viikko 2: ti 12, to 12, pe 12
 - ▶ Ohjelmistotuotantoon liittyvää käsitteistöä ja teoriaa
 - ▶ Vierailuluennot (2 viimeistä viikkoa)
- ▶ Laskarit
 - ▶ teoriaa kertaavat *monivalintatehtävät*, deadline su 23.59
 - ▶ viikon 1 deadline poikkeaa
 - ▶ versionhallintaa, testaamista ja ohjelmistojen konfigurointia käsitlevät, deadline *maanantaina klo 23:59*
 - ▶ oletettu kuormittavuus on noin 8 tuntia ensimmäisen kahden viikon aikana, viikolla 3 noin 6 tuntia ja 4 tuntia sen jälkeen
 - ▶ monivalintoihin vastaaminen nopeaa, jos...
- ▶ Miniprojekti
 - ▶ alkaa kurssin 3. viikolla
 - ▶ yhdistää teorian ja käytännön

Miniprojekti

- ▶ kurssin viikoilla 3-7
- ▶ ryhmätyö: koodataan hiukan, harjotellaan projektinhallintaa sekä eräitä laadunhallintatekniikoita

Miniprojekti

- ▶ kurssin viikoilla 3-7
- ▶ ryhmätyö: koodataan hiukan, harjotellaan projektinhallintaa sekä eräitä laadunhallintatekniikoita
- ▶ ryhmässä 4-6 opiskelijaa, ryhmillä on myös asiakas, jota tavataan viikoittain
- ▶ ensimmäisellä viikolla asiakastapaamiseen tulee varata 90 minuuttia, jälkimmäisillä 30 minuuttia
- ▶ kurssin lopussa on miniprojektien yhteinen 2h kestoinen demotilaisuus

Miniprojekti

- ▶ kurssin viikoilla 3-7
- ▶ ryhmätyö: koodataan hiukan, harjotellaan projektinhallintaa sekä eräitä laadunhallintatekniikoita
- ▶ ryhmässä 4-6 opiskelijaa, ryhmillä on myös asiakas, jota tavataan viikoittain
- ▶ ensimmäisellä viikolla asiakastapaamiseen tulee varata 90 minuuttia, jälkimmäisillä 30 minuuttia
- ▶ kurssin lopussa on miniprojektien yhteinen 2h kestoinen demotilaisuus
- ▶ miniprojekteissa työskentelyyn tulee varata aikaa noin 6 tuntia viikossa

Kurssin läpäisyyn edellytyksenä on hyväksytysti suoritettu tai hyväksiluettu miniprojekti

Miniprojektiin hyväksilukeminen

Vähintään kolmen kuukauden työkokemus tiimityönä tehtävästä ohjelmistokehityksestä

Lähetä emailia sen jälkeen kun olet palauttanut viikon 1 tehtävät

Kurssin arvostelu

Jaossa yhteensä 40 pistettä

- ▶ laskarit 10 pistettä
 - ▶ monivalintatehtävät 2 pistettä
 - ▶ viikoittaiset ohjelointi/versionhallinta/konfigurointitehtävät 8 pistettä
- ▶ miniprojekti 11 pistettä
- ▶ koe 18 pistettä
- ▶ osallistuminen vierailuluuennoille 1 piste (0.33 pistettä per luentokerta)

Arvosanaan 1 riittää 20 pistettä, arvosanaan 5 tarvitaan 36 pistettä.

Läipääsy edellyttää lisäksi miniprojektin hyväksyttyä suoritusta (tai hyväksilukua) ja vähintään puolia kokeen pisteistä

Luennot - laskarit - miniprojekti

Luennoilla ohjelmistokehityksen teoriaa ja käsitteistöä

- ▶ laskarien *monivalintatehtävät* liittyvät kunkin viikon luentoihin

Luennot - laskarit - miniprojekti

Luennoilla ohjelmistokehityksen teoriaa ja käsitteistöä

- ▶ laskarien *monivalintatehtävät* liittyvät kunkin viikon luentoihin
Versionhallintaa, konfigurointia, testausta ja ohjelmointia
käsittelevien **teknisempien laskarien** aihepiirejä ei paljoa käsitellä
luennoilla

Luennot - laskarit - miniprojekti

Luennoilla ohjelmistokehityksen teoriaa ja käsitteistöä

► laskarien *monivalintatehtävät* liittyvät kunkin viikon luentoihin

Versionhallintaa, konfigurointia, testausta ja ohjelmointia
käsittelevien **teknisempien laskarien** aihepiirejä ei paljoa käsitellä
luennoilla

Miniprojekti yhdistää luentojen teoria ja laskareissa käsitellyt
teknisemmät asiat, ja soveltaa niitä käytännössä

Luennot - laskarit - miniprojekti

Luennoilla ohjelmistokehityksen teoriaa ja käsitteistöä

► laskarien *monivalintatehtävät* liittyvät kunkin viikon luentoihin
Versionhallintaa, konfigurointia, testausta ja ohjelmointia
käsittelevien **teknisempien laskarien** aihepiirejä ei paljoa käsitellä
luennoilla

Miniprojekti yhdistää luentojen teoria ja laskareissa käsitellyt
teknisemmät asiat, ja soveltaa niitä käytännössä

Kokeessa suurin painoarvo teoriassa ja sen soveltamisessa
käytäntöön

- laskareiden teknisimpiä asioita ei kokeessa tulla kysymään
- tarkemmin kokeesta kurssin viimeisellä luennolla

Versionhallinta 1 op

Saat suoritusmerkinnän tekemällä kaikki kurssin
versiohallintatehtävät ja suorittamalla hyväksytysti miniprojektin

Kurssimateriaali

<https://ohjelmistotuotanto-hy.github.io/>

Kurssipalaute

Kurssipalaute

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

Kurssipalaute

- ▶ Kurssipalaute
 - ▶ Kurssilla lopussa kerättävän palautteen lisäksi ns. jatkuva palaute <https://norppa.helsinki.fi>
- ▶ Kurssipalaute **vaikuttaa**
- ▶ Edellisen kurssin palautteen “moitteita”
 - ▶ Miniprojekti
 - ▶ tekniset haasteet: boilerplate
 - ▶ ryhmätyöskentely: ohjeita
 - ▶ Kokeen painoarvo ja ylipäättäään merkitys
 - ▶ Materiaalin navigoitavuus

