Acta Universitatis Tamperensis
ser B vol 34

Tietokone Suomessa 30 vuotta

Näkökulmia tietotekniikan tutkimukseen

Toimittaneet Erkki Mäkinen ja Kari-Jouko Räihä

Tampereen yliopisto
Tampere 1990

ISBN 951-44-2666-5
ISSN 0355-5232

Tampere 1990, Kirjapaino Hermes Oy

Sisällys

Luentosarjan aiheet

7.3.	•	Tietotekniikan tutkimuksen synty ja kehittyminen Suomessa: miltä se 60-luvulla näytti
	•	Tietojenkäsittelyoppi eilen, tänään ja huomenna
14.3.	•	Mullistaako rinnakkaislaskenta tietotekniikan?
	•	Pitääkö tietoa mallittaa?
21.3.	•	Tietotekniikan teorian tutkimuskohteita eri aikoina
	•	Al Khwarizmin perintö
4.4.	•	Atk:n rutiinilla ehtii tehdä enemmän virheitä
	•	Tietotekniikka organisaatioissa – kustannussäästäjästä muutoksen agentiksi
11.4.	•	Tietotekniikka tunkee tehtaaseen
	•	Superkone tulee – mikä muuttuu?
18.4.	•	Tarvitseeko tietotekniikka humanisteja?
	•	Tietotekniikka ja ihmiskuva
25.4.	•	Tietotekniikan tutkimus teollisuudessa
	•	Mikroelektroniikka – elektronien vallankumous

Alkusanat

Keväällä 1989 järjestettiin Helsingin yliopistossa yleisluentosarja aiheesta "Tietokone Suomessa 30 vuotta: Näkökulmia tietotekniikan tutkimukseen". Aloitteen luentosarjan järjestämiseksi teki Tietotekniikan liitto, ja luentosarjan käytännön järjestelyistä vastasi Tietojenkäsittelytieteen Seura.

Luentosarjan tarkoituksena oli valottaa alan tutkimuksen kehittymistä Suomessa, esitellä erilaisia nykyisiä tutkimussuuntauksia – niin tietojenkäsittelytieteessä kuin sen lähitieteissäkin – sekä tarjota näkemyksiä tietotekniikan tutkimuksen kehittymisestä tulevaisuudessa. Sarjaan kuului 14 esitelmää. Luettelo esitelmien otsikoista on viereisellä sivulla.

Kahdeksan luennoijaa luovutti esityksestään kirjallisen version Tietojenkäsittelytieteen Seuran julkaistavaksi. Kirjoitukset on koottu tähän teokseen. Tietojenkäsittelytieteen Seura kiittää kaikkia kirjoittajia.

Tampereella ja Eugenessa 6.3.1990
Toimittajat

Tietotekniikan tutkimuksen synty ja kehittyminen Suomessa: Miltä se 60-luvulla näytti

REINO KURKI-SUONIO, TAMPEREEN TEKNILLINEN KORKEAKOULU

1. Johdanto

Tätä esitystä arkistojeni parissa valmistellessani olen koettanut mielessäni palata 1960-luvulle voidakseni välittää teille niitä näkemyksiä ja näkyjä, joiden varassa tietotekniikan tutkimus Suomessa alkoi. Tasapuoliseen ja täydelliseen historiankirjoitukseen en pyri, vaan otan vapauden katsella asioita omasta näkökulmastani ja omien subjektiivisten kokemusteni valossa. Monia kokonaisuuden kannalta tärkeitäkin seikkoja jää tämän vuoksi koskettelematta, kun taas henkilökohtaiset ainekset korostuvat. Pelkkien muistikuvien sijasta pyrin kuitenkin antamaan puheenvuoron erilaisille dokumenteille.

Siirtykäämme siis ajassa taaksepäin, aikaan jolloin tietokoneista ei juuri kirjoiteltu eikä puhuttu, eihän koko sanaakaan vielä ollut. Maaliskuussa 1958 oli Suomen Kuvalehdessä kuitenkin poikkeuksellinen kirjoitus [1] otsikolla *ESKOSTA UUSI SAMPO*. Se alkoi kertomalla, että "Suomen ensimmäinen elektroninen laskukone – joita tavallisessa puheessa sanotaan 'sähköaivoiksi' – on parhaillaan valmistumassa Teknillisen Korkeakoulun ullakkohuoneessa", ja siitä selvisi lisäksi mm. seuraavaa:

> ESKO on matematiikkakonekomiteamme esikoinen, ja se on tarkoitettu tieteelliseen käyttöön ... Kun ESKO valmistuu, matemaatikot ja matematiikkaa tarvitsevat tiedemiehemme selviytyvät nopeasti

laskutehtävistään. Ja kun he nyt suunnittelevat uusia laskuja, he voivat antaa kekseliäisyydelleen vapaan vallan, sillä ESKO suorittaa tunneissa laskutehtäviä, jotka muuten vaatisivat vuosien työn.

Siitä on helppo ymmärtää, kuinka hyödyllinen ESKO on. Se on robotti, joka auttaa tiedemiehiä voittamaan ennen ylipääsemättömiä aineellisia vaikeuksia ...

ESKO on ... pieni kone, mutta hyvin käyttökelpoinen tieteellisessä työssä ... Kun isot matematiikkakoneet saattavat käsitellä yhtäaikaisesti miljoonia yksityisiä tietoja, ESKO muistaa vain 1840 asiaa ... Se pystyy suorittamaan kaksikymmentä laskutoimitusta sekunnissa, mikä on hirveän vähän, koska 20 000 laskutoimitusta sekunnissa voidaan suorittaa nopeimmilla koneilla ...

ESKOn kaltainen matematiikkakone tarvitsee myös hyvin taitavia matemaatikkoja hoitajikseen. Onneksi meillä Suomessa ollaan matematiikan alalla korkeimmalla kansainvälisellä tasolla.

Perspektiiviä näihin ja myös myöhempiin tapahtumiin saamme prof. Olli Lehdon (nykyisin Helsingin yliopiston kansleri) kirjoituksesta [9] kahdeksan vuotta myöhemmin:

Ensimmäiset tietokoneet suunniteltiin ja rakennettiin yliopistoissa pyrittäessä ratkaisemaan numeerisesti tiettyjä matemaattisia tehtäviä. Tietokonealalla ovatkin korkeakoulut useissa maissa olleet aloitteentekijöinä ja tiennäyttäjinä. Suomessa kehitys on kuitenkin kulkenut hieman toisin. Täälläkin tosin lähdettiin liikkeelle yliopistojen piirissä. Vuonna 1954 perustettiin matematiikkakonekomitea, jonka eräänä päätehtäväksi ... esitettiin lähinnä korkeakoulujen käyttöön soveltuvan tietokoneen rakentaminen ... Kotimainen kone Esko saatiin kyllä alkuperäisestä aikataulusta melkoisesti myöhästyneenä valmiiksi vuonna 1960. Se oli kuitenkin jo valmistuessaan monessa suhteessa vanhentunut, eikä siitä sen käyttäjälle, Helsingin yliopiston laskentakeskukselle, sen parivuotisen toimintakauden aikana ollut kovinkaan suurta hyötyä.

Tällä välin johtoasema tietojenkäsittelyalalla Suomessa olikin siirtynyt koneiden maahantuojien haltuun. Aluksi IBM oli alalla käytännöllisesti katsoen yksin, mutta jo ... viisi vuotta sitten oli uudeksi varteenotettavaksi tekijäksi noussut Suomen Kaapelitehdas ... Palvelukseen pyrittiin saamaan parhaat mahdolliset voimat ja näille koetettiin järjestää paras mahdollinen koulutus. Ensimmäisen tavoitteen suhteen aika oli otollinen, koska korkeakoulujen opettajakunnan kasvu oli lähes pysähdyksissä ...

(Kaapelitehtaan koulutustoiminnan valtakunnallisen merkityksen) voi helposti todistaa oikeaksi tarkastelemalla tietojenkäsittelyopin tämänhetkistä tilaa maan korkeakouluissa, joihin alan korkein opetus ja alkava tutkimus

> nyt vihdoinkin on juurtumassa. Ensimmäisenä on Tampereen yliopisto liittänyt ohjelmaansa laajamittaisen tietojenkäsittelyalan opetuksen. Sen suunnittelun pani alulle tohtori Seppo Mustonen, ja hänen lisäkseen sitä kehittää nyt edelleen tohtori Reino Kurki-Suonio. Nämä Tampereen yliopiston vt. professorit ovat kumpikin Kaapelitehtaalla tietokonekoulutuksen saaneita. Ensi vuonna myös Helsingin yliopisto toivoo voivansa perustaa tietojenkäsittelyopin laitoksen ja saavansa alulle systemaattisen opetuksen. Valmistelutöitä on hoitanut kolmihenkinen toimikunta aktiivisimpana jäsenenään Kaapelitehtaan tohtori Martti Tienari.

Oma työsuhteeni Kaapelitehtaaseen alkoi kesäkuussa 1960. Aluksi olin puolisen vuotta oppia saamassa Tanskassa, missä ohjaajani oli Algol 60 -kieleen liittyvistä ansioistaan näihin aikoihin tunnetuksi tullut tri Peter Naur (nykyään professori Kööpenhaminan yliopistossa). Jatko-opintojen ja tutkimuksen poikkeuksellinen arvostus Kaapelitehtaalla oli varmastikin sen tietokoneosastoa johtaneen Olli Lehdon itsensä ansiota.

2. Tietokoneen yleisen merkityksen ymmärtäminen

Mutta eipä vielä kiirehditä Lehdon kirjoituksen ajankohtaan, vaan palataan aikaan, jolloin tietokonealan hegemonia Suomessa oli IBM:llä ja Kaapelitehtaalla. Tammikuussa 1961 Suomen Teknillinen Seura järjesti ensimmäisen tietokoneita käsitelleen täydennyskoulutuskurssin [2]. Prof. Pentti Laasosen (sittemmin Teknillisen korkeakoulun rehtori) avausesitelmä [3] ilmentää hyvin senhetkistä tilannetta:

> Kun maahamme ruvettiin seitsemän vuotta sitten rakentamaan ensimmäistä elektronista tietokonetta ja siten haluttiin päästä osallisiksi uudenaikaisten tietokoneiden luomasta tekniikasta, ei varmaankaan arvattu, miten äkillisen kehityksen edessä meilläkin tuolloin jo oltiin. Valaiskoot seuraavat pari numeroarvoa tämänhetkistä tilannetta. Suomessa on nyt tieteellis-teknilliseen laskutyöhön sopivia automaattisia numerokoneita käytössä 5 ja tämän vuoden lopulla ainakin 9. Kaupallisiin tarkoituksiin valmistettuja nykyaikaisia koneita on tuolloin vieläkin useampia ... Ei voi välttää sitä epäilystä, että liian nopea alan kehitys, etten sanoisi ryöstäytyminen, sen nykyiseen laajuuteen saattaa sisältää vaaran voimien ja varojen tuhlaamisesta osittain tehottomaan käyttöön.

Tulevaisuuden näköaloja luotaava fil. tri Kari Karhusen (sittemmin nykyisen Suomi-Salaman toimitusjohtaja) päätösesitelmä [4] puolestaan osoittaa kaukonäköistä tietokoneiden yleisen merkityksen ymmärtämistä. Hän havainnollistaa tilannetta vertaamalla sitä autoliikenteen kehittymiseen:

> Tällaisessa vaiheessa on esim. kysymys auton käytön taloudellisuudesta tai epätaloudellisuudesta vähintään hämärä ja monesti kokonaan vailla sisältöä. Käsitykseni on, että tietokoneiden kohdalla olemme lähiaikoina suunnilleen samassa kehitysvaiheessa. Ne avaavat mahdollisuuksia sellaisille inhimillisen toiminnan muodoille, tieteellisille ja käytännöllisille, joita ilman ihmiskunta ei halua olla. Ne tulevat kuulumaan niihin välineisiin, joita nimitämme "välttämättömiksi" ja joiden kohdalla kysymys ns. taloudellisuudesta menettää sisältönsä. Tällä hetkellä, jolloin tietokoneet eivät vielä ole perheenjäseniämme, vaan vasta palveluksiaan tarjoavia toimenhakijoita, on sana taloudellinen luvallinen, jopa ajankohtainen ... Joka tapauksessa tietokoneet lähiaikoina ovat yhtä luonnollisia apuvälineitämme kuin nyt kirjoitus- ja laskukoneet.

Seuraavana vuonna pidettiin vastaavanlainen kurssi, jonka päätösesitelmässä dipl. ins. Hans Andersin IBM:ltä (nykyään automaatiotekniikan professori Teknillisessä korkeakoulussa) kirjoittaa [6]:

> Ehkä 10-20 vuoden kuluttua tietokone kuuluu ihmisen tarvikkeisiin yhtä luonnollisena kuin tänään televisio ja auto. Silloin voidaan ajatella valmistettavan miljoonasarjoina tulitikkurasian kokoisia koneita alle miljoonan markan hintaan.

Saman vuoden alussa oli Arkhimedes-lehti julkaissut oman artikkelini [5], jossa teknisen sisällön ohella koetin myös valottaa yleisempiä näkökohtia. Otsikkoni oli *Tietokone, tutkimustyötä innoittava apuväline*, ja kirjoitin mm. seuraavasti:

> Tietokone ei ole pelkkä apuväline, kuten laskutikku tai funktiotaulut. Se eroaa aikaisemmista laskentavälineistä niin mullistavasti, että on aivan uudelleen asennoiduttava siihen kysymykseen, mitkä tehtävät ovat yleensä laskettavissa ...
>
> Tietokone on myös mielenkiintoinen tutkimuskohde. Sen tekniikka ja ohjelmointi kaipaavat vielä paljon teoreettista pohdintaa. Mitään irrallisia kohteita nämä eivät ole, vaan niillä on runsaasti liittymäkohtia muihin tieteisiin. Logiikka on kaiken perustana, ja ennen kaikkea juuri sitä tarvitaan uusien ideoiden löytämiseksi ...

> Kauan ei korkeakouluissamme riitä pelkkä tietokonealan yleissivistyksen jakaminen ja "klassillisen" ohjelmointitekniikan opetus. Ala on niin rikas tutkimuskohde ja liittyy niin kiinteästi muiden alojen nykyiseen tutkimukseen, että tutkimustyön ja alan erikoisopetuksen on vähitellen aika alkaa yliopiston piirissä.

Karhusen esitelmän tavoin koetin itsekin eri yhteyksissä tuoda esille tietokoneen ja tietojenkäsittelyn yleistä merkitystä kulttuurille [10]:

> Tietokoneen ja automaattisen tietojenkäsittelyn keskeinen asema kulttuurimme kehityksessä on kiistämätön. Useiden sovellutusalueiden piirissä tietokoneen käyttö on jo niin vakiintunut, että suorastaan ihmettelemme, miten aikaisemmin on tultu toimeen ilman sitä ... Myös jokapäiväinen työmme ja elämämme näyttää tulevan yhä enenevässä määrin tietokoneista riippuvaiseksi.
>
> Tietokonetta pidetään usein muihin teknillisiin apuneuvoihin verrattavana välineenä, joka säästää kustannuksia ja ihmisten aikaa. Tällöin ei tietokoneen mahdollisuuksia ja merkitystä kuitenkaan vielä tajuta. Vaikka ... voimme vasta aavistella sen todellisen merkityksen laajuutta, on käynyt täysin ilmeiseksi, että se tulee olemaan kulttuurimme keskeisimpiin apuvälineisiin, kieleen ja matematiikkaan verrattava apuneuvo. Se merkitsee yhtä olennaista laajennusta ihmisen mahdollisuuksiin kuin mitä esimerkiksi kirjoitettu kieli on merkinnyt.

3. Korkeakouluissa alkaa tapahtua

Omien jatko-opintojeni motiivina ei ollut korkeakoulu-uralle tähtääminen, vaan yleiset arvostustekijät. "Hyödyttömän" matematiikan opiskelussa olin oppinut pitämään mielenkiintoisten probleemoiden tutkimista sinänsä tärkeänä, enkä tuntenut vetoa myynti- tai esimiestehtäviin. Prof. Lehtoa ja johtaja Wikstedtiä saan kiittää siitä, ettei työnantajani koettanut ahtaa minua toisenlaiseen muottiin, vaan antoi suuren vapauden "asiantuntijatehtäviin". Näissä oloissa kiinnostukseni suuntautui ohjelmoinnista ja ohjelmointikielistä nouseviin teoreettisiin ja algoritmisiin kysymyksiin.

Ensimmäisiä alueita, joilla orastavan tietojenkäsittelytieteen "tieteellisyyttä" ei tarvinnut ujostella, oli formaalien kielten ja automaattien teoria sekä sen soveltaminen ohjelmointikielten määrittelyyn ja toteutukseen. Algol 60:llä oli tässä suhteessa aivan erityinen merkitys — ei niinkään

käytännön ohjelmointikielenä kuin tieteellisenä merkkipaaluna, joka ratkaisevasti vaikutti sekä kysymyksenasettelujen että tutkimusmenetelmien kehittymiseen. Sen inspiroimana itsekin lähdin liikkeelle, vaikken työni tieteelliseen sisältöön voinutkaan saada ohjausta. Olin tuolloin myös tietämätön Arto Salomaan (nykyisin matematiikan professori Turun yliopistossa) Turussa aloittamista formaalien kielten tutkimuksista.

Väiteltyäni keväällä 1964 pääsin tutkijaksi Yhdysvaltoihin. Paikka oli nykyinen Carnegie-Mellon -yliopisto, joka jo silloin oli alan parhaita, vaikka omaa laitosta ei sinne vielä ollutkaan perustettu, ja alan systemaattinen opetuskin oli aivan uutta. Siellä vaikuttivat jo tuolloin tunnetut professorit Alan J. Perlis, Allen Newell ja Herbert Simon, joista etenkin ensiksi mainitun inspiroivat ajatukset muovasivat voimakkaasti käsityksiäni tämän uuden tieteenalan olemuksesta. Sinne sain kuulla Tampereen yliopiston (tuolloin vielä Yhteiskunnallinen Korkeakoulu) suunnitelmista Seppo Mustoselta (nykyisin tilastotieteen professori Helsingin yliopistossa). Helmikuussa 1965 päivätystä, ilmeisesti rehtori Paavo Kolin aloitteesta laaditusta muistiosta [7] sain lukea mm. seuraavaa:

> Laskelmissa on päädytty siihen, että lähemmän viiden vuoden aikana tarvitaan vuosittain n. 200 akateemisen koulutuksen saanutta henkilöä, joilla on vähintäin tietojenkäsittelyopin peruskoulutus (= approbatur-taso). Peruskoulutuksen tavoitteena on kouluttaa ATK-alan suunnittelijoita ja ohjelmoitsijoita käytännön työhön. Samalla se muodostaa pohjan jatkoopinnoille.

Korkeakoulu oli perustamassa taloudellis-hallinnollista tiedekuntaa. Siihen tulisi myös tietojenkäsittelyopin professorin virka, ja Seppo oli valtuutettu tiedustelemaan halukkuuttani sen hoitamiseen. Yllätykseni oli täydellinen, sillä en ollut osannut kuvitella tietojenkäsittelyoppia itsenäisenä akateemisena oppiaineena ainakaan Suomessa. Virka olikin Pohjoismaiden ensimmäinen.

Valinta oli vaikea: toisaalta olisin halunnut käyttää hyväkseni tilaisuutta jatkaa työtä Carnegiessa, jotta olisin paremmin ehtinyt omaksua sitä uutta, mitä siellä oli tarjolla, mutta houkutus opetuksen aloittamiseen Suomessa oli suuri. Pelkäsin myös, että opetus saattaisi Suomessa muuten lähteä linjoille, joita en vasta omaksumieni ajatusten valossa olisi voinut pitää

onnistuneina. Mitään yleisesti hyväksyttyjä suuntaviivojahan ei alan korkeakouluopetukselle tuolloin ollut, oppikirjoista puhumattakaan.

Päätin ottaa haasteen vastaan. Tiesin itseni epäkypsäksi, mutten uskonut muita sen kypsemmiksi. Yhdessä Miikka Jahnukaisen (sittemmin valt. tri ja ATK-instituutin johtaja, nykyään konsultti) kanssa huolehdimme tutkintovaatimusten suunnittelusta ja alkuvuosien opetuksesta. Taloudellis-hallinnollisen tiedekunnan muodostamassa ympäristössä päädyimme eräänlaiseen kaksijakoisuuteen, joka näkyi myös henkilöissämme: Miikka oli systeeminsuunnittelututkimuksen uranuurtaja Suomessa. Katsoimme, että oppiainetta tuli lähestyä sekä ohjelmointiteorian että systeeminsuunnittelun lähtökohdista. Tutkintovaatimuksillamme ja kokemuksillamme oli ilmeinen vaikutuksensa myös muihin alan laitoksiin, kun ne pari vuotta myöhemmin aloittivat toimintansa.

Seppo Mustosella oli Tampereella kuitenkin paljon suurempia visioita tutkijakoulutukseen suuntautuvasta tiedekunnasta, jota nykytermein voisi sanoa tietoteknisesti painottuneeksi. Suunnitelman kulmakiviä olivat matematiikka, tilastotiede, tietojenkäsittelyoppi ja filosofia (logiikka ja tietoteoria), ja huomiota kiinnitettiin mm. sellaisiin tieteittenvälisiin aloihin kuin kommunikaatio- ja informaatioteoria, matemaattinen käyttäytymisteoria, matemaattinen talousteoria, operaatioanalyysi, psykolingvistiikka, päätösteoria, strukturaalinen kielitiede ja symbolinen logiikka. Ennen kaikkea haluttiin löytää vaihtoehto matematiikan perinteiselle kytkemiselle luonnontieteisiin [8]:

> Ehdotettu ... tiedekunta edustaa rakenteeltaan perinteellisestä poikkeavaa linjaa. Merkitseehän se esim. poikkeamista totunnaisesta matemaattis-luonnontieteellisen tiedekunnan ajatuksesta. Matematiikan ja luonnontieteitten kytkeminen samaan tiedekuntaan on kuitenkin käytännössä johtanut siihen, että matematiikan sovellutusalueina tässä ympäristössä on nähty etupäässä luonnontieteet ja yhteydet muille aloille ovat jääneet heikommiksi.

Kun nämä suunnitelmat rehtori Kolin innostuksesta huolimatta kariutuivat, Seppo siirtyi Helsingin yliopistoon jatkamaan uraauurtavaa työtään tilastollisten tietojenkäsittelyjärjestelmien kehittämisessä.

