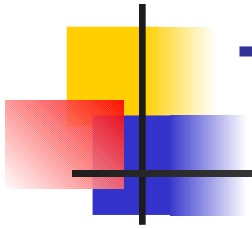
Tarkvaratehnika: fundamentaalsed probleemid metoodika valikul

Asko Seeba

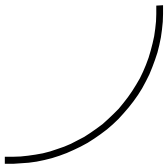
sügis 2007



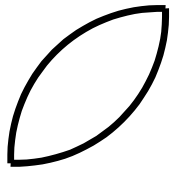
Teadmine ja suhtlemine

- Alistair Cockburn:
  - *Q: Can you ever know what you are experiencing, and can you ever communicate it?*
  - *A: No, you can't*
- Mida siis teha?
  - Ära püüa perfektselt suhelda
  - Õpi suhtluse ebatäielikkust juhtima

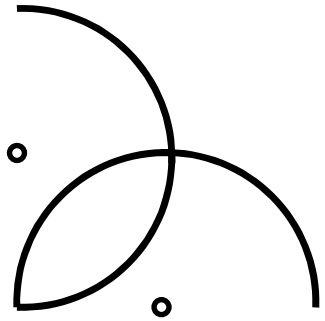
Puudulik suhtlemine



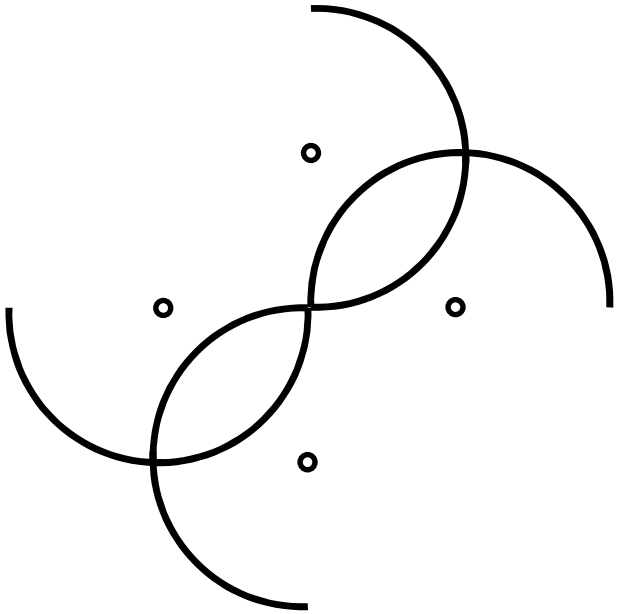
Puudulik suhtlemine



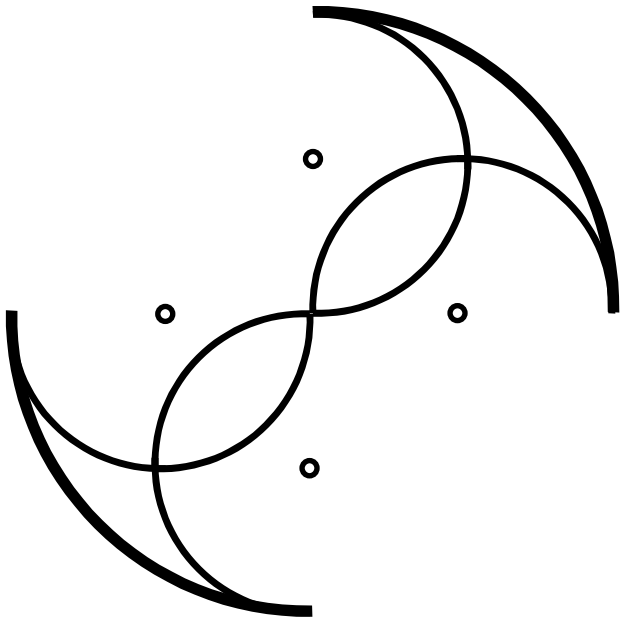
