HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu

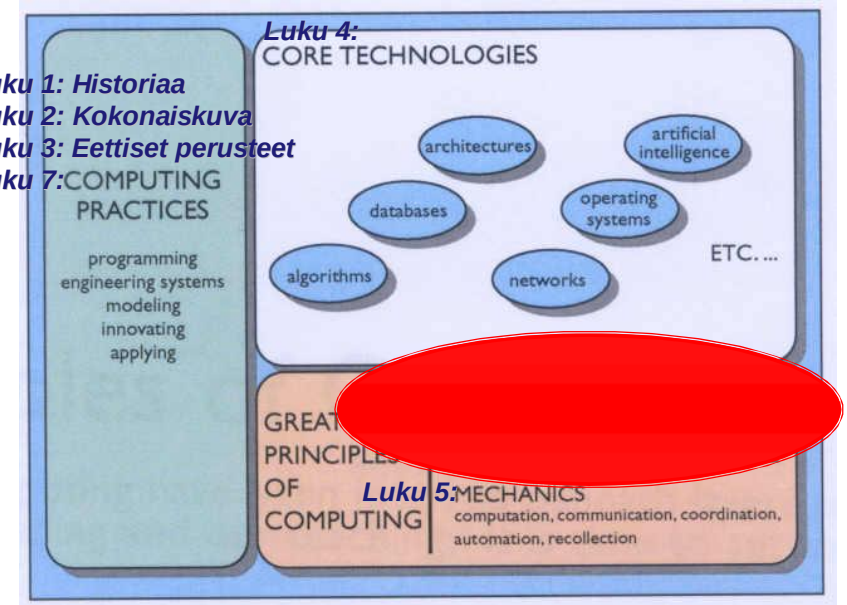
Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Kurssin sisältö

Lähde: Peter J. Denning: Great Principles of Computing (Communications of the ACM, 46, 11, marraskuu 2003, sivut 15-20).

Luku 1: Historiaa
Luku 2: Kokonaiskuva
Luku 3: Eettiset perusteet
Luku 7: COMPUTING PRACTICES
programming
engineering systems
modeling
innovating
applying



Suunnittelu

Tietojen-käsittelyn keskeiset periaatteet

Suunnittelun periaatteet:

- yksinkertaisuus
- suorituskyky
- luotettavuus
- kehitettävyys
- tietoturva

Tietojenkäsittelyn mekaniikat

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu
6.1 Yksinkertaisuus

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu
6.2 Suorituskyky

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Suorituskyvyn mitattavia suureita

- Suoritusteho (*throughput*).
  - Kuinka paljon hyödyllistä työtä tehdään aikayksikössä.
- Vasteaika (*response time*).
  - Kuinka kauan tehtävän suorittaminen kestää.
- Käyttöaste (*utilization*).
  - Osuus ajasta, jonka systeemi on aktiivisena.

Suorituskyvyn suureiden mittaustapoja

- Mitataan valmiin järjestelmän toimintaa.
- Simuloidaan (jäljitellään ohjelmalla) järjestelmän toimintaa likimääräisesti.
- Mallinnetaan järjestelmää matemaattisin menetelmin ja lasketaan mallin avulla keskeiset suorituskykysuureet.
  - Jonomallit.
  - Jonoverkkomallit.
  - jne

Suorituskyvyn suunnittelu

- Tavoite: ohjelmiston suunnitteluvaiheessa arvioidaan tulevan järjestelmän tehokkuutta ja tarvittavia laitteistoresursseja.
- Suorituskyky tietyn työkuorman vallitessa:
  - Käyttäjä kokee.
  - Järjestelmä tarjoaa.

Ohjelmiston vaatimukset

- Toiminnalliset (*functional*).
- Ei–toiminnalliset (*non-functional*).
  - Tietoturva (*security*).
  - Saatavuus (*availability*).
  - Luotettavuus (*reliability*).
  - Suorituskyky (*performance*).
- Asennevamma ohjelmistotuotannossa: jako toiminnallisiin ja ei-toiminnallisiin vaatimuksiin.
  - Korjaa-myöhemmin (*fix-it-later*) –asenne suorituskyvyn suhteen osoitti, että se ei kuulunut ohjelmiston suunnitteluvaiheeseen.

