

Nykyaikaiset ohjelmistokehityksen mallit

MallinnusOSY:n pienseminaari 17.9.2007

>  
**accenture**

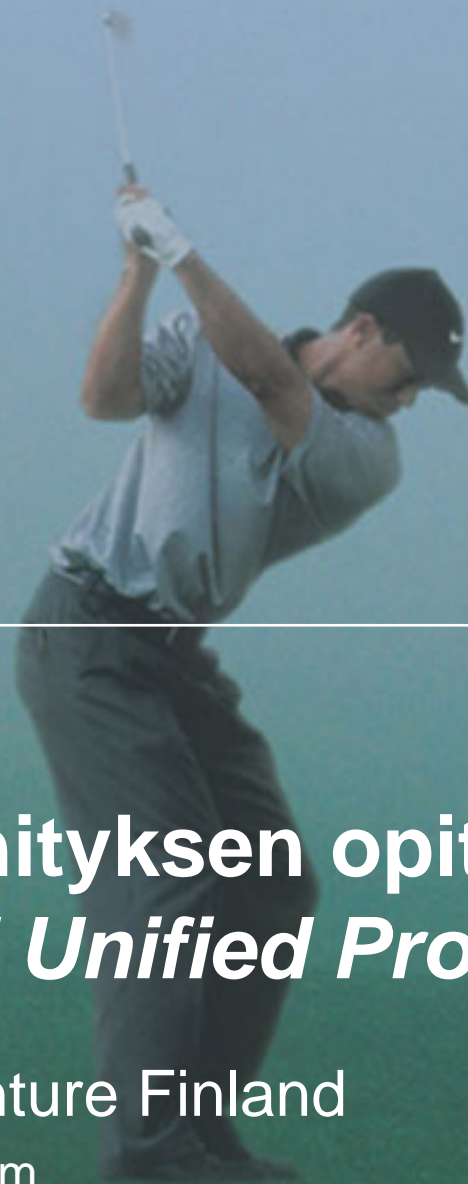
*High performance. Delivered.*

## **Ohjelmistokehityksen opit ja RUP (*Rational Unified Process*)**

**Allan Halme**, Accenture Finland

allan.halme@accenture.com

allan@iki.fi



# Ohjelmistokehityksen ongelmista



[Kruchten 2004]