4. Tieteenalan muotoutumista

Tietojenkäsittelyopin korkeakouluopetuksen alkaminen ei tietenkään perustunut tieteellisiin näkökohtiin vaan käytännön tarpeisiin. Opetuksen kehittämisessä pyrittiin tämän vuoksi pysymään suhteellisen lähellä käytännön kysymyksiä. Siitä huolimatta oppiainetta pidettiin 1960-luvulla liike-elämän taholla turhan akateemisena ja käytännön todellisuudelle vieraana. Ilmeisesti vasta tätä oppiainetta opiskellut sukupolvi alkoi arvostaa alan tietojenkäsittelyopillisia valmiuksia. Näitä kysymyksiä koskevat lukuisat keskustelut heijastuvat myös virkaanastujaisesitelmässäni [12]:

> On ... avoimesti myönnettävä, ettei tällä uudella oppiaineella ole tukevia juuria akateemisessa ympäristössämme. Se ei ole kehittynyt yliopistoissamme niiden tieteitten kiinteässä yhteydessä, joihin sen tulisi olla henkisesti juurtunut, vaan se on lähinnä käytännön sovellutusten pakottamana tullut korkeakouluihin. Opetuksen aloittamisesta huolimatta ei vielä ole riittävästi syvennytty siihen mikä tämän oppiaineen aseman tulisi yliopistoissa olla. Eräs huomattava vaikeus on myös siinä, että saatavilla oleva opettajavoima on käytännön työssä koulutuksensa saanutta, eikä käytäntö opeta tietojenkäsittelyn teoreettisia ja loogis-matemaattisia perusteita. Kun lisäksi tiedetään, että tietojenkäsittelyn käytännön tehtävissä on jatkuvasti suuri henkilökunnan puute, ja monet muut oppiaineet kaipaavat tietojenkäsittelyoppia tukiaineekseen, on oppiaineella suuri vaara jäädä pintapuolisen tietojenkäsittelytekniikan opetukseksi tai kokoelmaksi irrallisia, erilaisiin sovellutuksiin käyttökelpoisia menetelmiä.

Opetuksen keskeisiä kohteita kaavailin seuraavaan tapaan kirjoituksessa [10], jossa kokeilen myös tanskalaisten käyttöönottaman termin 'datalogia' istumista suomenkieleen:

> Datalogian keskeisiä tarkastelukohteita ovat tietorakenteet ja niiden esittäminen tietokoneessa, algoritmiset menetelmät, ohjelmointikielet ... ja mutkikkaat tietojenkäsittelyprosessit. Erityisen keskeisiä tietojenkäsittelyprosesseja ovat ns. systeemiohjelmien ... suorittamat prosessit. Sovellutusten kannalta katsottaessa nämä saattavat tuntua teknillisiltä yksityiskohdilta, joissa voitaisiin tyytyä tietokoneitten valmistajien ja maahantuojien tarjoamiin standardiratkaisuihin.

Nykyhetken valossa on ehkä vaikea ymmärtää, että tuolloin joutui tavan takaa puolustelemaan, miksi opetettiin esimerkiksi ohjelmointikielten kääntämiseen ja ajoaikaiseen tukeen taikka käyttöjärjestelmien perusteisiin liittyviä asioita, kun pienessä maassa olisi voitu keskittyä pelkästään sovellusten kehittämiseen. On huomattava, ettei tietokoneiden valmistusta tuolloin myöskään pidetty edes potentiaalisena mahdollisuutena tämän kokoisessa maassa. Vuosikymmen ei kuitenkaan ehtinyt päättyä, kun Strömberg jo alkoi tietokoneen kehittämisen. Tietotekniikkamme jo tuolloin hyviä valmiuksia osoittavat tämän Selco 1000 -koneen omaperäinen ja erittäin uudenaikainen käyttöjärjestelmä sekä tehokas Algol-kääntäjä.

'Tietotekniikka' on vasta tämän vuosikymmenen termi, joka integroi mm. tietojenkäsittelyn, tietokoneet, mikroelektroniikan ja tietoliikenteen. Tämän käsitteen kannalta ala oli 1960-luvulla melkoisen sirpaloitu. Esimerkiksi tietojenkäsittelyopin ja tietokonetekniikan välillä ei juurikaan ollut yhteyksiä. Teknillisen korkeakoulun fysiikan osastolla rakennettiin tuolloin Reflac-konetta, ja vuosikymmenen vaihtuessa prof. Teuvo Kohonen aloitti siellä sittemmin kansainvälistä huomiota saaneet tutkimuksensa assosiatiivimuisteista.

Tietojenkäsittelyopin sisällä pyrittiin luomaan umpeen niitä kuiluja, joita käytännössä oli eri sovellusalueiden välille syntynyt. Niinpä prof. Lehto saattoi jo vuonna 1966 todeta [9], että "vielä viisi vuotta sitten selvänä pidetty jako tieteellis-teknilliseen ja kaupallis-hallinnolliseen tietojenkäsittelyyn on häviämässä". Yhteyksiä sovellettuun matematiikkaan ja tilastotieteeseen pyrittiin myös lähentämään. Hyvistä tarkoituksista huolimatta tässä oli omat vaaransa, joita opettajavoimien puute vielä kärjisti. Edellä siteeraamassani IBM Katsauksen artikkelissa [10] kirjoitan:

> Tietokonealan suuri henkilötarve ei missään tapauksessa saa johtaa sellaiseen yliopisto-opetuksen alennustilaan, että itsenäisen oppiaineen nimellä kulkeva opetus olisi epämääräinen ja pintapuolinen sekoitus ohjelmointia, matemaattisia menetelmiä, operaatiotutkimusta, systeeminsuunnittelua, organisaatio-oppia, eri sovellutusalojen erikoiskysymysten tarkastelua jne.

Suhde puhtaaseen matematiikkaan sen sijaan laimeni. Turussa prof. Salomaan johdolla kansainvälisestikin hyvin ansiokkaasti tutkitusta

formaalien kielten teoriasta tuli eräillä käytännön tahoilla suorastaan kirosana. Formaalien kielten tutkimusmotivaatiot olivatkin jo etääntyneet tietojenkäsittelystä, ja ajankohtaisemmiksi olivat tietojenkäsittelyopissa nousseet mm. ohjelmointikielten semantiikkaan ja käyttöjärjestelmiin liittyvät probleemat, jotka vaativat toisenlaisia teoreettisia perusteita.

Kun alan opetus luotiin 1960-luvulla tyhjästä, ja opiskelijamäärät olivat tavattoman suuret, on luonnollista, että tutkimustyö jäi tietojenkäsittelyopin laitoksissa suhteellisen vaatimattomaksi. Tutkijat olivat vielä myös kokemattomia, ja heidän tutkimusintressinsä hajaantuivat moniin suuntiin.

5. Tietotekniikan tutkimuksen konetarpeet

Tietotekniikan opetuksen ja tutkimuksen tietokonetarpeet eivät 1960-luvulla saaneet osakseen kovinkaan suurta ymmärtämystä. Vielä ei ollut se Karhusen [4] ennustama aika, jolloin tietokoneen "välttämättömyys" ymmärretään ja "kysymys ns. taloudellisuudesta menettää sisältönsä". Palatkaamme tässä kohden Suomen Kuvalehden kirjoitukseen [1] maaliskuussa 1958, jossa jo ihmeteltiin, miksi tietokoneita tarvitaan niin paljon:

> Mielenkiintoa herättää kuitenkin se, että samanaikaisesti maahan vuokrataan ulkomaisia matematiikkakoneita suorittamaan virastojen toivomia töitä. Oliko todella mahdotonta laatia yhteisiä suunnitelmia ja toteuttaa ne yhdessä?

Valtion tietokonekeskus (VTKK), joka 1960-luvulla toisaalta kauppasi koneaikaa korkeakouluille, toisaalta antoi lausunnot niiden konehankintaesityksistä, näki alan laitokset lähinnä akuutin työvoimatarpeen tyydyttäjinä. Visioita tietotekniikan tutkimuksen merkityksestä ei vielä ollut. Muistelmissaan [14] VTKK:n ylijohtaja Otto Karttunen tokaiseekin paljastavasti:

> Korkeampi opetus oli saanut leikkikalunsa, joita se niin hanakasti vaati. Kyseessä oli myös status: mikä se sellainen yliopisto on, jolla ei ole edes omaa tietokonetta!

Nykyään on selviö, että tietotekniikan opetus ja tutkimus tarvitsevat laboratoriovälineikseen tähän tarkoitukseen hyvin soveltuvia tietokoneita. Ne olivat 1960-luvulla kuitenkin kovin kalliita, eikä laitoksilla ollut

rohkeutta esittää omien koneiden hankintaa ulkomaisten esikuvien tapaan. Alan professorien kesken joskus kyllä pohdittiin sitä, miksi esimerkiksi hiukkaskiihdyttimet olivat ainakin periaatteessa hyväksyttäviä laboratoriovälineitä, mutta tietokoneet eivät.

Tampereen yliopistossa lähdettiin vuonna 1967 liikkeelle siitä, että tuolloin ajanmukaisella osituskäyttötietokoneella voitaisiin riittävässä määrin yhdistää sekä yleisen tietojenkäsittelypalvelun että tietojenkäsittelyopin laboratoriokoneen tarpeet. Vuoden 1968 lopulla saatu VTKK:n lausunto [11] osoittaa, minkälaisessa umpikujassa olivat yritykset luoda edellytyksiä tietotekniikkaa kehittävälle tutkimukselle:

> Kone soveltuu erityisesti mm osituskäyttöön sekä käyttöjärjestelmien tutkimiseen. Mikäli Tampereen yliopistolla on tarkoitus suunnata tietojenkäsittelyopin opetus ja tutkimus käyttöjärjestelmien kehittämiseen ja tietokoneen hyväksikäyttöön tieteellis-teknisessä käyttöympäristössä sekä mikäli yliopiston tietokonekapasiteetin lisäämisasia ratkaistaan yksinomaan yliopiston piirissä tapahtuvana kehittämisenä, on ehdotettu ... hankinta puollettavissa. Kun kuitenkin ...

Jatkossa VTKK esitti koneajan ostamista sen Tampereen aluekeskukselta. Yhteisten selvitysten perusteella yliopisto totesi tämän sille epätaloudelliseksi tavaksi hankkia toisenlaista tietokonekapasiteettia kuin mitä opetus ja tutkimus tarvitsivat, mutta lopulliseen lausuntoon [13] tämä ei vaikuttanut:

> Koska valtion tietokonekeskuksen Tampereen aluekeskuksessa on käyttämätöntä kapasiteettia, ... tietokonekeskus ei puolla Tampereen yliopiston esitystä ... Koska ... IBM S360/30 laitteistolla voidaan suorittaa tutkimusta ja opetusta, joka olisi erittäin hyödyllistä yhteiskunnalle ... niin tietokonekeskuksen mielestä ei ... haitta ole korvaamaton yhteiskunnalle eikä tietojenkäsittelyopin opetukselle.

Vielä tulikin kulumaan kauan, ennenkuin tarkoituksenmukaiset työvälineet tulivat tietotekniikan tutkimuksen ulottuville.

6. Loppukommentteja

Tarkoitukseni on ollut välittää joitain näkymiä tietotekniikan tutkimuksen alkuajoilta maassamme, ei arvioida tuolloin tehtyä työtä tai arvostella sen

ajan päätöksiä. Pari yleisluontoista kommenttia lienee nykyhetken valossa kuitenkin paikallaan.

Tietotekniikan valtavasta potentiaalisesta vaikutuksesta yhteiskuntaan ja kulttuuriin puhuttiin jo varhain. Suuria muutoksia saatiin kuitenkin odotella, ja mikrojen myötä alkanut todellinen vallankumous tuli aikanaan sittenkin jonkinlaisena yllätyksenä. Käytännön koulutustarpeeseen yhteiskunta reagoi nopeasti 1960-luvulla, mutta vain välitön hyväksikäyttö nähtiin tärkeäksi, ei alan syvällisempi kehittäminen. Tietojenkäsittelyalan omatkin visiot olivat rajoittuneita: se vähätteli teoreettista tietoa ja tietotekniikan tutkimusta, ellei pitänyt näitä suorastaan käytännölle haitallisina. Nähtiin, ettei esimerkiksi Kaapelitehtaan tutkimusmyönteisyys ollut johtanut toivottuun taloudelliseen tulokseen, ja kädet olivat täynnä välittömästi tehtävää arkista työtä.

Vuonna 1958 saatettiin Eskosta puhuttaessa huudahtaa [1], että "onneksi meillä Suomessa ollaan matematiikan alalla korkeimmalla kansainvälisellä tasolla". Matemaatikot olivatkin näkyvästi mukana: itse Rolf Nevanlinna toimi matematiikkakonekomitean puheenjohtajana 1950-luvulla, ja Olli Lehdolla oli laajakantoinen vaikutus 1960-luvun alussa, jolloin myös lukuisia nuoria matemaatikkoja siirtyi tietojenkäsittelyalalle. Lehdon huomio [9] johtoaseman siirtymisestä maahantuojille osoittaa kuitenkin epäsuorasti, etteivät lähtökohdat tietotekniikan tutkimukselle siinä vaiheessa olleet kovinkaan vahvat. Vaikka esimerkiksi STS:n luentomonisteesta [2] vuodelta 1961 ilmenee, että tietokoneiden kanssa työskenteli aktiivisesti myös hyviä soveltavia matemaatiikkoja, esimerkiksi tri Kari Karhunen, prof. Pentti Laasonen ja prof. Olli Lokki, oli matematiikan tämä puoli selvästikin ollut aliarvostettua puhtaan matematiikan rinnalla. Tällä on ilmeinen yhteys myös siihen, että kaupallis-hallinnollinen tietojenkäsittely sai meillä dominoivamman aseman kuin yleensä muualla.

Lehdon vaikutus Kaapelitehtaan kautta tuli jälkikäteen ajatellen varsin kriittisessä vaiheessa. Kaiken kaikkiaan päästiin nyt siihen, että korkeakoulujen voimistuessa niillä oli käytettävissään henkilöitä, joilla oli sekä käytännön kokemusta että halua ottaa käytännön näkökohdat huomioon opetuksessa ja tutkimuksessa. Tämän eräänä seurauksena ei kaupallis-hallinnollinen tietojenkäsittely jäänyt korkeakouluissamme eroon tietojenkäsittelyopin kansainvälisistä valtavirtauksista, niinkuin monissa

muissa maissa näytti samaan aikaan tapahtuvan. Uuden tieteenalan identiteetti pääsi myös kehittymään ilman liian vahvoja sidoksia numeeriseen laskentaan.

Keskustelu tietotekniikan eri osa-alueiden olemuksesta ja niiden keskinäisistä suhteista on 1990-lukua lähestyttäessä edelleen ajankohtaista. Nyt sille on 1960-lukuun verrattuna kuitenkin aivan toisenlainen pohja, kun tietotekniikan tutkimus on vauhdittunut, ja soveltavat tieteenalat ovat sen suhteen kutakuinkin omavaraisia.

Viitteet aikajärjestyksessä

1. Osmo Mäkeläinen, Eskosta uusi Sampo. *Suomen Kuvalehti* n:o 11, 15.3.1958, 11-13.
2. *Tietokoneet*. Suomen Teknillisen Seuran täydennyskoulutuskurssi n:o 26, tammikuu 1961.
3. Pentti Laasonen, Tietojenkäsittelyn alueen kartoitusta. Viitteessä 2.
4. Kari Karhunen, Tulevaisuuden näköalat. Viitteessä 2.
5. Reino Kurki-Suonio, Tietokone, tutkimustyötä innoittava apuväline. *Arkhimedes* n:o 1, 1962, 18-23.
6. Hans Andersin, Tulevaisuuden näkymiä. Luento Suomen Teknillisen Seuran täydennyskoulutuskurssilla n:o 38 (STS:n julkaisu n:o 40), *Perustietoja tietokoneista*, syyskuu 1962.
7. Yhteiskunnallisen korkeakoulun taloudellis-hallinnollisen tiedekunnan, hallinnon instituutin, tutkimuslaitoksen, tutkijakoulun ja tietokonekeskuksen kehityssuunnitelmat, 15.2.1965.
8. Yhteiskunnallisen Korkeakoulun kehittämissuunnitelmat vv. 1966-1980, 15.10.1965.
9. Olli Lehto, Huomatuksia tietojenkäsittelyopin kehityksestä maassamme. *Abacus* (Suomen Kaapelitehdas Oy:n elektroniikka- ja tietokoneosastojen tiedotuslehti) n:o 3, 1966.
10. Reino Kurki-Suonio, Keskeisenä kulttuurimme kehityksessä – tietojenkäsittelyoppi eli datalogia itsenäisenä oppiaineena ja tieteenhaarana. *IBM Katsaus* n:o 2, elokuu 1967, 6-8.

11. Otto Karttunen, VTKK:n lausunto Tampereen yliopiston tietokonehankkeesta, 3893/Y/68, 11.11.1968.
12. Reino Kurki-Suonio, Tietojenkäsittely yliopistossa. Virkaanastujaisesitelmä Tampereen yliopistossa 30.11.1968. *Suomalainen Suomi Valvoja* n:o 4, 1969.
13. Otto Karttunen, VTKK:n lausunto Tampereen yliopiston tietokonehankkeesta, 3893/Y/68, 13.8.1969.
14. Otto Karttunen, *Avainpaikalla tietotekniikan kehityksessä*. Suomen ATK-Kustannus Oy, 1986, 110.

PROFESSORI REINO KURKI-SUONIO
TAMPEREEN TEKNILLINEN KORKEAKOULU
OHJELMISTOTEKNIIKAN LAITOS
PL 527
33101 TAMPERE
rks@tut.fi

Tietojenkäsittelyoppi eilen, tänään ja huomenna

MARTTI TIENARI, HELSINGIN YLIOPISTO

Kun 1970-luvulla opetushallinnon piirissä Suomessa yritettiin ohjata yliopistoja vaatimalla tavoitteellisuutta opetuksessa ja tutkimuksessa, yritettiin myös Helsingin yliopiston matemaattis-luonnontieteellisen osaston piirissä sorvailla tieteiden sisällöllisiä määritelmiä. Matemaatikot eivät aluksi päässeet paljoa pitemmälle kuin toteamukseen "Matematiikka on sitä, mitä matemaatikot tutkivat ja opettavat". Fysiikka ja kemia ovat jo hieman helpommin määriteltävissä. Miten on tietojenkäsittelyopin laita?

Laitoksemme opinto-oppaassa 1988-89 hahmotetaan tietojenkäsittelyoppia sanomalla: "Tietotekniikan käyttö tieteessä, tekniikassa, hallinnossa ja elinkeinoelämässä aiheuttaa koulutus- ja tutkimustarvetta tietojen automaattisen käsittelyn alalla. Yliopistollinen oppiaine, tietojenkäsittelyoppi, pyrkii tyydyttämään tätä tarvetta". Oppiaineen luonnehdinta ei ole tältä osin paljon muuttunut vuodesta 1969, joskin tuolloin vastaava teksti alkoi pioneerihengessä: "Tietokoneiden voimakkaasti laajeneva käyttö ...". Tosiasia on, että tietojenkäsittelyoppi on saanut alkunsa yhteiskunnan tarpeista. Oppiaineen vahva asema perustuu juuri atk-alan erilaisten asiantuntijoiden tarpeeseen, joka tuntuu yhä kasvavan sekä määrän että laatuvaatimusten osalta.

Edellä hahmottelemani lähtökohta tietojenkäsittelyopin kehittymiselle itsenäiseksi tieteenalaksi on käsittääkseni yleisesti hyväksytty. Tietojenkäsittelyopin luonnehtimista sovelletun matematiikan, elektroniikan tai hallintotieteiden erikoisalaksi tapaa nykyisin yhä harvemmin. Alan

kehityksen alkuaikoina 1960-luvulla tietojenkäsittelyopin itsenäisyys ja omaleimaisuus ei kuitenkaan ollut mitenkään itsestään selvää.

Jos tietojenkäsittelyoppia pyrkii luonnehtimaan edellä esitettyä syvällisemmin, joutuu jo tiettyihin vaikeuksiin. Meidän opinto-oppaamme on jo yli 20 vuotta väittänyt: "Tietojenkäsittelyoppi ei ole pelkästään yhdistelmä tietokoneen sovelluksista eri probleemoihin. Aineen keskeisin osa on matematiikan tavoin sovelluksista riippumaton ja melko abstrakti". Tätä erään amerikkalaisen 'computer science' -pioneerin luonnehdintaa oppiaineestamme eivät kaikki Suomessa hyväksy. Monet oppilaittemme välittömiä ammatillisia valmiuksia painottavat tahot haluaisivat atk:n sovelluksille voimakkaamman aseman opetuksessa ja tutkimuksessa kuin mitä edelläoleva luonnehdinta edellyttää.

Olemme toistaiseksi kosketelleet vasta epäsuoria luonnehdintoja tietojenkäsittelyopille. Voisiko tutkimuskohdettamme määritellä suoraan? Opinto-oppaamme sisältää kaksi kansainvälisistä lähteistä peräisin olevaa luonnehdintaa. Vaikka en ole näihin määrittelyihin enää kovin tyytyväinen – 1970-luvulla ne tuntuivat vielä osuvilta – esitän ne tässä kuitenkin parempien määritelmien puuttuessa: (1) Tietojenkäsittelyoppi on systemaattista ja monitieteellistä tiedon rakenteeseen, talletukseen, siirtämiseen ja muuntamiseen kohdistuvaa tutkimusta. (2) Tietojenkäsittelyoppi tutkii tietokoneella toteutettavien algoritmien suunnittelua, analyysiä, esitystapaa ja sovelluksia.

1. 1960-luvun muistoja

Kun tietojenkäsittelyopin ensimmäisiä professuureja perustettiin Suomeen, vuonna 1965 Tampereen yliopiston taloudellis-hallinnolliseen tiedekuntaan, vuonna 1967 Helsingin yliopiston matemaattis-luonnontieteelliseen osastoon, vuonna 1967 Jyväskylän yliopiston yhteiskuntatieteelliseen tiedekuntaan ja vuonna 1968 Teknillisen korkeakoulun koneinsinööriosastoon, oli lähtökohtana asiantuntijoiden kouluttaminen nopeasti yleistyvän hallinnollisen tietojenkäsittelyn tarpeisiin. Vuonna 1964 perustetun Valtion tietokonekeskuksen aktiivinen ja vaikutusvaltainen johtaja Otto Karttunen oli mukana taustavoimana tukemassa

korkeakoulujen professuuriesityksiä. Mainittakoon, että Helsingin yliopistossakin oltiin kahden vaiheilla, perustaako uusi professuuri valtiotieteelliseen tiedekuntaan tilastotieteen ja hallintotieteiden yhteyteen vai matemaattis-luonnontieteelliseen osastoon.