Kurssipalaute

- ▶ Kurssipalaute
 - ▶ Kurssilla lopussa kerättävän palautteen lisäksi ns. jatkuva palaute <https://norppa.helsinki.fi>
- ▶ Kurssipalaute **vaikuttaa**
- ▶ Edellisen kurssin palautteen “moitteita”
 - ▶ Miniprojekti
 - ▶ tekniset haasteet: boilerplate
 - ▶ ryhmätyöskentely: ohjeita
 - ▶ Kokeen painoarvo ja ylipäättäään merkitys
 - ▶ Materiaalin navigoitavuus
- ▶ Monivalintoihin kohdistunut kritiikki vähentynyt
 - ▶ haastavampia väittämiä paranneltu/poistettu

Monivalintojen korvaaminen luentopäiväkirjalla

Monivalintojen korvaaminen luentopäiväkirjalla

- ▶ Kurssin viikoilla 1-5 viikon luentoja kertaavat monivalinnat (2/40 kurssin pisteistä)

Monivalintojen korvaaminen luentopäiväkirjalla

- ▶ Kurssin viikoilla 1-5 viikon luentoja kertaavat monivalinnat (2/40 kurssin pisteistä)
- ▶ Monivalinnat herättäneet tunteita, en ole itsekään niiden fani
- ▶ Hyödyt ovat kiistattomat, kokeet menevät nykyään paremmin kuin monivalintoja edeltävänä aikana

Monivalintojen korvaaminen luentopäiväkirjalla

- ▶ Kurssin viikoilla 1-5 viikon luentoja kertaavat monivalinnat (2/40 kurssin pisteistä)
- ▶ Monivalinnat herättäneet tunteita, en ole itsekään niiden fani
- ▶ Hyödyt ovat kiistattomat, kokeet menevät nykyään paremmin kuin monivalintoja edeltävänä aikana
- ▶ Tarjolla vaihtoehto: **viikoittainen luentopäiväkirja**
- ▶ Noin A4:n kokoinen omin sanoin tehty yhteenvetö viikon luentojenasioista
 - ▶ voi olla tehty käsin, joko tekstiä, lista ranskalaisia viivoja tai esim. mind map
 - ▶ ei kuitenkaan missään tapauksessa plagiaatti tai ChatGPT:llä generoitut (opintovilppi: käsitellään HY:n prosessin mukaan)

Monivalintojen korvaaminen luentopäiväkirjalla

- ▶ Kurssin viikoilla 1-5 viikon luentoja kertaavat monivalinnat (2/40 kurssin pisteistä)
- ▶ Monivalinnat herättäneet tunteita, en ole itsekään niiden fani
- ▶ Hyödyt ovat kiistattomat, kokeet menevät nykyään paremmin kuin monivalintoja edeltävänä aikana
- ▶ Tarjolla vaihtoehto: **viikoittainen luentopäiväkirja**
- ▶ Noin A4:n kokoinen omin sanoin tehty yhteenvetö viikon luentojenasioista
 - ▶ voi olla tehty käsin, joko tekstiä, lista ranskalaisia viivoja tai esim. mind map
 - ▶ ei kuitenkaan missään tapauksessa plagiaatti tai ChatGPT:llä generoitut (opintovilppi: käsitellään HY:n prosessin mukaan)
- ▶ Deadline kunkin viikon sunnuntai klo 23:59, lähetetään emailitse matti.luukkainen@helsinki.fi (poikkeus vko 1 jonka deadline samaan aikaan kuin vko 2)
- ▶ Arvostelu, ks kurssimateriaalin osa 0

Ohjelmistotuotanto engl. software engineering

The IEEE Computer Society defines software engineering as:

- ▶ *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software*

Ohjelmistotuotanto engl. software engineering

The IEEE Computer Society defines software engineering as:

- ▶ *Systemaattisen, kurinalaisen ja mitattavissa olevan lähestymistavan soveltaminen ohjelmistojen kehittämiseen, käyttöön ja ylläpitoon, eli "insinöörityön" soveltaminen ohjelmistoihin*

Ohjelmistotuotanto engl. software engineering

The IEEE Computer Society defines software engineering as:

- ▶ *Systemaattisen, kurinalaisen ja mitattavissa olevan lähestymistavan soveltaminen ohjelmistojen kehittämiseen, käyttöön ja ylläpitoon, eli "insinöörityön" soveltaminen ohjelmistoihin*

Lähde SWEBOK eli *Guide to the Software Engineering Body of Knowledge*

- ▶ ison komitean yritys määritellä mitä ohjelmistotuotannolla tarkoitetaan ja mitä osa-alueita siihen kuuluu
- ▶ uusin versio ilmestyi 15.11.2024!

Ohjelmistotuotannon osa-alueet SWEBOK:in mukaan

- ▶ Software requirements eli **vaatimusmäärittely**
- ▶ Software design eli **suunnittelu**
- ▶ Software construction eli **toteutus/ohjelointi**
- ▶ Software testing eli **testaus**
- ▶ Software maintenance eli **ylläpito**

Ohjelmistotuotannon osa-alueet SWEBOK:in mukaan

- ▶ Software requirements eli **vaatimusmäärittely**
- ▶ Software design eli **suunnittelu**
- ▶ Software construction eli **toteutus/ohjelointi**
- ▶ Software testing eli **testaus**
- ▶ Software maintenance eli **ylläpito**

- ▶ Software configuration management eli **konfiguraatiot**
- ▶ Software engineering tools and methods eli **työkalut**
- ▶ Software quality eli **laadunhallinta**
- ▶ Software engineering process eli **ohjelmistotuotantoprosessi**
- ▶ Software engineering management eli **projektien johtaminen**

Ohjelmistotuotannon osa-alueet SWEBOK:in mukaan

- ▶ Software requirements eli **vaatimusmäärittely**
- ▶ Software design eli **suunnittelu**
- ▶ Software construction eli **toteutus/ohjelointi**
- ▶ Software testing eli **testaus**
- ▶ Software maintenance eli **ylläpito**

- ▶ Software configuration management eli **konfiguraatiot**
- ▶ Software engineering tools and methods eli **työkalut**
- ▶ Software quality eli **laadunhallinta**
- ▶ Software engineering process eli **ohjelmistotuotantoprosessi**
- ▶ Software engineering management eli **projektien johtaminen**

- ▶ Software architecture eli **arkkitehtuuri**
- ▶ Software engineering operations eli **operointi tuotannossa**
- ▶ Software security eli **tietoturva**
- ▶ Software engineering economics eli **talous**
- ▶ Software engineering professional practice eli **ammattietiikka ja ryhmäryynamiiikka**

Ohjelmiston elinkaari (software lifecycle)

Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

Ohjelmiston elinkaari (software lifecycle)

Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

Vaiheista muodostuu ohjelmiston “elinkaari”

Ohjelmiston elinkaari (software lifecycle)

Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

Vaiheista muodostuu ohjelmiston “elinkaari”

Käytetty **ohjelmistotuotantoprosessi** määrittelee miten vaiheet suhtautuvat toisiinsa

Ohjelmiston elinkaari (software lifecycle)

Riippumatta tyylistä ja tavasta, jolla ohjelmisto tehdään, käy ohjelmisto läpi seuraavat *vaiheet*

- ▶ Vaatimusten analysointi ja määrittely
- ▶ Suunnittelu
- ▶ Toteutus
- ▶ Testaus
- ▶ Ohjelmiston ylläpito ja evoluutio

Vaiheista muodostuu ohjelmiston “elinkaari”

Käytetty **ohjelmistotuotantoprosessi** määrittelee miten vaiheet suhtautuvat toisiinsa

Eri vaiheiden sisältöön palaamme myöhemmin tarkemmin

Alussa (ja osin edelleen) code'n'fix

Historian alkuaikoina laitteet maksoivat paljon, ohjelmat olivat laitteistoihin nähdyn "triviaaleja"

- ▶ ohjelmointi konekielellä
- ▶ sovelluksen käyttäjä ohjelmoi itse ohjelmansa

Alussa (ja osin edelleen) code'n'fix

Historian alkuaikoina laitteet maksoivat paljon, ohjelmat olivat laitteistoihin nähdyn "triviaaleja"

- ▶ ohjelointi konekielessä
- ▶ sovelluksen käyttäjä ohjelmoi itse ohjelmansa

Vähitellen ohjelmistot alkavat kasvaa ja kehitettiin korkeamman tason ohjelointikieliä (Fortran, Cobol, Algol)

- ▶ sovellusalue laajenee monille elämänaloille

Alussa (ja osin edelleen) code'n'fix

Historian alkuaikoina laitteet maksoivat paljon, ohjelmat olivat laitteistoihin nähdyn "triviaaleja"

- ▶ ohjelointi konekielessä
- ▶ sovelluksen käyttäjä ohjelmoi itse ohjelmansa

Vähitellen ohjelmistot alkavat kasvaa ja kehitettiin korkeamman tason ohjelointikieliä (Fortran, Cobol, Algol)

- ▶ sovellusalue laajenee monille elämänaloille

Pikkuhiljaa homma alkaa karata käsistä:

- ▶ budjetit ylittyivät ja projektit myöhästyivät aikatauluista
- ▶ ohjelmistot olivat tehottomia, niiden laatu oli huono ja ne eivät toimineet käyttäjien tarpeiden mukaan
- ▶ koodin ylläpito ja laajentaminen oli vaikeaa
- ▶ usein ohjelmistoja ei hyvästä aikeista huolimatta saatu ollenkaan toimitettua

Ohjelmistokriisi

Termin Software crisis lanseerataan kesällä 1968

- ▶ In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs.

Ohjelmistokriisi

Termin Software crisis lantseerataan kesällä 1968

- ▶ In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs.

Edsger Dijkstra:

- ▶ **as long as there were no machines, programming was no problem at all**

Ohjelmistokriisi

Termi Software crisis lanseerataan kesällä 1968

- ▶ In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs.

Edsger Dijkstra:

- ▶ **as long as there were no machines, programming was no problem at all**
- ▶ when we had a few weak computers, programming became a mild problem

Ohjelmistokriisi

Termin Software crisis lantseerataan kesällä 1968

- ▶ In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs.

Edsger Dijkstra:

- ▶ **as long as there were no machines, programming was no problem at all**
- ▶ when we had a few weak computers, programming became a mild problem
- ▶ **now we have gigantic computers, programming has become an equally gigantic problem.**

Kriisi ei ole ohi (syksy 2023)

Kaupunki | Koulut

Helsingiltä paljastui taas massiivinen epäonnistuminen: Uusi järjestelmä on susi

Koulujen ja päiväkotien ASTI-järjestelmällä piti korvata lopulta esimerkiksi Wilma, mutta 32 miljoonaa syönyt tietojärjestelmä jää keskentekoiseksi.



Helsingin kaupungintalo. KUVA: SAARA MANSIKKAMÄKI / HS

Software development as Engineering

Software development as Engineering

Termin **software engineering** määritellään ensimmäistä kertaa 1968:

- ▶ *The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines*

Software development as Engineering

Termin **software engineering** määritellään ensimmäistä kertaa 1968:

- ▶ *The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines*

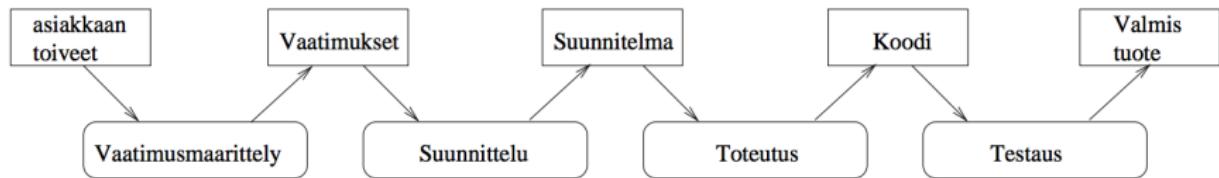
Ajatus siitä, että ohjelmistojen tekemisen tulisi olla kuin mikä tahansa muu insinöörityö

- ▶ ensin rakennettava artefakti *määritellään* (requirements)
- ▶ ja *suunnitellaan* (design) aukottomasti
- ▶ tämän jälkeen *rakentaminen* (construction) on melko suoraviivainen vaihe