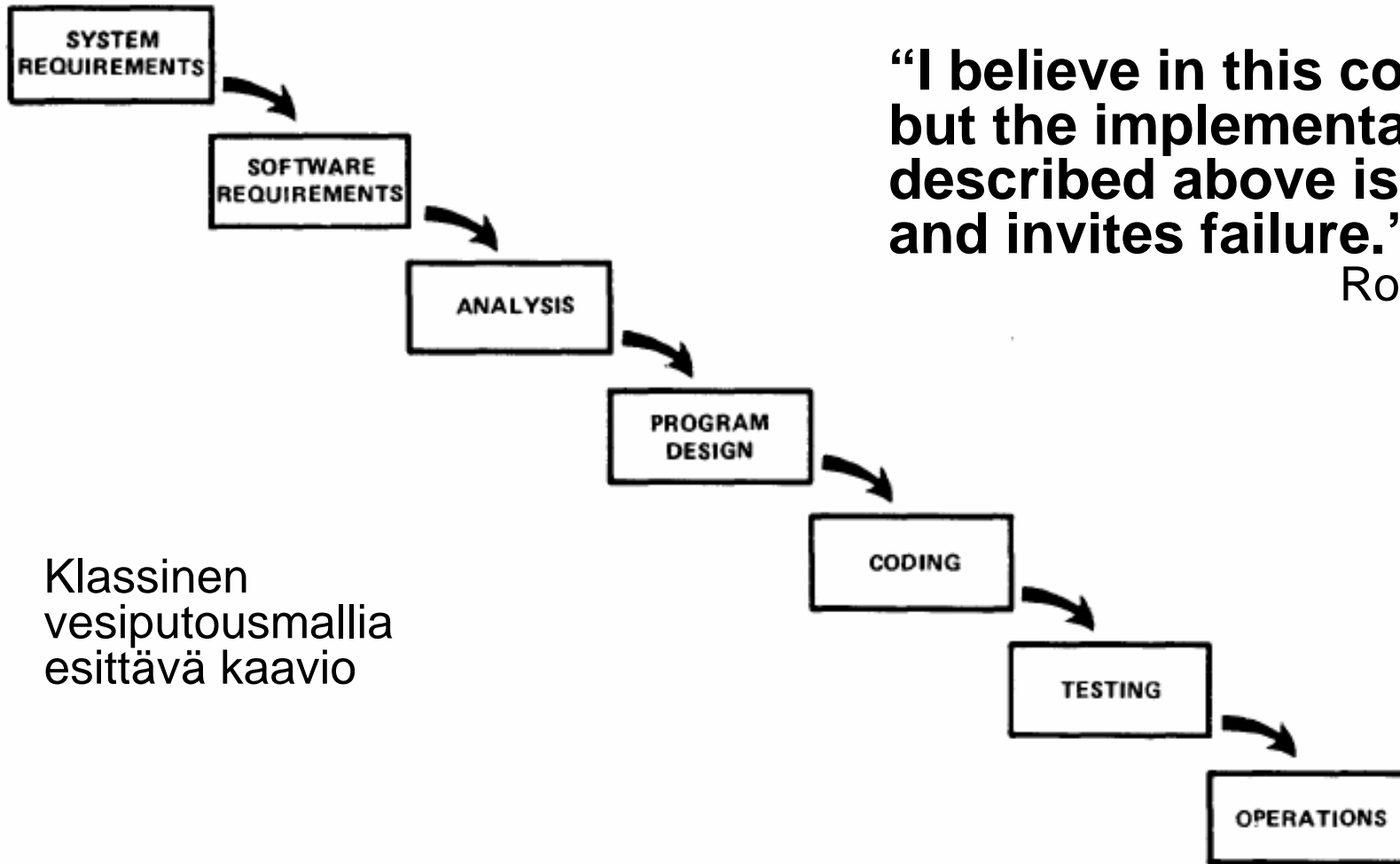
## Oireita

- Loppukäyttäjien tarpeiden puutteellinen ymmärtäminen
- Kykenemättömyys käsitellä muuttuvia vaatimuksia
- Yhteensopimattomat moduulit
- Vaikeasti ylläpidettävät tai laajennettavat ohjelmat
- Vakavien projektiongelmiin myöhäinen havaitseminen
- Heikko laatu
- Kelpaamaton suorituskyky
- Tiimin jäsenten työ ei ole sujuvaa ja jäljitettävää
- Epäluotettava kokoonpano- ja julkaisuprosessi

## Perimmäisiä syitä

- *Ad hoc* -vaatimustenhallinta
- Epäselvää ja epätarkkaa tiedonvälitystä
- Hauraat arkkitehtuurit
- Ylivoimainen kompleksisuus
- Huomaamatta jääneet epäjohtonmukaisuudet vaatimuksissa, suunnitelmissa, ja toteutuksissa
- Riittämätön testaus
- Projektin statuksen subjektiivinen arviointi
- Riskien riittämätön estäminen
- Muutosten hallitsematon leviäminen
- Riittämätön automatisointi

# Vesiputous



**“I believe in this concept,  
but the implementation  
described above is risky  
and invites failure.”**

Royce, 1970

Klassinen  
vesiputousmallia  
esittävä kaavio

Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

# “Järkevä” lähestymistapa



Vesiputousmalli vaikuttaa järkevältä lähestymistavalta:

1. Ymmärrä ongelma ja vaatimukset, ja dokumentoi ne.
2. Suunnittele vaatimukset toteuttava ratkaisu.
3. Toteuta suunnitelma parhaalla mahdollisella tavalla.
4. Varmista että toteutus täyttää annetut vaatimukset.
5. Järjestelmä on nyt valmis.