Alan alkuvaiheissa 1960-luvulla vallitsi atk-alalla selvä kahtiajako. Puhuttiin kaupallis-hallinnollisesta atk:sta ja tieteellis-teknisestä atk:sta. Edellinen ala ryhtyi käyttämään ohjelmointiin COBOL-kieltä, jälkimmäinen FORTRAN-kieltä. Toisinajattelijat harrastivat lisäksi Algol60-kieltä! Ruotsissa tämä kahtiajako jopa sinetöitiin professuurinimikkeisiin 'informationsbehandling särskilt administrativ databehandling' ja 'informationsbehandling särskilt numerisk analys'. Tietojenkäsittelyoppi nähtiin tuolloin pääasiassa tiettyjen sovellusten atk-tarpeiden valossa, ei vielä itsenäisenä tieteenä. Suomessa käytetty väljempi professuurien alan määrittely on osoittautunut alan nopeasti kehittyessä joustavammaksi kuin Ruotsissa sovellettu rajoittavampi menettely.

Suomessa oltiin sopivan aikaisin liikkeellä tietojenkäsittelyopin professuureja perustamassa. Ei voi väittää, että olisimme tuolloin viivytelleet kansainvälisen kehityksen seuraamisessa. Tosin meiltä puuttuivat USA:n ja Englannin 1940-luvulla käynnistyneet tietokoneprototyyppien kehitysprojektit, joita vastaavat hankkeet Ruotsissa ja Tanskassa käynnistyivät jo 1950-luvun alussa. Akateemikko Rolf Nevanlinnan aloite 1950-luvun puolivälissä Matematiikkakonekomitean perustamiseksi ja sen toimesta rakennettu ESKO-tietokone olivat merkittäviä ensi askeleita tietokonetekniikan tietämyksen kehittymisessä maassamme.

Tärkeää osaa tietojenkäsittelyalan tiedemiesten kasvattamisessa näytteli vuonna 1960 perustettu Suomen Kaapelitehtaan (nykyisin Nokia Oy) elektroniikkaosasto, jossa moni alan ensimmäisistä professoreista (mm. Kurki-Suonio, Mustonen, Peltola, Tienari) pääsi akateemikko Olli Lehdon johdolla alkuun atk-alalla. IBM oli 1960-luvulla mahtitekijä, suhteellisesti vielä mahtavampi kuin nykyisin. Sen piirissä hankkivat atk-tietoa ja kansainvälistä näkemystä atk-professoreistamme ainakin Andersin (joka on myös Matematiikkakonekomitean pioneereja) ja Kerola.

Atk-sovellukset, pääasiassa tieteellis-tekniset, olivat oman mielenkiintoni kohteena 1960-luvulla. Olin oppinut soveltamaan matematiikkaa professori Olli Lokin opastuksella; myös fysiikan opinnoistani oli työssäni

paljon hyötyä. Kehittelin Kaapelitehtaan laskentakeskuksessa yhdessä Seppo Mustosen kanssa tilastomatemaattisia ohjelmistoja, joilla palvelimme sekä yliopistojen että elinkeinoelämän tutkijoita. Ohjelmointi oli 1960-luvun alussa kovin työlästä. Vähitellen saimme vuosikymmenen puoliväliin mennessä ohjelmointikielet käyttöömme. Talven 1966-67 vietin Stanfordin yliopiston Computer Science -laitoksessa oppimassa tietojenkäsittelytiedettä. Se oli silloin vielä pääasiassa numeerista analyysiä. Stanfordista palattuani ryhdyin 1.9.1967 alkaen hoitamaan Helsingin yliopiston uutta tietojenkäsittelyopin professuuria. Tutkimusalakseni muotoutui viiden ensimmäisen professorivuoteni ajaksi numeeristen algoritmien pyöristysvirheteoria.

2. 1970-luvun muistoja

Tietojenkäsittelyoppi kehittyi voimakkaasti 1970-luvulla. Nousi esiin laajoja tutkimusaloja, joilla ei ole suoranaista palveluasemaa sovelluksiin, kuten ohjelmointikielet ja kääntäjät, algoritmit ja tietorakenteet, käyttöjärjestelmät ja tietokannat. Saksalaiset nimittivät näitä aloja nimellä 'Kerninformatik' ('ydintietojenkäsittelyoppi'), Skandinaviassa ryhdyttiin käyttämään sanaa 'datalogi'. Suomessa tälle tietojenkäsittelyopin suuntaukselle oli teoreettista pohjaa luonut erityisesti Turun yliopiston matematiikan laitos professori Arto Salomaan johdolla. Tampereella professori Reino Kurki-Suonio suuntautui jo 1960-luvulla datalogisesti.

Helsingin yliopistossa kehitettiin 1970-luvulla tietojenkäsittelyopin opetusta ja tutkimusta datalogisessa hengessä pitäen silmällä aineen itsenäistä asemaa tieteenä. Hallinnollisten atk-sovellusten pariin työelämään pyrkiviä tuettiin järjestämällä ammatillisesti suuntautuneita erikoiskursseja (mm. COBOL-ohjelmointi, tietojärjestelmät, atk-toiminnan johtaminen). Pääideana oli kuitenkin painottaa tietojenkäsittelyoppia itsenäisenä tieteenä ja jättää atk:n soveltamistaidot suurelta osin sivuaineopintojen ja työpaikkakoulutuksen varaan. Atk-sovellusten moninaisuus tekee niiden syvemmän huomioinnin tietojenkäsittelyopin opetuksessa vaikeaksi.

Omaksi opetus- ja tutkimusalueekseni muotoutui 1970-luvulla ohjelmointikielet ja niiden kääntäjät. Tämä valinta oli alan kansainvälisen

kehityksen kannalta ehkä hieman jälkijättöinen, mutta osoittautui laitoksen opetuksen ja tutkimuksen kehittämisen kannalta onnistuneeksi. Saimme luoduksi tällä tutkimusalueella hyvät kansainväliset kontaktit. Suomen Akatemian tuki kääntäjätekniikan tutkimusryhmällemme ja nuoret lahjakkaat jatko-opiskelijat, jotka ansaitsivat kannuksensa tässä tutkimustyössä, tekivät mahdolliseksi nousta suhteellisen nopeasti kansainvälisen tutkijayhteisön tietoisuuteen tällä yhä edelleen elinvoimaisella tietojenkäsittelyopin tutkimusalalla.

Tietojenkäsittelyoppi muodostui 1970-luvulla varsin suosituksi oppiaineeksi. Opiskelijamääriä jouduttiin voimakkaasti rajoittamaan elinkeinoelämän laajasta henkilöstötarpeesta huolimatta. Alan laitosten opettajamäärien kasvu pysähtyi tuolloin pitkäksi ajaksi valtion menotalouden voimakkaiden säästötoimien vuoksi. 1970-luku oli tietojenkäsittelyopin sisäisen voimakkaan kehityksen aikaa: opetuksen ja tutkimuksen taso nousi ja ala vakiinnutti asemansa korkeakouluissa. Tietojenkäsittelyoppi saavutti vähitellen merkittävän aseman myös sivuaineena. Luulen tällä olleen Suomessa alan kehittymisen kannalta vahintään yhtä suuren merkityksen kuin pääaineopiskelulla.

3. 1980-luvun kehityksestä

1980-luvun alussa aloittivat henkilökohtaiset mikrotietokoneet voittokulkunsa Suomessa. Atk-ala muuttui nopeasti. Suurten keskitettyjen laskentakeskusten sijasta kehitys alkoi suuntautua kohti hajautettua tietojenkäsittelyä. Tietoliikenne tuli elimelliseksi osaksi atk:ta. Valmisohjelmistot tulivat yhä tärkeämmiksi. Kaikki tämä merkitsi valtavaa atk-henkilökunnan lisätarvetta ja myös tietojenkäsittelyopille uusia haasteita. Valtiovaltakin havahtui, tosin melkoisella viiveellä. Vasta valtiovarainministeriön järjestelyosaston piirissä toimivan Tietotekniikan neuvottelukunnan kiinnitettyä Opetusministeriön huomiota atk-opetuksen huomattavaan laajentamistarpeeseen saatiin lisää resursseja tietojenkäsittelyopin opetukseen ja tutkimukseen.

Suomessa olivat teknilliset korkeakoulut ja kauppakorkeakoulut vielä 1970-luvulla yliopistoihin verrattuina yllättävän pidättyviä

soveltamista. Tietojenkäsittelytieteen peruskysymys on: 'Mitä voidaan automatisoida (tehokkaasti)?'".

Amerikkalainen työryhmä pelkistää tietojenkäsittelytieteen keskeisen sisällön yhdeksään osa-alueeseen, joilla heidän käsityksensä mukaan on kullakin jo identifioitavissa omat tutkijayhteisönsä. Näillä kullakin alueella on jo tietty yhtenäinen käsitteistö, riittävä teoreettinen pohja, kehittyneet yleisesti hyväksytyt mallit asioiden abstraktiin käsittelyyn sekä pitkälle kehittynyt käytännön suunnittelu- ja toteutustaito. Kultakin yhdeksältä alalta he pelkistävät raportissaan mielenkiintoisella tavalla yksityiskohtaisiksi luetteloiksi alan teoriataustaa, yleisesti hyväksyttyjä malleja ja abstraktioita sekä toteutuksissa ja suunnittelussa käytettäviä vakiintuneita menetelmiä.

ACM:n työryhmän määrittelemät yhdeksän tietojenkäsittelytieteen perusaluetta ovat seuraavat:

1. Algoritmit ja tietorakenteet
2. Ohjelmointikielet
3. Laitteistoarkkitehtuuri
4. Numeerinen ja symbolinen laskenta
5. Käyttöjärjestelmät
6. Ohjelmointimetodologia ja -tekniikka
7. Tietokannat ja tiedon jäljitys
8. Tekoäly ja robotiikka
9. Ihmisen ja koneen yhteistyö

Luettelossa kiinnittää huomiota, paitsi informaatiojärjestelmien vähäinen painotus, se, että hajautusta ja rinnakkaisuutta ei ole identifioitu omaksi tutkimuksen pääalueekseen. Työryhmä mainitsee, että he katsovat rinnakkaisuuden sisältyvän useaan heidän määrittelemistään tutkimusaloista. Heidän määrittelemänsä tutkimusala 'numeerinen ja symbolinen laskenta' on Suomessa pääasiassa sovelletun matematiikan professorien hoidossa.

Suomessa edellä siteerattu amerikkalainen raportti tulee epäilemättä vaikuttamaan tietojenkäsittelyopin opetuksen ja tutkimuksen muotoutumiseen. Meillähän pyritään yleensä tieteessä seuraamaan kansainvälisiä virtauksia. Suomessa alan tutkimuksen tuleviin painotuksiin vaikuttaa

paljon myös se, miten suomalainen tietotekninen teollisuus tulevaisuudessa menestyy maailmalla. Asian voi ilmaista näin, mutta sen voisi ilmaista myös päinvastoin: Tietojenkäsittelytieteen tutkimusta pitää kehittää niin, että suomalainen tietoteknillinen teollisuus menestyy maailmalla. Sen vuoksi tarvitaan yhä vielä kohtuullisessa määrin lisää resursseja sekä tutkimukseen että koulutukseen.

Viitteet

1. Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, Computing as a discipline. *Comm. ACM* **32**, 1 (January 1989), 9-23.

PROFESSORI MARTTI TIENARI
HELSINGIN YLIOPISTO
TIETOJENKÄSITTELYOPIN LAITOS
TEOLLISUUSKATU 23
00510 HELSINKI
tienari@cs.Helsinki.FI

Pitääkö tietoa mallittaa?

HEIKKI MANNILA, HELSINGIN YLIOPISTO

1. Johdanto

Tiedon[1] mallittamisella tarkoitetaan tietotekniikassa tietojen rakenteen kuvaamista. Rakennetta voidaan käyttää tietojen systemaattisessa käsittelyssä ja ylläpidossa.

Tarkastellaan esimerkkinä pienen kirjaston lainaustoimintaan liittyviä tietoja. Oletetaan, että näiden tietojen käsittelyyn ja talletukseen käytetään tietojenkäsittelyjärjestelmää. Esimerkkejä kirjastossa käsiteltävistä ja tallennettavista tiedoista ovat seuraavat.

V. Virtanen on lainannut teokset Seitsemän veljestä ja Nummisuutarit 7.3.1989.

L. Lahtinen on lainannut teoksen Rautatie 2.2.1989.

V. Virtanen on tilannut teoksen Rautatie 10.3.1989.

---

[1]Lienee syytä mainita terminologinen ongelma, joka usein tulee esiin. Sanojen 'tieto' ja 'informaatio' merkitykset ovat tietotekniikassa ja yhteiskuntatieteissä harmillisesti vastakkaiset. Tietotekniikkaan on juurtunut käytäntö, jossa 'tieto' tarkoittaa tulkitsematonta tietoa: kirjaimia ja numeroita paperilla tai tietokoneen muistissa. Tämä käyttötapa esiintyy esimerkiksi sanassa tietokone. Informaatiolla taas tarkoitetaan tietotekniikassa sellaista tietoa, joka on jollakin ihmisellä; tieto muuttuu informaatioksi, kun joku vastaanottaa tiedon. Yhteiskuntatieteissä on tapana käyttää samoja termejä täsmälleen päinvastaisissa merkityksissä. Informaatio on tulkitsemattomia merkkejä, tieto jonkun ihmisen tietämää informaatiota. Hyvä yleisesitys käsitteistöstä on Ilkka Niiniluodon teos *Informaatio, tieto ja yhteiskunta* [1].

V. Virtanen palautti teoksen Nummisuutarit 15.3.1989.

Kirjastossa tarvitaan siis tietoja lainauksista, palautuksista ja varauksista. (Todellisessa kirjastossa tarvitaan toki tietoja monista muistakin asioista, kuten hankinnoista, kirjojen tekijöistä ym.)

Tarkastelen seuraavassa tiedon mallittamisen perusteita ja annan joitakin esimerkkejä tämän alueen suuntauksista.

2. Mikä on tietomalli?

Tietomalli on tapa ryhmitellä ja kuvata tietoja, joita tietojenkäsittelyjärjestelmässä talletetaan ja käsitellään. Lisäksi malliin kuuluvat perustoiminnot, joilla tietoa haetaan ja muutetaan.

Yksi hyvin tunnettu ja laajalti käytetty tietomalli on ns. *relaatiomalli*. Sen perusajatus on tietojen ryhmittely taulukoiksi, joissa on nimetyt sarakkeet.

Kirjastoesimerkin lainatiedot voitaisiin esimerkiksi ryhmitellä taulukkoon, jossa on kolme saraketta: lainaaja, lainattu kirja ja lainauspäivä.

Lainaaja	Teos	Lainauspäivä
V. Virtanen	Seitsemän veljestä	7.3.1989
V. Virtanen	Nummisuutarit	7.3.1989
L. Lahtinen	Rautatie	2.2.1989

Lainaajien osoitteet voitaisiin tallettaa taulukkoon, jossa on vain kaksi saraketta.

Lainaaja	Osoite
V. Virtanen	Kalliolinnantie 8
L. Lahtinen	Puistokatu 2

Palautustiedot voitaisiin tallettaa taulukkoon, jossa olisi kolme saraketta: palauttaja, palautettu teos, ja palautuksen päivämäärä.

Palauttaja	Teos	Palautuspäivä
V. Virtanen	Nummisuutarit	15.3.1989
L. Lahtinen	Rautatie	15.4.1989

Relaatiomalliin liittyvillä tiedonkäsittelyoperaatioilla voidaan valita taulukoista tietyt ehdot täyttäviä rivejä, jättää sarakkeita pois ja yhdistellä eri taulukoiden tietoja. Yhdistämällä lainaus- ja palautustiedot voidaan esimerkiksi etsiä ne teokset, jotka on lainattu mutta joita ei ole palautettu. Yhdistämällä näin saatuun (jälleen taulukkomuotoiseen) tietoon lainaajien osoitetiedot saadaan postituslista esimerkiksi karhukirjeitä varten .

Toinen tunnettu tietomalli on ns. *hierarkkinen malli*. Sitä käytettäessä tiedot ryhmitellään luetteloiksi. Esimerkissämme voisimme vaikkapa ryhmitellä lainaustiedot pitkäksi luetteloksi, jossa on peräkkäin kunkin lainaajan lainaamien kirjojen nimet ja niiden lainauspäivät:

LAINAT
- V. Virtanen
  - Seitsemän veljestä 7.3.1989
  - Nummisuutarit 7.3.1989
- L. Lahtinen
  - Rautatie 2.2.1989

Hierarkkisessa mallissa lainaajan tiedot ja lainatut kirjat ovat jossakin määrin eri asemassa; relaatiomallin mukaisessa tietojen ryhmittelyssä ne ovat symmetrisiä toisiinsa nähden.

Tietomallin tärkeimpiä ominaisuuksia ovat luonteviuus ja helppokäyttöisyys. Mallilla pyritään kuvaamaan todellisen maailman tietoja: ryhmittelemään niitä siten, että tietojen ylläpito ja käsittely olisi helppoa. Jotta näin olisi, mallin täytyy perustua yksinkertaisille ja luonteville käsitteille. Relaatiomalli on hyvä esimerkki tällaisesta mallista: sen peruskäsite, taulukko, on kaikille ruutupaperin käyttäjille tuttu. Samoin hierarkkisen mallin perusajatus on selkeä.

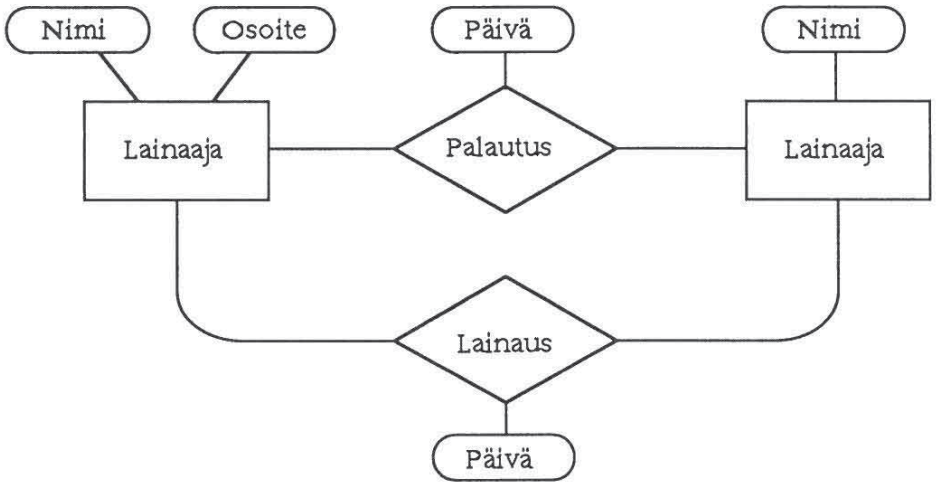
Toinen tietomallin hyvä ominaisuus on voimakkuus, eli kyky kuvata tietojen monimutkainenkin rakenne, ja kyky määritellä yksinkertaisesti monimutkaisiakin operaatioita tiedoilla. Relaatiomalliin kuuluu viisi perusoperaatioita, joita yhdistelemällä voidaan ilmaista suurin osa haluttavista toimenpiteistä.[2]

---

[2]Käytännössä usein esiintyvistä perusoperaatioista ns. transitiivinen sulkeuma on ainoa, jota ei voida ilmaista relaatiomallin peruskäsitteillä.

Tietomalleja on kehitetty lukuisia. Kolme 1970-luvulla vakiintunutta mallia ovat relaatiomalli, hierarkkinen malli ja ns. verkkomalli, joka on hierarkkisen mallin eräänlainen yleistys. Verkkomalli oli 1970-luvulla kenties käytetyin tietomalli; ainakin sitä käyttävät tiedonhallintajärjestelmät olivat suosittuja. Relaatiomallin suosio on kasvanut voimakkaasti 1980-luvulla, kun henkilökohtaiset tietokoneet ovat tehneet mallien yksinkertaisuudesta entistä tärkeämmän tekijän ja kun relaatiomalliin perustuvat järjestelmät on opittu toteuttamaan tehokkaasti.

Tietojärjestelmien suunnittelussa on laajasti käytössä ns. *olio-suhde -malli* (eli ER-malli, sanoista 'entity' ja 'relationship'). Tässä mallissa tiedot kuvataan olioina ja olioiden välisinä suhteina. Tietojen rakenne esitetään usein kuvana. Kirjastoesimerkissä voidaan kirjoja ja lainaajia pitää olioina, joiden välillä on kaksi suhdetta: lainaus ja palautus. Lainaajien osoitteet, kirjojen nimet sekä lainaus- ja palautuspäivät ovat olioiden ja suhteiden attribuutteja. Kuvassa 1 on esitetty kirjaston tietojen kuvaus ER-mallia käyttäen. Laajan tietokokoelman mallittamisessa käytetään usein ensin ER-mallia, ja sen jälkeen siirrytään esimerkiksi relaatiomalliin.



Kuva 1. Kirjaston lainaustietoja kuvaava ER-kaavio.

3. Uudet tietomallit

Perinteiset tietomallit sopivat hyvin kirjastojen tai varastokirjanpidon kaltaisten järjestelmien tietojen mallintamiseen. Näissä tiedot ovat useimmiten lukuja, ehkä joitakin lyhyitä kirjainjonoja, kuten henkilöiden tai tuotteiden nimiä. Tietoja voi olla hyvinkin monta eri tyyppiä, mutta tyyppikokoelma on kohtuullisen stabiili; uudenmuotoisia tietoja ei järjestelmään jouduta ottamaan mukaan kovinkaan usein.

Tähän asti olemme lähinnä tarkastelleet tietoja, jotka ovat muodoltaan yksinkertaisia väitelauseita: "V. Virtanen lainasi Nummisuutarit". Yleisesti tällainen lause sanoo, että jollakin oliolla on tietty ominaisuus tai että kahden[3] olion välillä vallitsee tietty suhde. Tietojenkäsittelyterminologiassa tällaisia lauseita kutsutaan faktoiksi.