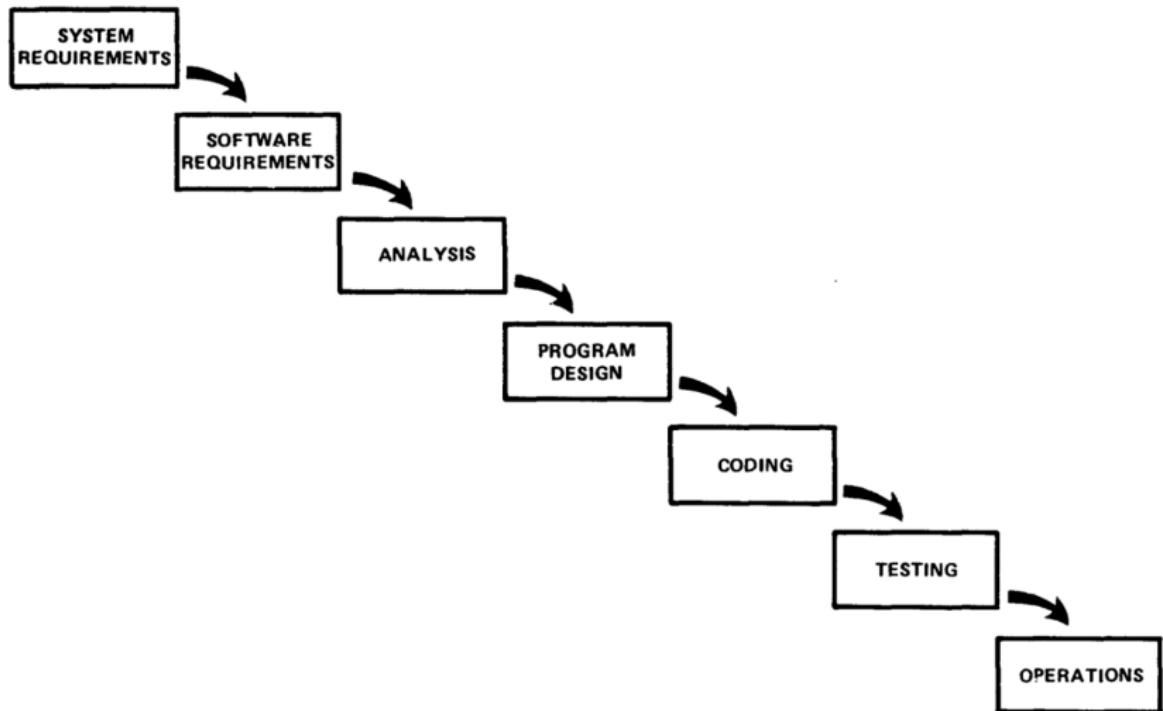
Vesiputousmalli

Winston W. Royce: Management of the development of Large Software, 1970

Sivulla 2 Royce esittelee yksinkertaisen prosessimallin, jossa elinkaaren vaiheet suoritetaan lineaarisesti peräkkäin:



Vesiputousmalli Roycen artikkelista



Vesiputousmallin suosion taustaa

Vesiputousmalli saavutti nopeasti suosiota

Yhdysvaltain puolustusministerö rupesi vaatimaan kaikilta alihankkijoiltaan vesiputousmallin noudattamista

- ▶ Standardi DoD STD 2167

Vesiputousmallin suosion taustaa

Vesiputousmalli saavutti nopeasti suosiota

Yhdysvaltain puolustusministerö rupesi vaatimaan kaikilta alihankkijoiltaan vesiputousmallin noudattamista

- ▶ Standardi DoD STD 2167

Muutkin ohjelmistoja tuottaneet tahot ajattelivat, että koska DoD vaatii vesiputousmallia, tapa kannattaa omaksua itselleen

Vesiputousmallin oletuksia

Vesiputousmalli perustuu vahvasti siihen, että *erit vaiheet ovat erillisten tuotantotuumien tekemiä*

- ▶ Tämän takia kunkin vaiheen tulokset *dokumentoidaan tarkoin*

Vesiputousmallin oletuksia

Vesiputousmalli perustuu vahvasti siihen, että *eri vaiheet ovat erillisten tuotantotuumien tekemiä*

- ▶ Tämän takia kunkin vaiheen tulokset *dokumentoidaan tarkoin*

Vaiheet tehdään peräkkäin

- ▶ esim. tekninen suunnittelu aloitetaan kun vaatimusmäärittely on valmis

Vesiputousmallin oletuksia

Vesiputousmalli perustuu vahvasti siihen, että *eri vaiheet ovat erillisten tuotantotuumien tekemiä*

- ▶ Tämän takia kunkin vaiheen tulokset *dokumentoidaan tarkoin*

Vaiheet tehdään peräkkäin

- ▶ esim. tekninen suunnittelu aloitetaan kun vaatimusmäärittely on valmis

Vesiputousmallin mukainen ohjelmistoprosessi on yleensä etukäteen *tarkkaan suunniteltu, resursoitu ja aikataulutettu*

Vesiputousmallin oletuksia

Vesiputousmalli perustuu vahvasti siihen, että *eri vaiheet ovat erillisten tuotantotuumien tekemiä*

- ▶ Tämän takia kunkin vaiheen tulokset *dokumentoidaan tarkoin*
Vaiheet tehdään peräkkäin
- ▶ esim. tekninen suunnittelu aloitetaan kun vaatimusmäärittely on valmis

Vesiputousmallin mukainen ohjelmistoprosessi on yleensä etukäteen tarkkaan *suunniteltu, resursoitu ja aikataulutettu*

Vesiputousmallin mukainen ohjelmistotuotanto ei ole osoittautunut erityisen onnistuneeksi

Vesiputousmallin ongelmia

Asiakkaan *vaatimukset muuttuvat* usein matkan varrella:

- ▶ Asiakas ei tiedä tai osaa sanoa mitä haluaa/tarvitsee
- ▶ Asiakkaan tarve muuttuu projektin kuluessa
- ▶ Asiakas alkaa haluta muutoksia kun näkee lopputuotteen

Vesiputousmallin ongelmia

Asiakkaan vaatimukset muuttuvat usein matkan varrella:

- ▶ Asiakas ei tiedä tai osaa sanoa mitä haluaa/tarvitsee
- ▶ Asiakkaan tarve muuttuu projektin kuluessa
- ▶ Asiakas alkaa haluta muutoksia kun näkee lopputuotteen

Vaatimusmäärittelyn, suunnittelun ja toteutuksen erottaminen ei järkevää

- ▶ Ohjelmaa on mahdotonta suunnitella siten, että toteutus on suoraviivaista
- ▶ Toteutusteknologiat vaikuttavat suuresti määriteltyjen ominaisuuksien hintaan
- ▶ Osa suunnittelusta tapahtuu pakosti vasta ohjelmoitaessa

Vesiputousmallin ongelmia

Asiakkaan vaatimukset muuttuvat usein matkan varrella:

- ▶ Asiakas ei tiedä tai osaa sanoa mitä haluaa/tarvitsee
- ▶ Asiakkaan tarve muuttuu projektin kuluessa
- ▶ Asiakas alkaa haluta muutoksia kun näkee lopputuotteen