... paitsi että pilvenpiirtäjiä ja siltoja on rakennettu satoja ja tuhansia vuosia, mutta ohjelmia tehty vasta muutama kymmenen ...

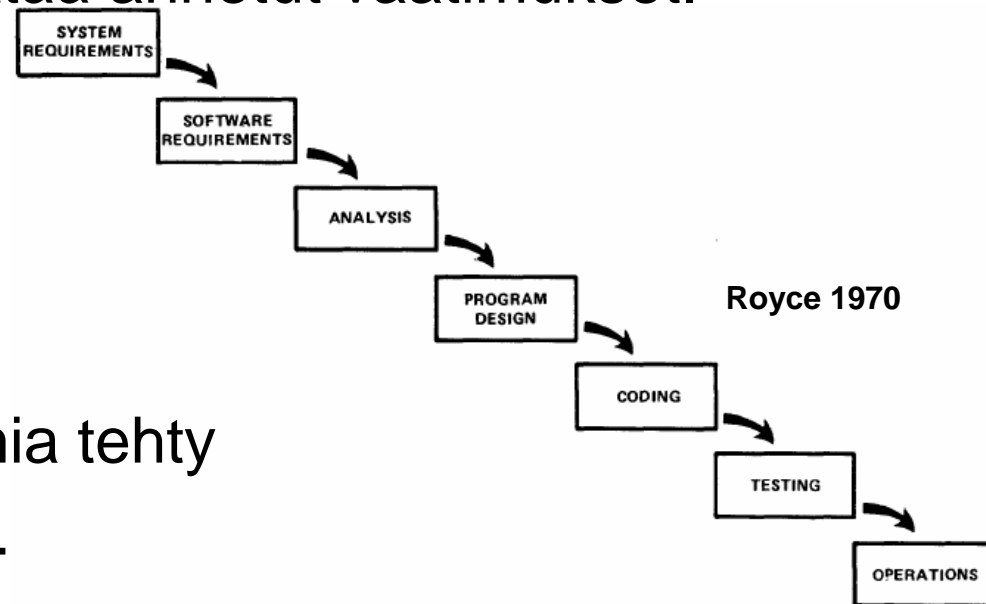
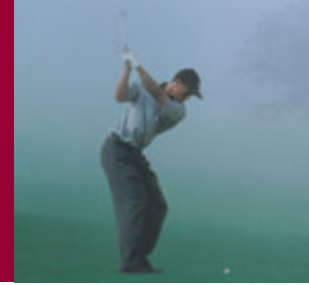


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

# Muutama virheellinen oletus



1. **“Vaatimukset voidaan jäädyttää”** – paitsi että ...
  - käyttäjät muuttuvat
  - ongelma muuttuu (mm. IKIWISI)
  - teknologia muuttuu
  - markkinat muuttuvat
  - vaatimuksia ei voida kiinnittää riittävän tarkasti ja yksityiskohtaisesti
2. **“Suunnittelu voidaan onnistuneesti tehdä paperilla ennen kuin jatketaan”** – paitsi että ...
  - oikeellisuutta ei käytännössä voida verifioida analyttisesti
  - sillan rakentaminen pohjautuu fysiikan lakeihin, mutta mitään vastaavaa pohjaa ei ole ohjelmistotuotannossa

