

## Ohjelmistotuotanto

Ohjelmistoprosessi  
Ohjelmiston elinkaari

1

## Ohjelmistoprosessi

- Ohjelmiston rakentamisen vaiheet ja niiden tulokset
- Ohjelmiston elinkaaren määrittely
- Yleisrakenne sille, miten ohjelmisto tuotetaan
  
- Yleensä prosessilla ymmärretään toistettavaa toimintamallia – voiko uniikkituotannossa olla prosesseja?

© Harri Laine, Jukka Paakki

2

## Ohjelmistoprosessimalli

- Prosessimalli on ohjelmiston elinkaaren rakenteen määrittely ts. kuvaus sille, millaisten vaiheiden kautta ohjelmisto kehittyi ideasta haetaan
  - mahdollisimman yleisesti sovellettavissa oleva ohjeisto ohjelmistojen tuottamiseen
  - ei konkreettisia yksityiskohtaisia toimintasääntöjä vaan pelkkä yleisrakenne
  - malli muokattavissa organisaatiolle ja sovellusalueelle sopivaksi

© Harri Laine, Jukka Paakki

3

## Yleiset ohjelmistotuotannon osatehtävät

- Määrittely
  - Suunnittelu
  - Ohjelmointi
    - Testaus
    - Käyttö ja ylläpito

© Harri Laine, Jukka Paakki

4

## Elinkaaren vaihejako

- On keskeisin apuväline prosessimallissa
- Määrittelee prosessin tärkeimmät osavaiheet
- Määrittelee osavaiheiden suoritusjärjestyksen
- Määrittelee osavaiheiden keskinäiset vaikutussuhteet
- Määrittelee osavaiheiden keskinäisiin rajapintoihin liittyvät toimenpiteet kuten dokumentoinnin, tarkastukset, yms.

© Harri Laine, Jukka Paakki

5

## Vaihejaon ominaispiirteitä

- jokainen vaihe tuottaa jonkin määritellyn tuloksen
- vaiheen tulos toimii syötteenä seuraavalle vaiheelle
- kumuloituvat osatulokset ohjaavat seuraavan vaiheen suoritusta
- vaiheen tulokset ovat jollakin määritellyillä kriteereillä hyväksyttävissä tai hylättävissä
- jokaisen vaiheen alku ja loppu ovat selvästi havaittavissa
- usein vaihe tarkoittaa sitä edeltäneiden vaiheiden tuloksia

© Harri Laine, Jukka Paakki

6

## Vaihejaon ominaispiirteitä

- keskeistä useissa prosessimalleissa on vaiheen toisto ja silmukallinen iterointieli toisto
- käytännössä prosessi on aina iteratiivinen
- prosessiin liittyy implisiittisesti joukko yleisiä, kaikille osavaiheille yhteisiä tukitoimenpiteitä
  - dokumentointi
  - laadunvarmistus
  - tuotteenhallinta
  - projektinjohto, ...



## Prosessityypit

- **Rakentaminen**
  - peräkkäiset työvaiheet, joista jokaisen tuloksena syntyy jokin 'vaihetuote'
  - määrittely - suunnittelu - toteutus
  - edellinen vaihe pohjana seuraavalle
- **Kasvattaminen**
  - pikaisesti toimivaan systeemiin, jota muokataan, kunnes tulos on tyydyttävä
- **Kokoaminen**
  - kootaan uudelleenkäytettävistä osista

## 1. Lineaarinen malli (vesiputous, waterfall)

- Ohjelmiston tuottaminen on sarjallinen, järjestyksessä etenevä prosessi, jossa on seuraavat vaiheet (huom. vaiheet vaihtelevat eri lähteissä)
- **järjestelmäsuunnittelu** (system engineering)
  - etsitään halutusta kokonaisjärjestelmästä ohjelmistokomponentit ja hahmotetaan niiden rooli järjestelmässä
  - selvitetään koko järjestelmän tehtävä ja sen jako osiin
  - tulos dokumentoidaan
- **vaatimusmäärittely** (analysis)
  - määritellään ohjelmisto kiinnittämällä sen päätoiminnot, ulkoiset vaatimukset ja rajoitukset, liittymät suorituskykyvaatimukset
  - tuloksena vaatimusdokumentti