Puudulik suhtlemine



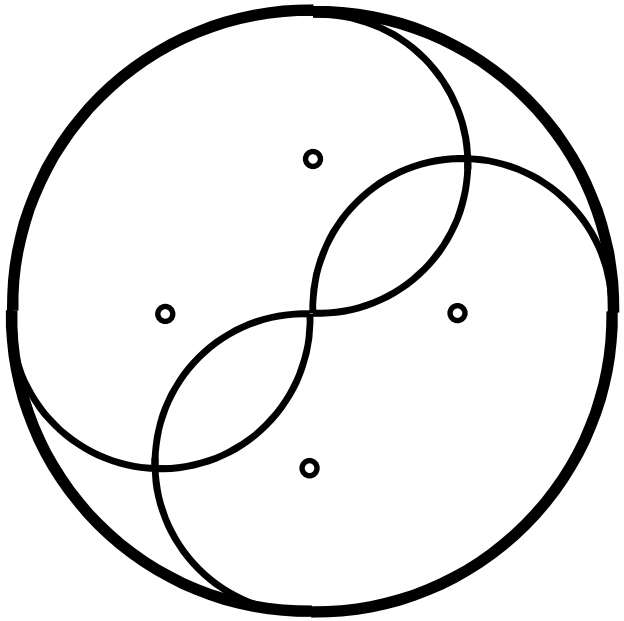
Puudulik suhtlemine



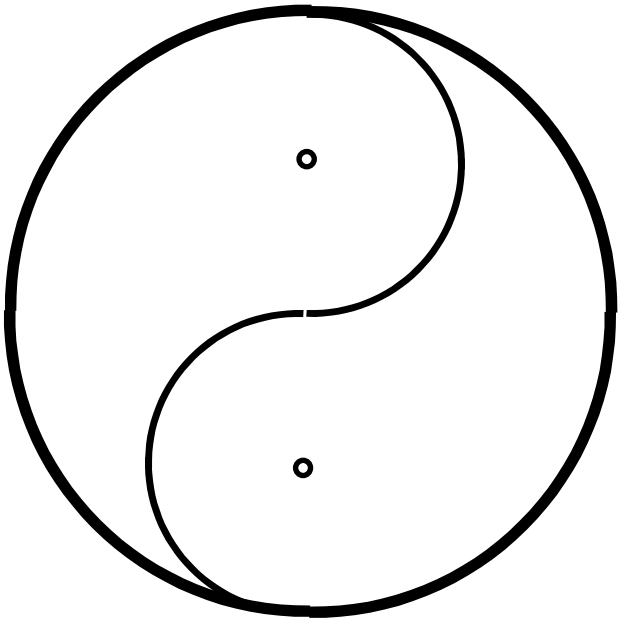
Puudulik suhtlemine

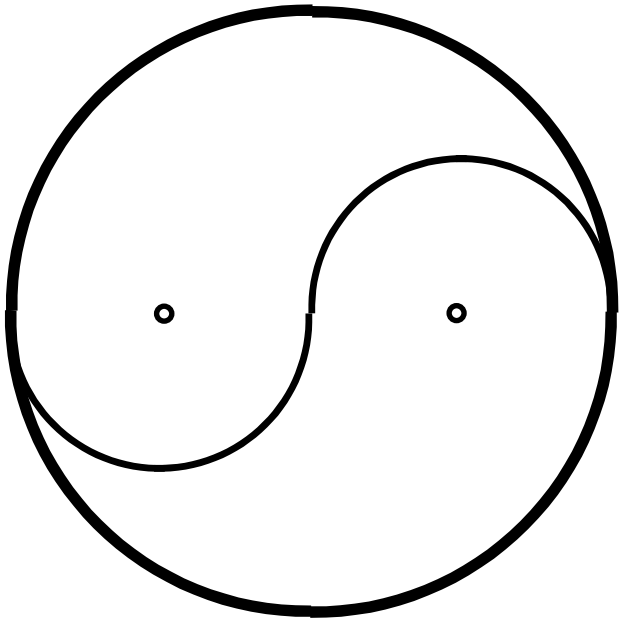


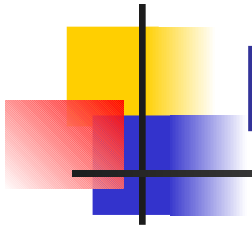
Puudulik suhtlemine



Puudulik suhtlemine

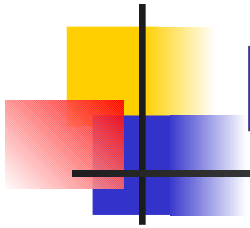






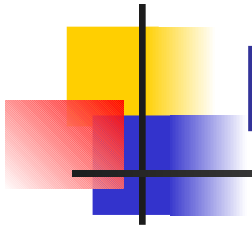
Puudulik suhtlemine

- 1, 3, 7, 11, 13, ...



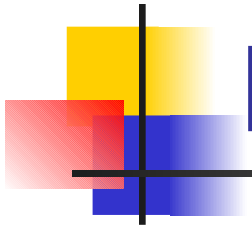
Puudulik suhtlemine

- 1, 3, 7, 11, 13, ...
- 1, 3, 7, 11, 13, 15, ...



Puudulik suhtlemine

- 1, 3, 7, 11, 13, ...
- 1, 3, 7, 11, 13, 15, ...
- 1, 3, 7, 11, 13, 15, 5, 9, ...