Miksi suorituskyvyn suunnittelu ei ole osa ohjelmiston suunnitteluvaihetta

- Menascén havainnot:

1. Tieteellisten mallien ja periaatteiden puute.
2. Koulutus.
3. Tietojenkäsittelyn työvoima.
4. Yhden käyttäjän ajattelu.
5. Pienen tietokannan ajattelu.

1. Tieteellisten mallien ja periaatteiden puute.

- Ohjelmistotekniikassa ei ole yleisesti käytetä malleja suorituskyvyn suunnittelussa ohjelmiston elinkaaren aikana.
  - Malleja ohjelmistotuotannon hallitsemiseen muuten on.
- Perinteisissä insinööritaidoissa (esim. rakentamisen eri muodot) käytetään matematiikkaan, fysiikkaan ja laskennallisiin tieteisiin perustuvia tieteellisiä periaatteita ja malleja.

2. Koulutus.

- ACM:n/IEEE:n tkt:n opetussuosituksissa ei ole tietokonejärjestelmien suorituskyvyn analysoinnin pakollista kurssia.
  - Suorituskyvystä vain hajatunteja käyttöjärjestelmä- ja tietoliikennekursseilla.
  - Ohjelmistotekniikka ei sisällä mainintaa suorituskyvystä.
- Miksi?
  - Opettajat eivät osaa.
  - Tieteellisten periaatteiden ja mallien puute.
  - Muutosvastarinta.
  - Kaikki hyödyllinen ei mahdu tutkintoon.

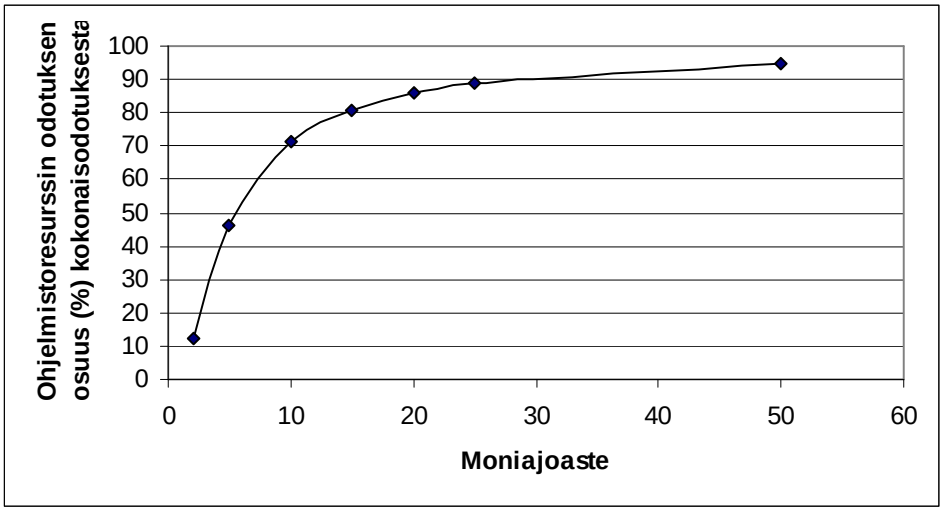
3. Tietojenkäsittelyn työvoima.

- Suunnittelu- ja ohjelmointiväessä paljon henkilöitä vailla tietojenkäsittelyn koulutusta.

4. Yhden käyttäjän ajattelu.

- Suunnittelijat ja ohjelmoijat luulevat, että järjestelmää käyttää vain yksi käyttäjä kerrallaan.
  - Samanaikaisia käyttäjiä on kuitenkin yleensä useita.
  - Samanaikaisuus aiheuttaa kilpailua (ja odotusta) sekä laitteistoresursseista
    - prosessorit, muistit, talletusvälineet, tietoliikenneyhteydet,…
  - että ohjelmistoresursseista
    - tietokantalukot, kriittiset alueet, ohjelmistosäikeet, …

Esimerkki ohjelmistoresurssikilpailun vaikutuksesta – 33% käsittelystä kriittisellä alueella



5. Pienen tietokannan ajattelu.

- Tietokannan käytön ohjelmoinnissa ei yleensä oteta tietokannan kokoa huomioon.
  - Kysely 1000 rivin tietokantaan voidaan yleensä tehdä eri tavalla kuin 1 000 000 rivin tietokantaan.

Menascén johtopäätökset.