## Lineaarinen malli (vesiputous, waterfall)

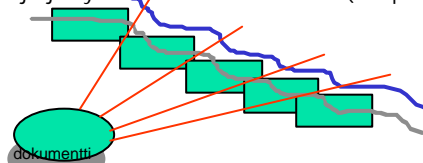
- **suunnittelu** (design)
  - suunnitellaan ohjelmiston tekninen rakenne, pääkomponentit, arkkitehtuuri, tietorakenteet, tietoliikenne, käyttöliittymä
  - tuloksena suunnitteludokumentti
- **toteutus** (implementation / coding)
  - suunnitelman realisointi toimivaksi ohjelmistoksi
  - tuloksena ohjelmat ja toteutusdokumentti
- **testaus** (testing)
  - toimivuuden ja vaatimusten täyttymisen varmistaminen ohjelmaa suorittamalla
  - virheiden korjaus ja dokumentointi
  - tuloksena testausdokumentti

## Lineaarinen malli (vesiputous, waterfall)

- **käyttöönotto**
  - käyttäjien koulutus, asennukset, tietojen siirrot, ...
- **ylläpito** (maintenance)
  - haluttujen muutosten tai tarvittavien korjausten toteuttaminen, sopeuttaminen
- Vaiheet voidaan jakaa edelleen alivaiheisiin, esim.
  - testaus = {yksikkötestaus, integrointitestaus, järjestelmätestaus}

## Lineaarinen malli (vesiputous, waterfall)

- Idealistinen tilanne: prosessi etenee järjestyksessä vaiheesta toiseen (vesiputous)



## Lineaarinen malli (vesiputous, waterfall)

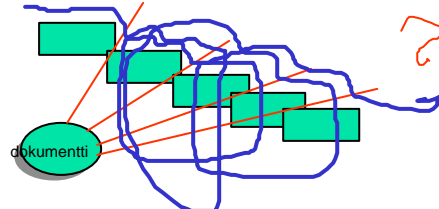
- selkeä, yksinkertainen, perinteinen malli
- tulodokumentti päättää vaiheen
- vasta vaiheen lopussa olevan tarkistuspisteen jälkeen voidaan edetä seuraavaan vaiheeseen
- käytännössä:
  - vaiheilla riippuvuuksia
  - vaiheen suoritus saattaa paljastaa edellisessä vaiheessa tehtyjä virheitä
    - prosessia tulee peruuttaa -> prosessi muuttuu iteratiiviseksi
  - palattava toistamaan aikaisempia vaiheita etenemisjärjestys muutoin säilyttäen

© Harri Laine, Jukka Paakki

13

## Lineaarinen malli (vesiputous, waterfall)

- Todellinen tilanne



© Harri Laine, Jukka Paakki

14

## Lineaarinen malli - ongelmia

- iteratiivisuus on huonosti kuvattu
- kiinnittää tarkistuspisteet ja dokumentit vaiheiden rajapinnoille -> käytännön peruutus hankalaa
- prosessin loppupäässä käynnistyvä peruutus (esim. ylläpito --> suunnittelu) saattaa olla todella kallis operaatio (suuri osa vaiheista ja niiden tuloksista uusittava)
- toimiva järjestelmä saadaan asiakkaan tutkittavaksi hyvin myöhään

© Harri Laine, Jukka Paakki

15

## Dokumentointi lineaarisen mallin yhteydessä

- vaatimusmäärittely
  - esitutkimusraportti
  - vaatimusdokumentti
  - toimintojen kuvaukset
  - hyväksymistestaussuunnitelma
  - luonnos käyttöohjeesta
- suunnittelu
  - järjestelmäarkkitehtuuri
  - järjestelmätestaussuunnitelma
  - liittymämäärittelyt
  - moduulisuunnitelmat
  - integroitintestaussuunnitelma