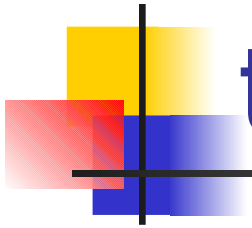


Puudulik suhtlemine

- 1, 3, 7, 11, 13, ...
- 1, 3, 7, 11, 13, 15, ...
- 1, 3, 7, 11, 13, 15, 5, 9, ...

Inimeste olemasolu tarkvaraprojektis on oluline projekti tulemust mõjutav faktor

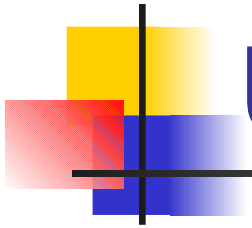
Ühine kogemus – aeglane tempo



- Algajad õpivad programmeerimist:
  - alamklassid, polümorfism, järgnevusskeemid, kardinaalsus, projekteerimismallid
- Algajaga suhtlev kogenud projekteerija:
  - vaatab läbi lähtekoodi, teeb paarisprogrammeerimist, joonistab tahvlile jutu käigus UML-skeeme, ...

Ühine kogemus – kiirus kasvab

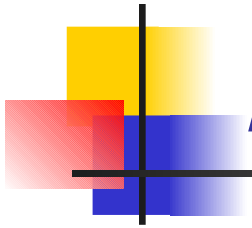
- Kogenud projekteerijad, kes pole varem koos töötanud, omavahel:
  - "... Siia tuleb mingi *Composite*, vaated *Decorator*'itena." "... Peavad olema eraldi *.h*-failidena saadaval." jne.
- Koos töötanud kogenud projekteerijad omavahel
  - "... Protokolli teeme sama transpordikihi otsa, nagu projektis XYZ, aga ilma autentimiseta..."



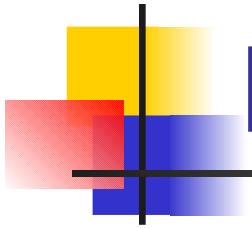
Ühine kogemus – praktika

- Pole aega iga kord suhtlemiseks baasterminoloogiat uuesti leiutada
- Efektiivne suhtlus toimub maksimaalse ühise kogemuse baasil
- Dokumentatsiooni kirjutamine:
  - Lõhe kirjutaja ja lugeja kogemuste vahel tuleb katta
  - Ülejäänu on ressursi raiskamine

Oskuste kolm taset – *Shu-Ha-Ri*

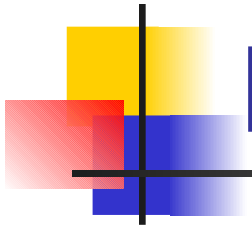


1. **Järgimine** – otsitakse üht protseduuri, mis töötaks
2. **Eraldumine** – hakatakse mõistma meetodi piiranguid, otsitakse alternatiive
3. **Vaba valdamine** – küsimus, kas järgitakse mingit (kirjanduses tuntud) metoodikat, ei puutu enam asjasse. Saadakse aru projekti eesmärgist ja lihtsalt jõutakse selleni



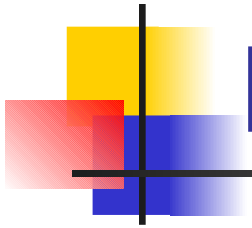
Kolm taset ja metoodikad

- Tase 1
  - The Science of Programming (Gries 1983)
  - The Discipline of Programming (Humphreys 1991)
  - Rational Unified Process detailsed mallid
- Tase 2
  - The Art of Programming (Knuth 1997)
- Tase 2 ja 3 kombinatsioon
  - The Pragmatic Programmer (Hunt 2000)



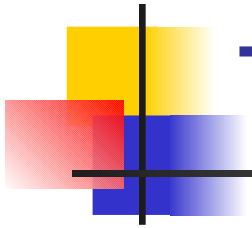
Kolmas tase (I)

- Oskused
  - Sa tee mida iganes, mis toimib
  - Sa pole teadlik, et Sa seda teed
  - Sa teed seda ainult senikaua, kui see toimib
- *Crystal Clear* kolmanda taseme lugeja jaoks kirjeldatuna
  - Pane 4-6 arendajat ja kasutaja ühte ruumi koos arvutite ja tahvlitega. Lase neil iga kahe kuu tagant tarnida kasutajatele töötav ja testitud tarkvara.