Tietojärjestelmässä tarvitaan ja käytetään myös monimutkaisempaa tietoa, esimerkiksi sääntöjä: "kirjojen laina-aika on kuukausi". Relaatiomallin tasolle vietynä tämä sääntö saa muodon "lainauksia koskevassa taulukossa olevan lainapäivän ja palautus-taulukossa olevan palautuspäivän väli saa olla enintään 30 päivää".

Tietämyksen esittämisen perusajatus on, että säännöt ovat tietoa siinä kuin faktatkin. Niitä voidaan käsitellä, ryhmitellä, tallettaa ja muuttaa samaan tapaan kuin yksinkertaisempiakin lauseita.

Tietämyksen esittäminen on yksi 1980-luvun uusi tietomalliajatus (vaikka idea sinällään on peräisin ainakin 1960-luvulta). Esimerkiksi ns. asiantuntijajärjestelmät (joita olisi parempi kutsua taitotukijärjestelmiksi) perustuvat siihen ajatukseen, että säännöt ovat tärkeä tapa esittää ja kuvata tietoa. Tietomallin kannalta asiantuntijajärjestelmissä ei ole mitään kovin erikoista.

Tietämyksen esittämisen ja sääntöpohjaisten järjestelmien suosion taustalla on tekoälytutkimuksen uusi nousu 1980-luvulla; alue oli 1960-luvulla suosittu, mutta pimennossa ja väheksytty 1970-luvulla.

---

[3]tai useamman

Epävarman tiedon esittäminen on kiintoisa tapa laajentaa tietomalleja. Moni todellisen maailman tiedoista on epävarma. Tietoihin voidaan liittää niiden varmuutta kuvaavia todennäköisyyksiä. Tällöin tosin tietokannan käsittely vaikeutuu: ei ole mitenkään selvää, millä tavalla todennäköisyyksiä pitää käsitellä tietoja yhdisteltäessä.

Tietomallin tehtävänä on tiedon jäsentäminen käsittelyä varten. Pelkkä tiedon rakenteen kuvaaminen ei tähän riitä. Tietoalkioille halutaan usein tehdä joitakin operaatioita: lukuja halutaan laskea yhteen, nimiä halutaan lajitella aakkosjärjestykseen jne. Perinteisissä tietomalleissa operaatiot olivat joko valmiiksi määriteltyjä (kuten yhteenlasku) tai sitten ne määriteltiin tiedon rakenteesta erillään, useimmiten tietoja käsittelevien ohjelmien yhteydessä. Jälkimmäisestä sopii esimerkiksi päivämäärien operaatio "onko annettu päivä pyhäpäivä". Tällainen tiedon käsittely vaatii erikoistietoa ympäristöstä.[4] Operaatioiden kuvaaminen tietojen rakenteesta erillään on hankalaa ja voi johtaa epäjohdonmukaisuuksiin. *Olio-suuntautuneet tietomallit* mahdollistavat tiedon rakenteen ja käsittelyoperaatioiden määrittelyn yhtäaikaa. Nämä tietomallit liittyvät olioperustaiseen ohjelmointiin, joka on samaten 1980-luvulla suosituksi tullut ohjelmointitapa. Ne ovat erityisen tarpeellisia esimerkiksi koneiden ja laitteiden tietokoneavusteisessa suunnittelussa, jossa erilaisia operaatioita on paljon.

Tekstitiedon mallittaminen on yksi tärkeä tiedonhallinnan ongelma. Suuri osa tietokoneilla käsiteltävästä tiedosta on nykyään tekstiä, jota kirjoitetaan, muokataan ja luetaan tekstinkäsittelyjärjestelmillä. Niissä teksti on yleensä vain jono kirjaimia ja numeroita, joilla ei ole minkäänlaista sisäistä rakennetta. Tämä voi tehdä tekstin käsittelyn vaikeaksi. Monissa teksteissä (vaikkapa sanakirjoissa, ohjelmistojen käyttöohjeissa tai valtion tulo- ja menoarvioesityksessä) on paljon muodollista rakennetta (jokaista sanakirjan sanaa seuraa ääntämisohje, merkityksen selitys, käyttöesimerkit jne.). Tällaista rakenteista tekstiä kirjoitettaessa on rakennetta seurattava, ja sitä käytetään hyväksi myös tietoa etsittäessä.

Hyperteksti on 1980-luvun lopussa suurta suosiota saanut ajatus, joka yksinkertaisesti sanottuna tuo viittausten käyttömahdollisuuden

---

[4]Esimerkiksi 6.12. on pyhäpäivä Suomessa, mutta ei monissa muissa maissa.

tekstinkäsittelyjärjestelmiin. Hypertekstijärjestelmissä käytetään siis tietomallia, jonka mukaan teksti koostuu kirjaimista ja viittauksista tekstin kohdista toiseen. Yleensä näissä järjestelmissä on kahdenlaisia viittauksia: hierarkkisen rakenteen kuvaavat viittaukset (esimerkiksi sisällysluettelosta kuhunkin kirjan lukuun) ja kommenttiviittaukset, jotka vastaavat tekstissä olevia ala- tai lähdeviitteitä tai muotoa "katso luku 3.4" olevia viittauksia.

4. Tiedon mallittaminen tietojenkäsittelytieteen tutkimusalueena

Tiedon mallittaminen on keskeinen tietojenkäsittelytieteen tutkimusalue. Sen merkitys on lisääntynyt, kun ohjelmoinnin tärkeys ja vaikeus ovat sovelluskehittimien ansiosta vähentyneet. Alan sovelluksia ajatellen voi tiedonhallinnan ja mallittamisen merkityksen kasvua ehkä havainnollistaa sillä, että yritysten atk-yksiköissä on yhä useammin tietohallintopäällikkö, joka on korvannut atk-päällikön.

Tiedonhallinta ja tietomallit sisältävät tutkimuksen kannalta paljon erilaista materiaalia. Esimerkiksi tiedon eri esitysmuotojen tiiviys ja operaatioiden vaatiman ajan laskenta on hyvin selkeästi matemaattista ja algoritmista aluetta. Tietomallien helppokäyttöisyyden arviointi ja organisaation tietohallinnon perusteiden pohtiminen taasen ovat selkeästi toisenlaisia kysymyksiä.

Tiedonhallintaa voi tutkia joko yhden tietomallin sisällä tai mallien välisin tarkasteluin. Mallien väliset kysymykset voivat esimerkiksi koskea mallien helppokäyttöisyyttä, ilmaisuvoimaa tai niiden operaatioiden nopeuden vertailuja. Mallien väliset tarkastelut ovat harvinaisia.

Yhden tietomallin sisäinen tutkimus on paljon vilkkaampaa. Kun mallin peruskäsitteet on kiinnitetty, on esimerkiksi päätetty käyttää relaatiomallia ja esittää tieto siis taulukkoina, niin voidaan kysyä, mistä löydetään juuri ne taulukot, jotka esittävät tiedon hyvin ja luontevasti? Tällöin tarkastellaan tietokannan suunnitteluksi kutsuttua laajaa aluetta. Suurin ongelma tällä alueella on se, että hyvän rakenteen löytäminen edellyttää tietokannan kohteen varsin tarkkaa ja näkemyksellistä ymmärtämistä.

Monien epäonnistuneiden atk-järjestelmien takana on puutteellinen kohteen tuntemus.

Tietomallin operaatioiden valitseminen eli mallin kyselykielen määrittely on toinen tiedonhallinnan esimerkkiongelma. Se on hyvin läheistä sukua ohjelmointikielten tutkimukselle. Tietomallin operaatioiden tehokas toteutus on samaten merkittävä tutkimuskohde, joka lähenee jo algoritmitutkimusta ja mahdollisesti myös tietokoneiden rakentamista.

Tiedonhallinnan tutkimuskohteissa ilmenee selkeästi tietotekniikalle ominainen teemojen sekoittuminen: erillisiltäkin tuntuvilla osa-alueilla on monesti paljon yhteistä.

Viitteet

1. Ilkka Niiniluoto, *Informaatio, tieto ja yhteiskunta*. Valtion Painatuskeskus, 1989.

PROFESSORI HEIKKI MANNILA
HELSINGIN YLIOPISTO
TIETOJENKÄSITTELYOPIN LAITOS
TEOLLISUUSKATU 23
00510 HELSINKI
mannila@cs.Helsinki.FI

Al-Khwarizmin perintö – mitä on algoritmitutkimus?

ESKO UKKONEN, HELSINGIN YLIOPISTO

1. Johdanto

Algoritmien ja laskennallisen kompleksisuuden tutkimus on tietojenkäsittelytieteen päävirtauksia. Sillä on kahdenlaisia tavoitteita. Konstruktiivisena tavoitteena on kehittää erilaisille tietojenkäsittelyongelmille tehokkaita ratkaisualgoritmeja. Analyyttisenä tavoitteena on arvioida eri algoritmien vaatimia laskentaresursseja kuten laskenta-aikaa ja muistitilaa sekä sitä, onko jokin algoritmi tehokkain mahdollinen ongelman ratkaisu.

Algoritmien optimaalista tehokkuutta tarkasteltaessa tulee tietojenkäsittelyongelmista analyysin varsinaisia kohteita. Tällöin puhutaan vaativuus- tai kompleksisuusteoriasta. Ongelman analysointi verrattuna yksittäisen algoritmin analyysiin on oleellisesti vaikeampaa, koska silloin pyritään arvioimaan ongelman kaikkien mahdollisten ratkaisualgoritmien käyttäytymistä. Toisaalta kysymyksenasettelutkin ovat kiintoisia: tavoitteena on ymmärtää, miksi joidenkin ongelmien algoritminen ratkaisu vaatii radikaalisti enemmän laskentaresursseja kuin joidenkin toisten ongelmien ratkaisu.

Tässä kirjoituksessa esitetään ensin pieni katsaus algoritmitutkimuksen historiaan, minkä jälkeen esitellään tutkimusalan peruskäsitteitä ja ongelmanasetteluja. Lopussa on joitakin huomautuksia algoritmitutkimuksen luonteesta ja kehitysnäkymistä.

2. Eukleides, al-Khwarizmi, Babbage

Vanhin tunnettu nimenomaan algoritmiksi kutsuttu algoritmi lienee Eukleideen nimiin laitettu menetelmä kahden luvun suurimman yhteisen tekijän määräämiseksi.

Sana *algoritmi* tulee kuitenkin persialais-arabialaisen matemaatikon al-Khwarizmin ("Khwarizmilainen") nimestä. Al-Khwarizmi, jonka täydellinen nimi oli Muhammed ibn Musa Abu Abdullah al-Khwarizmi al-Madjusi al-Qutrubulli, eli noin vuosina 780-850. Hän syntyi Aral-järven itäpuolella sijainneessa muinaisessa Khwarizmissa. Elämäntyönsä hän kuitenkin teki Bagdadissa, "Viisauden taloksi" kutsutun tutkimuslaitoksen palveluksessa. Laitosta ylläpiti kalifi al-Mamun, joka oli Tuhannen ja yhden yön kertomuksista tunnetun kalifi Harun ar-Rashidin poika.

Al-Khwarizmi osallistui intialaisen matematiikan välittämiseen länsimaihin. Sen keskeistä sisältöä olivat desimaalijärjestelmän käyttö lukujen merkitsemiseen, aritmetiikka ja trigonometria. Näitä kaikkia al-Khwarizmi käsitteli kirjoituksissaan, jotka sisältävät esimerkiksi myös hyvin selkeitä algoritminomaisia menetelmiä toisen asteen yhtälöiden ratkaisemiseksi. On kuitenkin epäselvää, mikä on intialaista perintöä ja missä näkyy al-Khwarizmin oma luova panos.

Keskiajan matematiikassa al-Khwarizmi eli "Algorismus" näkyy koulukuntajaossa abakistit - algoristit. Edelliset käyttivät laskuapuna helmitaulua, jälkimmäiset turvautuivat desimaalijärjestelmään ja aritmeettisiin menetelmiin.

Myöhempää kehitystä kuvattaessa on erityisesti mainittava englantilainen Charles Babbage (1792-1871). Hän oli aikaansa edellä oleva näkijä, jonka suunnittelema "analyyttinen kone" on nykyisten tietokoneiden varhainen edeltäjä. Analyyttinen kone, joka tosin ei koskaan valmistunut, oli suunniteltu kutomakoneen inspiroimana. Sen vallankumouksellinen piirre oli ohjelmoitavuus. Ohjelma oli tarkoitus antaa reikäkorteilla. Samaan tapaan kuin reikäkortit ohjasivat kutomakonetta, piti korttien ohjata myös analyyttisen koneen matemaattisia toimintoja. Vaihtamalla kortit sama

kone saataisiin suorittamaan erilaisia ohjelmia kuten kutomakonekin pystyi tuottamaan erikuosisia kankaita.

Vaikka kone jäi rakentamatta, Babbage tajusi kirkkaasti ideansa merkityksen. Nykyisen algoritmitutkimuksen huoneentauluksi sopiikin Babbagen omaelämäkerrassaan hiukkasen mahtipontisesti muotoilema arvio:

> As soon as an Analytical Engine exists, it will necessarily guide the course of the science. Whenever any result is sought by its aid, the question will then arise — By what course of calculation can these results be arrived at by the machine in the shortest time?

Algoritmikäsitteen matemaattisesti täsmällinen määrittely tehtiin 1930-luvulla. Näistä erästä, Turingin konetta, käsitellään tässä kirjoituksessa vielä tarkemmin. Turingin koneen ohella kehitettiin useita muita määrittelytapoja, jotka osoittautuivat keskenään samanarvoisiksi.

Elektronisen tietokoneen synty 1940-luvulla ja sitä seurannut tietokonelaitteistojen ripeä kehitys johti algoritmien kysynnän räjähdysmäiseen kasvuun. Yhtäkkiä algoritmitutkimus oli sovelluskelpoista, olihan onnistuttu rakentamaan laite, joka pystyi ainakin periaatteessa suorittamaan kuinka työläitä ja monimutkaisia algoritmeja tahansa.

Samalla alkoi laaja algoritmisen ajattelun läpimurto kaikkialla tutkimuksessa ja tekniikassa. Tämä on johtanut siihen, että yhä useammilla aloilla tavoitteeksi on tullut tiedon saattaminen algoritmeiksi tai ainakin algoritmisesti käsiteltävään muotoon. Sen, missä määrin tässä onnistutaan, voi katsoa monessa tapauksessa jokseenkin suoraan mittaavan asianomaisen alan saavuttamaa tiedon syvyyttä.

3. Algoritmin käsite ja Turingin kone

Koulusivistykseen kuuluu useita algoritmeja. Hyvä esimerkki on moninumeroisten lukujen kertolaskun suorittamiseksi opittu menettely. Jos on laskettava '49 kertaa 18', muodostuu tämän algoritmin kirjanpito seuraavaksi:

```
  18
 _49
 162
 _72
 882
```

Ensin 18 kerrotaan kertojan 49 ykkösillä, siis 9:llä, mistä saadaan 162, ja sitten kymmenillä, mistä saadaan 72 (oikeastaan 720, mutta 0 on tapana jättää kirjoittamatta). Lopuksi lasketaan välitulokset yhteen. Näin on syötteistä 49 ja 18 saatu tuloste 882.

Yleisesti algoritmeja voi luonnehtia toteamalla, että kukin algoritmi määrittelee äärellisen, täsmällisen operaatiojonon, joka tuottaa algoritmin saamista syötteistä halutut tulosteet. Jonon kunkin operaation täytyy olla äärellisessä ajassa suoritettavissa. Algoritmin määrittelemää toimintaa voi ajatella symbolien manipulointina, jonka voi periaatteessa aina suorittaa täysin tarkasti käsityönä äärellisessä ajassa kynällä ja paperilla.

Sama ongelma voidaan usein ratkaista monella eri algoritmilla, jotka saattavat poiketa syvällisesti toisistaan. Esimerkiksi kertolaskua varten tunnetaan myös "venäläinen" algoritmi. Edellisen esimerkin tapauksessa sen kirjanpito näyttäisi seuraavalta:

```
 49   18 +
 24   36
 12   72
  6  144
  3  288 +
 _1  576 +
      882
```

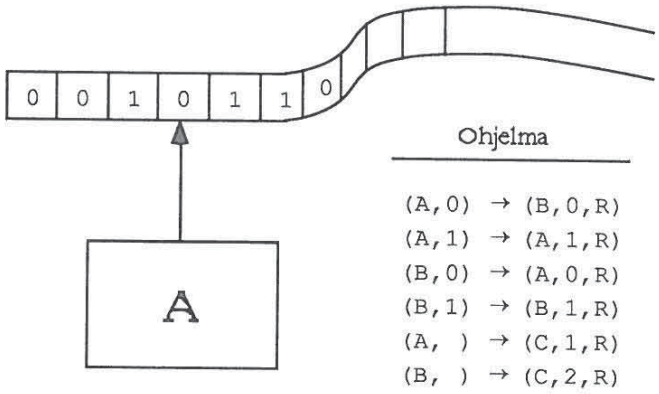
Ensimmäisellä sarakkeella olevat luvut on saatu jakamalla kertoja 49 toistuvasti luvulla 2 ja ottamalla tuloksen kokonaisosa. Toisella sarakkeella olevat luvut on puolestaan saatu kertomalla kerrottava 18 toistuvasti luvulla 2. Lopputulos 882 saadaan laskemalla yhteen ne kerrottava-sarakkeen luvut (merkitty +:lla), joita vastaa kertoja-sarakkeella pariton luku. Tämä algoritmi, joka käyttää perusoperaationaan luvulla 2 jakamista ja kertomista sekä yhteenlaskua, sopii hyvin tietokoneille, koska ne käyttävät 2-kantaista lukujärjestelmää.

Algoritmi saa täsmällisen muodon kun se kirjoitetaan ohjelmaksi jollakin ohjelmointikielellä. Ohjelmointikielten joukko on tunnetusti laaja ja koko ajan kasvava. Eri kielet voivat poiketa ulkoasunsa, perusoperaatioidensa ja kieleen liittyvän ohjelmointiajattelun osalta erittäin syvällisestikin toisistaan; eroilla on käytännöllistä merkitystä lähinnä siinä, miten helppoa ja luontevaa erityyppisten tietojenkäsittelymenetelmien ja algoritmien muotoilu kussakin kielessä on. Kielten teoreettinen ilmaisuvoima on

yleensä kuitenkin sama: kaikilla pystytään lopulta laskemaan täsmälleen samat asiat.

Koska kielet ovat tässä mielessä samanarvoisia, algoritmiteoreettisten tarkastelujen perustana kannattaa usein käyttää mahdollisimman primitiivistä ohjelmointikieltä, jolla ohjelmoidaan äärimmilleen pelkistettyä abstraktia tietokoneen mallia. *Turingin kone* ja sen ohjelmointikieli sopivat tähän tarkoitukseen.

Yksinkertaisin Turingin kone käyttää muisti-, syöttö- ja tulostuslaitteena yhtä "nauhaa". Nauha on jaettu muistipaikkoihin. Näitä on rajattoman paljon, koska nauhan ajatellaan jatkuvan äärettömiin. Kussakin paikassa voi olla talletettuna yksi merkki, tai paikka voi olla tyhjä. Koneessa on lisäksi tietysti "prosessori", jolla voidaan käsitellä nauhan sisältöä.



Kuva 1. Turingin kone.

Käsittely tapahtuu koneeseen kuuluvan ohjelman mukaan. Prosessori pystyy kunakin ajanhetkenä tarkastelemaan yhtä nauhapaikkaa. Yhdessä ohjelma-askeleessa prosessori voi ensin muuttaa tarkastelemansa paikan sisällön kirjoittamalla sinne jonkin merkin ja siirtyä sitten tarkastelemaan välittömästi vasemmalla tai oikealla puolella olevaa nauhapaikkaa. Lisäksi prosessorilla on oma sisäinen äärellistilainen muisti, jonka sisältöä eli prosessorin "tilaa" ohjelma-askel voi myös muuttaa. Koneen koko toiminta muodostuu kuvatuista hyvin yksinkertaisista askelista. Toiminnan alkaessa koneen saama syöte on kirjoitettu nauhan alussa oleviin peräkkäisiin

muistipaikkoihin, toiminnan päättyessä koneen antama tuloste voidaan lukea nauhan muistipaikoista.

Kuvassa 1 on eräs Turingin kone. Se tarkastaa, onko syötteenä annetussa bittijonossa 0010110 parillinen vai pariton määrä merkkejä 0. Ohjelman ensimmäinen lause (A,0) → (B,0,R) voitaisiin lukea: "Jos kone on tilassa A ja tarkasteltu nauhapaikka sisältää merkin 0, siirry tilaan B, kirjoita nauhapaikkaan 0 ja siirry tarkastelemaan seuraavana oikealla puolella olevaa nauhapaikkaa." Kuvan tilanteessa tämä askel otettaisiin seuraavaksi, koska koneen tila todella on A ja tarkasteltu nauhamerkki 0.

On selvää, että Turingin kone on teoreettinen konstruktio, jolla ei ole varsinaista käytännön merkitystä. Edellä kuvattujen kertolaskualgoritmien ohjelmoiminen Turingin koneeksi olisi kyllä mahdollista, mutta jokseenkin työlästä. Se kelpaa kuitenkin algoritmikäsitteen matemaattiseen määrittelyyn: algoritmisesti laskettavissa olevaa on täsmälleen se mitä pystytään laskemaan Turingin koneilla.

Onko näin asetettu laskettavuuden määritelmä oikein? Toisin sanoen, vastaako Turingin koneina formalisoitu laskettavuus intuitiivista, esimerkiksi kynällä ja paperilla tapahtuvaan symbolien käsittelyyn tai mikroprosessoreiden ominaisuuksiin perustuvaa käsitystämme algoritmisista menetelmistä? *Church – Turing teesi* lausuu sen, että näin on. Teesiä ei tietenkään voi matemaattisesti todistaa. Käytännön kokemus kuitenkin tukee sitä. Tämä voi tuntua yllättävältä, merkitseehän teesi esimerkiksi sitä, että kehittyneimmänkin digitaalisen tietokoneen toimintaa voidaan tietyssä mielessä tarkasti simuloida yksinkertaisella Turingin koneella.