## Lisäksi ongelmia aiheuttavat –

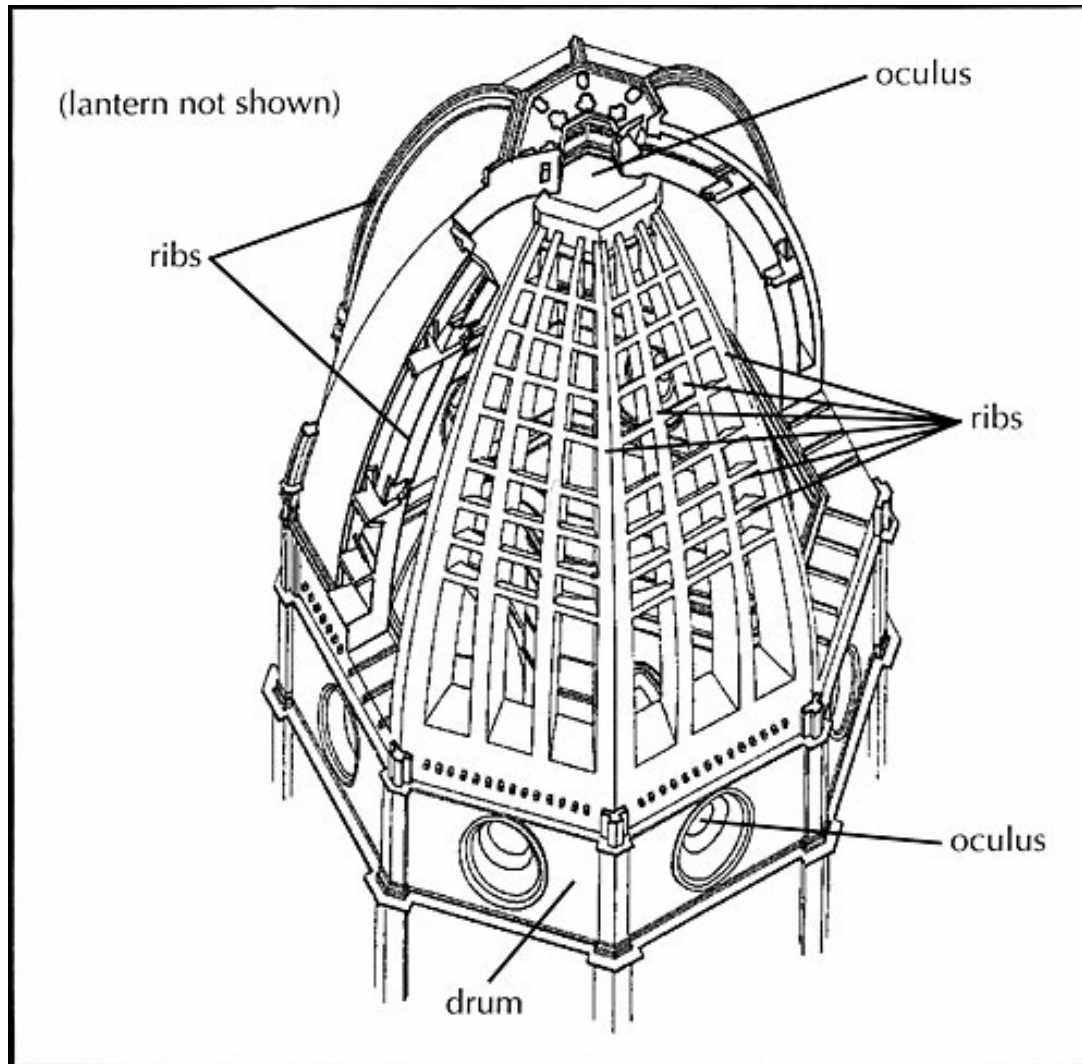
Riskit, pidemmät projektit, dokumenttien pyörittäminen, aika-  
vs-laajuus

**Firenzen  
katedraalin  
kupoli**



**Filippo Brunelleschi (1377-1446),  
arkkitehti**

# “Arkkitehtuurikaavio”



# Staattinen rakenne: Prosessin kuvaus



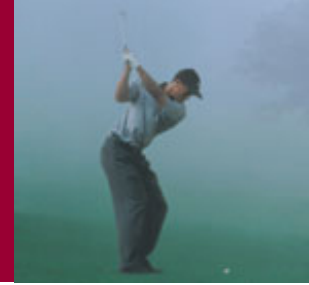
RUPia esitetään kuudella keskeisellä käsitteellä