Kolmas tase (II)

- Kent Beck XP-st erinevatel tasemetel
  1. Tee kõik nii, nagu kirjas
  2. Pärast seda katseta variatsioone
  3. Lõpuks ära enam mõtle sellele, kas Sa teed XP-d või mitte



Tarkvara ja *Engineering*

- Tarkvaraarendus peaks välja nägema nagu *engineering*?
- Tavatähendus: Tarkvaraarendus peaks sarnanema statistilistest kvaliteedinäitajate abil tehase juhtimisele
- Reaalsus:
  - Silla ehitamisel **ei leiutata** uusi struktuure – kasutatakse *code book*'e
  - Tarkvaraarenduseks ei ole *code book*'e – alustehnoloogiad muutuvad liiga kiiresti

Informatsioonivoog ja projekti kulud

- Elmar vajab infot, mis on Kadril olemas
- Projekti kulud sõltuvad
  - Ajast, mis kulub Elmaril tuvastamaks, et Kadri teab midagi kasulikku
  - Energiast, mille kulutavad Elmar ja Kadri info kandmiseks Kadrilt Elmarile
- Väikestel muutustel suur mõju: paarisprogrammeerijad – 100 vastatud küsimust päevas. Kuluta ainult 1 lisaminut iga küsimuse vastamisele – mis on tulemus?

Kaotatud võimalused – suurim kuluartikkel

- Tuleb mängu ebamugavate suhtluskanalite korral
- Hinda suurendavad komponendid
  - Mõttekäikude taastamine
  - Ebaõnnestumine partneriga kontakti saamisel motiveerib järgmisel korral enam mitte üritama – selle asemel:
    - kasutatakse eeldusi, sh. vigaseid
    - palju aega kulub vigade leidmisele ja kõrvaldamisele

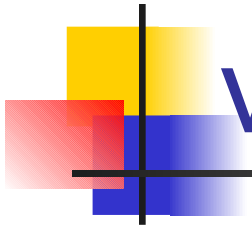
Suhtluskiirus kahanevas järjekorras

- Elmar ja Kadri kasutavad paarisprogrammeerimist
- Elmar ja Kadri istuvad samas ruumis kõrvuti arvutite taga, näevad vabalt teineteise ekraani
- Elmar ja Kadri istuvad samas ruumis vastas seintes teineteise poole seljaga
- Elmar ja Kadri istuvad teineteisest seinaga eraldatud ruumides – kaotatud võimaluste hind võib siin ilmnema hakata
- Elmar ja Kadri istuvad eraldi linnades, isegi eri ajavööndites

Metoodika dokumenteerimine

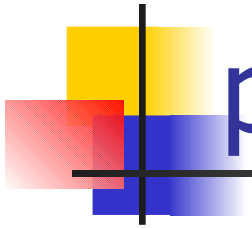
- Metoodika tekstid kasvavad kergesti suureks
  - Nt. “väike” metoodika kirjeldus – 4 rolli, 4 tehist rolli kohta ja 3 tähtpunkti tehise kohta: 68 (4 + 16 + 48) kirjeldatavat olukorda
  - Nt. XP, algselt 200 lehekülge (Beck 1999) läheneb 1000 leheküljele kui selle osadele lisada täiendavaid juhendeid (Jeffries 2000, Beck 2000, Auer 2001, Newkirk 1001)
- Dokumenteerimisel 2 probleemi:
  - dokumenteerimine vs. aru saamine
  - vajadused muutuvad pidevalt

Metoodika kirjelduste vähendamine



- Reaalajas saadavad, loetavad ja muudetavad tehiste näited (projekti plaan, riskiloend, kasutusmall, klassiskeem, testijuhtum, funktsiooni päis, koodinäide)
- Tegevuste kirjeldamisest hoidumine – viidata raamatutele ja kursustele
- Struktureeritud tekstid rollipõhiselt
- Miniatuursed protsessiemulatsioonid

Metoodika projekteerimise probleemid

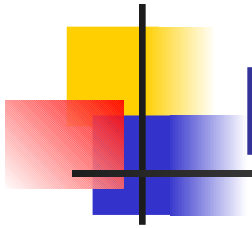


- Inimesed on erinevad – asendatavuse probleem
- Projektid on erinevad – metoodika rakendatavuse probleem
- Pikad silumisperioodid – metoodika testimine kestab kuid ja aastaid
- Tehnoloogiad muutuvad – metoodika peab muutuma

Peamised vead

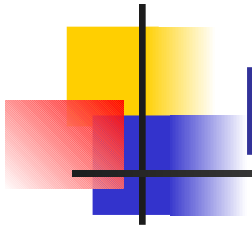
- Üks suurus kõigi projektide jaoks
- Sallimatus
- Raskekaalulisus
- Kaunistused (vihje: “peaks”)
- Läbi proovimata
- On korra kasutatud

Metoodika projekteerimise põhimõtted (I)