Algoritmikäsitteen vaihtoehtoisista formalisointitavoista Turingin kone ei ehkä ole elegantein. Algoritmitutkimuksen kannalta se on kuitenkin käyttökelpoinen, koska siinä on luonnolliset mitat laskennan vaatimille resursseille: laskenta-ajan mittana voidaan käyttää laskennan ottamien ohjelma-askelien lukumäärää, tilan mittana niiden muistipaikkojen määrää, joiden kautta laskenta on kulkenut. Usein tarvitaan kuitenkin kehittyneempiä malleja, koska Turingin kone on liian kaukana todellisista tietokoneista. Näiden laskenta-askeleet vastaavat paremmin oikeiden tietokoneiden perusoperaatioita.

4. Algoritmien analysointi

Miten keksitään hyviä algoritmeja? Kaiken kattavia suunnittelusääntöjä on arvatenkin mahdotonta antaa, siinä määrin yksilöllisiä monet merkittävät algoritmit ovat. Joitakin hyviksi havaittuja periaatteita on kuitenkin eristetty. Näitä ovat muiden muassa hajoita ja hallitse -tekniikaksi kutsuttu menettely ja dynaaminen ohjelmointi. Esimerkiksi hajoita ja hallitse -algoritmit jakavat ratkaistavan tehtävän keskenään samanlaisiin osatehtäviin, jotka ratkaistaan erikseen ja osaratkaisuista kootaan kokonaisratkaisu. Tietorakenteiden kehittely on usein oleellinen osa hyvän algoritmin muodostamista.

Kun algoritmi on konstruoitu, teoreettisen analyysin keinoin pyritään arvioimaan sen tehokkuutta, lähinnä algoritmin toiminnan vaatimaa aikaa ja muistitilaa. Tehtävä voi tuntua hankalalta: on ilmeistä, että sama algoritmi voi käyttäytyä eri syötteillä hyvinkin eri tavoin. Tilannetta on sopivasti yksinkertaistettava. Vakiintunut tapa on arvioida resurssivaatimuksia algoritmin syötteen pituuden n funktiona. Kaikki saman n pituiset syötteet arvioidaan siis yhdessä. Yleensä tarkastellaan, miten resurssivaatimukset pahimmassa tapauksessa kasvavat, kun syötteen pituus n kasvaa rajatta.

Tällaisen *pahimman tapauksen asymptoottisen analyysin* ratkaisevana etuna on analyysituloksen riippumattomuus algoritmia suorittavan tietokoneen nopeudesta ja toisarvoisista yksityiskohdista. Myös analyysilaskut tulevat suhteellisen yksinkertaisiksi.

Haittana voi olla epätarkkuus: jos pahimman tapauksen antava syöte on käytännössä harvinainen ja keskimäärin algoritmi käyttäytyy huomattavasti pahinta tapausta tehokkaammin, analyysillä on saatu liian huono kuva. Asymptoottisen arvion ilmaisema käyttäytyminen voi myös tulla esiin vasta niin suurilla arvoilla n, että ne eivät tule käytännössä kyseeseen. Silloin algoritmien vertailu vain asymptoottisten arvioiden perusteella voi johtaa harhaan.

Palataan kertolaskun peruskoulualgoritmiin. Kun sillä kerrotaan kaksi n-numeroista lukua, ei ole vaikea osoittaa, että laskenta-ajan tarve

kasvaa pituuden n kasvaessa kuten n^2. Silloin algoritmin asymptoottista aikavaatimusta merkitään $O(n^2)$. Tällä kertaa analyysin tulos on varsin tarkka, koska pahimmalla ja parhaalla tapauksella ei ole suurta eroa.

Tunnetaan myös hajoita ja hallitse -tyyppinen kertolaskualgoritmi, jonka aikavaatimus on $O(n^{1.59})$, siis edellistä pienempi. Asymptoottisesta paremmuudestaan huolimatta tämä algoritmi on siinä määrin monimutkainen, että eräässä kokeessa se saatiin toimimaan peruskoulualgoritmia nopeammin vasta kun n oli yli 600! Syynä on monimutkaisuuden aiheuttama yleisrasite, joka dominoi kun n on pieni. Algoritmilla näyttäisi kuitenkin olevan käyttöä yli 600-numeroisten lukujen kertolaskua tarvitsevissa erikoissovelluksissa.

Ei ole aivan harvinaista, että asymptoottiset parannukset saadaan aikaan algoritmeilla, joiden käytännön arvo tällä hetkellä on rajoittunut. Kuitenkin niillä on pysyvää merkitystä. Onhan mahdollista, että tulevaisuuden tietokoneissa algoritmi voidaan toteuttaa niin, että se pääsee oikeuksiinsa jo kun n on pieni. Toisaalta uusissa sovelluksissa yhä suuremmat n voivat olla tarpeen.

5. Optimaaliset algoritmit

Algoritmitutkimuksen eräs luonnollinen tavoite on selvittää, onko ongelman annettu ratkaisualgoritmi jossakin mielessä paras mahdollinen.

Tarkastellaan esimerkkinä etsintäongelmaa. On annettu pakka nimikortteja. Pakasta on löydettävä tietty kortti, esimerkiksi Hannes Hemulin kortti. Etsintä voidaan hoitaa ilmeisellä tavalla peräkkäisetsintänä seuraavasti: Tutkitaan kortit yksitellen siinä järjestyksessä kuin ne ovat pakassa. Kun Hannes Hemuli löytyy tai kaikki kortit on tuloksetta katsottu, tehtävä on suoritettu. Aikaa menee pahimmillaan $O(n)$, jos n on korttien lukumäärä. Kaikki n korttia tulevat nimittäin katsotuiksi ainakin siinä tapauksessa, että Hannes Hemulin korttia ei ole lainkaan pakassa.

Toisaalta tehtävästä ei voi pahimmassa tapauksessa selvitä nopeamminkaan kuin $O(n)$ askeleessa. Jos yksikin kortti jätetään katsomatta, juuri se voi olla etsitty Hannes Hemuli. Peräkkäisetsintä on tässä mielessä optimaalinen.

Näin hidas etsintämenetelmä on tarpeen kuitenkin vain silloin kun korttipakka on sekaisin. Jos kortit ovat järjestettynä aakkosjärjestykseen, huomattavasti nopeampi algoritmi tulee mahdolliseksi. Silloin soveltuu binääriseksi etsinnäksi sanottu menetelmä.

Binäärisessä etsinnässä katsotaan ensin pakan puolivälissä oleva kortti. Jos se on Hannes Hemulin, etsintä on valmis. Muuten jos Hannes Hemuli on aakkosjärjestyksessä ennen kortissa olevaa nimeä, jatketaan etsintää pakan alkupuoliskosta käyttäen samaa puolittamisperiaatetta, muuten jatketaan pakan loppupuoliskosta. Jokainen kortin tutkiminen puolittaa sen pakan osan, jossa Hannes Hemulin kortti voi olla. Vertailuja tarvitaan näin ollen korkeintaan korttien määrän logaritmin verran eli $O(\log n)$ kappaletta. Tämän ylärajan rinnalle voidaan asettaa alarajatulos: mikä tahansa algoritmi, joka etsii lajitellusta pakasta, joutuu tekemään pahimmassa tapauksessa ainakin $O(\log n)$ vertailua. Binäärinen etsintä on siis optimaalinen.

Näiden lupaavien esimerkkien jälkeen on huomautettava, että yleisesti asiain tila ei ole yhtä valoisa. Ei-triviaalien alarajojen todistaminen mielenkiintoisten ongelmien algoritmisen ratkaisun resurssivaatimuksille on osoittautunut usein erittäin vaikeaksi, ja monien tärkeiden ongelmien ratkaisualgoritmien optimaalisuutta ei ole pystytty lainkaan osoittamaan.

6. Ratkeavuus periaatteessa ja käytännössä

Algoritminen ratkeamattomuus on algoritmikäsitteen formalisoinnin yhteydessä paljastuneita ilmiöitä. Tunnetaan runsaasti ongelmia, joista voidaan osoittaa, että niiden ratkaiseminen algoritmilla on mahdotonta. *Pysähtyvyysongelma* on näistä tunnetuimpia. Tehtävänä on selvittää pysähtyykö annetun algoritmin tai tietokoneohjelman suoritus aina vai onko mahdollista että suoritus jää silmukkaan jatkuen loputtomasti. Ongelman ratkaisevaa yleistä algoritmia ei voi olla. Sivuhuomautuksena todettakoon, että tämä tulos asettaa vahvoja periaatteellisia rajoituksia mahdollisuuksille kehittää automaattisia ohjelmoinnin apuvälineitä.

Käytännön tarpeisiin ei kuitenkaan pelkkä algoritmin olemassaolo riitä. Algoritmin on oltava myös käyttökelpoisen tehokas. Esimerkiksi

sopii seuraava helposti syntyvä "kombinatorisen räjähdyksen" sisältävä tilanne. Tarkastellaan algoritmia, jonka suoritusaika kasvaa kuten 2^n, missä n on jälleen syötteen pituus. Voidaan yksinkertaisesti ajatella, että n merkkiä pitkän syötteen käsittely vaatii 2^n mikrosekuntia. Tämä algoritmi ehtii silloin yhden vuorokauden aikana käsitellä syötteen, jonka pituus on vaatimattomat 32. Entä jos aikaa on käytettävissä runsaasti? Siinä ajassa, joka on kulunut maailmankaikkeuden arvioidusta alkuräjähdyksestä, algoritmi olisi ehtinyt käsitellä syötteen, jonka pituus on 70!

Esimerkin sanoma on selvä. Tällainen *eksponentiaalinen algoritmi* on käyttökelvoton lukuunottamatta aivan lyhyitä syötteitä. Voidaan tosin ajatella että ongelma ratkeaisi tulevaisuuden hypernopeilla ultrarinnakkaisilla superkomputaattoreilla. Näin ei käy: jos esimerkkialgoritmimme suoritettaisiin koneella, joka olisi 1.000.000 kertaa nopeampi, kasvaisivat edellä annetut pituudet 32 ja 70 vain 20:llä, ja jos tällaisia koneita olisi 1.000.000 kappaletta rinnakkaisesti suorittamassa algoritmiamme, kasvaisivat pituudet parhaimmillaankin jälleen vain 20:llä. Koneiden nopeuttaminen ja niiden kytkeminen rinnakkain eivät auta oleellisesti.

Käytännössä ratkeavina pidetäänkin yleensä vain ongelmia, joilla on *polynomisessa ajassa* toimiva algoritmi. Tämä tarkoittaa, että algoritmin vaatima toiminta-aika on korkeintaan jokin syötteen pituuden polynomi. Näin lausuttu yleinen periaate ei tosin ole ilman poikkeuksia. Jos syötteen koko on pieni, voi eksponentiaalinenkin algoritmi riittää. Toisaalta on sovelluksia, joissa ajankäytön polynomisuus ei saa olla liian korkeaasteista. Reaaliaikaiset sovellukset voivat vaatia ajassa $O(n)$ toimivaa algoritmia, $O(n^2)$ saattaa olla jo liian hidas.

7. NP-täydellisyys

Algoritmitutkimuksen ja kompleksisuusteorian kuuluisin avoin kysymys liittyy NP-täydellisyyteen. Kysymys esitetään yleensä muodossa: Onko $P = NP$? Tässä P tarkoittaa käytännössä ratkeavien ongelmien luokkaa, siis ongelmia, joilla on polynominen ratkaisualgoritmi. Luokka NP sisältää muiden ohella myös kaikki NP-täydelliset ongelmat. Kysymys on siitä, voidaanko NP-täydelliset ongelmat ratkaista polynomisessa ajassa. Pian

kaksi vuosikymmentä jatkuneesta intensiivisestä tutkimuksesta huolimatta kysymys onko $P = NP$ pysyy sitkeästi avoimena.

NP-täydellisiä ongelmia tunnetaan runsaasti. Useat ovat tärkeitä monissa sovelluksissa, esimerkiksi n.s. kauppamatkustajan ongelma on tällainen. Tarkastellaan esimerkkinä osasummaongelmaa.

Osasummaongelmassa on annettu mielivaltainen joukko kokonaislukuja, vaikkapa luvut [15, 27, 16, 8, 24, 12, 19, 3, 31, 45, 33] ja lisäksi erityinen luku, esimerkiksi 50. Tehtävänä on selvittää, sisältääkö lukujoukko osajoukon, jonka lukujen summa on yhtä suuri kuin erityinen luku. Esimerkkitapauksessa olisi päätettävä, voidaanko annetusta joukosta poimia luvut, joiden summa on 50.

Osasummaongelmalle on helppo keksiä ratkaisualgoritmi. Poimitaan systemaattisesti vuoron perään kaikki lukujoukon osajoukot ja testataan, onko osajoukon summa yhtä suuri kuin erityinen luku. Esimerkkitapauksessa löytyisivät näin jossakin vaiheessa luvut 16, 3 ja 31, jotka antavat myönteisen ratkaisun, koska 16 + 3 + 31 = 50. Algoritmi on kuitenkin eksponentiaalinen: jos lukuja on n kappaletta, erilaisia osajoukkoja eli ratkaisuehdokkaita on 2^n, joten aikaa menee pahimmillaan ainakin $O(2^n)$. Oleellisesti tehokkaampaa algoritmia ei tunneta, mutta liioin ei ole kyetty osoittamaan etteikö sellaista voisi olla.

Osasummaongelma on NP-täydellinen. Sillä on seuraava kaikille NP-täydellisille ongelmille luonteenomainen piirre: ratkaisun löytäminen on vaikeaa, mutta annetun ratkaisuehdokkaan tarkistaminen on helppoa. Esimerkkitapauksessa luvut 16, 3 ja 31 muodostavat erään ratkaisun. Vaikeus on siinä, miten juuri nämä suotuisat luvut löydetään nopeasti. Ainoa mahdollisuus näyttää olevan aikaavievä kaikkien eri vaihtoehtojen läpikäynti. Annetun ratkaisuehdokkaan tarkistus sensijaan sujuu nopeasti. Jos joku väittää, että ongelmalla on myönteinen ratkaisu ja antaa väitteensä tueksi ratkaisuehdokkaan, ehdokkaan luvut vain lasketaan yhteen ja katsotaan tuliko 50.

Toinen NP-täydellisten ongelmien oleellinen ominaisuus on, että ne ovat tavallaan yhtä vaikeita. Kaikki NP-täydelliset ongelmat nimittäin palautuvat toisiinsa. Jos yhdellekin tunnettaisiin tehokas algoritmi, sen avulla voitaisiin ratkaista tehokkasti myös kaikki muut. Algoritmin avulla voitaisiin myös ratkaista kaikki NP-täydellisiä helpommat ongelmat.

NP-täydelliset ongelmat ovat tässä mielessä täydellisen vaikeita ongelmia luokassaan.

Yleensä arvellaan, että luokka *NP* ei sisälly luokkaan *P* eli yhtäpitävästi, että osasummaongelmalla tai millään muulla NP-täydellisellä ongelmalla ei ole polynomisessa ajassa toimivaa algoritmia. Jos tämä arvelu osoittautuu oikeaksi, ei asian lopullisella todistuksella välttämättä ole suurta vaikutusta, vastaisihan tulos odotuksia. Jos toisaalta osoittautuisi että $P = NP$, laajentuisi käytännössä ratkeavien ongelmien joukko merkittävästi, millä voisi olla selviä taloudellisiakin seurauksia.

8. Vaikeiden ongelmien käsittely

Miten tullaan käytännössä toimeen laskennallisesti vaikeiden ongelmien kanssa? Näitä ongelmia, joille ei tunneta polynomista algoritmia, joudutaan jatkuvasti ratkaisemaan eri yhteyksissä, vieläpä varsin hyvällä menestyksellä.

Kuten edellä jo mainittiin, eksponentiaalinenkin algoritmi voi olla käyttökelpoinen, jos laskenta-ajan kombinatoriseen räjähdykseen johtavat syötteet ovat algoritmin käyttöympäristössä harvinaisia ja tavallisesti algoritmi on nopea. Tyyppiesimerkki tällaisesta algoritmista on lineaarisen ohjelmoinnin tunnettu Simplex-algoritmi. Samalle ongelmalle löydettiin jo yli 10 vuotta sitten huomattavan kohun saattelemana myös polynominen menetelmä, joka kuitenkaan ei ole pystynyt syrjäyttämään Simplex-algoritmia.

Toinen laajassa käytössä oleva tapa ratkoa laskennallisesti vaikeita ongelmia on tyytyä tarkan ratkaisun sijasta likiarvoon. Ongelma on usein optimointitehtävä: on löydettävä jonkin suureen maksimi tai minimi. Käytännössä saattaa hyvin riittää, että löydetään optimin likiarvo. Likiarvo saattaa löytyä nopealla algoritmilla. Tosin on osoittautunut, että monessa tapauksessa hyvää likiarvoa ei ole sen helpompi löytää kuin tarkkaa ratkaisuakaan. Siksi on ymmärrettävää, että optimointiongelmien ratkaisemisessa on vallalla varsin anarkistinen tila: likimääräisiä ratkaisuja etsitään algoritmeilla, jotka perustuvat erilaisiin huonosti ymmärrettyihin peukalosääntöihin. Ne ovat nopeita ja näyttävät toimivan luotettavan

tarkasti, mutta kunnollista laatutakuuta ratkaisun tarkkuudelle ei yleensä tunneta.

Toisaalta on huomattava, että laskennallisesti vaikeista ongelmista ei ole pelkästään harmia. Niillä on hyötykäyttöä salakirjoituksessa ja tiedonsuojausmenetelmissä, jotka perustuvat yksisuuntaisiksi funktioiksi kutsuttujen laskentaongelmien käyttöön. Funktion arvon laskenta on vaikeaa, mutta sen käänteisfunktion laskenta on helppoa.

9. Algoritmikäsitteen laajennuksia

Laskennallisen kompleksisuuden voittamiseksi on myös etsitty uudentyyppisiä, *rinnakkaisuutta* ja *satunnaisuutta* sisältäviä algoritmeja.

Tavallinen algoritmi on *peräkkäisalgoritmi*, jonka suorittajana on yksi suoritin eli prosessori. Yhteen suorittimeen perustuvassa tietokonearkkitehtuurissa suorittimen ja muistin välinen tietoliikenne muodostaa kokonaisnopeutta rajoittavan ahtaumakohdan, "von Neumannin pullonkaulan".

Pullonkaula avartuu, kun otetaan käyttöön useita rinnakkaisia suorittimia. Algoritmien kehittäminen tällaiselle yhteistä muistia käyttävälle moniprosessorikoneelle on osoittautunut haastavaksi. Yhteisen algoritmin jako rinnakkaisille suorittimille ja muistinkäytön organisointi aiheuttavat yleisrasitetta, johon rinnakkaisuudella saavutettu lisänopeus voi osittain haaskaantua. Yksinkertaisiin vektorointi- ja liukuhihnaperiaatteisiin perustuvaa rinnakkaisuutta käytetään menestyksellisesti supertietokoneissa. Teorian kehittelyn puolella tutkimus ei ole vielä päässyt yksimielisyyteen rinnakkaislaskennan formaalista mallista. Yhtenäisen mallin puuttuminen hidastaa algoritmiteoreettista kehitystä.

Jos eri prosessoreilla ei ole yhteistä muistia ja kukin prosessori toimii itsenäisesti omaan tahtiinsa, joudutaan ratkaisemaan epäsynkronisten moniprosessori- ja tietokoneverkkojen ja hajautettujen algoritmien ongelmia. Perustehtävänä on järjestää eri suorittimien välille niiden yhteistyön mahdollistava kommunikointi. Tällöin syntyy uudentyyppisiä algoritmisia ongelmia, esimerkiksi pitää varautua verkossa olevien suorittimien

tilapäiseen epäkuntoisuuteen. Paitsi tehokkuuden, jopa algoritmien oikeellisuuden ja luotettavuuden saavuttaminen on vaikeaa.

Suoraan elektronisina piireinä toteutettavat algoritmit sisältävät runsaasti alkeellisia prosessoreja käyttävää synkronista rinnakkaisuutta. Integroitujen piirien kehittyessä yhä useampia algoritmeja tullaan toteuttamaan tällä tasolla. Kytkentäpiirien algoritmiteoria liittyy luontevasti peräkkäisalgoritmien teoriaan.

Satunnaisuutta sisältävät algoritmit käyttävät satunnaislukuja. Havainnollisesti voidaan ajatella, että algoritmit perustavat jotkin toimintapäätöksensä lantinheittoon! Perinteinen algoritmi on deterministinen: samoilla syötteillä algoritmi tekee aina täsmälleen samat laskenta-askeleet ja tuottaa samat tulokset. Satunnaisalgoritmin toimintaan vaikuttaa se, mitä lukuja satunnaislukulähde on sattunut algoritmin toimiessa antamaan. Jos tällainen algoritmi toistetaan samoilla syötteillä, on mahdollista että algoritmi suorittaa eri kerroilla eri askeleet ja saa myös eri lopputulokset. Satunnaislukujen käyttö tehokkaiden algoritmien kehittämisessä perustuu siihen, että niiden avulla voidaan organisoida onnekkaita arvauksia, jotka johtavat algoritmin nopealle suoritusreitille.

Tarkastellaan esimerkkinä erästä reititysalgoritmia. Prosessoriverkon prosessorit lähettävät viestejä toisilleen verkkoa pitkin. Lähettäjäprosessori liittää viestiin osoitteeksi vastaanottajaprosessorin nimen, ja verkko huolehtii viestin viemisestä perille. Verkon ruuhkien tasaamiseksi sopii seuraava satunnaisalgoritmi: kun viesti lähtee, sille arvotaan satunnaislukujen avulla väliaikainen osoite, joka voi olla mikä tahansa verkon prosessoreista. Sen jälkeen viesti kulkee ensin väliaikaiseen osoitteeseen ja vasta sieltä varsinaiseen osoitteeseensa. Tämä yksinkertainen menettely purkaa ruuhkia ja tasaa kuormitusta tehokkaasti.

Erään luokituksen mukaan satunnaisalgoritmit jakautuvat Sherwood-, Monte Carlo ja Las Vegas -algoritmeihin. *Sherwood-algoritmin* lähtökohtana on tavallinen deterministinen algoritmi. Ideana on satunnaislukujen avulla tasoittaa lähtöalgoritmin pahinta tapausta. Tuloksena on algoritmi, joka millä tahansa syötteellä vaatii suurella todennäköisyydellä sen ajan minkä alkuperäinen algoritmi vaati keskimäärin.

Monte Carlo -algoritmissa sallitaan virheellinen tulos. Algoritmit ovat aina nopeita mutta tulos voi olla väärin. Virheen todennäköisyys

voidaan kuitenkin tehdä mielivaltaisen pieneksi. *Las Vegas -algoritmit* antavat aina oikean tuloksen, mutta ovat vain todennäköisesti nopeita. Pienellä todennäköisyydellä laskenta voi olla pitkä.

