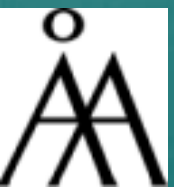
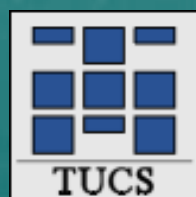
Miten tehdään virheettömiä ohjelmia

Ralph-Johan Back Joakim von Wright
Åbo Akademi TUCS
Formaalit menetelmät ohjelmoinnissa ryhmä

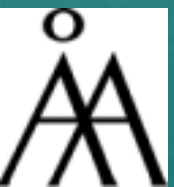


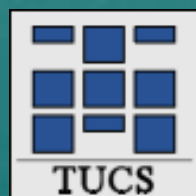


Esitelmät

Ohjelmoinnin matematiikka
Ralph Back

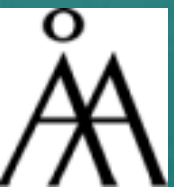
Matematiikan ohjelmointi
Joakim von Wright

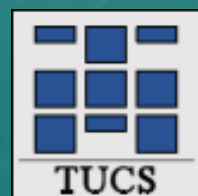




Ohjelmoinnin matematiikka

Ralph-Johan Back





Sisältö

- Ohjelmiston luotettavuus
- Formaalit menetelmät
- Ohjelman oikeellisuuden todistaminen
- Inkrementaalinen ohjelmointi

Tietokoneet, kommunikointi ja ohjelmistot

Ohjelmistot keskeisessä roolissa

- Ohjelmistot muodostavat suurimman osan tietojärjestelmistä
- Yhteiskunta erittäin riippuvainen ohjelmistoista
  - sulautetut järjetestelmät, hallinto, pc sovellutukset, ...
- Ohjelmistot ohjaavat kriittisiä toimintoja:
  - lentoliikennettä, sairaalalaitteita, teollisia prosesseja, ...
- Ohjelmistojen laadulla on vahva vaikutus yhteiskunnan ja yksityisten toiminnan luotettavuuteen, turvallisuuteen, tehokkuuteen ja tulosten laatuun

Mikä on hyvä tietokoneohjelma

- monipuolinen, tekee sitä mitä minä haluan
- toimii käytössä olevalla tietokoneella ja käyttöjärjestelmässä
- helppokäyttöinen, selkeä käyttöliittymä
- toimii tehokkaasti ja reagoi nopeasti
- toimii hyvin muiden ohjelmien kanssa
- luotettava, ei sisällä paljon virheitä
- käyttäytyy hyvin myös virhetilanteissa
- jne jne

Ohjelmien laadun kehitys

- Kehitys on kulkenut kohti parempilaatuisia ohjelmia
  - helppokäyttöisempiä ohjelmia, tehokkaampia, monipuolisempia, ...
- Paitsi: ohjelmistojen luotettavuus ei ole parantunut, vaan lähinnä huonontunut.

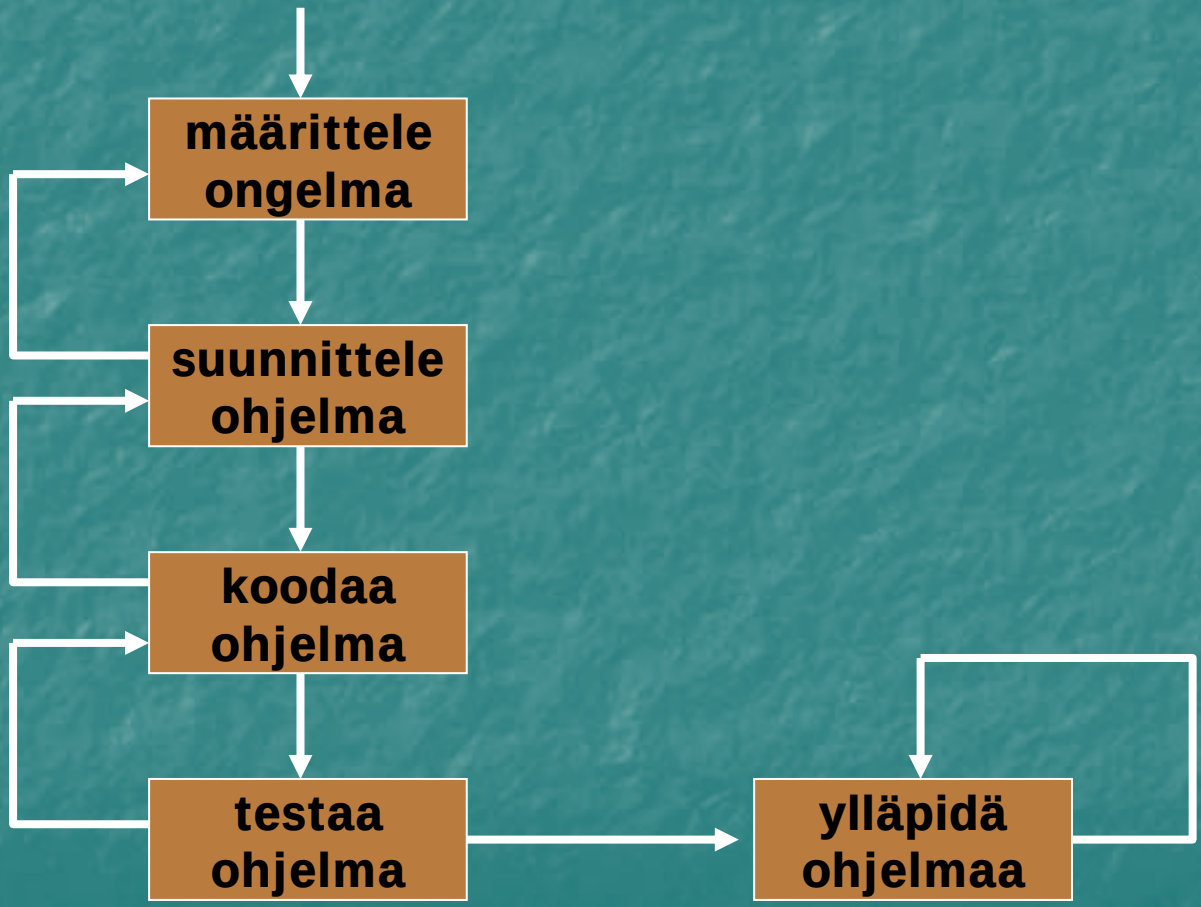
Mistä ohelmistojen virheet johtuvat

- Perussyy on ohjelmien kompleksisuuden kasvu
  - Ohjelmistojen koko on jatkuvasti kasvanut
  - Nykypäivän ohjelmistot ovat valtavan suuria (100 000 – 10 000 000 riviä koodia)
  - Ohjelmistot ovat hyvin monimutkaisia, ja niiden osien väliset vuorovaikutukset ovat vaikeasti hallittavissa
- Muut laatukriteerit (monipuolisuus, käytettävyys, ...) kasvattavat ohjelmien kokoa
- Markkinavoimat suosivat huonoja ohjelmistoja
  - Ohjelmistojen nopea valmistuminen (time to market) tärkeintä
  - Kuluttajat eivät vaadi luotettavia ohjelmia
  - Ohjelmistoille ei esim. anneta luotettavuustakuuta

Ohjelmistoteknologian nykytaso

- Ohjelmistotekniikka on kehittymätöntä verrattuna muihin teknologian aloihin
  - Perustutkimusta ei paljon hyödynnetä teollisessa ohjelmistotuotannossa
- Ohjelmointi tarvitsee toisenlaisen tieteellisen perustan kuin tavanomainen tekniikkaan
  - mallit ovat useimmiten diskreettejä, eivät jatkuvia
  - tärkein työkalu on logiikka ja algebra, ei analyysi
  - suurin osa tarvittavasta matematiikasta on kehitetty viimeisten 50 vuoden aikana

Ohjelmointiprosessi tänään



Testaaminen

- Suoritetaan ohjelma esimerkkidatalla, ja tarkistetaan ovatko tulokset oikeita
- Testauksessa joudutaan suorittamaan ohjelma hyvin monella eri datalla, jotta löydettäisiin mahdollisimman monta virhettä
- Testaamisella ei voida poistaa kaikki ohjelmavirheet, ainoastaan ne jotka löytyvät esimerkkien avulla

Virheiden poistaminen testaamalla

- Prosessin alkuvaiheessa tehdyt ohjelmavirheet pyritään loppuvaiheessa poistamaan testauksella.
- Testaaminen käy yhä työläämmäksi, mitä suuremmiksi ohjelmistot tulevat
- Testaaminen muodostaa yhä suuremman osan ohjelmistokehityksen kustannuksista
- Testaamisella ei kuitenkaan voida saavuttaa haluttu ohjelmistojen luotettavuuden taso

Formaalit menetelmät ohjelmoinnissa

- Formaalit menetelmät muodostavat osaksi vaihtoehdon testaamiselle, osaksi täydentävät testaamista
- Ensisijainen tavoite on parantaa ohjelmien luotettavuutta
- Ohjelman ominaisuuksia tutkitaan analysoimalla sitä matematikan ja logiikan avulla
- Esimerkki: todistetaan matemaattisesti että ohjelma toimii oikein (ts ei sisällä virheitä)