- Ohjelmiston monimutkaisuus aiheuttaa usein tehottomuutta.
- Monimutkaisuuden hallitsemiseksi kannattaa parantaa ohjelmoijien ammattitaitoa eikä kehittää heidän käyttämiään työkaluja.
- Tehokkain menettely hyvän suorituskyvyn ohjelmistojen tuottamiselle on suunnittelijoiden ja ohjelmoijien koulutus suorituskykyyn liittyvissä asioissa.
- Ohjelmiston hyvä suorituskyky riippuu enemmän hyvästä suunnittelusta kuin hyvästä ohjelmoinnista.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu
6.3 Luotettavuus

Matemaattis-luonnontieteellinen tiedekunta
Tietojenkäsittelytieteen laitos

Luotettavuus

- Päällekkäisyys tai toisteisuus (*redundancy*).
- Toipuminen (*recovery*).
- Tarkistuspisteiden käyttö eli varmistaminen (*checkpointing*).
- Eheys (*integrity*).
- Luottamus (*trust*).

Luotettavuuden tarina – toipuminen

- Candea ja Fox: Chrash-only software.
  - Erilainen näkökulma ohjelmiston toipumiseen kaatumisesta – ei ole tietojenkäsittelyn vakiintunutta käytäntöä.
- Tietojenkäsittelyjärjestelmä, joka ei koskaan kaadu, ei ole realistinen tavoite.
- Järjestelmän suunnittelussa on siis varauduttava kaatumiseen ja toipumiseen.
- Miksi suunnitella sekä hallittu alasajo ja siitä toipuminen että kaatuminen ja siitä toipuminen?

Crash-only ohjelmistojen perusajatus

- Kaatuminen (*crash*) on tehtävä turvalliseksi.
- Toipuminen (*recovery*) on tehtävä nopeaksi.
- Ainoa tapa lopettaa ohjelman suoritus on kaataa se.
- Ohjelma käynnistetään aina toipumisen kautta.

Toipumisesta

- Järjestelmän toipumisesta huolehtiva ohjelman osa käsittelee poikkeustilanteita. Sen on oltava virheetön.
- Poikkeukselliset tilanteet ovat hankalia käsitellä.
  - Esiintyvät harvoin, eikä niitä ole helppo tuottaa ohjelmiston kehitysvaiheessa, jolloin toipumista olisi testattava.
  - Toipumisen suorittava ohjelman osa on vaikea saada virheettömäksi.

Crash-only ohjelmistojen ominaisuuksia

- Kaikki ei-tilapäinen tilatieto on talletettava erityiseen tilatietomuistiin (*state store*).
- Ohjelmiston osien on varauduttava muiden osien kaatumiseen ja niiden palveluiden tilapäiseen puuttumiseen (*unavailability*).
- Keskeisiä ominaisuuksia:
  - Modulaarisuus.
  - Vahvat rajapinnat, joissa häiriöt hallitaan ilman vaikutuksia toisaalla.
  - Ajastimiin perustuva kommunikointi.
  - Laina-aikaan (*lease*) perustuva resurssien varaus.
  - Täydellisesti itsensä kuvaavat palvelupyynnöt.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu
6.4 Kehitettävyys

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Ohjelmistotekniikka on kriisissä

- Ollut jo 1960-luvun lopulta alkaen.
- Kriisi on äkillinen ja lyhytaikainen vakava hätätila.
- Ilmiö on vakava.
- Kyseessä ei ole kriisi vaan pitkäaikaishoitoa vaativa krooninen tauti.

Tilannekatsaus (Lehman 1998)

- Yhteiskunnan tietokone(ohjelmisto)riippuvuus kasvaa.
- Tietoteknologian käytön lisääntyminen lisää järjestelmien integroinnin tarvetta.
- Ympäröivä maailma muuttuu.
  - Y2K – vuosituhannen vaihtumista ei ohjelmistosuunnitelmissa.
  - Euro.
  - Puhelinnumeroiden piteneminen.
  - jne

Huomioita ohjelmistoista

- Ohjelmistot ovat monimutkaisimpia ihmisen aikaansaannoksia.
- Ohjelmisto itsessään on malli sovelluksesta, osallistujista (ihmiset, organisaatiot, laitteet,…) käyttöalueesta ja kyseisen alueen toiminnoista.
- Käyttöalue on moniulotteinen – käytännössä rajattoman laaja.
  - Ohjelmisto on rajallinen.
- Ohjelmisto on äärellinen ja epätäydellinen malli rajattoman käyttöalueen rajattomasta sovelluksesta.