1. Interaktiivne näost-näkku suhtlemine on odavaim ja kiireim viis vahetada informatsiooni
2. Liigne metoodika kaal on kulukas
3. Suuremad meeskonnad vajavad raskekaalulisemat metoodikat
4. Kriitilisemad projektid vajavad rohkem tseremooniat

Metoodika projekteerimise põhimõtted (II)



5. Rohkem tagasisidet ja suhtlemist vähendavad vajadust vahesaaduste järgi
6. Distsipliin, oskused ja arusaamine vähendavad vajadust protsessi, formaalsuste ja dokumentatsiooni järgi
7. Efektiivsust saab tõsta mitte-pudelikaeltes

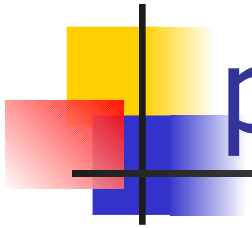
Vajadus metoodika järgi

- Protsessi tutvustamine uutele inimestele
- Inimeste asendamine – kuigi raske, tuleb ette
- Vastutuse määramine – mis kuulub ja mis ei kuulu inimese töö hulka
- Rahastajale mulje avaldamine – jõud, mis sunnib inimesi pakse dokumente kirjutama
- Progressi nähtavalt demonstreerimine
- Loengukursuste inspiratsiooni allikas

Väle protsess – väle, kuid piisav (I)

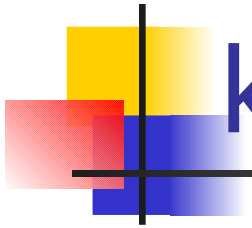
- Kergesti vastuollu sattuvad eesmärgid:
  - Esmane – tarnida tarkvara
  - Teisene – sobivate menetlusviiside (*practices*) rakendamine
- Idealistlikult:
  - Dokumenteerimisest hoidutakse niikaua, kui võimalik ja siis dokumenteeritakse nii vähe, kui võimalik – halb, kui dokumenteeritakse liiga vara ja/või liiga palju
  - Dokumentatsioon lõppeb seal, kus ta võimaldab lugejal astuda järgmise sammu

Väle protsess – väle, kuid piisav (II)



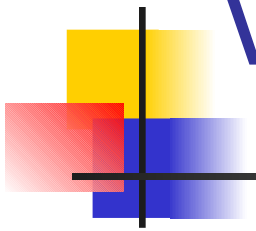
- Kergekaalulisus – tagab manööverdatavuse
- Piisavus – tagab mängus püsimise
- Vale küsimus: kas mingis olukorras saab mõnd väledat meetodit rakendada
- Õige küsimus: kuidas antud olukorras väle olla

Väleda arenduse ideaalne keskkond



- 2..8 inimest samas ruumis
- valdkonna spetsialist samas ruumis
- 1-kuused inkremendid
- Täisautomaatsed regressioon-testid
- Kogenud arendajad
- Toimiv konfiguratsioonihaldus
- Puuduvad olulised pudelikaelad

Virtuaalsed meeskonnad (I)