Åbo Akademin ohjelmointitekniikan tutkimusryhmä

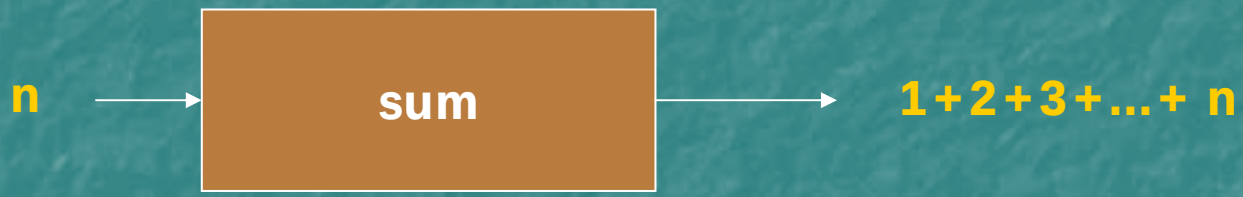
- Åbo Akademin tietojenkäsittelyopin laitos 1984--
- Keskittynyt formaalien menetelmien tutkimiseen ja kehittämiseen
- Suomen Akatemia tutkimuksen huippuyksikkö 1995 -99 (osana TUCSia) ja 2002 – 2007 (omana yksikkönä)
- Senioritutkijat
  - Ralph-Johan Back (johtaja)
  - Johan Lilius
  - Kaisa Sere
  - Joakim von Wright
- Ryhmässä noin 50 tutkijaa (noin puolet ulkomaalaisia)

Tutkimussuunnat

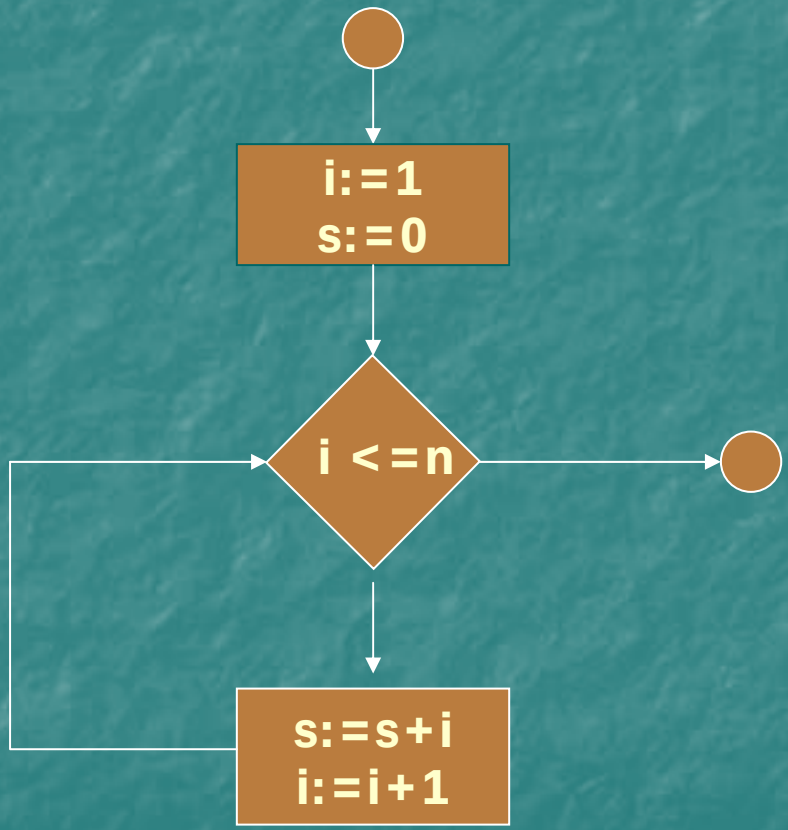
- Ohjelmoinnin matemaattinen perusta
  - tarkennuskalkyyli
  - aktiojärjestelmät
  - ohjelmistoarkkitehtuuri
- Uudet ohjelmointiparadigmat
  - rinnakkaiset ja hajautetut järjestelmät
  - vlsi-suunnittelu
  - olio-ohjelmointi
  - ohjelmistojen suunnittelu ja vaatimusmäärittelyt
- Ohjelmoinnin työkalut
  - ohjelmointiympäristöt
  - mekaaninen todistaminen
  - mallintarkistaminen
- Ohjelmointiprosessit

Esimerkki ohjelman oikeellisuuden todistamisesta

- Ohjelma joka laskee yhteen ensimmäiset n kokonaislukua: 1 + 2 + 3 + ... + n



Sum-ohjelman toimta



Ohjelmakoodi

```
def sum(n):
    i := 1
    s := 0
    while i <= n:
        s := s+i
        i := i+1
    return s

print sum(10)
```

laske 1+2+3+...+n

i on kierroslaskuri

s on osasumma

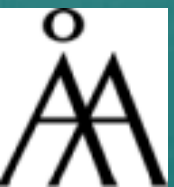
iteroidaan kun i <= n

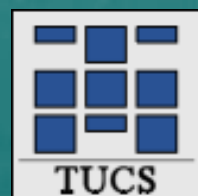
lisätään uusi i summaan s

kasvatetaan i:tä

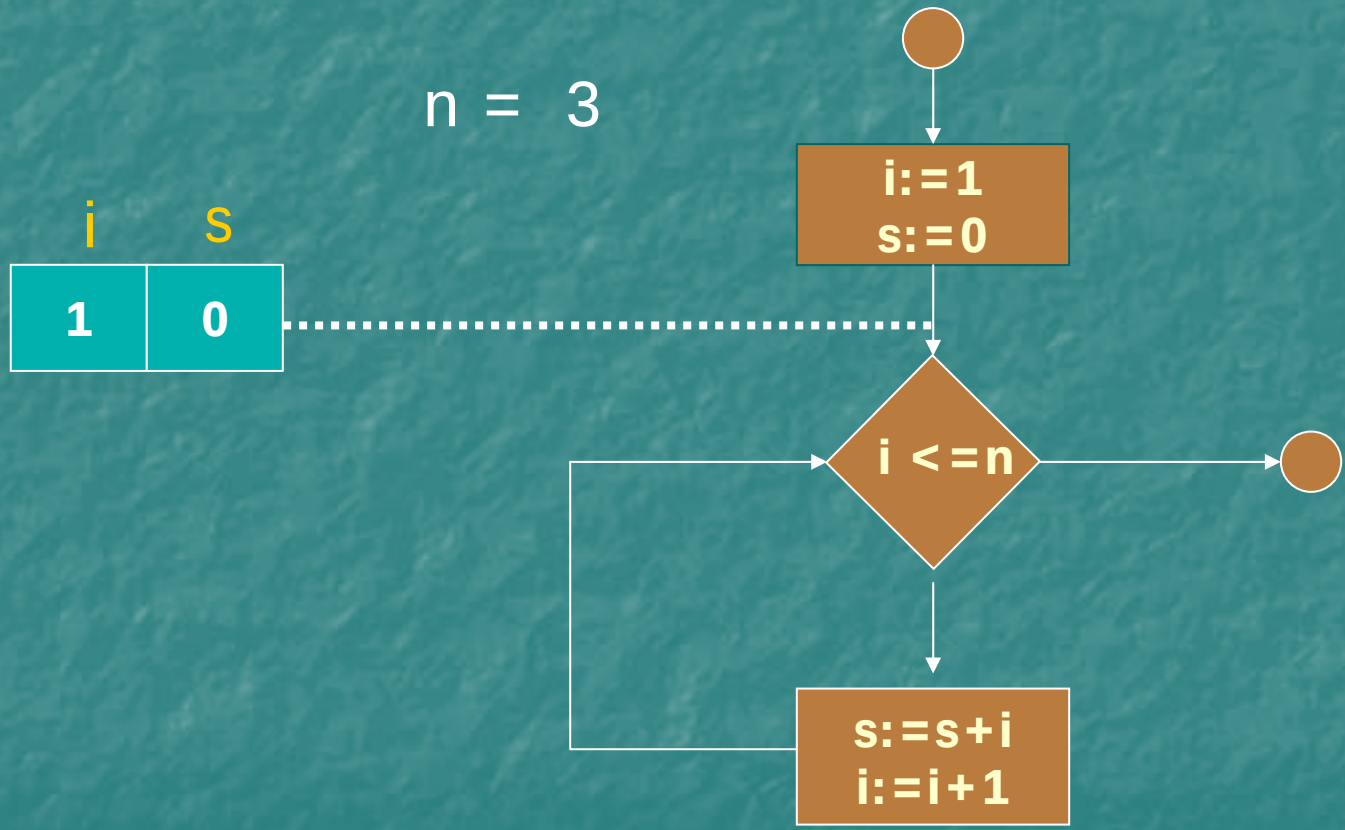
kun i = n+1, lopetetaan,
palautetaan summa s