© Harri Laine, Jukka Paakki

16

## Dokumentointi lineaarisen mallin yhteydessä

- toteutus
  - ohjelmakoodi
  - yksikkötestaussuunnitelmat
  - tekniset dokumentit
- testaus
  - yksikkötestausraportit
  - integroitintestausraportti
  - järjestelmätestausraportti
  - hyväksymistestausraportti

© Harri Laine, Jukka Paakki

17

## Dokumentointi lineaarisen mallin yhteydessä

- Tiukasti dokumentteihin sidotun etenemisen ongelmia:
  - dokumenttien sovittu aikataulu ei välttämättä vastaa prosessin aikataulua
  - iterointi hankaloituu (onko uusittavai tehtävä korjausliitteitä?)
  - dokumenttien I/O-olemus liian suoraviivainen näkemys prosessista
  - joskus tarkan dokumentin tekeminen on mahdotonta

© Harri Laine, Jukka Paakki

18

## Lineaarisen mallin plussat

- yksinkertainen
- tunnetuin
- yleisesti hyväksytty
- modularisoi hyvin ohjelmistotuotantoprosessin
- tarjoaa pohjan kehittyneemmille malleille

© Harri Laine, Jukka Paakki

19

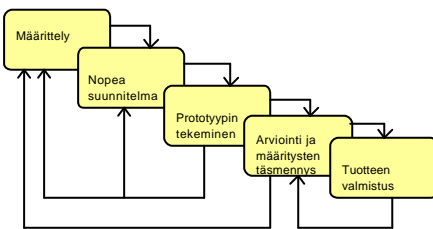
## 2. Prototyypimalli

- Prototyyppi = nopeasti toteutettu osa järjestelmästä, muokataan tarpeen mukaan
- Idea: järjestelmän keskeisiä osia kokeillaan prototyyppien avulla ennen toteutusvaihetta
  - epäselvien asiakasvaatimusten selventäminen
  - ratkaisuvaihtoehtojen selvittäminen
  - käyttöliittymän ja järjestelmän palvelujen esittäminen ja tarkentaminen asiakkaalle
  - nopeasti jotain toimivaa arvioitavaksi
  - edellyttää nopean kehityksen välineitä

© Harri Laine, Jukka Paakki

20

## Prototyypimalli



© Harri Laine, Jukka Paakki

21

## Prototyypimalli

- Kahdenlaisia prototyyppejä:
  - hylättävät prototyypit
    - prototyyppi ei siirry koskaan tuotantoon, vaan tätä varten tehdään erillinen järjestelmä, mahdollisesti kokonaan eri välinein
  - kehittyvät prototyypit
    - prototyypistä jalostetaan tuotantoversio
    - jalostus voi tapahtua monen prototyypin kautta

© Harri Laine, Jukka Paakki

22

## Prototyypimalli - ongelmia

- Asiakas voi tulkita prototyypin varsinaiseksi järjestelmäksi, eikä ymmärrä viivettä
  - miksei järjestelmää voi vielä käyttää, toimihan se jo kuukausia sitten?
- Milloin protoilu pitäisi lopettaa?
- Kehittyvien prototyyppien kohdalla epäkypsät suunnitteluratkaisut voivat siirtyä tuotteeseen -> tehottomuutta
- Hylättävät prototyypit -> tuplatoteutus, resurssien hukkakäyttö?