1. **Roolit** (*roles*) – *kuka* tekee
2. **Aktiviteetit** (*activities*) – *miten* tehdään
3. **Lopputuotteet** (*deliverables*) – *mitä* tehdään
4. **Työnkulut** (*workflows*) – *milloin* tehdään
5. **Työvaiheet** (*disciplines*) – *kokoaa* näitä yhteen
6. Elinkaarivaiheet (*phases*) – projektin elinkaarimalli

Lisäksi on

- Ohjeita (*guidelines*)
- Lopputuotepohjia (*templates*)
- Työkaluohjeita (*tool mentors*)
- Käsitteet (*concepts*)

# Dynaaminen rakenne: Iteratiivinen kehitysmalli



## Elinkaarivaiheet (phases)

Aloitus (*inception*)

Tarkennus (*elaboration*)

Rakennus (*construction*)

Siirtymä (*transition*)

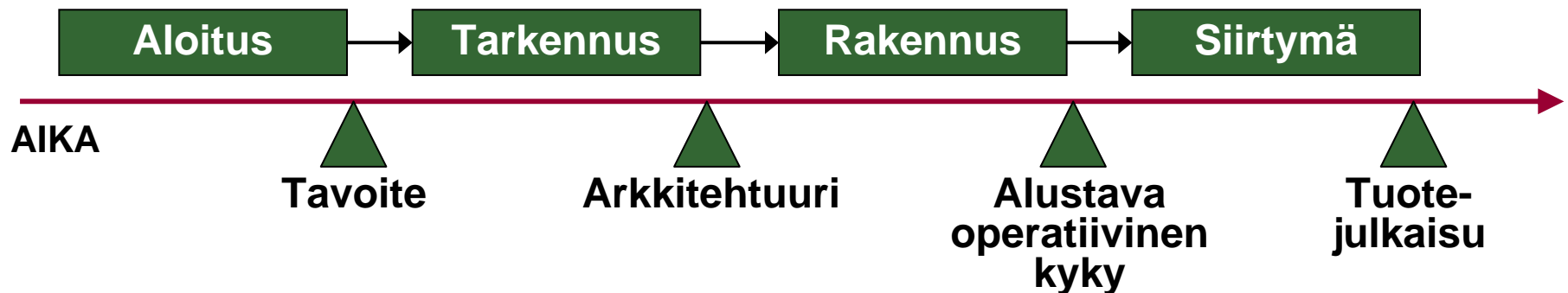
## Merkkipaalat (milestones)

Tavoite (*mitä* tehdään?)

Arkkitehtuuri (*miten* se tehdään?)

Alustava operatiivinen kyky ( $\beta$ )