lasketaan 1+2+3+...+10





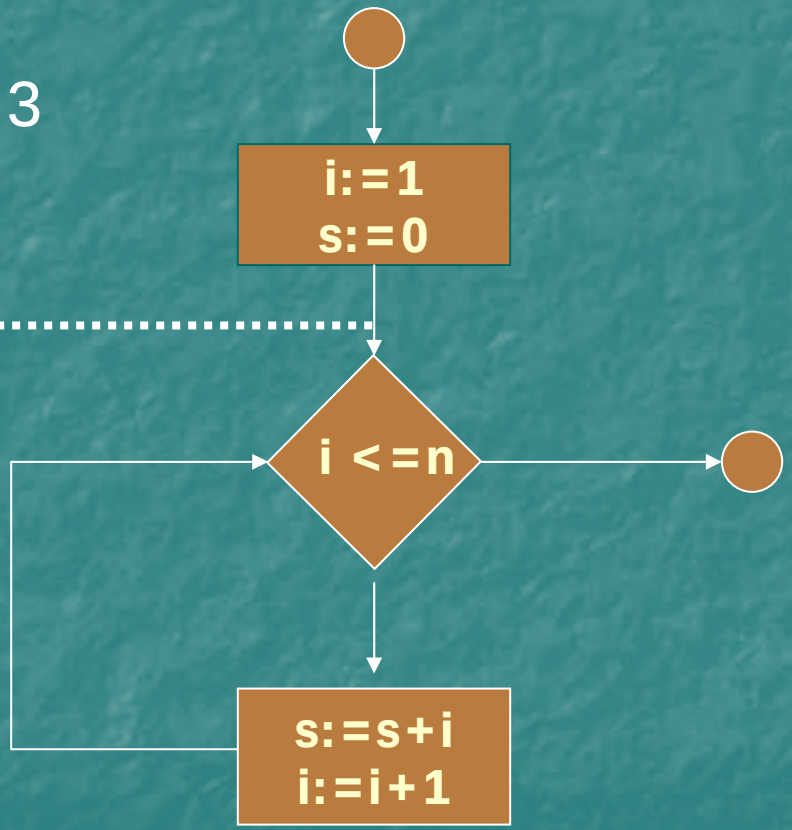
Ohjelman suoritus



Ohjelman suoritus

n = 3

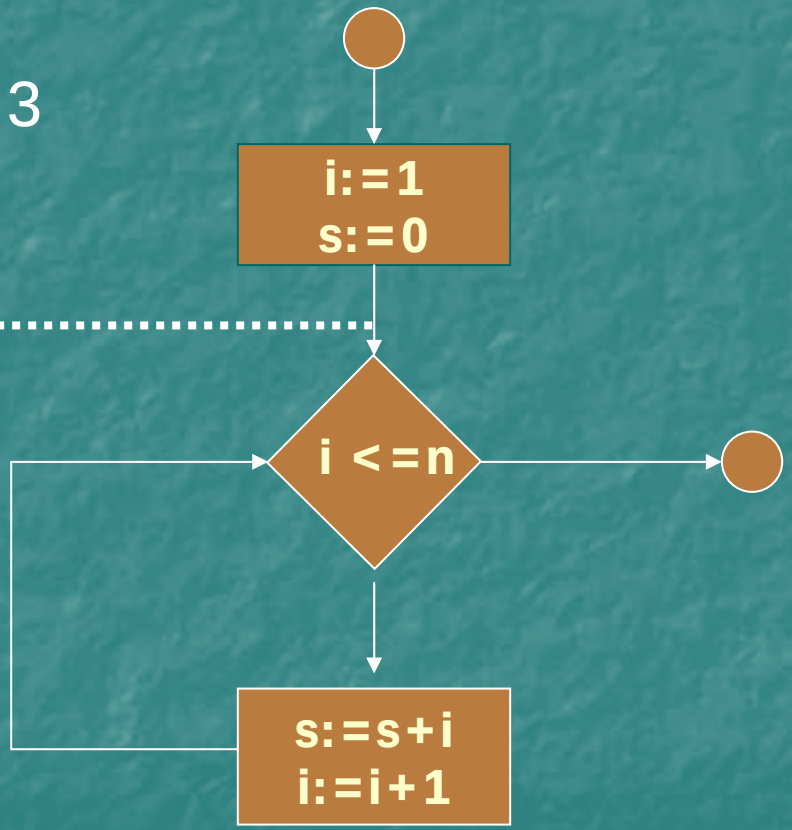
i	s
1	0
2	1



Ohjelman suoritus

n = 3

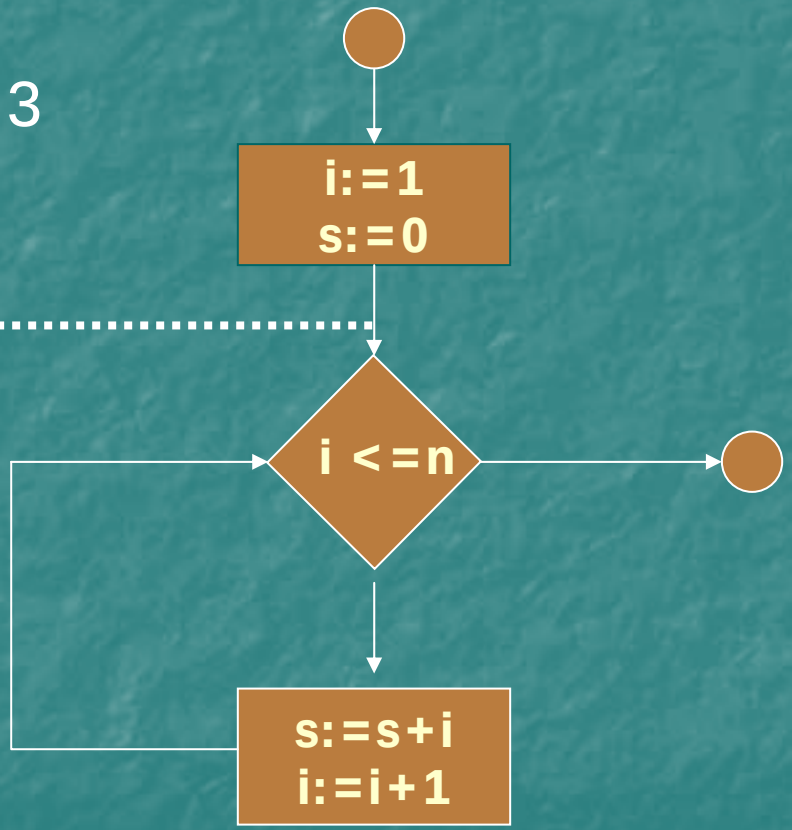
i	s
1	0
2	1
3	3



Ohjelman suoritus

n = 3

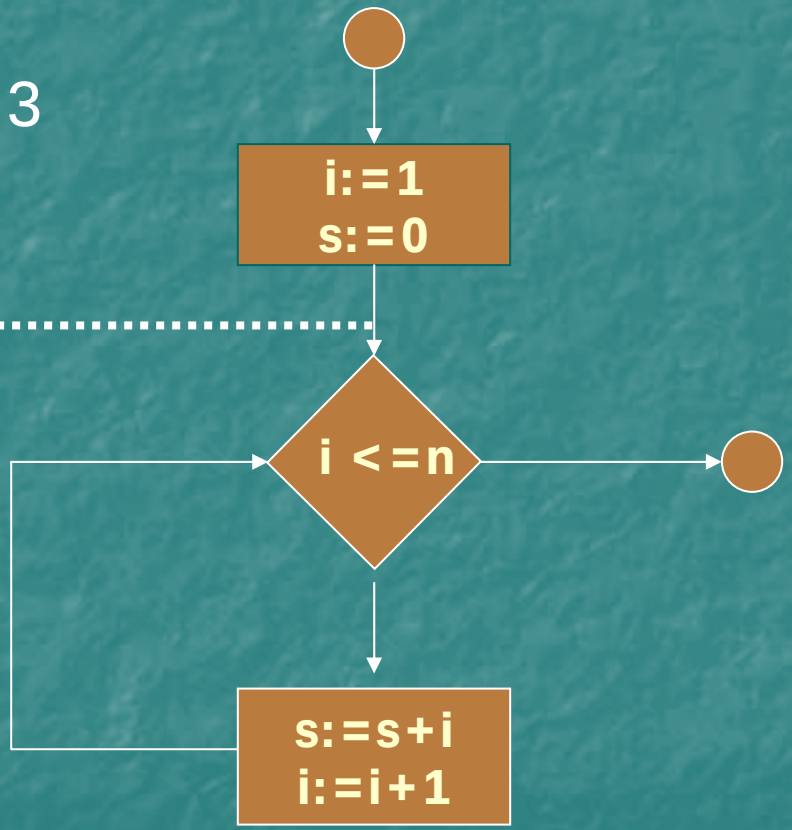
i	s
1	0
2	1
3	3
4	6



Induktiohypoteesi

n = 3

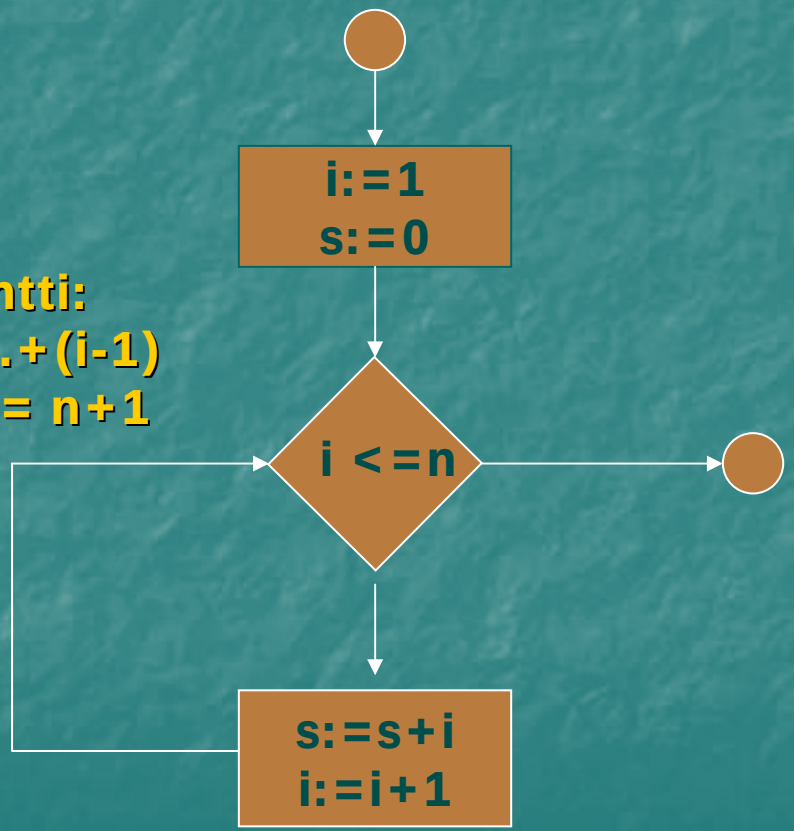
i	s
1	0
2	1
3	3
4	6



s =1+...+(i-1)
1<=i <= n+1

Väittämät

alkuehto:
0 <= n



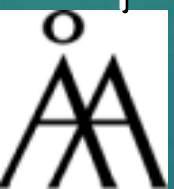
invariantti:
s =1+...+(i-1)
1<=i <= n+1

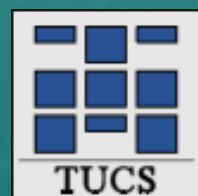
loppuehto:
s = 1+...+n

Väittämät ohjelmatekstissä

```
def sum2(n):                    # 0 <= n
    i := 1
    s := 0
    while i <= n:               # s =1+...+(i-1) & 1<=i <= n+1
        s := s+i
        i := i+1
    return s                    # s = 1+...+n


print sum2(10)
```

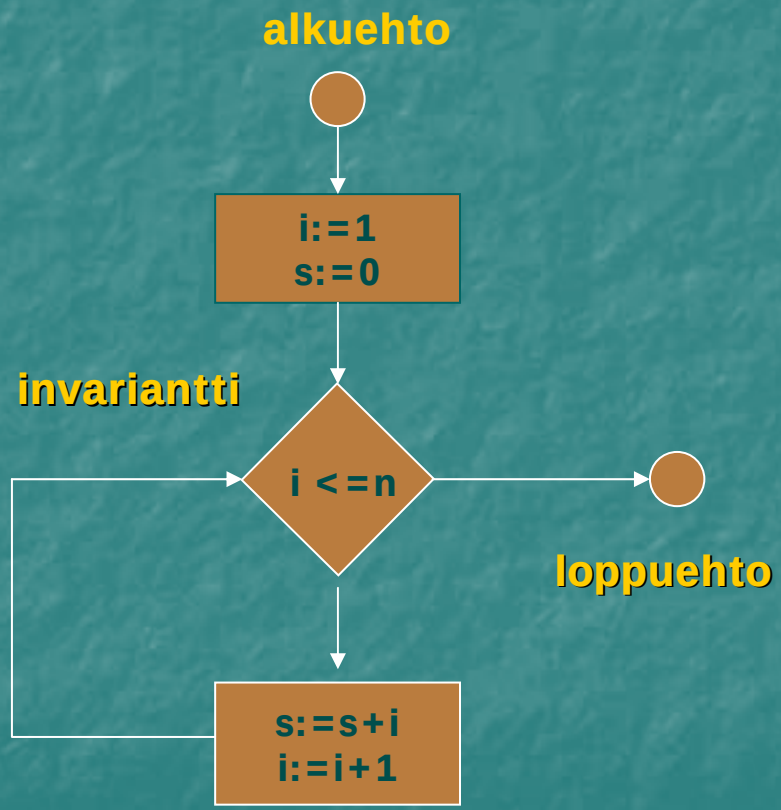




Sum-ohjelman oikeellisuus

- Invariantti on voimassa silmukan alussa
- Invariantti pysyy voimassa silmukkaa kierettäessä kerran
- Invarianttista seuraa loppuehto, kuin silmukka pysähtyy
- Silmukan suoritus pysähtyy

Sum-ohjelma toimii oikein



Oikeellisuuden todistamisen ongelmat

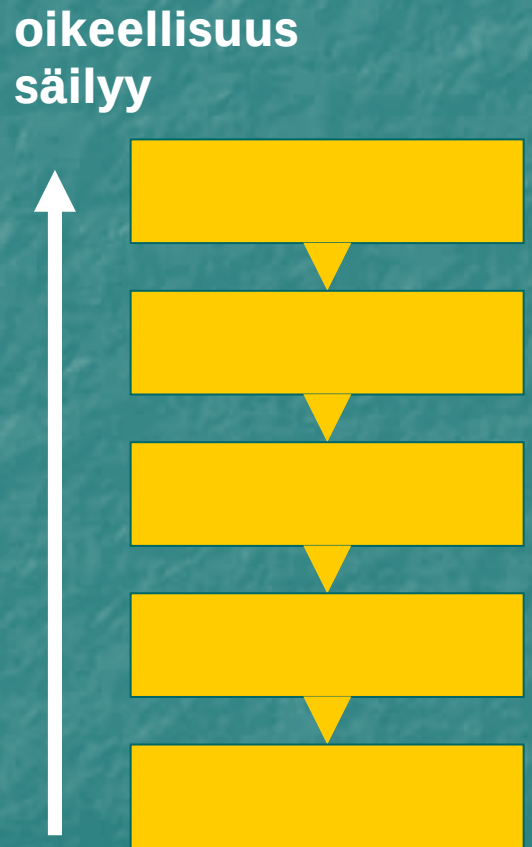
- Todistaminen vaatii matemaattista ajattelutapaa
- Kun ohjelman koko kasvaa, niin todistaminen tulee yhä vaikeammaksi
- Todistus itsessään voi sisältää virheitä