© Harri Laine, Jukka Paakki

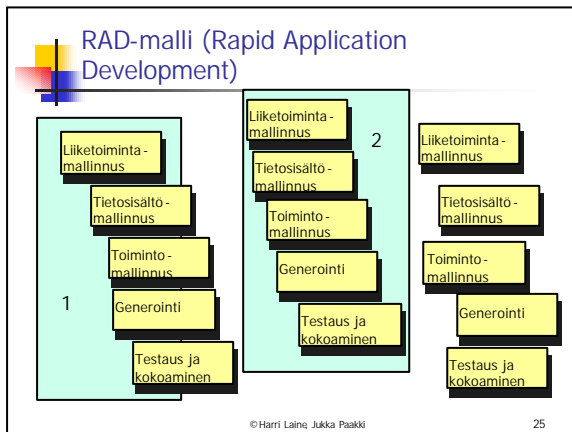
23

## 3. RAD-malli (Rapid Application Development)

- Järjestelmä jaetaan melko itsenäisiin komponentteihin
- Kutakin komponenttia kehitetään lineaarisen mallin mukaisesti
  - liiketoimintamalli (business model)
  - tietosisältömalli (information model – keskeinen)
  - toimintomalli (process model)
  - sovelluksen generointi (kehittimien käyttö)
  - testaus ja kokoaminen

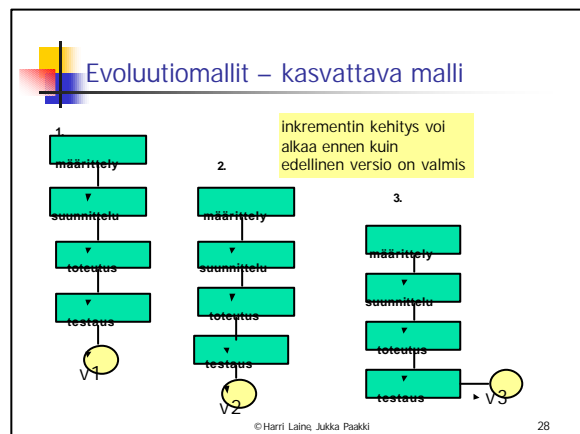
© Harri Laine, Jukka Paakki

24



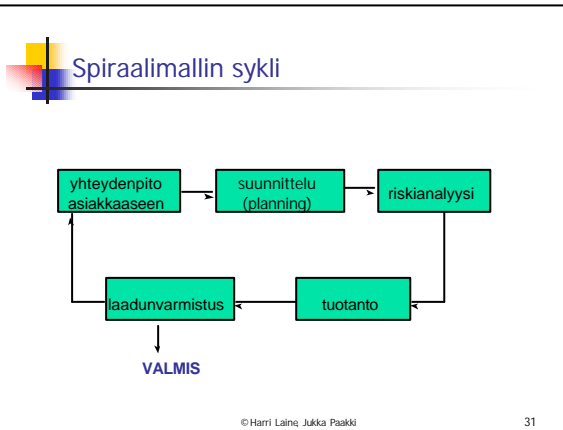
- ### RAD-malli (Rapid Application Development)
- Sopii, jos ohjelmiston rakenne selkeästi modulaarinen
  - Komponentin kehitys ei saisi viedä yli 3 kk
  - Asiakas ja kehittäjät tiiviissä yhteistyössä
  - Tarvitaan hyvät työkalut
- © Harri Laine, Jukka Paakki 26

- ### 4. Evoluutiomallit – kasvattava malli
- Suppean ydinjärjestelmän laajentaminen tuotantojärjestelmäksi pieninä täydennyksinä, ”inkrementteinä”
  - Sarja toistuvia vesiputouksia
  - Versio n laajentaa versiota n-1
  - Version n kehittämisessä voidaan ottaa huomioon versiosta n-1 saatu palaute
- © Harri Laine, Jukka Paakki 27