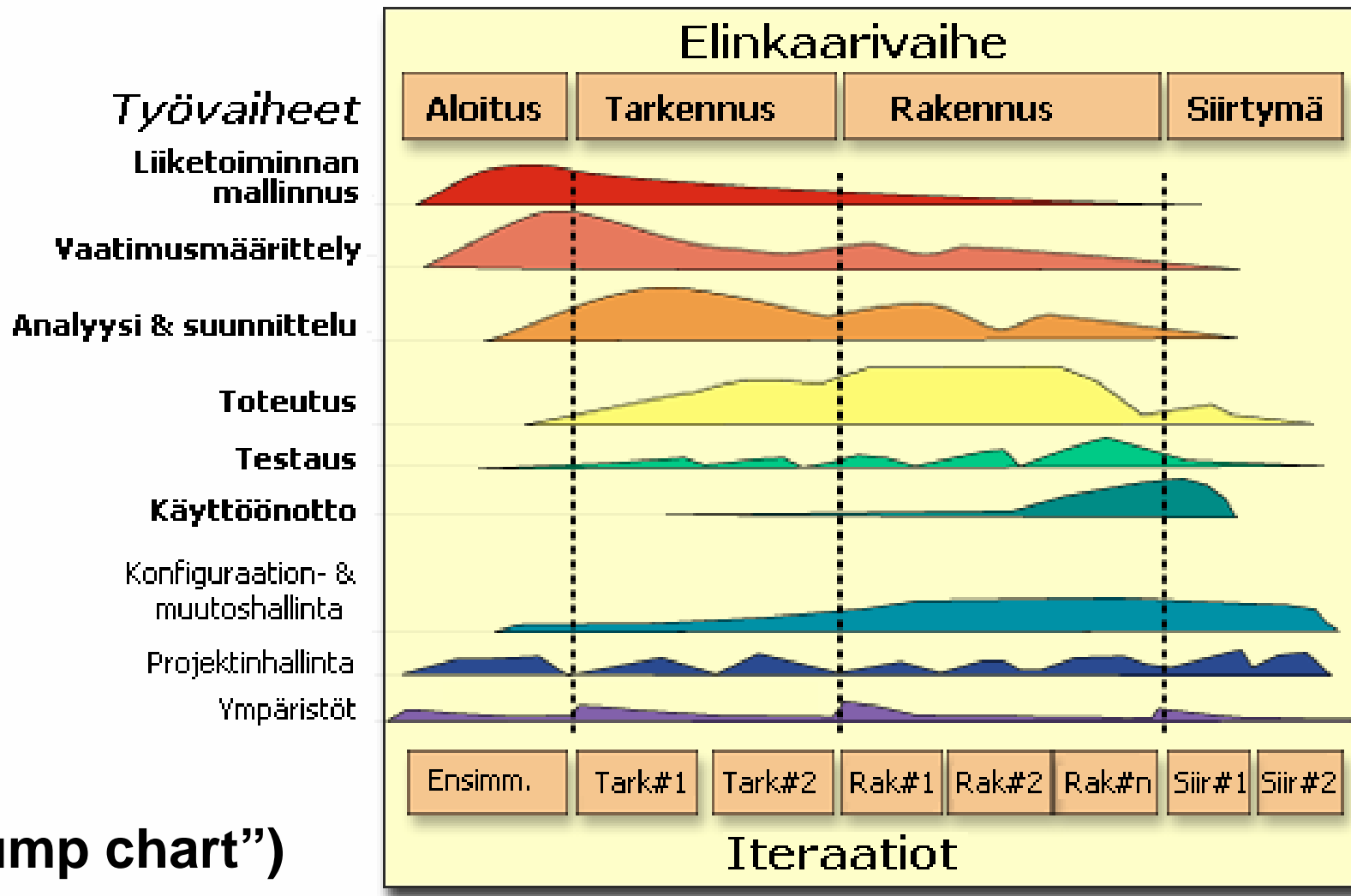
Tuotejulkaisu



[Boehm 1996, Kroll, RUP]



# Kyttyräkaavio. Koko totuus.



# Työvaiheet (*disciplines*)



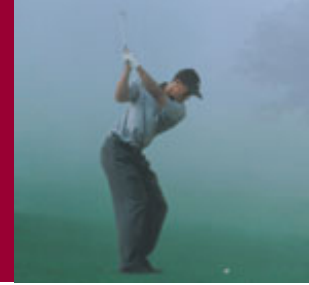
## Tekniset työvaiheet

1. **Liiketoiminnan mallinnus** (*business modeling*)
2. **Vaatimusmäärittely** (*requirements*)
3. **Analyysi ja suunnittelu** (*analysis and design*)
4. **Toteutus** (*implementation*)
5. **Testaus** (*test*)
6. **Käyttöönotto** (*deployment*)

## Avustavat työvaiheet

7. **Projektinhallinta** (*project management*)
8. **Konfiguraation- & muutoshallinta** (*configuration and change management*)
9. **Ympäristöt** (*environment*)

# Elinkaarivaiheet (1)



## 1. Aloitus (*Inception*)

### Tavoitteet

- Vahvistaa laajuus, reunaehdot, hyväksymiskriteerit, sisältö
- Erottaa keskeiset käyttötapaukset
- Esitellä vähintään yksi arkkitehtuuriehdotus
- Arvioida hinta ja aikataulu
- Arvioida riskit