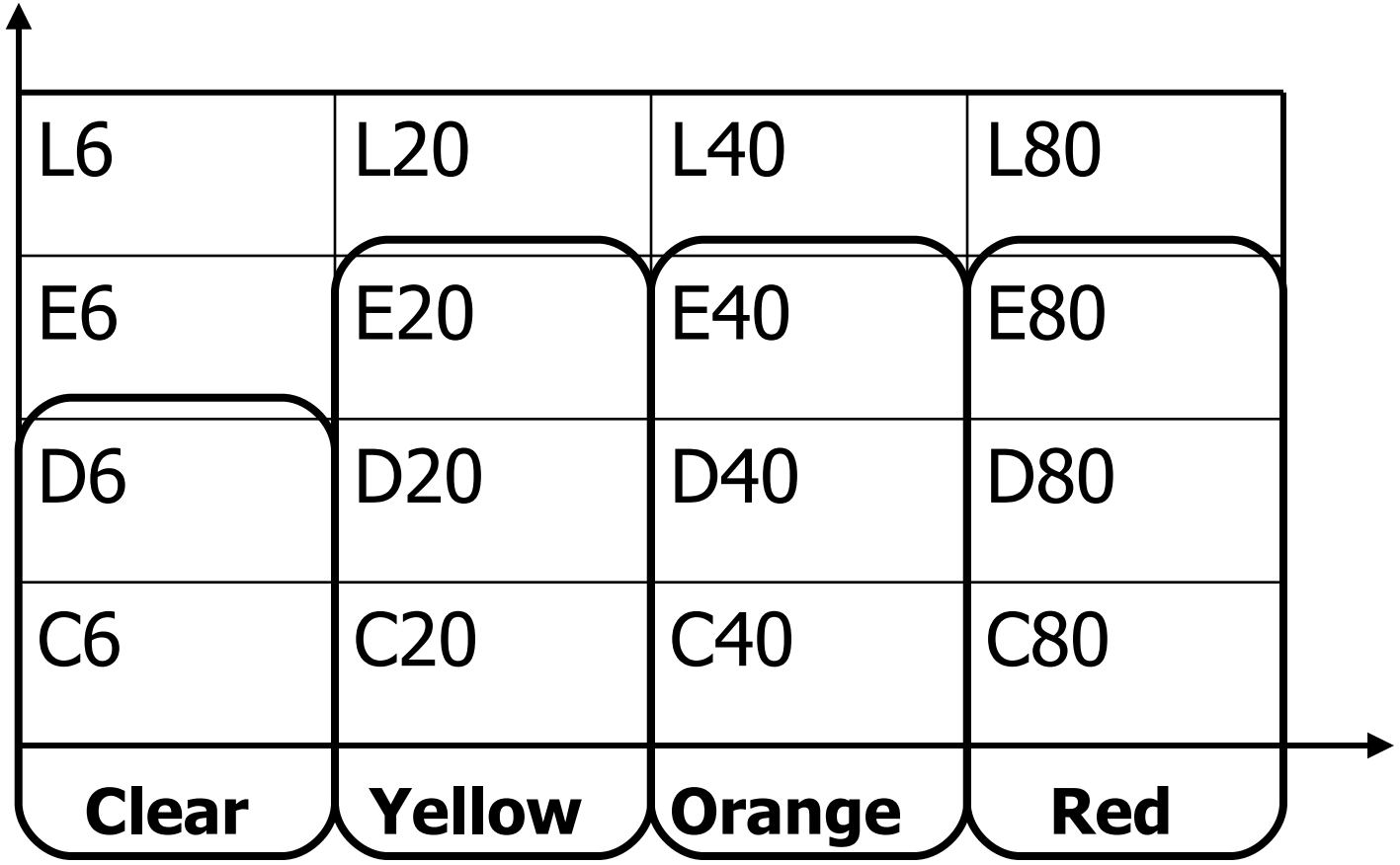


- Suur meeskond, mõned harukontorid – teostatav. Jälgida, et erinevate kontorite esindajad kohtuksid piisavalt tihti
  - Projekteerijad ühes kohas saadavad spetsifikatsioonid programmeerijatele teises kohas – lisaks ebaefektiivsele kommunikatsioonile rohkem konfliktialdis

Virtuaalsed meeskonnad (II)

- Hajus meeskond (palju asukohti, 1..2 inimest per asukoht) – muutub järjest tavalisemaks, aga ei ole efektiivne
- *Open Source* tarkvara arendus – ressursid (inimesed ja aeg) on piiramatud, keskendutakse ainult projekteerimisele ja koodi kvaliteedile

Crystal Methodologies



Agile Alliance (I)

- Manifesti sõnastamine – 17 kergekaaluliste arendusprotsesside evangelisti Snowbird'is, Utah'is veebruaris 2001
- Leppisid kokku 4 asjas:
  - Termin *Agile* – oluline on reageerida muutustele
  - Loetlesid 4 väärtushinnangut
  - Loetlesid 12 põhimõtet väärtushinnangute toetamiseks
  - Rohkemas kokkuleppimisest ei olda huvitatud

Agile Alliance (II)

- Esindatud metoodikad:
  - ASD, XP, Scrum, Crystal, Feature Driven Development, DSDM, Pragmatic Programming.
- Osalejad:
  - Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Kunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, John Kern, Brian Marick, Robert C. Martin, Stephen J. Mellor, Ken Schwaber, Jeff Sutherland ja Dave “Pragmatic” Thomas

The Agile Software Development Manifesto

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.

Process Description Examples – Extreme Programming (XP)

- Plaanimismäng
- Väikesed redaktsioonid
- Metafoor
- Lihtne disain
- Testimine
- Refaktoreerimine
- Paaris-programmeerimine
- Kollektiivne omand
- Pidev integreerimine
- 40-tunnised nädalad
- Klient on kohapeal
- Koodimisstandardid

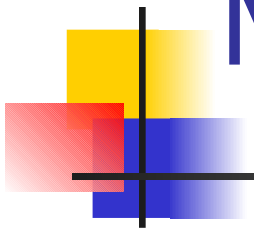
Plaanimismäng

- Äri poole otsustada
  - Ulatus (*scope*)
  - Prioriteedid
  - Redaktsioonide kooslus
  - Redaktsioonide ajad
- Tehnilise poole otsustada
  - Mahuhinnangud
  - Strateegiliste äriotsuste mõjud
  - Protsess
  - Detailne ajagraafik