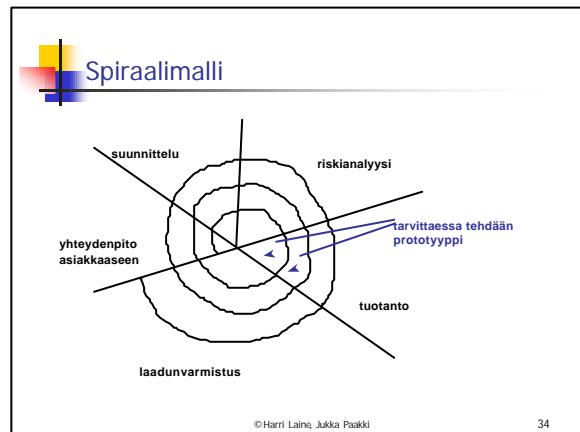
- ### Evoluutiomallit – kasvattava malli
- Pienet lisäykset kasvattavat monimutkaisuutta
  - Lopputulos riippuu version 1 ydinjärjestelmän ratkaisuista
  - Projektin- ja versionhallinta monimutkaistuu
  - Miten dokumentoidaan: jokaiselle versiolle täydellisenä vaiko inkrementaalaisesti - jos näin, löytyykö kokonaisuus?
- © Harri Laine, Jukka Paakki 29

- ### 4.1. Spiraalimalli
- Iteratiivinen malli, joka yhdistää vesiputouksimallin, prototyypimallin ja riskianalyysin
  - Kasvattava lähestymistapa
  - Tuote valmistuu sarjana vaihetuotteita
    - aikaiset vaihetuotteet voivat olla paperimalleja tai prototyyppejä
    - myöhemmät versiot laajentavat aiempaa
  - Mallissa suoritetaan toistuvasti toimintoklejä
- © Harri Laine, Jukka Paakki 30



- ### Spiraalimalli - riskianalyysi
- Riski: asia, joka voi mennä pieleen
    - esim. jonkin uuden ohjelmointikielen käyttöönotto
      - onko soveltuva ko. tehtävään?
      - onko käytettävissä osavia ohjelmoijia?
      - onko markkinoilla tehokkaita/toimivia toteutuksia?
    - riski havaitaan -> hankitaan puuttuva tieto
      - kokeillaan kieltä pilottiprojektissa
      - kartoitetaan ohjelmoijien osaaminen
      - tehdään markkinaselvitys
    - riskianalyysin perusteella tehdään käyttöönottopäätös
- © Harri Laine, Jukka Paakki 32

- ### Spiraalimalli - riskianalyysi
- Riskit vähenevät joka kierroksella
    - tosin esiin saattaa nousta uusia riskejä
      - esim ohjelmointikieliriski ei esiinny paperimallia tehtäessä
  - Tuote tarkentuu kierros kierrokselta
  - Asiakas arvioi vaihetuotetta ja antaa palautetta
- © Harri Laine, Jukka Paakki 33



- ### Spiraalimallin ongelmia
- asiakkaan toistuva aktivoiminen rasittaa
  - tarkentuvat prototyypit -> aikaavievää
  - käytännössä vajavaiseen riskiasiantuntemukseen nojautuminen
  - ei vakiintunut -> (yllättävän) vähän käytännön kokemuksia ja näyttöjä
- © Harri Laine, Jukka Paakki 35

- ### 4.2. Komponenttimalli
- Spiraalimallin muunnelmä, jossa perusideana on tehdyn työn hyödyntäminen
  - Jokaisen syklin alussa etsitään valmiita ratkaisuja esillä oleviin ongelmiin ja otetaan sellainen käyttöön, jos löytyy
  - Suunnitteluratkaisut, ohjelmakomponentit
  - Jos ratkaisua ei löydy valmiina, se tehdään itse
  - Ratkaisuvälikomponentit – komponenttikirjasto
- © Harri Laine, Jukka Paakki 36

## 5. Ketterät mallit

- Ns. **kevyet** (lightweight) tai **ketterät** (agile) prosessimallit:
  - projektinhallinta on kevyttä
  - dokumentointi on kevyttä
  - iteraatiot ovat lyhyitä ja ketteriä
  - inkrementit ovat pieniä ja kevyitä
  - soveltuvat pienehköille ja ketterille tiimeille (alle 10 henkeä)
  - soveltuvat ohjelmistoihin, joiden vaatimukset tai määritykset ovat epäselviä tai erityisen herkkiä muuttamaan jatkuvasti