### Aktiviteetit

- Projektin laajuuden muodostus
- Projektin perusteluiden ja hallinnon suunnittelu
- Ehdokas-arkkitehtuurin luominen

### Tulokset

- Visio: vaatimukset, toiminnot, rajaukset
- Käyttötapauskartoitus
- Alustava sanasto
- Projektin perustelut ja oikeutus (*business case*)
- Alustava riskianalyysi
- Projektisuunnitelma (vaiheet & iteraatiot)

[RUP]

## 2. Tarkennus (*Elaboration*)

### Tavoitteet

- Viitearkkitehtuurin määrittely, validointi, ja luonti
- Vision sitouttaminen
- Rakennusvaiheen tarkka suunnitteluluonnos
- Osoittaa, että viitearkkitehtuuri tukee visiota kohtuullisella kustannuksella kohtuulliseksi ajaksi

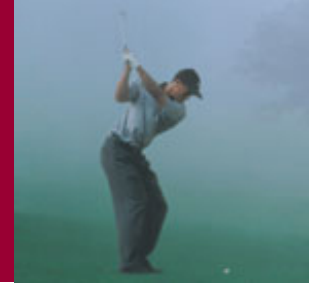
### Aktiviteetit

- Vision ja keskeisten käyttötapausten kattava tarkennus
- Prosessi, puitteet, työkalut, tuki on tarkennettu ja asennettu käyttöön
- Arkkitehtuuri on tarkennettu, komponentit integroitu, ja mahdollisesti uudelleen-suunniteltu opitun kokemuksen valossa

### Tulokset

- Käyttötapausmalli (80% valmis)
- Arkkitehtuurikuvaus ja lisävaatimukset
- Ajokelpoinen arkkitehtuuriprototyyppi
- Päivitetty riskianalyysi
- Projekti- ja prosessisuunnitelma
- Mahdollisesti alustava käyttöohje

# Elinkaarivaiheet (2)



## 3. Toteutus (*Construction*)

### Tavoitteet

- Minimoi kehityskustannukset; resurssien optimointi ja tarpeettoman työn ja korjailun välttäminen
- Riittävän laadun saavuttaminen ja käyttökelpoisten järjestelmäversioiden tuottaminen niin nopeasti kuin käytännöllistä

### Aktiviteetit

- Resurssien hallinta ja prosessin optimointi
- Komponenttien kehitys ja testaus
- Julkaisujen arviointi kriteerejä vasten

### Tulokset

- Integroitu ja toimiva ohjelmisto, joka on *käyttökelpoinen* (joskin *beta*), ja joka *annetaan käyttäjien käytettäväksi*
- Käyttöohjeet
- Julkaisun kuvaus

## 4. Siirtymä (*Transition*)

### Tavoitteet

- Käyttäjät pystyvät käyttämään tuotetta
- Hankkia sidosryhmiltä hyväksyntä toimitetulle järjestelmälle
- Saavuttaa lopullinen järjestelmäversio nopeasti ja kustannustehokkaasti

### Aktiviteetit

- Käyttöönoton toimenpiteitä
- Hienosäätöä, ml. vikojen korjausta sekä käytettävyyden ja suorituskyvyn parannukset
- Julkaisujen arviointi kriteerejä vasten

### Tulokset

- Valmis tuote tai järjestelmä

[RUP]

# Tyypillisiä virheitä RUPin käyttöönotossa



## Yleistä

- RUPista otetaan käyttöön liian laajaa osaa (ks.mm. [Larman 2003, s. 192])
- Kaikki omaksutaan kerralla, eikä iteratiivisesti
- RUPin käyttöönotto jätetään suunnittelematta
- Ohjelmistokehitysprosessin kehittäminen jätetään kytkemättä organisaation tavoitteisiin ja tuloksiin
- Liian suuri osa RUPista räätälöidään liian varhain
- RUPiin vain “verhoudutaan”