Ohjelmiston ja todellisen maailman välillä on kuilu

- Kuilua paikataan oletuksilla.
  - Algoritmien valinta.
  - Järjestelmän valinta, määrittely, suunnittelu ja toteutus sisältävät paljon oletuksia.
  - Osa oletuksista on selkeitä (*explicit*), kuten määrittelyssä tehdyt valinnat.
  - Osa oletuksista on epäsuoria (*implicit*), kuten valitun teorian mukana tulevat, algoritmin suunnittelun tuottamat, rajapinnan määrittelystä johtuvat jne.
- Lehman arvelee, että jokaista 10 ohjelmariviä kohti on olemassa yksi oletus. (Miljoona riviä ??? oletusta!)

FEAST (*feedback, evolution and software technology*)

- Ohjelmistoprosessi muodostaa
  - monitasoisen (*multilevel*) ja
  - monisilmukkaisen (*multiloop*) takaisinkytkentäjärjestelmän (*feedback system*)
- ja sitä on käsiteltävä sellaisena, jos haluamme saada huomattavaa parannusta sen suunnitteluun, ohjaukseen ja kehittämiseen.

Lehmanin suosituksia (lyhyt lista … 18 kohtaa)

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
6. Suunnittelu
6.5 Tietoturva

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Tietoturva (*security*)

- Pääsynvalvonta (*access control*).
- Salassapito (*secrecy*).
- Yksityisyys (*privacy*).
- Todennus (*authentication*).
- Eheys (*integrity*).
- Turvallisuus (*safety*).

Tietoturvan tarina - turvallisuus

- Sädehoidon ohjausohjelmisto – Therac-25 laitteistossa.
- Vuosina 1985 – 1987 on kuusi tunnettua tapausta potilaan ylisäteilytyksestä.
- Pahimmillaan säteilyannos oli 10 000-kertainen tarkoitettuun annokseen verrattuna.
- Ainakin viiden potilaan kuolema osoitettiin johtuvaksi sädehoitolaitteen suunnittelu- ja ohjelmointivirheistä johtuvaksi.

Onnettomuuksien syistä

- Useimmiten onnettomuudet johtuvat monimutkaisista toisiinsa kietoutuvista tapahtumista, jotka johtuvat teknisistä, inhimillisistä ja työyhteisöön liittyvistä tekijöistä.
- Therac-25:n tapausten kaksi vakavaa virhettä:
  - Uskottiin, että onnettomuuden syyt oli poistettu ensimmäisen tapauksen jälkeen.
    - Johtopäätökselle ei ollut kestäviä perusteita.
    - Vaihtoehtoisia syitä ei selvitetty kuin ylimalkaisesti.
  - Oletettiin, että yhden ohjelmistovirheen korjaaminen estäisi uudet onnettomuudet.
    - Monimutkaisissa järjestelmissä löytyy lähes aina seuraava virhe.

Onnettomuuksien selityksiä

- Usein selitetään onnettomuuksien johtuneen yhdestä syystä – esim. inhimillisestä erehdyksestä.
  - Lähes kaikkien syiden voidaan katsoa olevan inhimillisiä erehdyksiä.
  - Jos onnettomuuden syy on laitteiston kuluminen, niin miksei kulunutta osaa vaihdettu ajoissa?
- Onnettomuuden selityksenä inhimillinen erehdys ei ole kovin hyödyllinen, ellei sitä tarkenneta riittävästi.
- Yhtä hyödytön selitys on laitteistovika tai ohjelmistovirhe.

Therac-25:n onnettomuuksien syistä

- Valmistajan hallinnoinnissa olleet epätarkoituksen-mukaisuudet ja raportoitujen onnettomuuksien käsittelyn puutteellisuudet.
- Liiallinen luottamus ohjelmistoon ja laitteistovarmistuksista luopuminen, minkä seurauksena ohjelmistosta tuli varmistamaton virhelähde.
- Ohjelmistotekniikan käytäntöjen ilmeiset puutteet.
- Epärealistinen riskien arviointi ja ylenpalttinen luottamus arvioinnin antamiin tuloksiin.