Väikesed redaktsioonid

- Iga redaktsioon nii väike, kui võimalik
- Sisaldab kõige väärtuslikumaid ärivajadusi rahuldavaid erisusi
- Redaktsioon tervikuna omab mõtet
- 1-2 kuud korraga plaanida on parem, kui 6 kuud või aasta
- Mahukat tarkvara väljalaskval firmal ei pruugi see õnnestuda, aga peaks siiski üritama tsüklit vähendada nii palju, kui võimalik

Metafoor

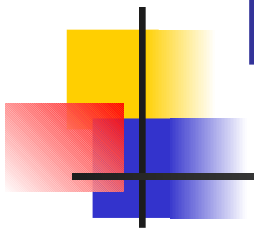


- Iga XP projekti juhttehis
- Kirjeldab süsteemi ja selle väljanägemist
- Areneb kogu projekti jooksul
- Peaaegu arhitektuur

Lihtne disain

- Õige disain
  1. Läbib kõik testid
  2. Ei sisalda dubleeritud loogikat
  3. Esitab kõiki programmeerijatele olulisi otsuseid
  4. Omab vähimat võimalikest klassidest ja meetoditest
- Lihtne disain
  - Võtab ära nii palju elemente, kui saab reegleid 1, 2 ja 3 rikkumata
  - Vastandub soovitusele “teosta tänase jaoks, projekteeri homse jaoks”

Testimine



- Erisust ilma automaatse testita pole olemas
- Programmeerijad kirjutavad *unit* teste
- Kliendid kirjutavad funktsionaalseid teste
- Sedasi tagatakse suutlikus muutusi aktsepteerida
- Teste pole vaja iga meetodi jaoks, vaid *production* meetodite jaoks
- Mõnikord tahad proovida, kas miski on võimalik. Proovid pool tundi. Jah, on. Nüüd viskad kõik minema ja alustad uuesti **testidega**.

Refaktoreerimine

- Erisust lisades vasta alati küsimusele, “kas olemasolevat programmi saab muuta nii, et seda erisust oleks lihtne lisada?”
- Peale erisuse lisamist vasta küsimusele, “kuidas teha programmi lihtsamaks nii, et see läbiks ikkagi kõik olemasolevad testid?”
- Ei refaktoreeri spekulatsioonidel, vaid siis, kui süsteem seda palub – kui süsteem nõuab koodi dubleerimist, tähendab see, et ta palub refaktoreerimist

Paarisprogrammeerimine

- Koodi kirjutatakse kahekesi 1 klaviatuuri ja 1 hiirega 1 arvuti taga istudes
- Hiire ja klaveriga paariline mõtleb parajasti käsiloleva meetodi parimast teostamisviisist
- Vaatlev paariline mõtleb strateegilisemalt
  - Kas kogu idee toimib?
  - Missugused testjuhtumid võivad veel mitte töötada?
  - Kas on võimalik tervet süsteemi lihtsustada nii, et antud probleem lihtsalt kaoks?
- Paaritumine on dünaamiline

Kollektiivne omand

- Igaüks, kes näeb suvalises koodi osas võimalust seda paremaks teha, peab seda tegema
- Vastand teistele mudelitele:
  - omandi puudumine – igaüks muudab vastavalt oma vajadusele sobivust muu olemasoleva koodiga arvestamata. Kood kasvab kiiresti, aga ebastabiilseks.
  - individuaalne omand – antud koodi osa tohib muuta ainult selle ametlik omanik. Kood on stabiilne, aga areneb aeglaselt. Ja kui omanik lahkub, siis ...
- Igaüks vastutab kogu süsteemi eest. Igaüks teab midagi iga osa kohta.

Pidev integreerimine

- Koodi integreeritakse ja testitakse iga paari tunni, kõige rohkem 1 päeva järel
- Integreerimise jaoks on 1 eraldi masin – kui masin on vaba, siis paar, kellel on kood integreerida
  1. Istub selle masina taha
  2. Laeb jooksva redaktsiooni
  3. Laeb oma muudatused (kontrollides ja lahendades kollisioonid)
  4. Käivitab teste ja parandab koodi seni, kuni testid läbitakse 100%
- Eelmine paar jättis maha 100% töötavad testid. Kui me ei saa hakkama, viskame oma muudatused minema ja teeme uuesti