Vaatimusmäärittelyn, suunnittelun ja toteutuksen erottaminen ei järkevää

- ▶ Ohjelmaa on mahdotonta suunnitella siten, että toteutus on suoraviivaista
- ▶ Toteutusteknologiat vaikuttavat suuresti määriteltyjen ominaisuuksien hintaan
- ▶ Osa suunnittelusta tapahtuu pakosti vasta ohjelmoitaessa

Lopuksi tapahtuva laadunhallinta paljastaa ongelmat liian myöhään

- ▶ Korjaukset mahdollisesti kalliita: voi paljastua ongelmia jotka pakottavat muuttamaan ohjelmiston vaatimuksia

Vesiputous oli väärinymmärrys

Paradoksaalista kyllä vesiputousmallin isänä pidetty Royce **ei suosittele** artikkelissaan suoraviivaisen lineaarisen mallin käyttöä

Vesiputous oli väärinymärrys

Paradoksaalista kyllä vesiputousmallin isänä pidetty Royce **ei suosittele** artikkelissaan suoraviivaisen lineaarisen mallin käyttöä Royce esittelee lineaarisen vesiputousmallin **sivulla 2**, mutta toteaa että se **ei sovellu** monimutkaisiin ohjelmistoprojekteihin

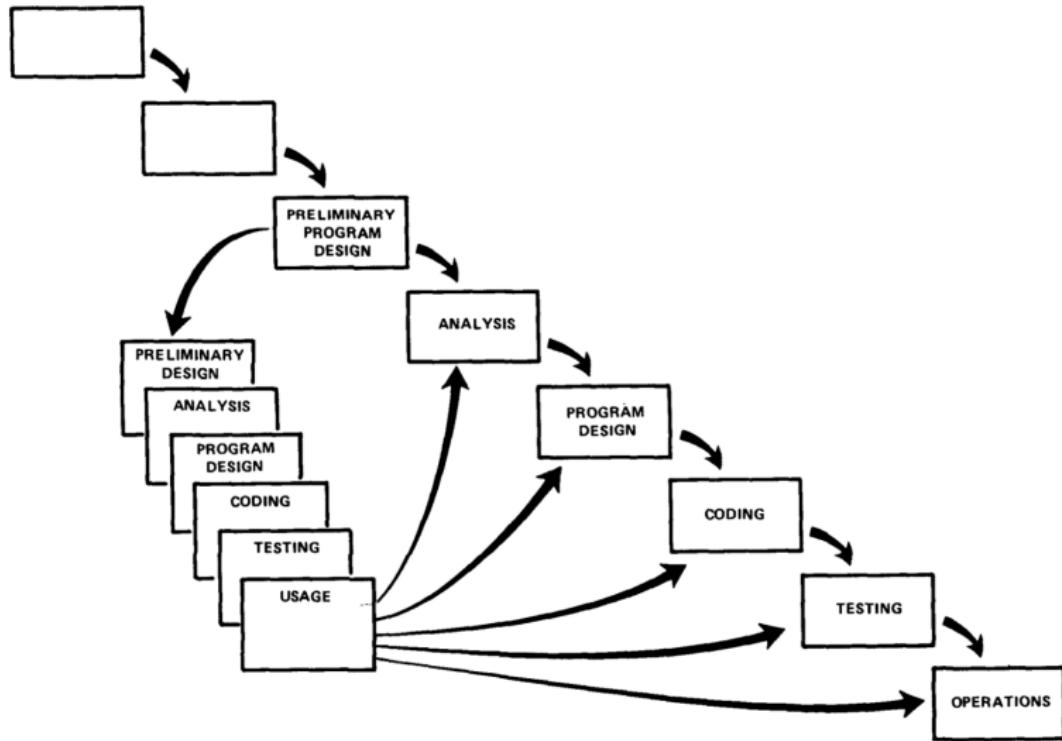
Vesiputous oli väärinymmärrys

Paradoksaalista kyllä vesiputousmallin isänä pidetty Royce **ei suosittele** artikkelissaan suoraviivaisen lineaarisen mallin käyttöä Royce esittelee lineaarisen vesiputousmallin **sivulla 2**, mutta toteaa että se **ei sovellu** monimutkaisiin ohjelmistoprojekteihin

Roycen mukaan

- ▶ ensin tulee tehdä prototyppi
- ▶ ja siitä saatujen kokemusten valossa suunnitellaan ja toteutetaan lopullinen ohjelmisto

Roycen kahden iteraation malli



Iteratiivinen ohjelmistokehitys

Vesiputousmallin ongelmiin reagoineen *iteratiivinen* tapa tehdä ohjelmistoja alkoi yleistyä 90-luvulla

- ▶ mm. spiraalimalli, prototyppimalli, Rational Unified Process

Iteratiivinen ohjelmistokehitys

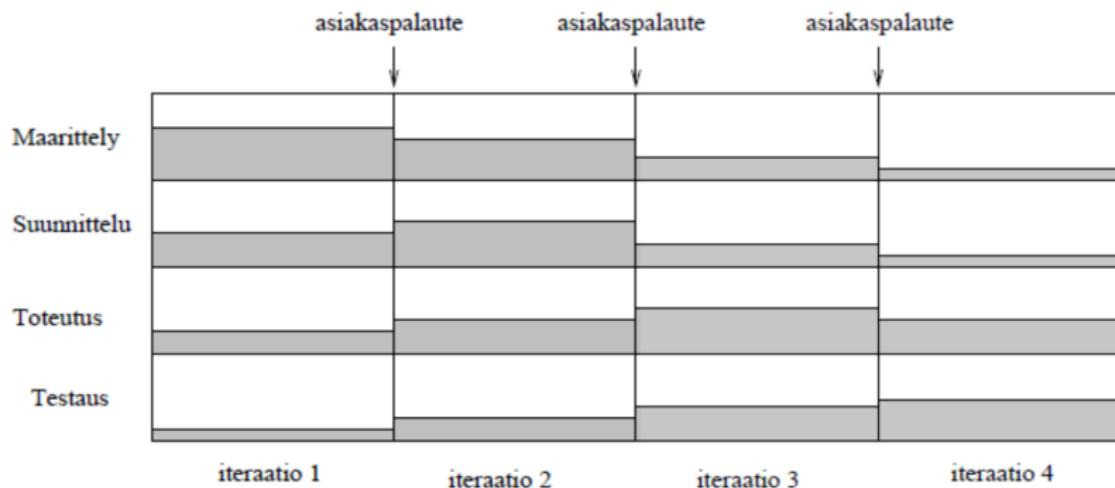
Vesiputousmallin ongelmiin reagoineen *iteratiivinen* tapa tehdä ohjelmistoja alkoi yleistyä 90-luvulla