- Usein vaikeata kuvata ennakkoehto ja loppuehto täsmällisesti
- Invariantin keksiminen hankalaa (vastaa induktiohypoteesin muodostamista)
- Todistukset rutiininomaisia, eivät erikoisen kiinnostavia tai matemaattisesti haastavia

Todistamisen lisäongelma

- Käytännössä ohjelmat sisältävät virheitä, joten niitä ei voida (eikä pitäisikään voida) todistaa oikeiksi
- Todistuksen mahdottomuus näyttää virheen paikan
- Ennen oikeellisuuden todistamista on kaikki virheet paikallistettava ja korjattava
- Todistaminen ja testaaminen antavat kaksi erilaista tapaa hakea virheitä ohjelmasta
- Päätelmä: valmiiden ohjelmien testaaminen hankalaa eikä erikoisen mielekästä.

Oikein toimivien ohjelmien rakentaminen

- Oikeellisuutta ei voida lisätä jälkeenpäin
- Oikeellisuus on rakennettava sisään alusta lähtien
- Ohjelma on rakennettava pala kerrallaan, siten että oikeellisuutta ei koskaan menetetä
- Ryhmämme on keskittynyt tällaisten inkrementaalisten oikeellisuuden säilyttävien ohjelmointimentelmien kehittämiseen ja tutkimiseen



Inkrementaalinen ohjelmointi

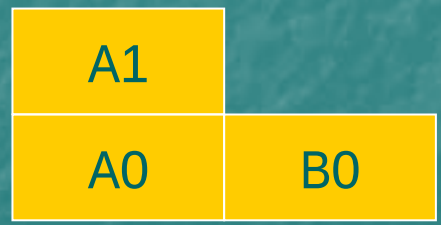
- Rakennetaan ensimmäinen pieni palikka A0
- Todistetaan se oikeaksi

Inkrementaalinen ohjelmointi

- Rakennetaan seuraava palikka B0
- Todistetaan se oikeaksi

Inkrementaalinen ohjelmointi

- Parannellaan A0:aa, lisäämällä sen käyttökelpoisuutta
- Todistetaan uusi palikka A1 oikeaksi
- Todistetaan, että A1 säilyttää A0:n toiminnot