## Iteratiivisen kehittämisen hallinta

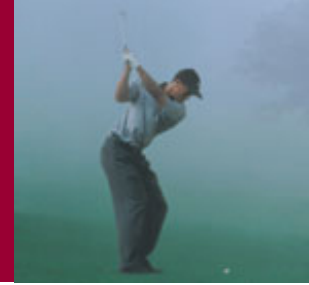
- Organisaatio on jaettu toiminnoittain ja on erikoistunut
- Sidosryhmillä on virheelliset odotukset tai hankintamalli on vanhanaikainen
- Projektin alussa on liikaa kehittäjiä mukana
- Helpot ongelmat ratkaistaan ensin
- Ensimmäinen iteraatio pitkittyy
- Iteraatiot ovat päällekkäisiä
- Projektin loppuvaiheessa sallitaan liikaa muutoksia

## Systemityö

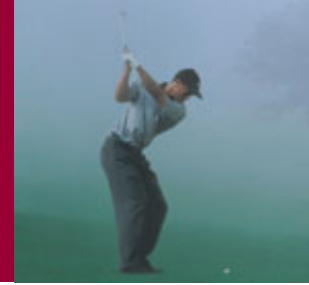
- Käyttötapauksia luodaan liikaa
- Analyysiparalyysi
- Vaatimukset sisältävät suunnittelupäätöksiä
- Kaikilta sidosryhmiltä ei ole hankittu hyväksyntä vaatimuksille
- “Not Invented Here” -syndrooma
- Tarkennusvaihe lopetetaan ennen kuin arkkitehtuuri on riittävän vakaa
- Keskittyminen katselmuksiin toimivan ohjelman arvioinnin sijasta

[Kroll, Bergström, Råberg]

# Iitalukemistoa



- Larman, ***Agile & Iterative Development: A Manager's Guide***
- Kroll & Maclsaac, ***Agility and Discipline Made Easy: Practices from OpenUP and RUP***, 2006
- Royce, ***Software Project Management: A Unified Framework***
- Bittner & Spence, ***Managing Iterative Software Development Projects***, 2007
- Jacobson, Booch, Rumbaugh, ***The Unified Software Development Process***
- Larman & Basili, ***Iterative and Incremental Development: A Brief History***, IEEE Software, June 2003
- Arlow & Neustadt, ***UML 2 and the Unified Process***, 2nd edition, 2005
- Kruchten, ***The Rational Unified Process: An Introduction***, 3rd ed., 2004
- Kroll, Kruchten, ***The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP***, 2003
- Bergström, Råberg, ***Adopting the Rational Unified Process: Success with the RUP***, 2004
- Pollice, Augustine, Lowe, Madhur, ***Software Development for Small Teams: A RUP-centric Approach***, 2004
- Ambler, Nalbone, Vizdos, ***The Enterprise Unified Process: Extending the Rational Unified Process***, 2005
- Boehm, ***Anchoring the Software Process***, IEEE Software, July 1996
- Royce, ***Managing the Development of Large Software Systems***, Proceedings, IEEE WESCON, 1970
- King, ***Brunelleschi's Dome: How a Renaissance Genius Reinvented Architecture***, 2001
- Viralliset nettisivut, <http://www-306.ibm.com/software/awdtools/rup/>
- IBM developerWorks, <http://www.ibm.com/developerworks/rational/products/rup>
- Wikipedia, [http://en.wikipedia.org/wiki/Rational\\_Unified\\_Process](http://en.wikipedia.org/wiki/Rational_Unified_Process)
- EPF ja **OpenUP**, <http://www.eclipse.org/epf/>
- Software Program Manager's Network, <http://www.spmn.com>



# Allan Halme

[allan.halme@accenture.com](mailto:allan.halme@accenture.com)

[allan@iki.fi](mailto:allan@iki.fi)