Järjestelmän rakentaminen

- Yleinen virhe on luottaa liikaa ohjelmistoihin.
- Ohjelmiston suunnitteluvirheiden löytäminen ja estäminen on huomattavasti hankalampaa kuin laitteiston kulumisesta johtuvien virheiden.
- Laitteiston virhekäyttäytymisen muotoja on vain muutama.
  - Niitä vastaan suojautuminen on yleensä olennaisesti helpompaa kuin ohjelmistovirheistä vastaan suojautuminen.

Therac-25:n opetuksia

- Ehkä tärkein: laitteistovarmistuksista ei pidä luopua, kun järjestelmän ohjaamiseen aletaan käyttää ohjelmistoa.
- Trendi on ollut vähentää laitteistovarmistuksia.
  - Niissäkin tapauksissa, missä laitteistovarmistuksia käytetään, niitä yhä useammin ohjataan ohjelmistolla.
  - Ehdotonta turvallisuutta vaativissa järjestelmissä ei saa olla varmistamattomia virhelähteitä.
- Järjestelmää ei saa suunnitella siten, että yksittäinen ohjelmistovirhe voi aiheuttaa katastrofin.

Ohjausjärjestelmien ohjelmistovirheistä

- Onko kyseessä tilapäinen laitteistovirhe?
- Ohjausohjelmisto lukee arvoja tunnistimista (*sensors*) ja lähettää komentoja säätimille (*actautors*).
- Hyvin vaikea (ellei mahdotonta) päätellä
  - antoiko tunnistin väärää tietoa,
  - lähettikö ohjelmisto väärän komennon vai
  - toimiko säädin tilapäisen laitevirheen vuoksi väärin.

Therac-25:n virhediagnostiikka

- Potilaiden oireet olivat ainoat todelliset indikaattorit järjestelmän virheistä.
- Järjestelmässä ei ollut riippumattomia toiminnan oikeellisuuden tarkistuksia.
  - Therac-25 ei voinut havaita antamaansa säteilyn määrää.
- Keskeinen opetus: ohjausjärjestelmät on suunniteltava pahimman toiminnan varalle.
  - Ehdotonta turvallisuutta vaativiin järjestelmiin on rakennettava jäljitysmekanismit (*audit trails*) sekä poikkeavien tilanteiden analysointimekanismit.

Loppuhuomioita

- Turvallisuus on pystyttävä takaamaan järjestelmätasolla (laitteistovarmistukset) mahdollisista ohjelmistovirheistä riippumatta.
- Therac-25:n edeltäjässä oli sama ohjelmistovirhe, mutta laitteistovarmistus esti onnettomuudet.
  - Usein naivisti oletetaan, että ohjelman uudelleen käyttö lisää turvallisuutta, koska ohjelma on ollut jo pitkään käytössä.
- Turvallisuus on koko järjestelmän ominaisuus – ei pelkästään ohjelmiston ominaisuus.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7. Tietojenkäsittelyn käytännöt

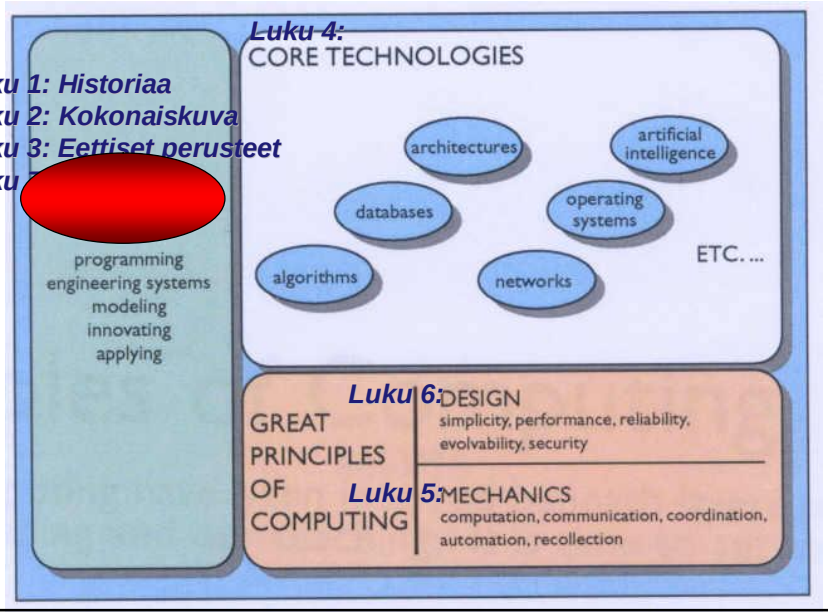
Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Kurssin sisältö

Lähde: Peter J. Denning: Great Principles of Computing (Communications of the ACM, 46, 11, marraskuu 2003, sivut 15-20).