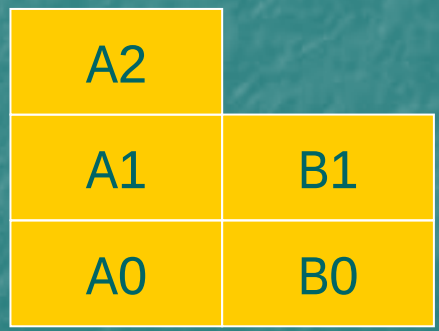
Inkrementaalinen ohjelmointi

- Parannellaan B0:aa, lisäämällä sen käyttökelpoisuutta
- Todistetaan uusi palikka B1 oikeaksi
- Todistetaan, että B1 säilyttää B0:n toiminnot

A1	B1
A0	B0

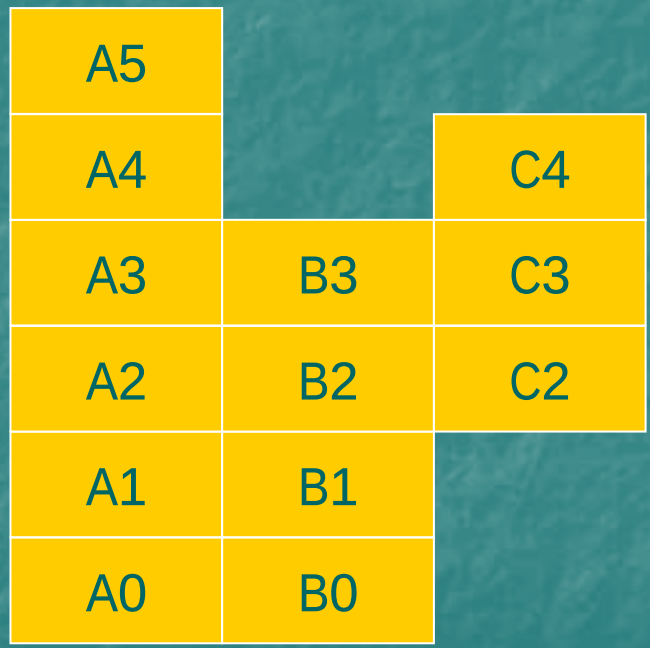
Inkrementaalinen ohjelmointi

- Parannellaan taas Ata



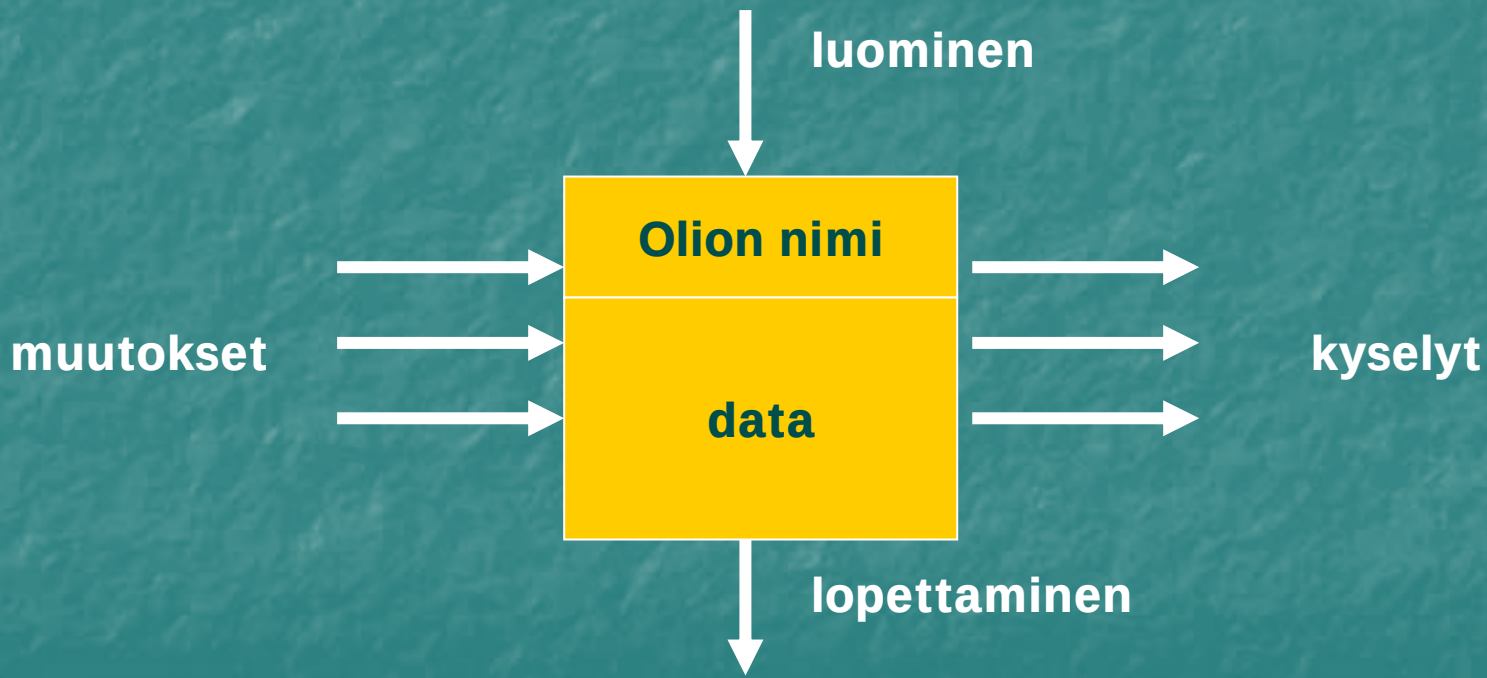
Inkrementaalinen ohjelmointi

- jne, kunnes ollaan saatu rakennettua ohjelmaan kaikki toivottu toiminnallisuus



Olipohjainen ohjelmointi

- Olio on sopiva rakennuspalikka:



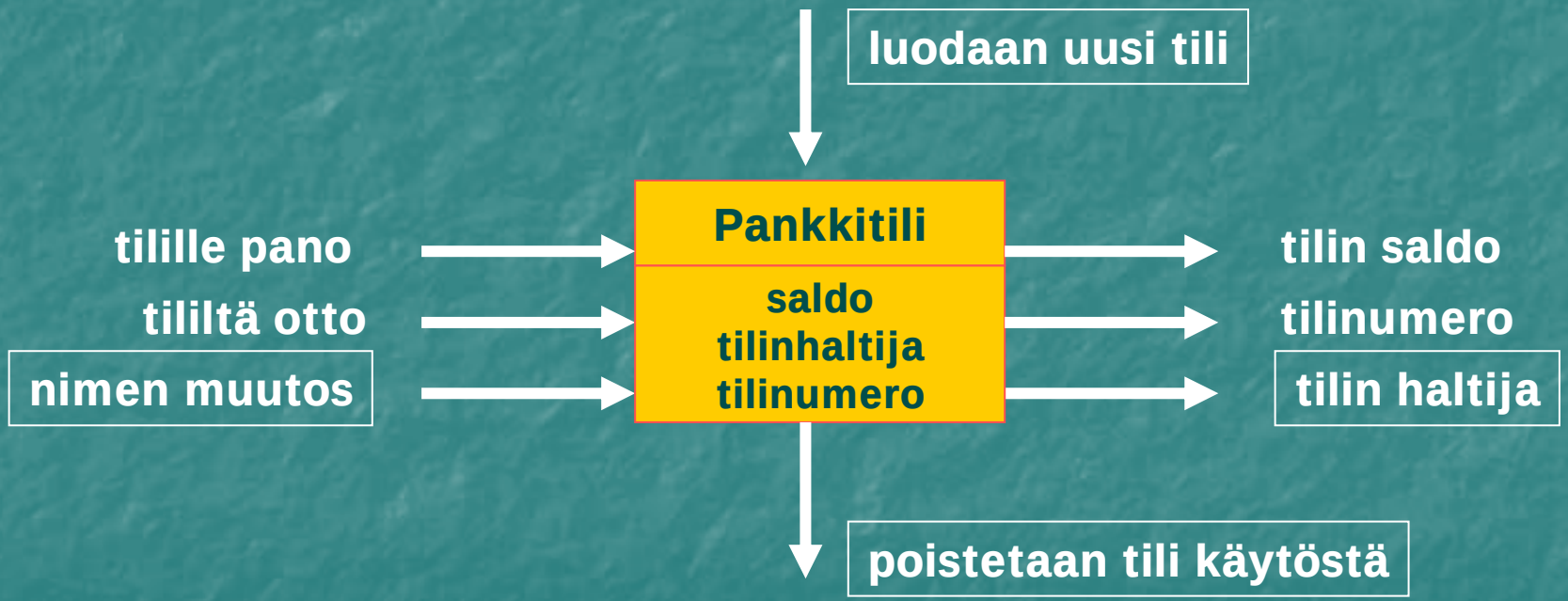
Pankkitili oliona

- Data:
  - Saldo, tilinumero ja tilin haltija
- Luominen:
  - Luodaan pankkitili (tilinumero ja asiakkaan nimi)
  - Tilillä ei aluksi ole yhtään rahaa
- Muutokset:
  - Pankkitilille voidaan panna rahaa
  - Pankkitililtä voidaan ottaa rahaa
  - Muutetaan asiakkaan nimi
- Kyselyt:
  - Paljonko rahaa on tilillä juuri nyt
  - Kuka on tilinhaltija, mikä on tilinumero
- Lopettaminen:
  - Pankkitili poistetaan käytöstä