- ▶ mm. spiraalimalli, prototyppimalli, Rational Unified Process

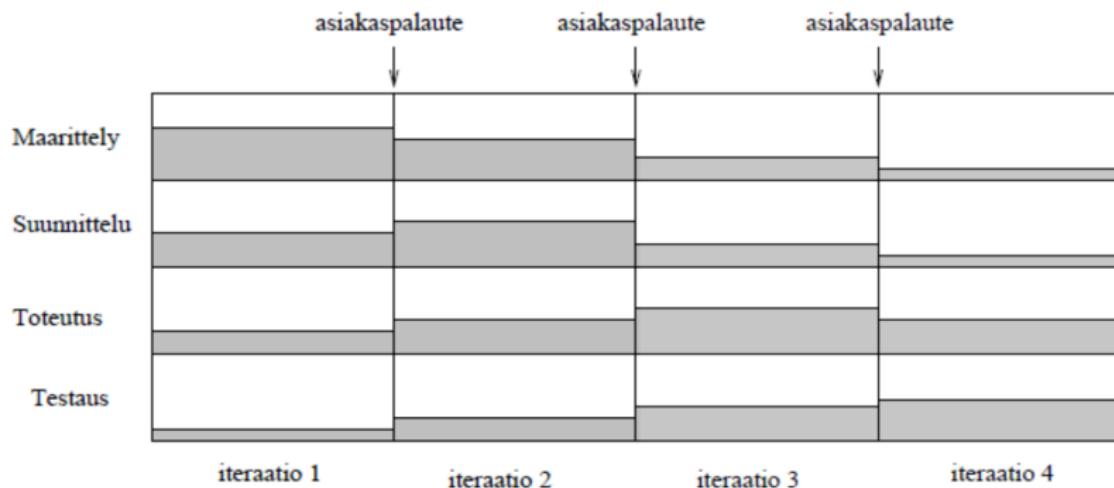
Iteratiivisessa ohjelmistokehityksessä

- ▶ ohjelmistotuotanto jaetaan jaksoihin, eli *iteraatioihin*
- ▶ jokaisen iteraation aikana määritellään, suunnitellaan toteutetaan ja testataan ohjelmistoa
- ▶ eli ohjelmisto kehittyy vähitellen (inkrementaaliseksi)

Iteratiivinen ohjelmistokehitys



Iteratiivinen ohjelmistokehitys



- ▶ asiakasta tavataan jokaisen iteraation välissä
- ▶ asiakas näkee sen hetkisen version ohjelmasta ja pystyy vaikuttamaan seuraavien iteraatoiden kulkuun

Iteratiivinen ohjelmistokehitys ei ole uusi keksintö

Yhdysvaltojen puolustusministeriön 2000 julkaisema standardi (MIL-STD-498) alkaa suositella iteratiivista ohjelmistoprosessia:

- ▶ There are two approaches, evolutionary (*iterative*) and single step (*waterfall*), to full capability. An *evolutionary approach is preferred...*

Iteratiivinen ohjelmistokehitys ei ole uusi keksintö

Yhdysvaltojen puolustusministeriön 2000 julkaisema standardi (MIL-STD-498) alkaa suositella iteratiivista ohjelmistoprosessia:

- ▶ There are two approaches, evolutionary (*iterative*) and single step (*waterfall*), to full capability. An *evolutionary approach is preferred...*

Iteratiivinen ohjelmistokehitys on paljon vanhempi idea kun vesiputosmalli

- ▶ esim. NASA:n ensimmäisen amerikkalaisen avaruuteen vieneen Project Mercuryn ohjelmisto kehitettiin iteratiivisesti
- ▶ avaruussukkuloiden ohjelmisto tehtiin vesipitousmallin valtakaudella, mutta iteratiivisesti
 - ▶ 8 viikon iteraatioissa, 31 kuukaudessa

Iteratiivinen ohjelmistokehitys ei ole uusi keksintö

Yhdysvaltojen puolustusministeriön 2000 julkaisema standardi (MIL-STD-498) alkaa suositella iteratiivista ohjelmistoprosessia:

- ▶ There are two approaches, evolutionary (*iterative*) and single step (*waterfall*), to full capability. An *evolutionary approach is preferred...*

Iteratiivinen ohjelmistokehitys on paljon vanhempi idea kun vesiputosmalli

- ▶ esim. NASA:n ensimmäisen amerikkalaisen avaruuteen vieneen Project Mercuryn ohjelmisto kehitettiin iteratiivisesti
- ▶ avaruussukkuloiden ohjelmisto tehtiin vesiputosmallin valtakaudella, mutta iteratiivisesti
 - ▶ 8 viikon iteraatioissa, 31 kuukaudessa

Roycen artikkelikin (vuonna 1970) ehdotti *kahden iteraation menetelmää* ohjelmistojen tekemiseen

Ketterien menetelmien synty...

Perinteisissä prosessimalleissa korostettiin

- ▶ huolellista projektisuunnittelua
- ▶ formaalia laadunvalvontaa
- ▶ yksityiskohtaisia analyysi- ja suunnittelumenetelmiä
- ▶ täsmällistä, tarkasti ohjattua ohjelmistoprosessia

Ketterien menetelmien synty...

Perinteisissä prosessimalleissa korostettiin

- ▶ huolellista projektisuunnittelua
- ▶ formaalia laadunvalvontaa
- ▶ yksityiskohtaisia analyysi- ja suunnittelumenetelmiä
- ▶ täsmällistä, tarkasti ohjattua ohjelmistoprosessia

Tukivat erityisesti laajojen, pitkäikäisten ohjelmistojen kehitystyötä

- ▶ pienien ja keskisuurten ohjelmistojen tekoon turhan jäykkiä

Ketterien menetelmien synty...