Luku 1: Historiaa
Luku 2: Kokonaiskuva
Luku 3: Eettiset perusteet
Luku



Tietojenkäsittelyn käytäntöjen pääalueet

1. Ohjelmointi (*programming*).
2. Järjestelmien rakentaminen (*engineering systems*).
3. Mallintaminen ja validointi (*modeling and validation*).
4. Innovointi (*innovating*).
5. Soveltaminen (*applying*).

Ohjelmointi

- Järjestelmän käyttäjien kanssa määritellyn ohjelmiston toteuttaminen ohjelmointikieliä käyttäen.
- Tietojenkäsittelyn ammattilaisen on hallittava useita eri ohjelmointikieliä ja osattava valita tarkoituksenmukaisin kuhunkin ongelmanratkaisutilanteeseen.

Järjestelmien rakentaminen

- Tietoverkossa toimivien hajautettujen järjestelmien suunnitteleminen ja toteuttaminen ohjelmisto- ja laitteistokomponenteista.
- Tietojenkäsittelyn ammattilaisella on oltava taidot osallistua laajojen (tuhansia moduuleja, miljoonia ohjelmarivejä) järjestelmien toteuttamiseen.

Mallintaminen ja validointi

- Järjestelmän mallintaminen ja sen käyttäytymisen ennustaminen erilaisissa tilanteissa ja olosuhteissa.
- Kokeiden (*experiment*) suunnittelu algoritmien ja järjestelmien validoimiseksi .

Innovointi

- Johtajuuden käyttäminen pysyvien muutosten aikaansaamiseksi ryhmien ja yhteisöjen toimintatavoissa.

Soveltaminen

- Työskentely sovellusalueiden ammattilaisten kanssa näitä palvelevien tietojenkäsittelyjärjestelmien toteuttamiseksi.
- Työskentely muiden tietojenkäsittelyn ammattilaisten kanssa useita erilaisia sovelluksia palvelevien ydinteknologioiden kehittämiseksi.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7. Tietojenkäsittelyn käytännöt
7.1 Ohjelmointi

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Ohjelmoinnin tarina

- Donn Seeley
- How Not to Write Fortran in Any Language
- Maailmassa on nähty niin monia huonoja Fortran-ohjelmia, että Fortranista on tullut huonon ohjelmakoodin synonyymi.
- Syyt historiallisia.
  - Useimmat Fortran-ohjelmat eivät olleet tietojenkäsittelyn ammattilaisten kirjoittamia vaan fyysikoiden ja muiden luonnontieteilijöiden kirjoittamia.
  - Ammattitaidottomat ohjelmoijat eivät olisi saaneet hyvää ohjelmakoodia kirjoitettua millään kielellä.

Hyvä ohjelmakoodi ja ohjelmointikieli

- Hyvän ohjelmakoodin ominaisuudet ovat varsin riippumattomia ohjelmointikielestä.
- Hyvin suunniteltu ohjelma voidaan kirjoittaa lähes millä tahansa ohjelmointikielellä siten, että ohjelmakoodi on selkeärakenteinen ja helposti ymmärrettävä.
- Mikään käyttökelpoinen ohjelmointikieli ei pysty estämään huonon ohjelmakoodin kirjoittamista.
- Käytetyn ohjelmointikielen vaikutuksia ohjelmakoodin laatuun on yliarvioitu.