© Harri Laine, Jukka Paakki

37

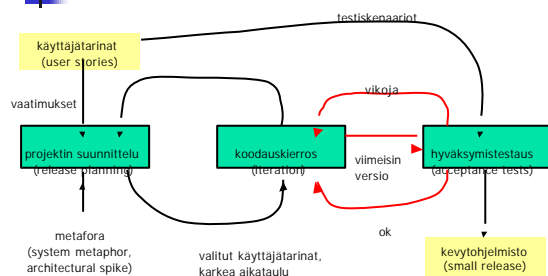
## 5.1. Extreme Programming (XP)

- Tunnetuin kevytmalli
- "Extreme programming":
  - ohjelmakoodi on tärkein työväline
  - koodi on ohjelmiston tärkein dokumentti
  - koodin laatu (oikeellisuus, ymmärrettävyys) keskeisessä asemassa
- Lähteitä:
  - K. Beck: *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.
  - <http://www.extremeprogramming.org/>

© Harri Laine, Jukka Paakki

38

## Extreme Programming (XP) – prosessi



© Harri Laine, Jukka Paakki

39

## Extreme Programming (XP) – käytännöt

1. Metafora (metaphor)
  - vertauskuvallinen kuvaus järjestelmästä
  - korvaa tavanomaisen arkkitehtuurin
2. Suunnittelupeli (planning game)
  - jokaisen iteraatiokierroksen (1-3 viikkoa) aloitus
  - valitaan iteraatiossa toteutettavat käyttäjätarinat ja arvioidaan toteuttamisen aikataulu
3. Paikalla oleva asiakas (on-site customer)
  - asiakas koko ajan mukana
  - asiakas suunnittelee ja suorittaa hyväksymistestit

© Harri Laine, Jukka Paakki

40

## Extreme Programming (XP) – käytännöt

4. Yksinkertainen suunnittelu (simple design)
  - mahdollisimman yksinkertaiset tekniset ratkaisut
  - toteutetaan vain suunnittelupelissä valittuja käyttäjätarinoita, ei mitään ylimääräistä
5. Koodin yhteisöomistus (collective ownership)
  - kuka tahansa voi muuttaa mitä tahansa osaa koodista
  - muuttajan vastuulla uusien testien laatiminen
6. Pariohjelmointi (pair programming)
  - kaksi kehittäjää (koodaaja, katselmoija) tuottaa koodia saman tietokoneen ääressä
  - XP:n kiistanalaisin käytäntö

© Harri Laine, Jukka Paakki

41

## Extreme Programming (XP) – käytännöt

7. Koodin parantelu, "refaktorointi" (refactoring)
  - rakenteellisesti huonolaatuisen koodin korjaaminen selkeämmäksi ja hyvien tapojen mukaiseksi
  - koodin toiminnallisuus ei saa muuttua
8. Ohjelmointistandardi (coding standard)
  - yhteisten ohjelmoinnin tyylisääntöjen noudattaminen
  - mahdollistaa yhteisöomistuksen ja kommunikoinnin
9. Testaus (testing)
  - yksikkö- ja hyväksymistestit, suunnitellaan ennen koodausta
  - automatisoidut testit ⇒ iterointi, regressiotestaus

© Harri Laine, Jukka Paakki

42



## Extreme Programming (XP) – käytännöt

10. Pienet versiot (small releases)
  - kasvattava ohjelmankehitys pienin inkrementein
11. Jatkuva integrointi (continuous integration)
  - uusi, yksikkötestit läpäissyt koodi liitetään heti ohjelmistoon
  - aina käytettävissä viimeisin versio, mutta jatkuvan regressiotestauksen kustannuksella
12. 40 tunnin työviikko (40-hour week)
  - ylityötä vain poikkeustilanteissa
  - tavoitteena yhteisen työtehon pitäminen korkeana