Perinteisissä prosessimalleissa korostettiin

- ▶ huolellista projektisuunnittelua
- ▶ formaalia laadunvalvontaa
- ▶ yksityiskohtaisia analyysi- ja suunnittelumenetelmiä
- ▶ täsmällistä, tarkasti ohjattua ohjelmistoprosessia

Tukivat erityisesti laajojen, pitkäikäisten ohjelmistojen kehitystyötä

- ▶ pienien ja keskisuurten ohjelmistojen tekoon turhan jäykkiä

Pyrittiin työtä tekevän yksilön merkityksen minimoimiseen

- ▶ yksilö on “tehdastyöläinen”, joka voidaan helposti korvata toisella ja tällä ei ole ohjelmiston kehittämiseen vaikutusta

Ketterien menetelmien synty

Ristiriita/turhauma synnytti joukon **ketteriä prosessimalleja** (agile)

- ▶ korostivat itse ohjelmistoa sekä ohjelmiston asiakkaan ja toteuttajien merkitystä yksityiskohtaisen suunnittelun ja dokumentaation sijaan

Ketterien menetelmien synty

Ristiriita/turhauma synnytti joukon **ketteriä prosessimalleja** (agile)

- ▶ korostivat itse ohjelmistoa sekä ohjelmiston asiakkaan ja toteuttajien merkitystä yksityiskohtaisen suunnittelun ja dokumentaation sijaan
- ▶ Useita samanhenkisiä menetelmiä 90-luvun loppupuolelta lähtien

Ketterä manifesti 2001

Ketterä manifesti 2001

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Ketterä manifesti 2001

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- ▶ **Individuals and interactions** over processes and tools
- ▶ **Working software** over comprehensive documentation
- ▶ **Customer collaboration** over contract negotiation
- ▶ **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more

Ketterä manifesti 2001

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

- ▶ **Individuals and interactions** over processes and tools
- ▶ **Working software** over comprehensive documentation
- ▶ **Customer collaboration** over contract negotiation
- ▶ **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more

Manifestin laativat ja allekirjoittivat 17 ketterien menetelmien varhaista pioneeria, mm. Kent Beck, Robert Martin, Ken Schwaber ja Martin Fowler

Ketterät periaatteet, osa 1

Ketterät periaatteet, osa 1

Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**

Ketterät periaatteet, osa 1

Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Ketterät periaatteet, osa 1

Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Business people and developers must **work together daily** throughout the project

Ketterät periaatteet, osa 1

Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Business people and developers must **work together daily** throughout the project

The most efficient and effective method of conveying information to and within a development team is **face-to-face** conversation

Ketterät periaatteet, osa 1

Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Business people and developers must **work together daily** throughout the project

The most efficient and effective method of conveying information to and within a development team is **face-to-face** conversation

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

The best architectures, requirements, and designs emerge from **self-organizing teams**

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

The best architectures, requirements, and designs emerge from **self-organizing teams**

At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

The best architectures, requirements, and designs emerge from **self-organizing teams**

At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly

Simplicity – the art of maximizing the amount of work not done – is essential

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

The best architectures, requirements, and designs emerge from **self-organizing teams**

At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly

Simplicity – the art of maximizing the amount of work not done – is essential

Continuous attention to **technical excellence and good design** enhances agility

Ketterät periaatteet, osa 2

Build projects around motivated individuals. Give them the environment and support they need, and **trust them to get the job done**

The best architectures, requirements, and designs emerge from **self-organizing teams**

At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly

Simplicity – the art of maximizing the amount of work not done – is essential

Continuous attention to **technical excellence and good design** enhances agility

Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely

Ketterät menetelmät

Sateenvarjotermi useille ketterille prosessimalleille

- ▶ Näistä tunnetuimpia Scrum ja eXtreme programming eli XP
- ▶ Molempiin, erityisesti Scrumiin tutustutaan kurssin aikana

Ketterät menetelmät

Sateenvarjotermi useille ketterille prosessimalleille

- ▶ Näistä tunnetuimpia Scrum ja eXtreme programming eli XP
- ▶ Molempien, erityisesti Scrumiin tutustutaan kurssin aikana

Ketterä ohjelmistotuotanto on ottanut vaikutteita myös Toyota production systemin taustalla olevasta *lean-ajattelusta*

Viime vuosina syntynyt joukko “lean-menetelmiä”

- ▶ Kanban
- ▶ Scrumban

Ketterät menetelmät

Sateenvarjotermi useille ketterille prosessimalleille

- ▶ Näistä tunnetuimpia Scrum ja eXtreme programming eli XP
- ▶ Molempien, erityisesti Scrumiin tutustutaan kurssin aikana

Ketterä ohjelmistotuotanto on ottanut vaikutteita myös Toyota production systemin taustalla olevasta *lean-ajattelusta*

Viime vuosina syntynyt joukko “lean-menetelmiä”

- ▶ Kanban
- ▶ Scrumban

... sekä laajan mittakaavan leaniin ja ketterään kehitykseen tarkoitettuja menetelmiä kuten SaFe

Myös nämä kuuluvat kurssin aihepiiriin

Viikon 1 laskarit

Viikon 1 laskarit

Ohjelmistokehityksen käytännön työkaluja

- ▶ Versionhallinta: Git
- ▶ Automatisoitu testaus: Unittest
- ▶ Projektin riippuvuuksienhallinta ja “buildaus”: Poetry
- ▶ CI- ja build-palvelinohjelmisto: Github Action

Tavoitteena mahdolistaa hallittu- ja toistettavissa oleva ohjelmistokehitys

Viikon 1 laskarit

Ohjelmistokehityksen käytännön työkaluja

- ▶ Versionhallinta: Git
- ▶ Automatisoitu testaus: Unittest
- ▶ Projektin riippuvuuksienhallinta ja “buildaus”: Poetry
- ▶ CI- ja build-palvelinohjelmisto: Github Action

Tavoitteena mahdolistaa hallittu- ja toistettavissa oleva ohjelmistokehitys

Deadline maanantaina klo 23:59

AI: viekö se koodarin työt?

Koodareille pysäyttävä viesti: ohjelmointi loppuu – edessä täysin uusi rooli

Aleksi Kolehmainen 22.8.2024 12:44 | päivitetty 27.8.2024 14:56 OHJELMISTOKEHITYS DIGITALOUS TEKOÄLY

AWS:n Matt Garmanin mukaan tulevaisuudessa kehittäjät keskittyvät yhä enemmän käyttäjien tarpeiden ymmärtämiseen sekä luovuuteen ja innovaatioihin.



Mikä on juniorikoodarin tulevaisuus

Uutinen

Koodareiden työpaikat vaarassa – "juniorit lähtevät ensimmäisinä"

Anna Helakallio 23.9.2024 14:36 [TEKOÄLY](#) [OHJELMISTOKEHITYS](#)

It-johtajat ennustavat tekoälyn mullistavan koodarin työn, mutta osa johtajista on skeptisiä.