Seeleyn lopputoteamukset

- Hyvän ohjelmakoodin kirjoittaminen ei ole kovinkaan paljon vaativampaa kuin huonon.
- Hyvän ohjelmakoodin hyödyt tulevat selkeästi esille ohjelman ylläpidon aikana.
- Ei ole mitään järkeä kirjoittaa muuta kuin hyvää ohjelmakoodia.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7. Tietojenkäsittelyn käytännöt
7.2 Järjestelmien rakentaminen

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Vasa-laivan tarina

- Tilaus tammikuussa 1625.
  - Kaksi 108-jalkaista ja kaksi 135-jalkaista laivaa.
- Rakentaminen alkoi 1626.
- Marraskuussa 1626 108-jalkaiset 120-jalkaisiksi.
  - Puuta oli yhteen 111-jalkaiseen ja yhteen 135-jalkaiseen.
- Kun 111-jalkaisen kölirakenne oli tehty, niin alus päätettiin pidentää 135-jalkaiseksi ja kaksikantiseksi.
  - Laivaa levennettiin puoli metriä, mutta vain yläosasta.
  - Aseistusta muutettiin siten, että kanuunoiden yhteispaino kasvoi noin 50 % (n. 75 tonniin).
  - Paljon veistoksia ja ornamentteja painavasta tammesta aluksen yläosiin.
- Vuonna 1627 hankkeessa työskenteli n. 400 ihmistä.
- Neitsytmatka 10. elokuuta 1628.

Kymmenen tautia ja joitakin lääkkeitä

1. Aikataulupaine.
   - Objektiiviset arviot, resurssien lisääminen, resurssien parantaminen, vaatimusten priorisointi, tuotoksen vaiheistaminen.
2. Tavoitteiden muuttuminen.
   - Iteratiivinen ohjelmistokehitys, perusratkaisun hallinta.
3. Teknisten määritysten puuttuminen.
   - Alustavien määritysten tekeminen, määritysten päivittäminen, määritysten hallinnointi.
4. Dokumentoidun projektisuunnitelman puuttuminen.
   - Alustavan suunnitelman tekeminen, suunnitelman toistuva päivittäminen, projektisuunnitelman hallinnointi, projektipäällikön nimeäminen.

Kymmenen tautia ja joitakin lääkkeitä

5. Yletön ja
6. toissijainen innovointi.
   - Perusdokumenttien hallinnointi, vaikutusten analysointi, jatkuva riskien hallinta, nimetty ohjelmistoarkkitehti.
7. Vaatimusten luisuminen.
   - Alustava vaatimusten versio, versioiden hallinnointi, riskien hallinta, nimetty ohjelmistoarkkitehti.
8. Tieteellisten menetelmien puuttuminen.
   - Prototyyppien tekeminen, vaiheittainen kehittäminen, suorituskykymittaukset.

Kymmenen tautia ja joitakin lääkkeitä

9. Olennaisen unohtaminen.
   - Karkeat (*back-of-the-envelope*) laskelmat, opittujen opetusten sulauttaminen.
10. Epäeettinen käyttäytyminen.
    - Eettinen työympäristö ja työtavat, henkilökohtainen eettisten säännösten noudattaminen.

- Epärealistisen kiireinen aikataulu on yleisin ohjelmistoprojektien epäonnistumisen syy (yleisempi kuin muut syyt yhteensä).

Projektisuunnitelmasta

- Tehtävän työn jaottelu osatehtäviksi.
- Vaatimusten sijoittaminen osatehtäviin.
- Aikataulu tarkistuspisteineen ja virstanpylväineen.
- Kunkin osatehtävän tuotokset määräaikoineen.
- Tarvittavien ohjelmistojen hankintasuunnitelma.
- Alihankintojen hallinnointisuunnitelma.
- Vastuiden selkeä kirjaaminen.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7. Tietojenkäsittelyn käytännöt
7.3 Mallintaminen ja validointi

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Mallintamisen ja validoinnin tarina

- Charles E. Knadler Jr.
- The robustness of Separable Queueing Network Models
- Artikkelissa selvitetään suorituskykyanalyysissä käytettyjen jonoverkkomallien herkkyyttä oletusten rikkomiselle.
- Jonoverkkomallien matemaattinen käsittely edellyttää oletuksia separoituvista jonoverkoista. Käytännössä oletukset eivät täysin toteudu. Ovatko mallin antamat tulokset silti käyttökelpoisia käytännössä?