Pankkitili

pankin toiminnat

asiakkaan toiminnat



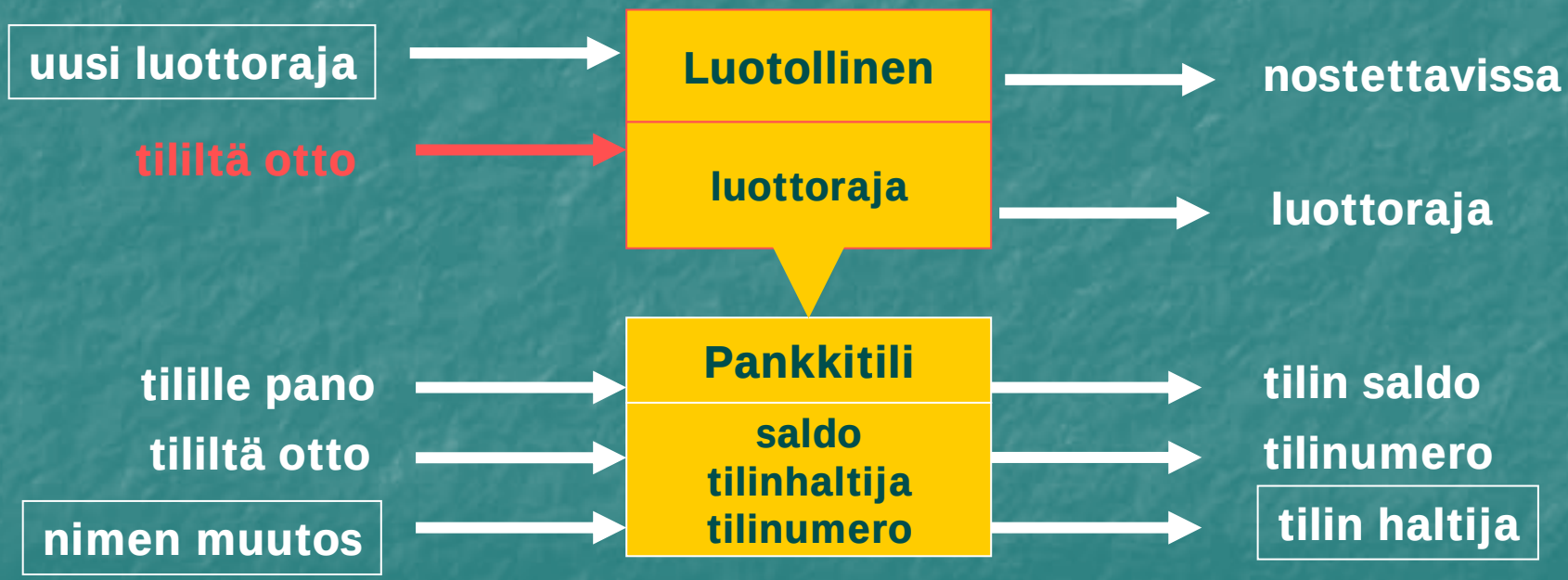
Pankkitili ohjelmana

```
class Account:
    def __init__(self, name, account):
        self.saldo := 0
        self.name := name
        self.account := account
    def insert(self, sum):
        self.saldo := self.saldo + sum
    def remove(self, sum):
        self.saldo := self.saldo - sum
    def getSaldo(self):
        return self.saldo
    .......
```

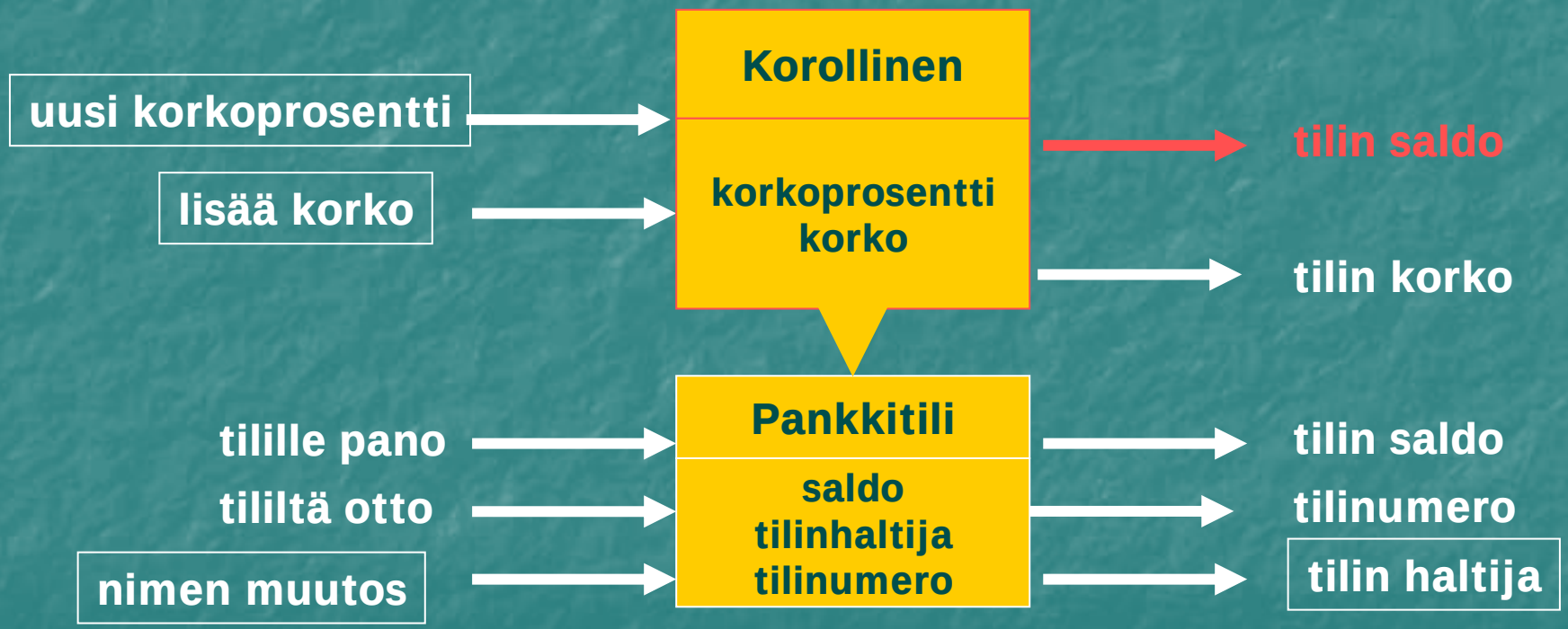
Pankkitilin lisäominaisuuksia

- Koron maksaminen
- Luotollinen pankkitili
- Pankkitiliin on liitetty tunnusluku
- Tapahtumien rekisteröinti
- Automaattinen laskujen maksu eräpäivänä
- Tilillä voi olla eri valuuttoja
- jne, jne ...