Satunnaisalgoritmien tutkimus jatkuu vilkkaana. Alaa rikastuttavat monet yhteydet eri tahoille kuten lukuteoriaan, kryptologiaan ja tiedonsuojaukseen sekä hajautettuihin algoritmeihin. Merkittäviin sovelluksiin johtavia tuloksia voi tältä suunnalta myös odottaa.

10. Lopuksi

Algoritmitutkimuksen arvomaailma muotoutui ainakin tavallisten peräkkäisalgoritmien osalta 1970-luvulla. Monien uudentyyppisten menetelmien, kuten rinnakkais- ja satunnaisalgoritmien, tutkimus kuplii kuitenkin edelleen varsin vapaasti, ja työtä on runsaasti jäljellä. Algoritmitutkimuksen asema on erityisen vahva amerikkalaisessa tietojenkäsittelytieteessä, mutta myös Suomessa on ylletty kansainvälisen tason tuloksiin.

Algoritmianalyysi ja kompleksisuusteoria vaatii täsmällistä matemaattista tutkimusotetta. Monesti tosin riittää varsin rutiininomainen matematiikan käyttö. Kuitenkin uusien algoritmisten ideoiden kehitys ja monet sitkeästi avoimina säilyvät vanhat ongelmat viittaavat siihen, että algoritmitutkija tarvitsee yhä laajempia matematiikan tietoja.

Algoritmitutkimuksen viehättävimpiä piirteitä on lyhyt ja mutkaton yhteys sovelluksiin. Yhteyden vaaliminen, mikä käytännössä tarkoittaa algoritmien kehittämistä ja analysointia konkreettisen ohjelmiston rakentamisen yhteydessä, on alan elinvoimaisuuden perusedellytys.

Kirjallisuutta

1. David Harel, *Algorithmics - The Spirit of Computing*. Addison-Wesley, 1987.

PROFESSORI ESKO UKKONEN
HELSINGIN YLIOPISTO
TIETOJENKÄSITTELYOPIN LAITOS
TEOLLISUUSKATU 23
00510 HELSINKI
ukkonen@cs.Helsinki.FI

Atk:n rutiinilla ehtii tehdä enemmän virheitä

PERTTI JÄRVINEN, TAMPEREEN YLIOPISTO

1. Testaus on joskus vaarallista

Teräsyhtiö OVAKOlle hankittiin vuonna 1963 tietokone, jossa oli ensimmäisenä Euroopassa levy-yksiköt ja niissä vaihdettavat levypakat. Uuden suorasaantitiedoston perustamisen yhteydessä piti levymuistista puhdistaa tiedostoalueen entinen sisältö. Sitä varten oli järkevää tehdä yleinen aliohjelma tai oikeammin makro, jolle annettaisiin puhdistettavan alueen rajat määrittävät kaksi osoitetta, ensimmäisen ja viimeisen sektorin osoite. Aliohjelman tehtävänä oli kirjoittaa tyhjää ensimmäisestä sektorista lähtien toistuvasti viimeiseen sektoriin asti. Osoitteet voitiin antaa joko aliohjelmaa kutsuvasta ohjelmasta tai konsolikirjoittimelta, laitteelta, jota nykyisin sanotaan päätteeksi.

Kun olin mielestäni saanut aliohjelman toimivaan kuntoon, halusin testata sitä. Konesalin operaattorit olivat juuri saaneet edellisen työn tehtyä ja kone oli hetken vapaana uusien ohjelmien kehittelyä varten. Katsoin, että levy-yksiköllä oli pyörimässä joku levy. Luulin, että se oli työlevy ja otin sen käyttöön testaustani varten. Annoin sen vuoksi konsolilta kaksi osoitetta ja käynnistin ohjelman. Listasin sitten levy-yksiköllä pyörivän levypakan sisältöä, jotta voisin todeta, toimiko ohjelma. Ohjelma näytti toimivan, mutta olin vahingossa kokeillut ohjelmaa levypakalla, johon oli juuri ennen kokeiluani ajettu viimeisin palkka-ajo. Olin tuhonnut muutaman työntekijän tietueet palkkatiedostosta, kun operaattori ei ollut ennen

innokasta kokeiluani vielä ehtinyt ottaa palkkatietojen levypakkaa levyyksiköltä eikä siirtää sitä lukittuun paloturvakaappiin. Sain tekosestani kahden viikon porttikiellon konesaliin.

Tässä tapauksessa virheen ja sen syyn välinen kausaaliketju oli selkeä. Voidaan sanoa, että oli sattunut paha lipsahdus (slip, [8]) rutiinitehtävässä. Todettakoon tässä yhteydessä, että olen antanut esitykselleni nimen juuri tästä syystä. Atk:n rutiinilla todella ehtii tehdä nopeasti ja paljon pahaa, jos niin haluaa, tai jos vahinko sattuu.

Käsitän rutiinitehtävän tässä Rasmussenin [5] mielessä tehtävänä, jonka suoritus on lähes automatisoitunut, ja joka siis tapahtuu ilman psyykkistä säätelyä. Rutiinitehtävien luokka eroaa kahdesta muusta tehtäväluokasta, sovellustehtävistä ja ongelmanratkaisutehtävistä juuri sen perusteella, missä määrin niissä tarvitaan tietoista psyykkistä kontrollia. Ongelmanratkaisutehtävässä erotetaan itse asiassa kaksi ongelmaa [2]: varsinainen ongelma, jota sanotaan kohdeongelmaksi, ja käsittelyongelma. Ensin on ratkaistava käsittelyongelma, eli on tuotettava algoritmi, jolla kohdeongelma saadaan ratkaistuksi. Jos sama tai samanlainen kohdeongelma esiintyy toisen, kolmannen jne. kerran, voidaan soveltaa aiemmin johdettua valmista algoritmia eli käsittelyongelman ratkaisua.

Rasmussen on analysoinut inhimillisten virheiden mahdollisuuksia eri tehtäväluokissa [7]. Niiden perusteella voidaan sanoa, että tietueiden tuhoutuminen johtui siitä, etten selvittänyt etukäteen tarpeeksi tarkasti testauksen mahdollisia sivuvaikutuksia. Mainitussa toisen sukupolven tietokoneessa ei ollut vielä käyttöjärjestelmää, joka olisi kontrolloinut yhteiskäyttöisten resurssien käyttöä. Nykyaikainen systeemi kysyy ennen tuhoamiskomennon toteutusta: "Aiotko todella hävittää kyseisen tietojoukon?"

Testaus voi joissakin muissakin tilanteissa olla vaarallinen toimenpide. Esimerkiksi jos virusohjelman tai tietokonemadon testaus onnistuu eli joku onnistuu ohjelmoimaan viruksen tai madon, niin tulokset voivat olla kohtalokkaat, kuten syksyltä 1988 muistamme. Tuolloin matoa kehitellyt amerikkalainen tutkija tarkoittamattaan aiheutti madollaan paljon harmia ja ylimääräistä työtä. Tällä hetkellä tiedetään, että virusten tuhoamisohjelmat voivat vahingossa saada itsekin viruksen luonteen. Siksi niiden kehittelyyn ja testaukseen liittyy huomattava riski [1].

Taylor [11] on kiinnittänyt huomiota siihen, että perinteiset kausaalisuhteiden ja kausaalilakien tunnistamiseen tähtäävät tutkimusotteet näyttävät pystyvän selittämään vain osan virheiden problematiikasta. Erityisesti riskien kohdalla, jolloin on otettava kantaa toimenpiteiden seurausvaikutuksiin, niiden laatuun ja vakavuuteen, ei voida välttää keskustelua arvoista ja moraalisista kysymyksistä.

Testauksen vaaroja kuvaava johdattava esimerkki on yli 25 vuoden takaa. Virheet ja häiriöt atk-systeemeissä eivät ole loppuneet, vaan aivan viime aikoinakin niitä on esiintynyt lukuisasti ja siksi niitä on syytä tarkastella lähemmin. Atk-alan tutkijat ovat perustaneet sähköisen ilmoitustaulun, jonne kootaan atk-systeemien virheitä eri maista opiksi ja varoitukseksi kollegoille. Software Engineering Notes -lehden avustaja Peter G. Neumann poimii ilmoitustaululta "parhaat" tapaukset ko. lehteen. Seuraavassa esitellyt tapaukset ovat kyseisestä lehdestä vuodelta 1988 [4].

2. Virheet muuttujan vaihteluvälin rajoilla

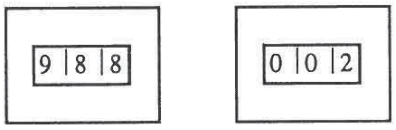
Pittsburghin kaupungin vesilaitos lähetti eräälle asiakkaalle laskun, jossa

edellinen lukema oli	988
uusi lukema oli	2
ja laskettu kulutus	0

Vesilaskutuksen atk-systeemi ei ollut varautunut siihen, että vesimittari "menee ympäri". Systeemi piti uutta lukemaa lukuvirheenä ja ohitti virheellisen tapauksen. Asiakas oli hyvin tyytyväinen, kun ohjelma ei ollut suorittanut vähennyslaskua nurinpäin eikä ollut laskuttanut 986 kuution kulutuksesta. Vastaavanlainen ongelma on aina olemassa, kun mittaria vaihdetaan.

Neumann [4] on syystä ihmeisssään, että vielä tänä päivänä pääsee tapahtumaan kyseinen virhe, jonka jäljet johtavat kyseisen systeemin suunnitteluun (mistake, [8]). Itse epäilisin, ettei ohjelmoija ollut koskaan nähnyt vesimittaria, eikä siis tiennyt, että mittarissa oli kolminumeroinen laskija. Ainakaan hän ei ollut pohtinut, mitä tapahtuu, kun laskija täyttyy. Voi olla, että ohjelmoija oli saanut laskutusohjelman spesifikaatiot suunnittelijalta. Ohjelman laatiminen spesifikaatioista koodiksi on

tapahtunut oikein, mutta spesifikaatio ei ole yhtäpitävä todellisuuden kanssa.



002-988 = -986 < 0

Kuva 1. Vesimittarin lukemat

Neumann mainitsee myös toisen rajaongelman, joka on tullut esille, kun USAsta on siirretty ohjelmisto Englantiin. Kun Ruotsista siirretään ohjelmisto Suomeen tai päinvastoin, on muutettava tekstit toiselle kielelle, mutta Neumannin tapauksessa ei ole kysymys tästä. Kyseinen ohjelmisto oli tarkoitettu palvelemaan lennonjohtajia, siis paikantamaan lähestyviä ja poislentäviä koneita. Kun paikanmääritys USAssa onnistuu yksiselitteisesti, niin Englannissa pulmana on Greenwichin nollameridiaani eli 0-pituuspiiri, josta lähtien lasketaan joko itäistä tai läntistä pituutta.

3. Yksilöivä tunniste

Kanadan Vancouverissa asuva Judi Sommer on elossa ja voi hyvin. Hän on noin 40-vuotias ja käy työssä. Hänellä on kuitenkin vaikeuksia saada verottaja uskomaan tätä. Vaikeudet alkoivat, kun hänen äitinsä kuoli 1986 ja siinä yhteydessä äidin kuolintodistukseen kirjoitettiin epähuomiossa tyttären henkilötunnus. Kesti kolme kuukautta, ennenkuin vuoden 1986 verotus saatiin tyttären kohdalta kuntoon. Näyttää siltä, että myös seuraavien vuosien verotuksen suhteen tulee ongelmia, sillä virheellinen tieto on levinnyt moneen rekisteriin.

Toinen tapaus henkilötunnusten problematiikasta koskee kahta Ann Marie O'Connoria, jotka molemmat asuvat New Yorkin osavaltiossa, ja joilla on sama henkilötunnus. He ovat molemmat yhtä pitkiä, heillä kummallakin on ruskeat hiukset ja ruskeat silmät. Molempien isän nimi ja

veljen nimi on Daniel. Valtion koneistolta meni 9 kuukautta nimen muutoksen kanssa, kun toinen Ann Marie O'Connor meni naimisiin.

Kolmas tapaus koskee kahta James Edward Tayloria, joilla lisäksi on sama syntymäpäivä (23.7.1919) ja jotka kumpikin ovat syntyneet Virginian osavaltiossa. Vaikka virhe huomattiin jo 1965, ja vaikka se on aiheuttanut jatkuvasti pientä harmia, niin sitä ei oltu saatu korjatuksi kahdeksan seuraavan vuoden aikana.

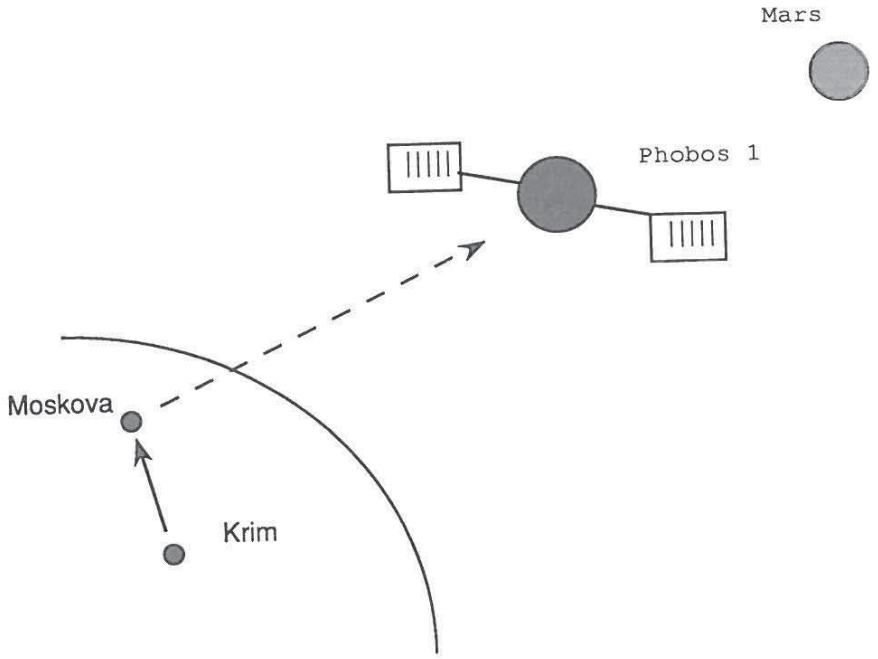
USAn henkilötunnukset on annettu em. henkilöille, kun manuaalijärjestelmästä on siirrytty atk-järjestelmään. Tällä hetkellä Suomessa annetaan kullekin lapselle oma uusi tunnuksensa. Mitä tulee kanadalaiseen tapaukseen, rekisterinpitäjien tulisi kuolemantapauksessa olla erittäin tarkkoja ja huolellisia. Jos virhe kuitenkin pääsee sattumaan, tulee se korjata hienotunteisesti. Minusta Shotter [9] varoittaa meitä osuvasti: Jos pidämme ihmistä yleistettynä koneena, on syytä pelätä, että muutumme itsekin koneiksi.

Suomessa on vast'ikään säädetty Henkilörekisterilaki (471/87), joka sallii kansalaisten rajoittaa henkilötunnuksen käyttöä mainontaan, markkinointiin, poimintatutkimuksiin jne. Lähetin tässä tarkoituksessa Autorekisterikeskukselle kirjeen, jossa pyysin, ettei ajoneuvorekisterissä olevia tietojani käytettäisi suoramainontaa, puhelinmyyntiä eikä osoitepalvelua varten. Mutta mainittu kirje ei ollut riittävä, vaan sain paluupostissa lomakkeen, joka minun oli täytettävä, ennenkuin asia voitaisiin hoitaa.

Minusta se, että asian hoitamiseksi vaadittiin kaksi kirjettä, tuntui liialliselta. Täytin kuitenkin annetun lomakkeen ja liitin mukaan erillisen kirjeen, jossa selitin, ettei lomakkeen tietojen vaatiminen ole ainakaan atk-teknisesti perusteltavissa, koska minun autoni ja monen muunkin autonomistajan ajoneuvotietueen yksilöintiin riittää nimi ja osoite. VTKK:lla on nimittäin kyseiseen tarkoitukseen Minttu-ohjelma, jolla voidaan poimia tarkasteluun tietueita antamalla poimintaehto. Kysyin lisäksi, mikä on se todellinen syy, jonka vuoksi markkinakielto on tehty niin vaikeaksi. Sain vastaukseksi atk-teknistä selitystä ja pyynnön asioida jatkossa puhelimitse.

Olen omalta kannaltani päätynyt siihen, että markkinakiellon asettaminen on tehty vaikeaksi siksi, että osoitetietojen myynti muodostaa Autorekisterikeskukselle merkittävän tulolähteen. Tämän rahahanan

ottaminen ja niiden käyttöönotto ovat yleensä normaalia jokapäiväistä toimintaa, jonka soisi muodostuvan rutiiniksi.



Kuva 2. Phobos-luotaimen ohjausaseman siirto..

6. Iranilaisen matkustajakoneen alasampuminen

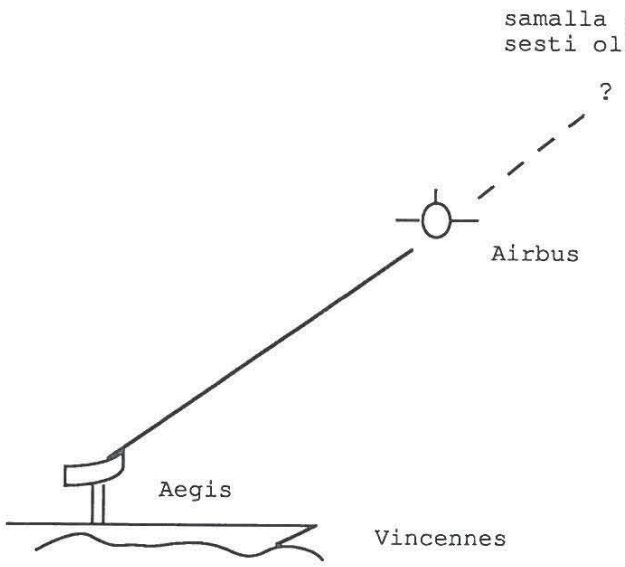
USAn risteilijä Vincennes ampui heinäkuun 3:na päivänä vuonna 1988 alas Iranin lentoyhtiön Airbus-matkustajakoneen. 290 ihmistä kuoli. Risteilijän tietokoneohjattu Aegis-systeemi oli tarkoitettu tunnistamaan lähestyviä ohjuksia eikä lentokoneita. Virallinen tutkimusraportti kertoi, että näyttöruudulla oli aivan oikea kuva, nimittäin lentokoneesta, joka oli nousussa oikealla nopeudella ja oikeassa kurssissa, ja että näyttöruudun kuva oli taistelutilanteessa tulkittu väärin, nimittäin kiihtyvällä nopeudella lentäväksi ja kurssistaan poikenneeksi laskevaksi koneeksi.

Atk:n riskien kannalta ei juuri ole eroa sillä, syytetäänkö tietokonesysteemiä vai ihmistä, joka tulkitsi tietokoneen näytölle kuvatut tiedot väärin, sillä tarkastelun kohteeksi on otettava koko järjestelmä, jossa ihminen on oleellinen osa systeemiä. Jos henkilö on sijoitettu tällaiseen järjestelmään siten, ettei hänellä ole riittävästi aikaa hankkia tai saada asianmukaisia tietoja valintojensa perusteiksi, niin systeemi on väärin suunniteltu. Tässä tapauksessa Aegis-systeemiä käytettiin tehtävään, johon sitä ei ollut tarkoitettu. Lisäksi se ei ollut selviytynyt erikoisen hyvin aikaisemmista testeistä, sillä Defense Week -lehti kertoi, että Aegis-testien takaiskut kuvaava julkinen raportti oli jätetty pois USAn kongressille toimitetusta aineistosta. Lisäksi Aegis-systeemin käyttöliittymä oli huonosti suunniteltu.

Atk-systeemien käytössä sotilaallisiin tarkoituksiin törmätään aina kohdealueen spesifioinnin ongelmaan [10]. Ei auta mitään, vaikka asejärjestelmää ohjaava ohjelma olisi todistettu oikein johdetuksi spesifikaatioista, jos spesifikaatioiden määrittämä malli ei vastaa todellisuutta.

Ihminen on tietokonetta paljon joustavampi mukautumaan ja sopeutumaan uusiin tilanteisiin. Rasmussen [6] katsookin, että ihmistä tarvitaan (rauhanomaisiin tehtäviin tarkoitetun) systeemin komponenttina silloin, kun on odotettavissa hankaluuksia, ts. kun ollaan epävarmoja, millaisiin tehtäviin ihminen-kone -yhdistelmää tullaan käyttämään. Mutta ihmisellä tulee tällöin olla realistinen ja totuudenmukainen kuva sekä uudesta tehtävästä että systeemin atk-osuudesta. Lisäksi hänen tulee tietää, mihin tehtäviin atk-systeemiä voidaan käyttää ja mihin ei.

Iranilaisen matkustajakoneen tapauksen ympärillä liikkuu monia tarinoita ja huhuja. Erään sellaisen mukaan Vincennesin tutka olisi tunnistanut sotilaskoneen samalla säteellä kuin matkustajakone. Jätän tämän ja muut tarinat pohtimatta ja esitän sen sijaan lopuksi kaksi yleisempää kysymystä: Oliko Airbus-matkustajakoneen pakko olla siellä, missä se oli?, ja Oliko Vincennesin pakko olla siellä, missä se oli?



Kuva 3. Iranilaiskoneen alasampuminen.

Viitteet

1. A. K. Dewdney, Computer recreations – of worms, viruses and Core War. *Scientific American*, March 1989, 90-93.
2. Kari T. Eloranta, Heuristiikat ja heuristisuus – käsittelyongelmista ja niiden ratkaisemisen metodologiasta hallinto-opin näkökulmasta. Tampereen yliopisto, 1974.
3. Pertti Järvinen, Problems of job design met in information system development and maintenance. *BIT* **20** (1980), 15-24.
4. Peter G. Neumann, Risks to the public in computers and related systems. *ACM Software Engineering Notes* **13**, 4 (1988), 3-20.
5. Jens Rasmussen, What can be learned from human error reports? In *Changes in Working Life*, Duncan, Gruneberg and Wallis (Eds.), Wiley, New York, 1980.