Tapahtumien simuloinnilla selvitetään matemaattisten tulosten herkkyyttä, kun oletukset eivät ole voimassa

- Simulointitulokset ovat kauniisti lähellä teoreettisia tuloksia, joten oletusten rikkominen ei tässä esimerkissä vaikuta ratkaisevasti tulosten käyttökelpoisuuteen.
- Simulointikokeiden suunnittelu on tärkeää: on perusteellisesti ymmärrettävä kysymykset, joihin vastauksia etsitään.
- Simulointeja pitäisi tehdä lisää ja tarkastella tuloksia tilastollisesti, jotta sattuman vaikutus vähenisi.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7 Tietojenkäsittelyn käytännöt
7.4 Innovointi

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Innovaatioiden tarina

- P. J. Denning.
- The Social Life of Innovation.
- Innovaatioiden käytännön voi oppia kunhan tietää mitä innovaatio on.
- Innovaatiot ovat uusia tapoja tehdä asioita.
- Käytäntöjen muuttaminen on paljon vaikeampaa kuin uusien teknologioiden keksiminen.

Innovaatio ja keksintö

- Innovoinnin ja keksimisen erottaminen eri käsitteiksi on olennaista.
- Keksinnöissä voidaan keskittyä teknologioihin.
- Innovaatioissa on otettava huomioon sosiaalinen yhteisö.
  - Mitä muut ihmiset arvostavat ja hyväksyvät otettavaksi käyttöön.

Innovaatioprosessin peruselementit

1. Mahdollisuuksien etsiminen.
2. Analysointi.
3. Kuunteleminen.
4. Keskittyminen.
5. Johtajuus.

Innovaatioiden lähteet

1. Odottamattomat tapahtumat.
2. Epäsuhdat.
3. Prosessin tarpeet.
4. Liike-elämän rakennemuutos.
5. Demografia.
6. Ilmapiirin ja asenteiden muuttuminen.
7. Uusi tietämys.
8. Marginaaliset käytännöt.

Neljä yleisintä väärinkäsitystä

1. Innovaatioiden on oltava isoja.
2. Innovaatiot ovat vain muutamien lahjakkuuksien työsarkaa.
3. Innovaatiot perustuvat uusiin ajatuksiin.
4. Innovaatioita tapahtuu vain elinkeinoelämässä.

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Johdatus tietojenkäsittelytieteeseen
7. Tietojenkäsittelyn käytännöt
7.5 Soveltaminen

Matemaattis-luonnontieteellinen tiedekunta

Tietojenkäsittelytieteen laitos

Soveltaminen

- Työskentely sovellusalueiden ammattilaisten kanssa näitä palvelevien tietojenkäsittelyjärjestelmien toteuttamiseksi.
- Työskentely muiden tietojenkäsittelyn ammattilaisten kanssa useita erilaisia sovelluksia palvelevien ydinteknologioiden kehittämiseksi.

Soveltamisen tarina

- Ahmed Seffah.
- Learning the Ropes: Human-Centered Design Skills and Patterns for Sofware Engineers' Education.
- Ohjelmistoammattilaisen tarvitsemat taidot ihmiskeskeisessä suunnittelussa.
- Ihmiskeskeinen suunnittelu on parantanut tuotteita.
  - Käyttäjillä ei useinkaan ole loistavassa suunnittelussa tarvittavaa näkemystä.

Ihmiskeskeisestä suunnittelusta

- Ihmiskeskeisen suunnittelun yksi keskeinen periaate on käyttäjän kuunteleminen.
- Ohjelmistoammattilaisen tulee myös tietää, miten käytettävyyttä (*usability*) mitataan ja miten havaintoaineiston perusteella tehdään päätöksiä.
- Seffah esittää 19 taitoa, joita tarvitaan menestyksekkäässä ihmiskekseisesä suunnittelussa.
  - Välttämättömät edellytykset (*prerequisite skills*).
  - Erityistaidot (*specific skills*).
  - Yleistaidot (*generic skills*).

Kertauksena kurssin oppimistavoitteet

- Kurssin suorituksen jälkeen osaat
  - selittää ja kuvailla maisterin tutkinnossa esiintyvät tietojenkäsittely(tietee)n
    - perusperiaatteet,
    - käytännöt ja
    - keskeiset teknologiat,
  - käyttää tietojenkäsittelyn käsitteistöä (terminologiaa),
    - englanti on valtakieli,
  - lukea alan artikkeleita ja tehdä niistä lyhyitä referaatteja (esseitä),
  - työskennellä ryhmässä yhteisen tavoitteen saavuttamiseksi ja
  - tunnistaa ja ratkaista alan eettisiä kysymyksiä.