Luotollinen pankkitili



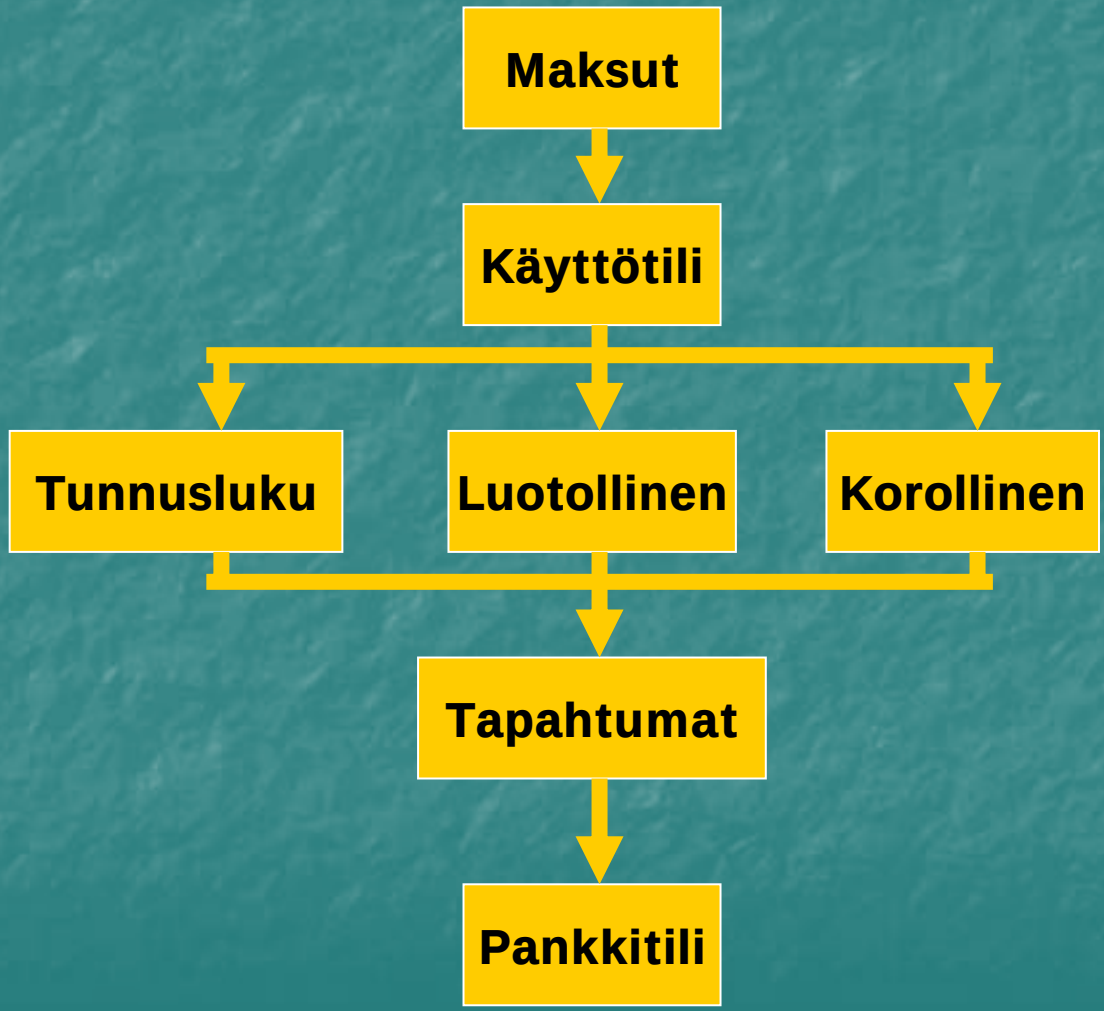
Korollinen pankkitili



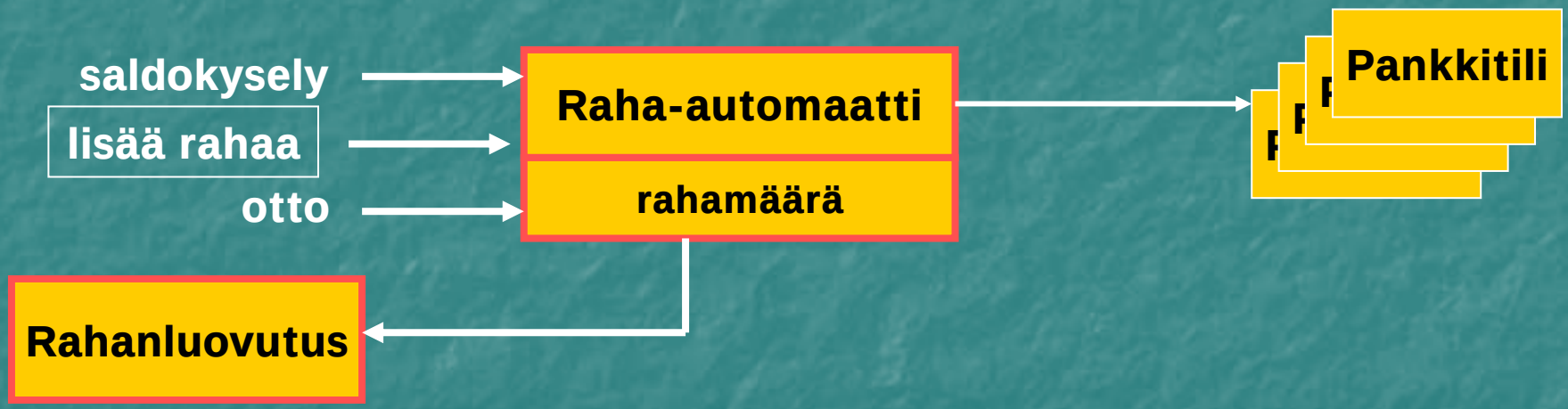
Todistaminen

- Uuden olion on toteutettava sille asetetut vaatimukset
- Uuden olion on säilytettävä alkkuperäisen olion ominaisuudet
- Esim.
  - nimenmuutos, tilinhaltija toimii Luotollinen-oliossa samalla tavalla kuin Pankkitili-oliossa
  - tililtä otto huomio luottorajan, toimii samoin kuin aiemmin jos luottoraja on 0.