6. Jens Rasmussen, Definition of human error and a taxonomy for technical system design. In Rasmussen, Duncan and Leplat (Eds.), *New Technology and Human Error*, Wiley, New York, 1987, 23-30.
7. Jens Rasmussen, Cognitive control and human error mechanisms. In Rasmussen, Duncan and Leplat (Eds.), *New Technology and Human Error*, Wiley, New York, 1987, 53-61.
8. J. Reason, The psychology of mistakes: A brief review of planning failures. In Rasmussen, Duncan and Leplat (Eds.), *New Technology and Human Error*, Wiley, New York, 1987, 45-52.
9. J. Shotter, Men the magicians: the duality of social being and the structure of moral worlds. In *Models of Man*, Chapman and Jones (Eds.), British Psychological Society, Leicester, 1980, 13-34.
10. Brian Smith, Limits of correctness in computers. Stanford University, Center of the Study of Languages and Information, Report CSLI-85-36, 1985.
11. D. H. Taylor, The hermeneutics of accidents and safety. In Rasmussen, Duncan and Leplat (Eds.), *New Technology and Human Error*, Wiley, New York, 1987, 31-41.

PROFESSORI PERTTI JÄRVINEN
TAMPEREEN YLIOPISTO
TIETOJENKÄSITTELYOPIN LAITOS
PL 607
33101 TAMPERE
pj@utacs.uta.fi

Tietotekniikka organisaatioissa - kustannussäästäjästä muutoksen agentiksi

KALLE LYYTINEN, JYVÄSKYLÄN YLIOPISTO

Abstrakti

Tietotekniikka on kehittynyt 30 vuoden aikana organisaatioiden taustaoperaatioita ja rutiineja hoitavasta kustannussäästäjästä merkittäväksi organisaatioiden tavoitteisiin, toimintakeinoihin ja rakenteeseen vaikuttavaksi tekijäksi. Muutos on tapahtunut 30 vuoden aikana usean vaiheen kautta. Artikkelissa kartoitetaan ja luokitellaan tietotekniikan omaksumisessa käytetyt vaiheistukset neljään luokkaan: edellytyksiin perustuvat vaiheistukset, muotoihin perustuvat vaiheistukset, soveltamiskohteisiin perustuvat vaiheistukset ja vuorovaikutusmallit. Vuorovaikutusmalleista tarkastellaan lähemmin integroivia periodisointeja ja Nolanin vaiheteoriaa. Lopuksi esitetään kolme ajankohtaista ja tuoretta tietotekniikan omaksumista organisaatioissa selittävää lähestymistapaa: transaktiokustannusteoria, Marchin carbage-can malli ja poliittiset teoriat.

1. Johdanto

> "We are in the midst of an information age. In most contemporary organizations information is climbing up the chain of dependency. In some the thrust of information is so powerful that the converse of Chandler's logic holds: Information systems beget strategy." [14]

Tietokoneen tunkeutuminen tehtaaseen, toimistoon ja kotiin on tapahtunut nopeasti — onhan kulunut vasta 30 vuotta siitä, kun maamme ensimmäinen tietokone hankittiin Postipankkiin vuonna 1958. Nykyisin valtaosa työikäisestä väestöstä on päivittäin tekemisissä tietokoneiden kanssa työssään ja jokainen meistä on välillisesti tietokoneiden vaikutuspiirissä käyttäessään julkisia tai yksityisiä palveluksia kuten puhelinta, pankkia, jne.

Tietotekniikka on alusta alkaen ollut nimenomaan "organisaatioteknologiaa", koska sillä on käyttöä laajassa mitassa vain osana modernien organisaatioiden hallintoa esim. ratkaistaessa resurssien allokointi- tai valintaongelmia, koordinaatio-ongelmia, tai tuotettaessa palveluja. Tämän kiinteän yhteyden vuoksi on tietotekniikan soveltamisen keinoja, päämääriä, sisältöä ja vaikutuksia vaikea ymmärtää, jos tietotekniikkaa ei tarkastele nimenomaan "tietotekniikkana organisaatioissa". "Tietotekniikka organisaatioissa" on kuitenkin vasta ensimmäinen edellytys tietokoneiden kasvavan käytön ymmärtämiselle. Toinen edellytys on, että erittelemme tarkoin kunkinhetkisiä omaksumisen edellytyksiä ja muotoja sekä tietotekniikan soveltamisen kohteita ja jäsennämme näiden ilmiöiden välisiä monimutkaisisia vuorovaikutussuhteita. Omaksumisen edellytyksillä tarkoitan tietotekniikan kehitystasoa ja hintaa, saatavilla olevia teknisiä valmiuksia soveltaa sitä sekä taloudellisia resursseja investoida tietotekniikkaan. Omaksumisen muodoilla tarkoitan organisatorisia ratkaisuja ja prosesseja, jotka edesauttavat tai jarruttavat tietotekniikan hyväksikäyttöä. Tietotekniikan soveltamiskohteilla tarkoitan sovellustyyppejä ja sovellusarkkitehtuurien määrittelyyn liittyviä ongelmia.

Tietotekniikan ja organisaatioiden muuttuvaa vuorovaikutusta olen pyrkinyt luonnehtimaan esitelmäni otsikon jälkiosassa "kustannussäästäjästä muutoksen agentiksi". Luentoni tarkoituksena onkin selvittää, millä

tavoin tietotekniikan soveltamisen edellytykset, muodot ja kohteet ovat organisaatioissa muuttuneet viimeisen 30 vuoden aikana ja mitkä mahdolliset tekijät ja piirteet voivat selittää näitä muutoksia.

2. Tietotekniikan soveltamisen kehityskausista

Tietokoneiden ensimmäinen soveltamiskohde löytyi vaativasta teknisestä ja tieteellisestä laskennasta. Ensimmäinen tietokone MARK I, jonka Harvardin matematiikan professori Howard Aiken kehitti IBM:n kanssa yhteistyössä oli nimenomaan automaattinen laskukone (Automatic Sequence Controlled Calculator). Tuona aikana ei tietokoneille nähty juurikaan muuta käyttöä kuin tehokkaina laskukoneina. Oiva esimerkki tästä on Aikenin aikoinaan laatima laskelma siitä, että maailmassa on tarve korkeintaan kuuteen tietokoneeseen, koska enempää tarvetta raskaaseen laskentaan ei ole olemassa.

1950-luvun puolivälissä tietokoneiden luotettavuus, hinta ja toiminnalliset ominaisuudet mahdollistivat tietokoneen tunkeutumisen hallintoon. Laskukone muuttui merkkien käsittelijäksi, varastoijaksi ja siirtäjäksi — funktiot, jotka yhdistyvät kaikissa hallinnollisissa sovelluksissa. Ensimmäinen hallinnollinen tietojärjestelmä toteutettiin vuonna 1954, kun erään Yhdysvaltain armeijan osaston palkanlaskenta siirrettiin tietokoneelle. Uusia sovelluksia rakennettiin tämän jälkeen nopeassa tahdissa ja ensimmäiset tietokoneistetut tietojärjestelmät näkivät Suomessakin päivänvalon ennen vuosikymmenen taitetta.

Tietokoneiden soveltamistahti hallintoon on ollut kiivas ja tietotekniikan vaikutukset organisaatioihin monipuoliset. Usein uuden tekniikan omaksuminen on ollut hallitsematonta ja vaikutusten tasoa, laajuutta ja suuntaa on ollut vaikea arvioida ja ennustaa (ks. esim. [1, 6]). Tämän vuoksi tieteellisessä yhteisössä on jatkuvasti käyty runsasta ja osin värikästäkin keskustelua tietotekniikan omaksumisen muodoista, sisällöstä ja suunnasta (ks. esim. [13, 17, 9]). On myös eritelty tekijöitä, joilla selitetään tietokoneiden soveltamistahdin ja organisaatioiden rakenteen ja toiminnan erityispiirteiden välisiä riippuvuuksia. Vaikutusten arviointi on vaihdellut spekulaatioista perusteellisiin, mutta vaikeasti tulkittaviin, tilastollisiin

analyyseihin. Tietokoneistumista selittävät tekijät ovat vaihdelleet teknologisesta determinismistä (tietokoneet ja niiden ominaisuudet työntävät organisaatioita tiettyyn suuntaan) erilaisiin strategisen valinnan teorioihin (tietokoneistumisen suunta määräytyy organisaation johtavan eliitin strategisista valinnoista). Välimaastossa ovat selityksiään tarjonneet prosessiteorioiden ja vuorovaikutusmallien kannattajat (tietokoneistumisen suunta voidaan selittää vain erittelemällä tarkasti kutakin omaksumisen prosessia ja sitä kuinka organisaatiot ja tietotekniikka "kohtaavat" tuossa prosessissa). Mitään yhtenäistä ja yksiselitteistä selitystä, teoriasta puhumattakaan, tietokoneistumisen ja organisaatioiden suhteista ei näiden 30 vuoden aikana ole kuitenkaan syntynyt.

Samanaikaisesti tutkimus on tuottanut erilaisia kuvauksia, vaiheistuksia tai malleja, joilla tietotekniikan omaksumisen tasoa organisaatioissa on pyritty kuvaamaan. Tyypillisiä ovat vaiheistukset, joissa periodisoidaan tietokoneistumisen vaiheet organisaatiossa jonkin tai joidenkin tekijöiden perusteella. Usein tällaiset vaiheistukset ovat luonteeltaan kuvaavia ja niiden pohjalta ei kyetä selittämään tietokoneistumisen suuntaa ja tasoa. Tietokoneistumisen vaiheistuksia on kehitelty useista lähtökohdista ja erilaisia tarkoituksia varten. Kirjallisuudesta löytää ainakin seuraavia vaiheistuksia:

(1) *tietokoneistumisen edellytyksiä kuvaavat vaiheistukset,*
(2) *omaksumisen muotoja ja prosessia kuvaavat vaiheistukset,*
(3) *tietokoneistumisen soveltamiskohteita kuvaavat vaiheistukset,* ja
(4) kohtia (1), (2) ja (3) yhdistelevät *vuorovaikutusmallit.*

Näistä vuorovaikutusmalleihin palaan hieman tuonnempana.

3. Yksinkertaiset vaiheistukset

Tietokoneistumisen edellytyksistä nousevat tarkastelut ottavat tarkastelun kohteeksi tietotekniikan sisäisen kehitystason tai muut tietokoneistumista ulkoisesti mittaavat tekijät, esim. atk-henkilöstön suhteellisen määrän yrityksessä. Tyypillinen edellytyksistä lähtevä tarkastelu on tietokoneistumisen vaiheistaminen tietokonesukupolvien mukaan. Tällöin jaamme

tietokoneistumisen vaiheet tyhjiöputkien, transistorien, piiri-integraation, hajautettujen järjestelmien jne. mukaan. Näillä tietokoneistumisen edellytyksien radikaaleilla muutoksilla katsotaan vastaavasti olevan radikaaleja vaikutuksia atk-toiminnan organisoitumisen muotoihin (esimerkiksi organisatoriset ratkaisut, joiden puitteissa johdetaan ja hallitaan organisaation tietojenkäsittelytoimintoa) ja soveltamiskohteisiin (uudet sovellukset, sovellusalueet ja arkkitehtuurit). Tunnetuin tämän suuntainen tarkastelu on Frederic Withingtonin vuonna 1974 Harvard Business Review -lehdessä esittämä vaihejako [28].

Omaksumisen muodoista ja prosesseista etenevät vaiheistukset koostuvat hyvin monipuolisesta ja vaihtelevasta joukosta historiallisia analyyseja. Omaksumisen muodoista ensimmäinen ja ilmiselvä vaiheistus on tarkastella *atk-toiminnan organisatorisen aseman muutoksia* yrityksissä. Alkuaikojen tilanteesta, jolloin atk-osasto muodosti pienen yksikön taloushallinnon ja/tai teknisen suunnittelun alaisuudessa, on atk-toiminto saavuttanut useissa yrityksissä tasaveroisen aseman muiden toimintojen kuten markkinoinnin tai tuotannon kanssa ja atk-toiminnon johtaja raportoi suoraan yrityksen ylimmälle johdolle (ks. [5, 3]).

Muita omaksumisen muotoihin kohdistuvia tarkasteluja ovat *atk-toiminnon aseman ja roolin vaiheistus* yrityksen sisällä, koska tietojenkäsittelyn organisatorisen aseman muutosta on usein seurannut myös toimintastrategian muutos. Alunperin atk-toimintoa käsiteltiin vain kiinteisiin kustannuksiin vaikuttavana kustannuspaikkana, jonka tehtävä ja rooli oli organisaation muita toimintoja tukeva. 80-luvun lopulla atk:sta on useissa yrityksissä muodostunut "business within business" [29], joka toimii omien sisäisten markkinoidensa puitteissa. Samalla atk-toiminnon rooli on suuntautunut kasvavasti ulospäin ja sitä tarkastellaan yhä enemmän palveluita tarjoavana yksikkönä, joka ratkaisevasti vaikuttaa yrityksen kilpailukykyyn, arvoketjun rakenteeseen ja johtaa yrityksen strategisten keinojen valintaa. Valaiseva esimerkki tästä on viime aikoina yleistynyt käytäntö perustaa yritysten atk-osastoista erillisiä yrityksiä, jotka myyvät palveluja entiselle emoyhtiölleen markkinoiden välityksellä.

Tyypillinen tähän luokkaan kuuluva vaiheistus on neljän oxfordilaisen tutkijan [7] laatima luokittelu atk-toiminnan johtamisessa tapahtuneista muutoksista. Tutkimuksessaan he jakavat atk-toiminnan johtamisen

kolmeen vaiheeseen: tuottamisvaihe, uudelleenorientaatio ja uudelleenorganisointi. Tuottamisvaiheessa atk-toiminto keskittyy tuottamaan kiinteitä tuotteita mahdollisimman halvalla ja sen johtamisessa huomio on sisäänpäin kääntynyt. Uudelleenorientaatiossa atk-toiminto järjestetään liiketoimintayksiköksi ja sitä johdetaan tavalla, joka on aktiivinen, ulospäinsuuntautunut ja pyrkii alistamaan tietotekniikan soveltamisen yritysstrategian osaksi. Uudellenorganisaatiovaiheessa pyritään atk-toiminnon sisäinen organisaatio, tehtävänjako ja atk-toiminnon johtamistapa sovittamaan yhteen atk-toiminnon uuden roolin kanssa.

Omaksumisen muotoja ja prosesseja koskevissa tarkasteluissa oman luokkansa muodostavat myös *tietotekniikan soveltamisprosessien luokittelu* aikakausittain. Kukin soveltamisprosessien luokka edustaa erilaista organisatorista puitetta ja säännöstöä, joka määrittelee kuinka ja missä tahdissa organisaatio omaksuu tietotekniikkaa. Tällöin voimme erottaa ainakin seuraavat vaiheet: *empiiriset kokeilut, formaalit vaihejakomallit, formaali sovellusten valinta ja ajoitusmenettely* ja *organisaationkattava sovellusarkkitehtuurin suunnittelu* (ks. esim. [3]).

Empiirisissä kokeiluissa organisaatio sallii tietotekniikan kokeilun ja omaksumisen ilman kattavaa säännöstöä ja tarkkoja kriteerejä kuinka sovelluksia tulee kehittää. Ominaista tälle omaksumisprosessille on riittävän löyhien resurssien olemassaolo sekä "championit" — henkilöt, jotka innostuvat ja "taistelevat" uuden tekniikan omaksumisen puolesta. Tämä oli tyypillinen menettelytapa tietojenkäsittelyn alkuaikoina, jolloin "atk-gurut" suunnittelivat ja toteuttivat sovelluksia ilman laajempaa muodollista kontrollia. Tämä omaksumistapa on myös tyypillinen useille muille uuden tekniikan omaksumisen esiasteille. Kokeellinen, yrityksen ja erehdyksen kautta tapahtuva omaksuminen on ollut tyypillistä mm. omavaraiskäytölle, päätöksenteontukijärjestelmille ja asiantuntijajärjestelmille, jotka kaikki ovat aikanaan edustaneet uutta tietoteknistä aaltoa.

Formaaleissa vaihejakomalleissa määritellään tarkemmin vaiheet ja askeleet, joita noudattaen uuden sovelluksen kehittäminen ja käyttöönotto tapahtuvat. Keskeinen asema vaihejakomalleissa on muodollisen päätöksentekomenettelyn soveltaminen, jolla pyritään (valitettavan usein ilman tuloksia) varmistamaan, että sovellus vastaa organisaation tarpeita ja että sovelluksen kehittämiseen ei käytetä liikaa organisaation resursseja.

Formaalisssa sovellusten valinta- ja ajoitusmenettelyssä tunnustetaan usean samanaikaisesti resursseista kilpailevan sovelluksen olemassaolo organisaatiossa. Tällöin on ensin ratkaistava mihin sovelluksiin organisaation niukat resurssit jaetaan ennen varsinaisten sovellusten kehittämishankkeitten käynnistämistä. Valinnassa joudutaan täten ottamaan huomioon muutkin seikat kuin yksittäisten sovellusten tarve käyttäjien näkökulmasta (tai atk-suunnittelijan näkökulmasta). Valintamenettely toteutetaan useimmiten erilaisten ohjausryhmien, johtamiskomiteoiden ja muodollisten päätösmenettelyjen virallistamisella. Päätösmenettelyssä pyritään rationaalisti punnitsemaan mm. sovellusten väliset riippuvuudet, teknologinen epäjatkuvuus ja teknisten ratkaisujen suljettu luonne jne. Toinen kilpaileva (ja usein parempi) selitys tämän organisaation sisäisen "kynnyksen" kehittymiseen tietotekniikan soveltamiselle on organisaatioiden eri osastojen ja ryhmien välinen kilpailu atk-resursseista ja niiden halu suunnata näiden resurssien käyttö palvelemaan omia rajoitettuja intressejään. Tällöin tietotekniikan johtamisryhmä muodostuu poliittiseksi areenaksi, jossa tietotekniikan soveltamiskohteet ja -tavoitteet ovat jatkuvan kaupankäynnin ja neuvottelun alaisia (ks. esim. [10]).

Organisaation kattavassa tietojenkäsittelysuunnittelussa ei sovelluksille anneta enää suurtakaan erillistä merkitystä. Sovellukset syntyvät ja kuolevat osana laajempaa koko organisaation kattavaa tietotekniikan soveltamis- ja omaksumissuunnitelmaa. Sovelluksille ei anneta erillistä ja itsenäistä asemaa muiden tietotekniikan omaksumista säätelevien suunnitelmien kuten tietoarkkitehtuurin, tietoteknisten ratkaisujen, ja tietoliikenneratkaisujen rinnalla ja ne muodostavat kokonaisuutena organisaation tietotekniikan soveltamis- ja omaksumispuitteet. Tästäkin omaksumisprosessien luokasta voidaan esittää myös tulkinta, jossa tietotekniikan soveltamis- ja omaksumissuunnitelma edustaa poliittista kompromissia ja sen sisällöstä ja suunnasta käydään jatkuvaa julkista tai piilotettua poliittista taistelua.

Kuten kaikissa vaiheistuksissa ei omaksumisprosessien kronologinen muotojen kehitys noudata missään puhtaasti edellä esitettyä kaavaa. Jokaisessa organisaatiossa on aina jäänteitä ja piirteitä aiemmista kehitysvaiheista. Lisäksi uudet teknologiset innovaatiot luovat aina epäjatkuvuutta kehityksen suuntaan ja polveiluja aiempiin omaksumismalleihin. Kuitenkin

edellä esitetty omaksumismallien vaiheistus tuntuu yleisesti pätevän — yksittäisistä uuden tekniikan kokeiluista edetään kohti organisaation kattavaa suunnittelua ja muodollista ohjausta. Sama kehityskaari toistuu usein myös uusien teknologisten innovaatioiden omaksumisessa. Niin kutsuttu itsenäiskäyttö tai omavaraiskäyttö on tästä hyvä esimerkki. Alunperin hallitsemattomasta mikrotietokoneilla toteutetusta taulukkolaskennasta ja tekstinkäsittelystä alkanut kokeilu on edennyt nyt vaiheeseen, jossa sitä useissa yrityksissä ohjataan ja suunnataan organisaationkattavan politiikan mukaisesti.

Kolmas tapa vaiheistaa tietotekniikan omaksumisastetta organisaatioissa on tarkastella *sovellusten tyyppejä* ja *tietotekniikan soveltamiskohteiden laadullisia muutoksia*. Tämä luokittelutapa on ollut hyvin suosittu niin tutkijoiden keskuudessa kuin käytännön konsultointi- ja opetustehtävissä toimivien piirissä. Tunnetut jaottelut erä-, tapahtuma-, ja hajautettuihin sovelluksiin — tai jako tiedostopohjaisiin ja tietokantapohjaisiin sovelluksiin ovat tästä tunnetuimpia esimerkkejä. Näiden luokittelujen ongelmana on yleisyys ja liika sidonnaisuus teknisiin ratkaisuihin — ne paremminkin sijaitsevat soveltamisen edellytysten luokittelun ja soveltamiskohteiden luokittelun välimaastossa.

Hyödyllisempi soveltamiskohteiden vaiheistus on käsittääkseni luokitella *lähestymistapoja* [25], joiden tuloksena "tuotetaan" tietynlainen tapa etsiä tietotekniikan soveltamiskohteita ja tuottaa ratkaisuja soveltamiskohteissa havaittuihin ongelmiin. Lähestymistavat muodostavat ikäänkuin silmälasit, joiden siivilöimästä kohteiden joukosta organisaatiot etsivät kunakin hetkenä sopivat soveltamiskohteet resurssien ja kypsyytensä mukaan. Lähestymistapa sisältää tällöin perustelut miksi tietotekniikkaa kannattaa/tulee soveltaa; käsitteet ja menetelmät, joilla identifioidaan soveltamiskohteet ja määritellään tietotekniikan soveltamisen tapa; sekä keinot, joilla tietotekniset ratkaisut muotoillaan ja toteutetaan. Tyypillisiä, ja laajalti käytössä olleita lähestymistapoja ovat olleet: *rutiinikeskeinen*, *tietokeskeinen*, *käyttäjäkeskeinen*, ja *palvelukeskeinen*. Yllä esitetty järjestys vastaa myös näiden lähestymistapojen kronologista järjestystä.