40-tunnised nädalad

- Kent Beck tahab olla
  - Värske ja täis hakkamist igal hommikul
  - Väsinud ja rahul igal õhtul
  - Piisavalt väsinud ja rahul reedel, et muretult mõelda järgneval kahel päeval millegi muu, kui töö peale
  - esmaspäeval tagasi tulla täis tuld ja ideid
- Erinevad inimesed suudavad kontsentreeruda 35-45 tundi, keegi ei suuda 60 tundi mitu nädalat järjest töötada ja olla sealjuures ikka veel värske, loominguline, hoolikas ja rahul
- Ületöötunnid projektis on tõsise probleemi sümptom
- XP reegel – teist nädalat järjest ületunnitööd ei tehta – probleemi, mis seda põhjustab, ei lahenda ületundidega

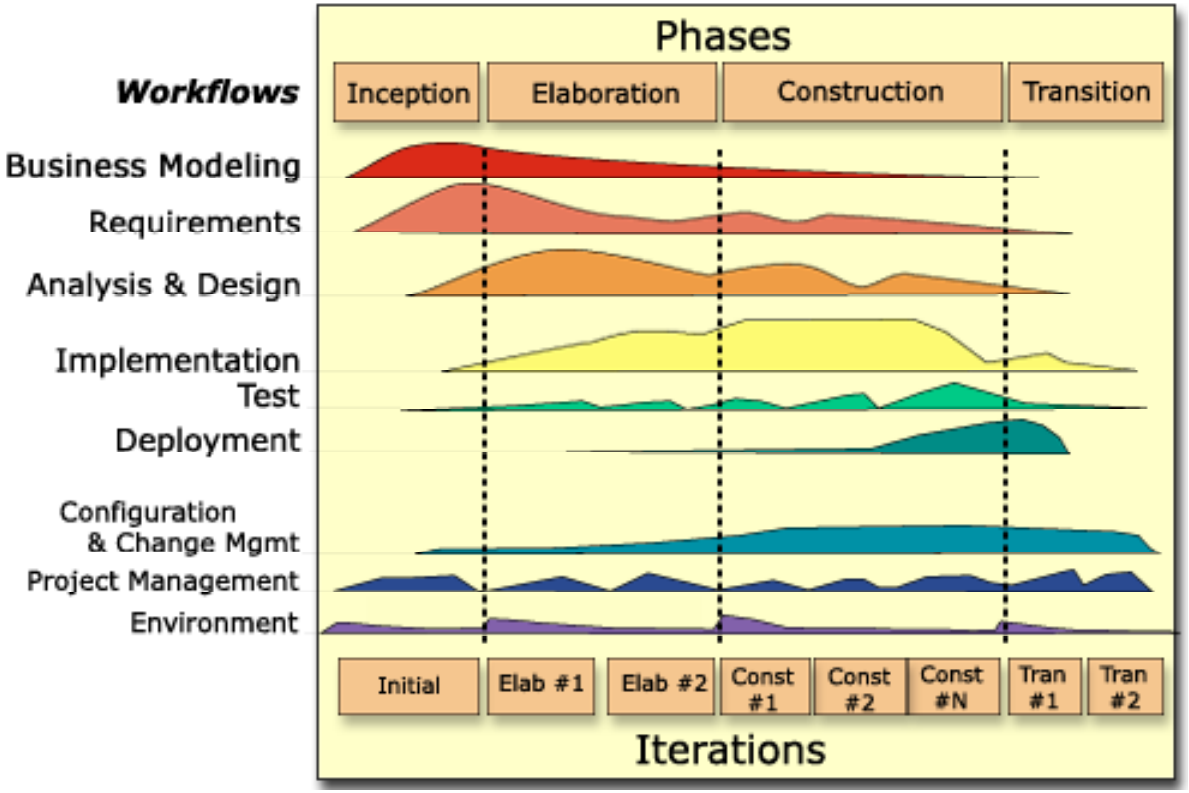
Klient on kohapeal

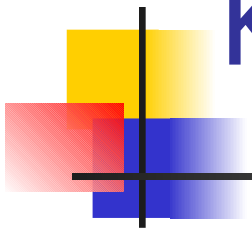
- “Tegelik” klient istub koos meeskonnaga, valmis
  - Vastama küsimustele
  - Lahendama vaidlusi
  - Määrama väikese ulatusega prioriteete
- Tuleb otsustada, kas väärtuslikum on rutem hästitöötav või üks töötav tööline. Kui viimane, siis pole äkki süsteemi vaja? Pealegi, ega programmeerijad ei suudagi iga nädal 40 tundi küsimusi genereerida.

Koodimisstandardid

- Muuhulgas võiks sisaldada *Once and Only* reegliga kooskõlalist vähima võimaliku töö nõuet

Process Description Examples – Rational Unified Process (RUP)





Küsimusi?