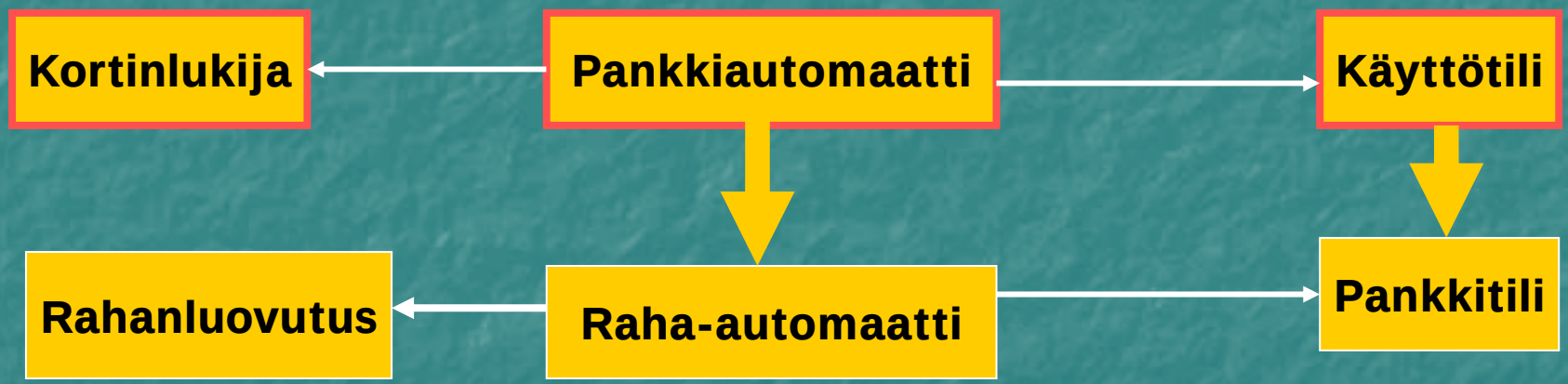
Pankkitilin rakentaminen



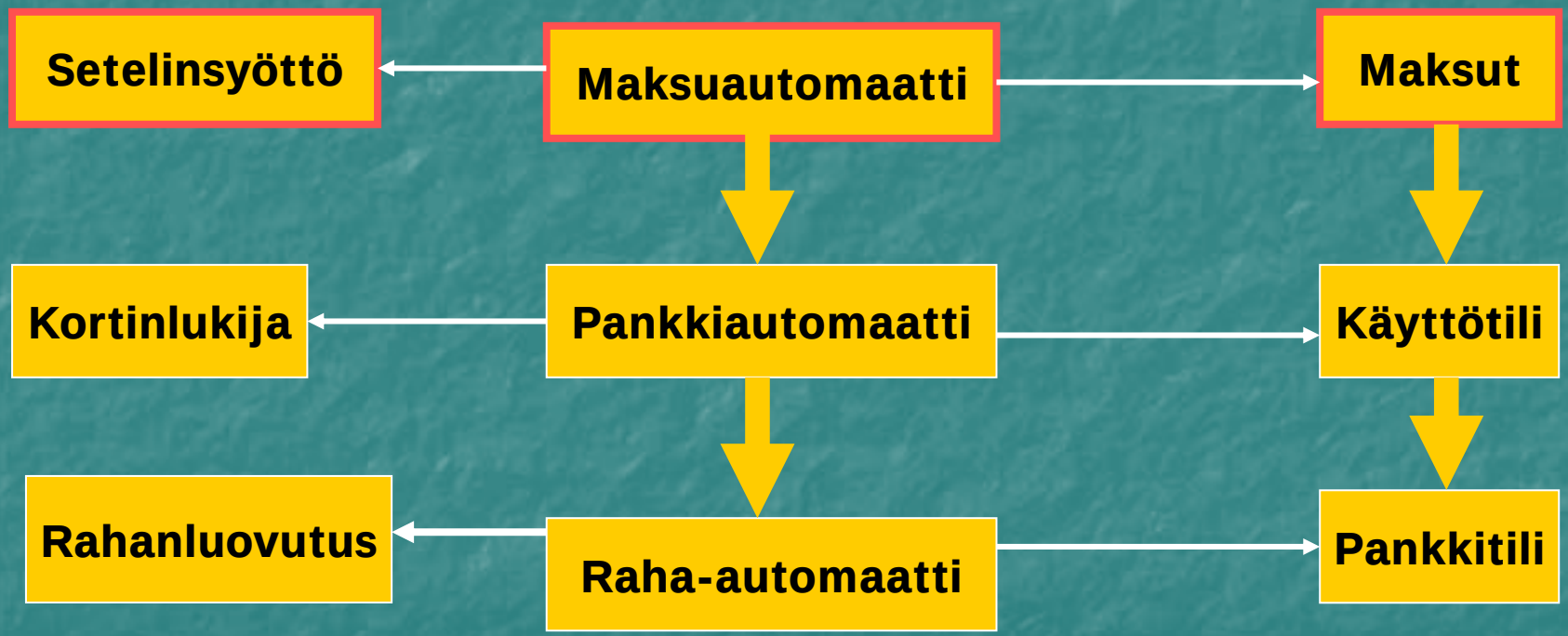
Raha-automaatti



Pankkiautomaatti



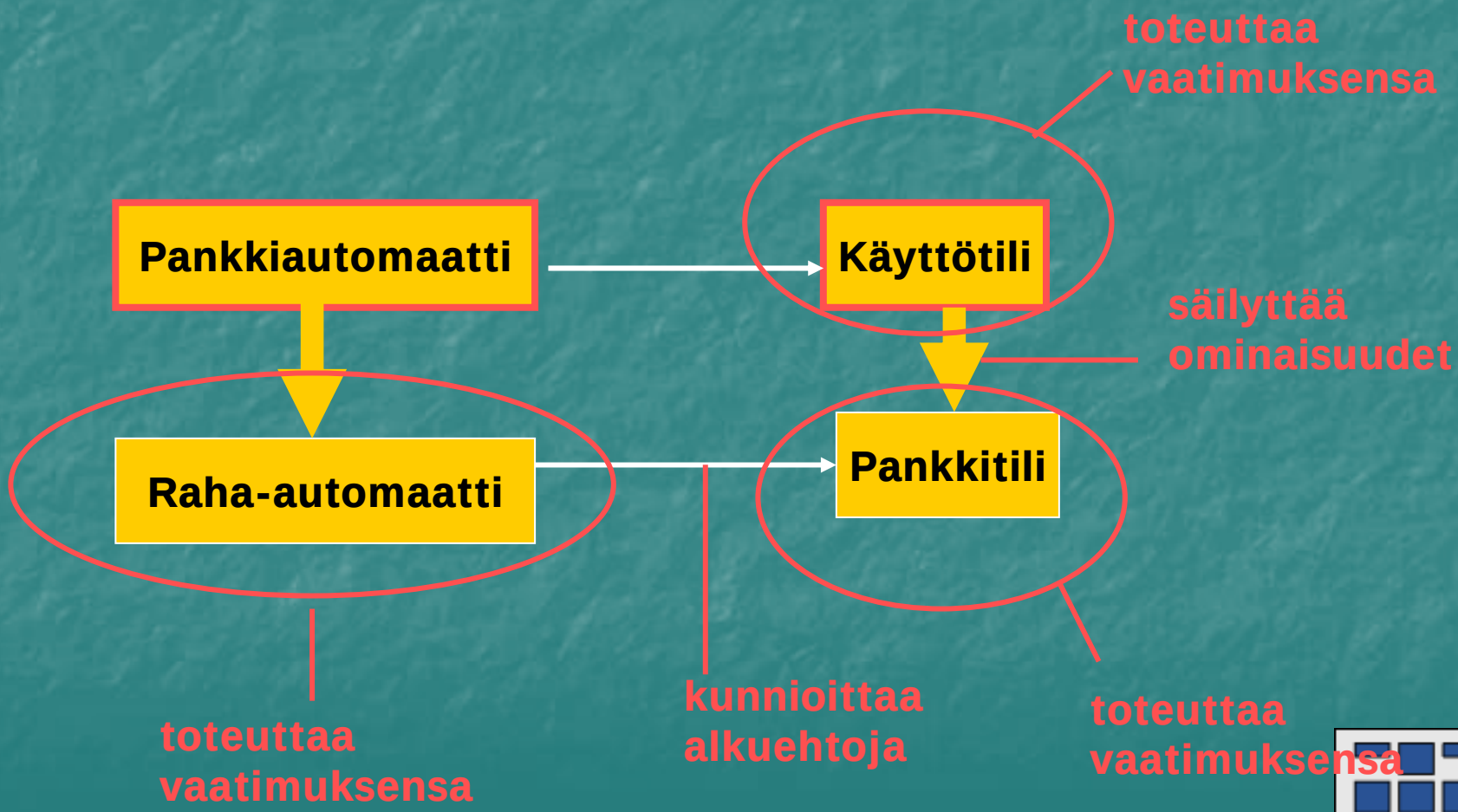
Maksuautomaatti



Miten todistetaan oikeellisuus

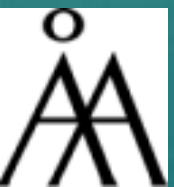
- Perusoliot on todistettava oikeiksi
- Olion laajennus on todistettava oikeaksi
  - laajennuksen on toteuttettava sille asetettavat vaatimukset
  - laajennuksen on säilytettäv aikaisemmat ominaisuudet
- Olion käyttö on tapahduttava oikein
  - olion menetelmää saa vain kutsua jos menetelmän alkuehto on voimassa

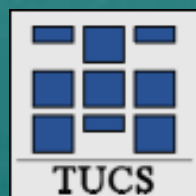
Todistus


toteuttaa
vaatimuksensa
Pankkiautomaatti
Käyttötili
säilyttää
ominaisuudet
Raha-automaatti
Pankkitili
toteuttaa
vaatimuksensa
kunnioittaa
alkuehtoja
toteuttaa
vaatimuksensa

Inkrementaalinen ohjelmointi

- Vaatii uutta lähestymistapaa ohjelmointiin
- Oikeellisuus on tarkistettava jatkuvasti ohjelmaa rakennettaessa
- Ohjelman arkkitehtuurin on tuettava inkrementaalista rakentamista
- Ohjelmointiprosessin on myös tuettava tätä lähestymistapaa
- Testaaminen yhä tarpeellinen, tapahtuu jokaisen laajennuksen yhteydessä



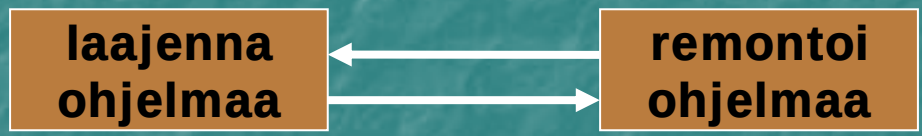


Rakenteen remontointi

- Kun suurta kokonaisuutta rakennetaan pala kerrallaan, voidaan joutua umpikujaan:
  - valittu arkitehtuuri ja rakennepalikat eivät tue uusia laajennuksia
- Tällöin ohjelman rakennetta on remontoitava
  - arkitehtuuria muutetaan
  - olioita tai olioiden toimintoja muutetaan
  - toiminnallisuutta ei muuteta
- Rakenteen remontti voi olla hankalaa, mutta se kannattaa

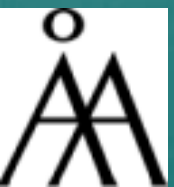
Ohjelmointiprosessi

- Inkrementaalisessa ohjelmoinnissa ohjelman laajentaminen ja ohjelman remontointi vuorottelevat



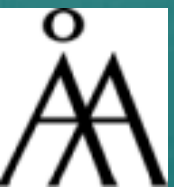
Inkrementaalisen ohjelmoinnin edut

- Iso ohjelma voidaan rakentaa pieninä laajennuksina
- Oikeellisuuden säilyminen voidaan tarkistaa jatkuvasti
- Käytössä on aina toimiva ohjelma (toiminnallisuus on vajaa, mutta se mikä siinä on toimii oikein)
- Ohjelmaa voidaan jatkuvasti verrata vaatimuksiin, ja tarvittaessa korjata ohjelmaa tai vaatimuksia
- Erillistä ohjelman ylläpitovaihetta ei ole, ylläpito on samanlaista kuin alkuperäisen ohjelman rakentaminen
- Oikeellisuuden jatkuva huomioiminen tekee ohjelman remontin mahdollisiksi



Inkrementaalisen rakentamisen rajoitukset

- Käytännössä liian suuria ohjelmia ei voida rakentaa tällä tavoin
- Ohjelmat on jaettava erikseen ohjelmoitaviin moduleihin, joilla on hyvin määritelly rajapinnat
- Eri ohjelmointiryhmät voivat ohjelmoida moduleja toisistaan riippumatta
- Inkrementaalinen rakentaminen ja modulaarinen rakentaminen täydentävät toisiaan
- Yhdessä ne mahdollistavat suurten ja luotettavien ohjelmien rakentamisen
- Virheettömyys edellyttää mekaanista ohjelmien oikeellisuuden todistamista



Inkrementaalisen rakentamisen projekteja ÅAssa

- Inkrementaalisen rakentamisen teoria (tarkennuskalkyylissä)
- Inkrementaalista rakentamista tukeva ohjelmointiympäristö (UML-pohjainen)
- Inkrementaalista rakentamista tukeva ohjelmointiprosessi (XP-pohjainen)
- Useita piloottiprojekteja, joissa rakennetaan suurempia ohjelmia tällä menetelmällä
- Inkrementaalinen rakentaminen eri ohjelmointiparadigmoissa (vlsi-suunnittelu, olio-ohjelmointi, hajautet järjestelmät, rinnakkaisohjelmat, jne)