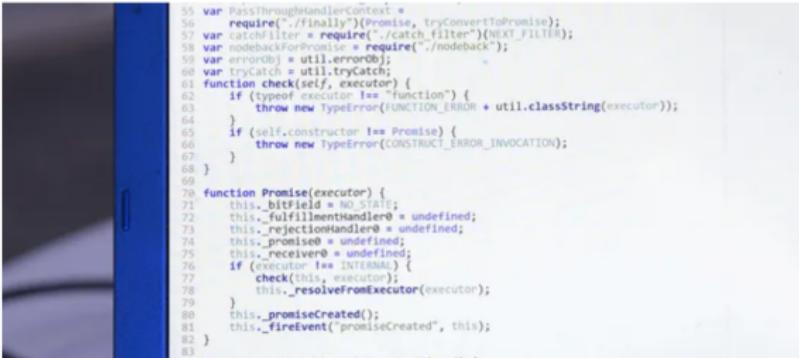
AI: voi olla itseasiassa aika huono

Uutinen

Koodareiden hehkutetut työkalut tyrmättiin: eivät paranna tuottavuutta – virheet lisääntyvät 41 %

Aleksi Kolehmainen 2.10.2024 22:45 | päivitetty 15.10.2024 18:14 TEKOÄLY OHJELMISTOKEHITYS

Tuore selvitys kyseenalaistaa GitHubin Copilotin kaltaisten tekoälyavusteisten koodaustyökalujen vaikutukset kehittäjien tuottavuuteen ja työuupumuksen ehkäisynn.



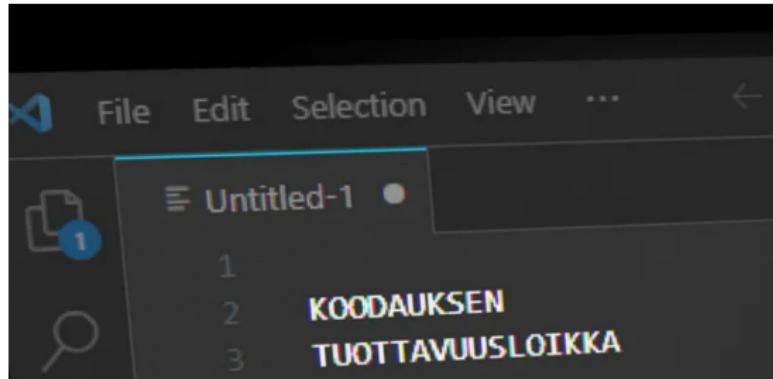
```
55 var PassThroughHandlerContext =  
56   require("./finally")({Promise, tryConvertToPromise});  
57 var catchFilter = require("./catch_filter")({NEXT_FILTER});  
58 var nodebackForPromise = require("./nodeback");  
59 var errorOrB = util.errorOrB;  
60 var tryCatchB = util.tryCatchB;  
61 function checkSelf(executor) {  
62   if (typeof executor != "function") {  
63     throw new TypeError(FUNCTION_ERROR + util.classString(executor));  
64   }  
65   if (self.constructor != Promise) {  
66     throw new TypeError(CONSTRUCT_ERROR_INVOCATION);  
67   }  
68 }  
69  
70 function Promise(executor) {  
71   this._bitField = NO_STATE;  
72   this._fulfillmentHandler0 = undefined;  
73   this._rejectionHandler0 = undefined;  
74   this._promised = undefined;  
75   this._receiver0 = undefined;  
76   if (executor != INTERNAL) {  
77     check(this, executor);  
78   }  
79   this._promiseCreated();  
80   this._fireEvent("promiseCreated", this);  
81 }  
82  
83
```

AI: voi myös olla uhka

Osa koodareista hylkäsi hypetetyn työkalun – ”varmistaa, ettet koskaan oikeasti opi koodaamaan”

Aleksi Kolehmainen 29.8.2024 21:10 | päivitetty 24.9.2024 13:25 TEKOÄLY OHJELMISTOKEHITYS

GitHub Copilot on saavuttanut suosiota ohjelmistokehittäjien keskuudessa, mutta kaikki eivät ole vakuuttuneita sen edusta.



AI: työn painopiste saattaa muuttua

Tällaisille koodareille riittää jatkossakin töitä – ”Tekoäly on vähän kuin innokas juniori”

Aleksi Kolehmainen 8.8.2024 07:11 | päivitetty 13.9.2024 17:03 TEKOÄLY OHJELMISTOKEHITYS

Vaikka tekoäly muokkaa ohjelmistokehityksen kenttää ennennäkemättömillä tavalla, ihmisiä tarvitaan ohjelmistokehittämisessä jatkossakin.



Viestintää. Matti Luukkainen uskoo, että tekoälyn tulo nostaa viestintätaitojen

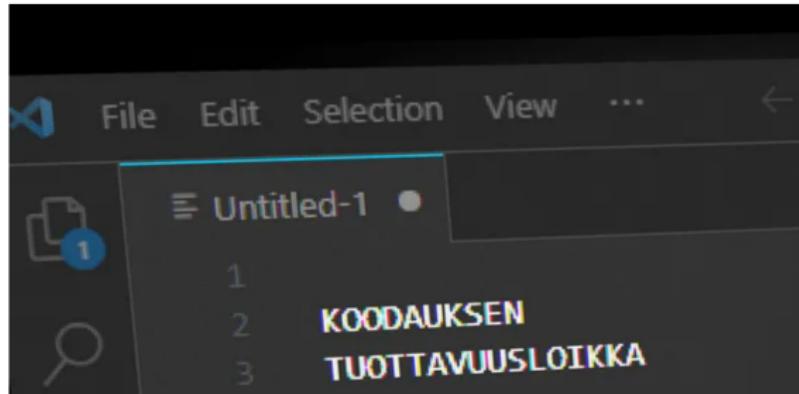
AI: uusia temppuja tarvitaan mutta ehkä duuni säilyy

Uutinen

80 % koodareista vaarassa jäädää jälkeen

Aleksi Kolehmainen 16.10.2024 22:03 | päivitetty 16.10.2024 22:18 TEKOÄLY OHJELMISTOKEHITYS

Tekoäly ja sen mukanaan tuomat muutokset ovat mullistamassa ohjelmistokehityksen tulevina vuosina. Gartnerin mukaan tekoäly ei kuitenkaan syrjäytä ohjelmistokehittäjiä, vaan saattaa itse asiassa kasvattaa heidän kysyntäänsä entisestään.



AI: hyvä renki, huono isäntä/emäntä



“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?”

– Brian Kernighan

AI: hyvä renki, huono isäntä/emäntä



“Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?”

– Brian Kernighan

<https://curre.helsinki.fi/chat/chats>