Rutiinikeskeisessä lähestymistavassa tietotekniikka muodostaa keinon automatisoida toiminnallisia rutiineja. Tietotekniikan käytön perusteluna käytetään kustannusten leikkaamista. *Tietokeskeisessä*

lähestymistavassa tietotekniikka muodostaa keinon integroida ja varastoida organisaation keskeiset tietovarannot. Perusteluna käytetään organisaatioiden toimintojen lisääntynyttä tehokkuutta ja tuottavuutta, koska saatavilla olevan tiedon laatu ja oikea-aikaisuus paranee integroinnin ansiosta. *Käyttäjäkeskeisessä* lähestymistavassa tietotekniikkaa käytetään tukemaan yksittäisten käyttäjien tai käyttäjäryhmien tehtäviä, erityisesti päätöksenteko-ongelmien ratkaisua. Tietotekniikan soveltamisen perusteluna käytetään parantunutta tehtävien tuottavuutta ja suurempaa käyttäjätyytyväisyttä. *Palvelukeskeisessä* lähestymistavassa tietotekniikkaa käytetään muotoilemaan/tuottamaan/välittämään ja/tai valvomaan organisaation tuottamia palveluja. Tietotekniikan soveltamisen perusteluna käytetään parantunutta palvelua ja sen kautta yrityksen/organisaation parempaa kilpailuasemaa/kykyä markkinoilla. Pitemmällä tähtäimellä pyritään erilaiset palvelut tuottamaan taloudellisesti tehokkaimmalla tavalla.

Edellä käsiteltyjen, ja muiden vastaavien kirjallisuudessa esitettyjen luokittelujen etuna on, että ne valaisevat tiettyjä laadullisia muutoksia tietotekniikan soveltamisessa organisaatioissa. Niitä voidaan käyttää siten ymmärtämään esimerkiksi tietokoneistumisen edellytyksissä, muodoissa ja kohteissa tapahtunutta kehitystä ja arvioimaan kehityssuuntaa. Yksittäisten vaiheistusten ongelmana on, että ne ovat liian yksinkertaisia kuvaamaan seikkaperäisesti ja monipuolisesti tietokoneistumisen ja organisaatioiden välisiä riippuvuuksia. Tarkempiin ja kokonaisvaltaisempiin kuvauksiin on päästy kehittämällä erilaisia *vuorovaikutusmalleja*. Tarkastelen seuraavassa kahta vuorovaikutusmallia: *integroivaa periodisointia* ja *Nolanin vaiheteoriaa*.

4. Vuorovaikutusmallit

Integroiva periodisointi tarkoittaa tietojenkäsittelyn evoluution periodisointia siten, että edellä käsitellyt muutokset edellytyksissä, muodoissa ja kohteissa integroidaan yhden mallin puitteisiin. Malli on luonteeltaan kuvaileva, eikä se käsittele eri mallin komponenttien (edellykset, muodot, kohteet) välisiä keskinäisiä riippuvuuksia ja vuorovaikutusmekanismeja.

Malli ei siten täytä varsinaisen vuorovaikutusmallin vaatimuksia, vaan lähinnä yhdistää eri komponenttien periodisoinnit yhden rakenteen alle.

Kirjallisuudessa on esitetty useita integroivia periodisointeja. Niissä käytetyt termit eri vaiheista vaihtelevat. Mielenkiintoista kyllä, lähes kaikki tutkijat ovat yleensä päätyneet kolmeen vaiheeseen. Esimerkiksi Mason [15] erottaa "tehokkuuden" (efficiency era), "vaikuttavuuden" (effectiveness era) ja "kokonaisvaikutuksen" (systemic era) kaudet. Somogyi ja Galliers [22] jakavat kaudet vastavaasti "tietojenkäsittelyn" (data processing), "johdon palvelujen" (management services) ja "informaation käsittelyn" (information processing) kausiin. Oltiinpa termien oikeaan osuvuudesta mitä mieltä tahansa, on jaottelu kuitenkin eri tutkijoiden kohdalla hyvin samansuuntainen ja johtopäätökset samankaltaiset.

Hyvän yhteenvedon integroivasta periodisoinnista tarjoavat Somogyin ja Galliersin pohjalta laaditut yhteenvetotaulukot (taulukot 1, 2 ja 3). Näissä taulukoissa voimme erottaa muutokset tietokoneistumisen

AIKAKAUSI	TEKNIIKAN LUONNE	SYSTEEMIOPERAATIOT	JÄRJESTELMÄN LUONNE
Tietojen-käsittely	- hankala ja työläs - epäluotettava - rajoitettu syöttö/tulostus	- keskitetty - etäinen	- eräajo - erillinen
Johdon palvelut	- main frame koneet - käyttöjärjestelmä - näyttöpäätteeet - TKHJ	- hajautettu käsittely - tapahtumaorientaatio	- tosiaika - vuorovaikutteinen - keskitetty tietokanta
Informaation käsittely	- konvergoituva tekniikka - miniaturisaatio - pakkaukset - PC/työasema	- käyttäjä/business-perustainen - information center - balanssi keskityksen ja hajautuksen välillä	- käyttäjäystävällinen - verkkopohjainen - sovellusten integrointi
Trendi	Erillinen | Paketoitu, Yhtenäistyvä	Vaikeasti saatavilla, Säädelty | Tukeva, Saatavilla	Joustamaton, Pirstottu | Joustava, Sopeutuva

Taulukko 1. Muutokset tietokoneistumisen edellytyksissä.

AIKAKAUSI	SYSTEEMITYÖN ALUE	SYSTEEMITYÖN LUONNE
Tietojen-käsittely	- Ohjelmointiprosessi	- Empiirinen - Kokeileva
Johdon palvelut	- Analyysi ja suunnittelu - Elinjakso - Lisääntyvä ylläpito - Tiedonhallinta	- Analyyttinen ja jäsentävä - Projektinhallinta
Informaation käsittely	- IRM - Kommunikaatio - Käyttäjien osallistuminen - Yhdistä liikestrategiaan	- Osallistuva - Omavaraiskäyttö - Prototyypit - Automatisoidut välineet - TJ suunnittelu/strategia
Trendi	Tekninen osaaminen | Liiketoiminnan tuki, Strateginen kysymys	Ad hoc | Metodinen, Sosiaalinen, Liikevetoinen

Taulukko 2. Tietokoneistumisen muodot.

edellytyksissä (tekniikan luonne, systeemioperaatioiden luonne, ja tietojärjestelmien luonne), tietokoneistumisen muodoissa (systeemityön alue, systeemityön luonne), ja tietotekniikan soveltamisen kohteissa (tietotekniikan käytön syyt, sovellustyypit, luonteenomainen tapa käyttää tekniikkaa).

Taulukot osoitttavat selvästi, että tietojärjestelmien merkitys yrityksille on läpikäynyt vallankumouksellisen muutoksen viimeisten 30 vuoden aikana ja että tuo kehitys on edennyt määrättyjen (joskin vaikeasti erotettavien) vaiheiden lävitse. Tuona aikana tietokoneistumisen edellytykset, tietokoneistumisen organisatoriset muodot ja kohteet ovat muuttuneet ratkaisevasti. Tekniikka on halventunut ja yhtenäistynyt. Sovellukset ovat monipuolistuneet ja organisaatiot ovat (usein vaikean läksyn kautta) oppineet tehokkaammin soveltamaan uutta tekniikkaa. Tietotekniikka on muuttunut kustannussäästäjästä organisaatioiden muutoksen agentiksi. Samanaikaisesti tekniikan hankinnasta, soveltamisesta ja hyväksikäytöstä vastaavien organisatorinen asema ja valta on tullut näkyväksi ja heidän kykynsä ja halunsa ajaa omia etujaan organisaatioiden sisällä on parantunut. Siten

AIKAKAUSI	SOVELLUSTYYPPI	TIETOTEKNIIKAN KÄYTÖN SYY	LUONTEENOMAINEN SOVELTAMISTAPA
Tietojen-käsittely	- Tieteellinen laskenta - Hallinnolliset taustatehtävät	- Economies of scale	- Automatisointi
Johdon palvelut	- MIS ja valvonta-järjestelmät - Yritystietokannat	- Tehokkaat toiminnot - Tehokas valvonta	- Johdon tuki - Operatiivinen ohjaus
Informaation käsittely	- Monipuolisuus: päätöstuki käyttäjämallit asiantuntija-järjestelmät rutiinikäsittely & strateginen sovellus	- Liiketoiminnan kehittäminen - Kilpailuetu/ välttämättömyys	- Liiketoiminnan tuki - Strateginen ase
Trendi	Operatiivinen Taktinen Strateginen	Kustannusten leikkaus Liiketoiminnan tuki	Teknologiavetoinen Liikevetoinen

Taulukko 3. Tietokoneistumisen kohteet.

tietokoneistumisesta on samalla tullut, kuten professori Rob Kling Kalifornian yliopistosta on todennut, sosiaalinen liike, jota sitoo yhtenäinen ideologia ja rajatut sektionaaliset intressit [11].

Jos tarkastelemme kriittisesti integroivaa periodisointia, voimme todeta, ettei se selitä kuinka eri tietokoneistumisen osatekijät vaikuttavat toisiinsa ja mikä on ollut se sisäinen mekanismi, joka on työntänyt organisaatiota kohti yhä laajempaa tietokoneistumista ja miksi organisaatiot ovat siirtyneet tietokoneistumisen vaiheesta toiseen. Ensimmäinen, ja tunnetuin yritys selittää niitä mekanismeja, jotka työntävät organisaatiot kohti laajenevaa tietokoneistumista on Nolanin vaiheteoria.

5. Nolanin vaiheteoria

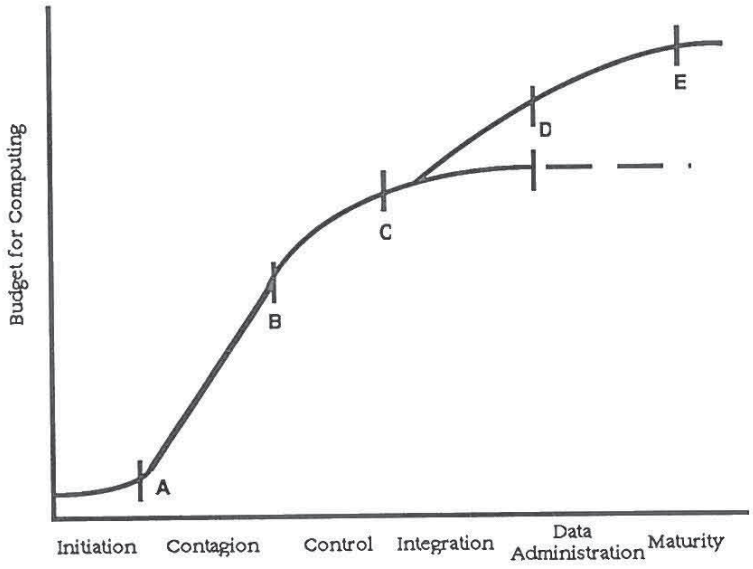
Nolanin vaiheteoria julkaistiin ensimmäisen kerran noin 15 vuotta sitten [18]. Ensimmäisissä tutkimuksissaan Nolan esitti vaiheteorian kasvuhypoteesinä tietojenkäsittelytoiminnnan kehittymisestä organisaatioissa. Myöhemmin hän tarkensi teoriaansa useaan otteeseen ja totesi samalla sen olevan empiirisesti vahvistettu (grounded) [20]. Johtopäätöksenä tutkimuksistaan Nolanin toteaakin vaiheteoriansa esittävän hyväksytyn näkemyksen siitä, kuinka organisaatioiden tietojärjestelmät *muuttuvat* ja myös kuinka niiden *tulisi muuttua*.

Tarkemmin tarkasteltaessa huomaamme, että Nolanin vaiheteoria ei muodosta mitään yhtenäistä teoriaa, vaan se koostuu sarjasta toisiaan täydentäviä selityksiä (malleja), jotka Nolan yhteistyössä usean tutkijan kanssa kehitti Harvardissa noin kymmenen vuoden aikana vuosien 1969-1979 välillä [18, 19, 20, 4]. Mallia täydennettiin tuona aikana useasti sitä vastaan esitetyn kritiikin ja saatujen kokemusten nojalla. Kuitenkin eräät mallin komponentit, kuten esimerkiksi S-muotoinen tietojenkäsittelyn kustannusten kasvukäyrä, ovat säilyneet muuttumattomina kaikissa teorian versioissa. Tarkoitukseni ei ole kuitenkaan seuraavassa tarkastella Nolanin teorian historiallista kehitystä – siitä on saatavilla erinomaisia esityksiä toisaalla – vaan kohdistaa huomio nimenomaan tapaan, jolla Nolan yritti teoriassaan analysoida tietokoneistumisen edellytysten, organisaatiomuotojen ja soveltamiskohteiden välisiä riippuvuuksia.

Nolanin teoria erottaa selvästi edellä tarkastellut kolme tietokoneistumisen komponenttia. Hänen lähtökohtanaan ovat *tietokoneistumisen edellytykset*, sillä teoria yrittää selittää miksi tietojenkäsittelytoiminnan kustannukset (edellytykset) noudattavat S-muotoista käyrää (kuva 1), kun kustannukset esitetään ajan funktiona. Tämän havainnon Nolan teki alunperin tutkittuaan kolmen yrityksen atk-osastojen budjettien kehitystä. Tämän jälkeen Nolan oletti, että *kustannuksia voidaan käyttää tunnuslukuna kuvattaessa runsasta ja vaihtelevaa joukkoa tietojenkäsittelyn muutoksia kuvaavia muuttujia kuten tekniset tekijät,*

toimialamuutokset, yritysten myyntiluvut, organisatorinen strategia, johtamiskäytännöt ja tietokoneiden käyttötavat.

Kustannuskäyrässä (kuva 1) esiintyvät käännepisteet A, B ja C jakavat tietojenkäsittelytoiminnan neljään erilliseen vaiheeseen. Tämä johti Nolanin tekemään toisen keskeisen vaiheteorian hypoteesin: *käännepisteet kustannuskäyrässä edustavat laadullisia siirtymäkohtia atk-toiminnan kasvussa.* Täten kullekin pisteiden origo-A, A-B, B-C, C- erottamalle tietojenkäsittelyn vaiheelle on ominaista tietty tietokoneistumisen muoto (organisatoriset ratkaisut ja tietokoneistumisen johtaminen) ja kohde (sovellustyypit ja lähestymistapa). Jos käännämme tarkastelumme takaperoiseksi paljastuu Nolanin vaiheteorian logiikka kirkkaana: pääaktiviteetit tietokoneistumisen muodoissa (suunnittelu, johtaminen, organisointitapa) ja tietokoneistumisen kohteet (sovellusjoukko) ovat laadultaan eroavat ja selvästi havaittavissa kussakin päävaiheessa. Kukin vaihe vastaa tiettyä stabiilia tilaa organisaation tietokoneistumisen kehityksessä, ja se voidaan rajata budjettikäyrän avulla. Kukin budjettikäyrän piste ja käyrän muoto on tulosta johdon reaktioista ympäristö- ja teknisiin muutoksiin.



Kuva 1. Budjettikäyrä ja Nolanin vaiheteoria.

Myöhemmin Nolan tarkensi teoriaansa siten, että kustannuskäyrän muutokset olivat pääosin seurausta tietotekniikan kehittymisestä [4]. Hän myös korosti, että syvät laadulliset muutokset teknisessä perustassa loivat pohjan uusien S-käyrien kehittymiselle (kuva 1: D-E). Täten S-muotoinen käyrä osoitti yleisesti teknisen muutoksen (syy) ja organisaation johdon reaktioiden (seuraus) välisen suhteen.

Nolanin vaiheteorian vaiheet voidaan lyhyesti kuvata seuraavasti:

Aloitus:

(Origo-A) (Initiation) Tietokoneiden käyttöönotto tyydyttää perustarpeet; hidas käytön kasvu; tietokoneistumisen ongelmien ilmaantuminen; johdon vähäinen kiinnostus ongelmiin; minimaalinen suunnittelutaso.

Laajentuminen:

(A-B) (Contagion) Käytön nopea kasvu, koska johto sitoutuu tekniikkaan ja käyttäjien odotukset kasvavat; johto pyrkii kontrolloimaan kasvua; keskittyminen alkaa; suunnittelu lisääntyy.

Kontrolli:

(B-C) (Control) Johdon kustannuskontrolli; suunnittelun priorisointi; tietojenkäsittelytoiminto keskitetään; atk-johdon asema organisaatiossa tunnustetaan; standardointi; veloitusjärjestelmät, jotka hillitsevät kasvua.

Integrointi/Kypsyys:

(C-E) (Integration/Maturity) Kontrollien uudelleenarviointi ja kokeilun lisääminen; suunnittelu laajalti hyväksytty; käyttäjien tietoisuus kasvaa; käyttö ja soveltaminen rationaalimpaa (kustannustietoisuus); sovelluskehityksen hajautus; keskitys/hajautus johdetaan liikestrategiasta; kustannusten hidas kasvu.

Viimeisessä artikkelissaan [20] Nolan täydensi teoriaansa toteamalla, että S-muotoinen käyrä osoittaa organisaatioiden yleistä oppimiskäyrää uuden tekniikan omaksumisessa: siten kustannus-akseli kuvaa tekniikan tunkeutumisen/omaksumisen astetta ja aika-akseli tietotekniikan kehitystä ja kustannuskäyrän muoto mallintaa johdon reaktioita uuden tekniikan leviämiseen. Organisaatio voi olla eri teknologioiden osalta myös eri vaiheissa oppimiskäyrää. Kokemukset toimistoautomaation tai vaikkapa

elektronisen postin osalta tuntuvat tukevan tätä väitettä — hyvin harvoin yritykset ovat kyenneet nojaamaan aiempaan atk-kokemukseen ottaessaan näitä ratkaisuja käyttöönsä (ks. [16]). Tämän vuoksi viimeisissä kirjoituksissaan Nolan tulkitsi malliaan normatiivisesti: S-käyrä osoittaa kuinka johdon kussakin tilanteessa tulee johtaa ja hallita uuden teknologian hajautumista organisaatioon.

Koska Nolanin teoria on tunnettu – onhan se intuitiivisesti puhutteleva – ja Nolan itse on väittänyt teoriansa olevan yleinen ja empiirisesti validoitu malli tietojenkäsittelytoiminnan kehityksestä organisaatioissa, on teorian pitävyyttä ja selitysvoimaa koeteltu ehkä enemmän kuin mitään muuta teoriaa tietojenkäsittelyn piirissä. Kuitenkin Nolanin oma anti teoriansa testaamisessa oli hyvin kyseenalainen — esimerkiksi budjettitiedot perustuivat hänen omiin konsultointikokemuksiinsa eikä niitä koskaan julkaistu. Hän ei myöskään myöhemmin vaivautunut perustelemaan väitteitään todellisilla empiirisillä aineistoilla. Siten ei ole ihme, että tehdyt empiiriset selvitykset ovat pala palalta osoittaneet teorian validisuuden ja selitysvoiman kyseenalaisuuden (ks. [8, 2]):

(1) Kustannuskäyrää ei voida käyttää mittarina aggregoimaan laajaa joukkoa muuttujia, ja yritysten atk-budjetit eivät kehity S-käyrän muotoisesti vaan lineaarisesti.
(2) Vaiheet eivät tosiasiassa ole erillisiä toisistaan siten, että ne kuvaisivat organisaation erilaista reagointitapaa tekniikkaan.
(3) Tietohallinto (kypsyys) ei kuvaa niinkään organisaation teknistä kypsyyttä kuin sen taloudellista kykyä investoida tietotekniikkaan.
(4) Organisaatiot, jotka soveltavat ns. "kypsiä" atk-politiikkoja kamppailevat suurimpien ongelmien parissa. Täten "kypsyys" on empiirisesti hyvin kyseenalainen termi.
(5) Teknologian omaksumista eivät määrittele ainoastaan tarjonnan muutokset (uudet nopeammat laatikot), vaan myös kysynnän vaihtelut, jotka johtuvat osin sattumanvaraisista tekijöistä (yritykset yli-investoivat tietotekniikkaan verotus yms. syistä) ja osin laskentatarpeen todellisesta laajenemisesta.
(6) Organisaatioiden ei ole helppoa sopeutua edes "kypsään" tekniikkaan, eivätkä muutokset tietojenkäsittelyn "rutiinissa" ole välttämättä

aina menestyksellisiä. Monet epäonnistumistarinat vahvistavat karvaalla tavalla tämän.

(7) Malli väittää organisaatioiden noudattavan jatkuvan teknisen muutoksen strategiaa. Kuitenkin käytännössä organisaatiot omaksuvat tekniikkaa sykleittäin ja pääosin vain tarkentavat olemassaolevaa teknistä perustaa.

Myöskään Suomessa tehdyt selvitykset [24] eivät vahvista Nolanin vaiheteorian paikkaansapitävyyttä — korkeintaan sitä voidaan käyttää referenssinä vertailtaessa eri organisaatiden keskimääräistä kehitystä.

Nolanin vaiheteorian empiirinen karilleajo täydentyy, jos tarkastelemme teorian eräitä keskeisiä oletuksia, joiden pätevyys on vähintäänkin kyseenalainen.

(1) Teoria on ns. *finaalisuus*teoria. Organisaation tietojenkäsittely kehittyy tiettyjen vaiheiden kautta kohti kehityksen korkeinta vaihetta, jonka Nolan suurta omintakeisuuta osoittaen nimitti kypsyydeksi. Mitään teoreettista perustelua tämän kaltaisen postulaatin sisällyttämiseen malliinsa ei Nolan esitä (ks. [8]).

(2) Teoria pätee *universaalisti*. Kaikki organisaatiot kaikkina aikoina noudattavat tätä mallia. On mielenkiintoista, että samaan aikaan, kun organisaatioteoriassa alettiin 70-luvulla avoimesti keskustella ns. kontingenssiteorioiden merkityksestä organisaatiotutkimuksessa, esittää Nolan erään organisaation keskeisen alueen kehityksestä täysin universaalin teorian.

(3) Teoria on *staattinen*. Nolan olettaa, että tietojenkäsittelyä tuottavat yksiköt, samoin kuin tietojenkäsittelypalveluja käyttävät yksiköt säilyvät tietojenkäsittelyn eri vaiheissa muuttumattomina — ainoastaan tietojenkäsittelyn volyymi kasvaa. Siten ei esimerkiksi tietojenkäsittelyn kohteiden kehitys vaikuta tietokoneistumisen muotoihin. Vain täten voi tulla ymmärretyksi Nolanin huomion kiinnittäminen vain atk-osaston budjetin kasvuun (vaikka se kattaa vain pienen osan organisaation tietojenkäsittelyn kokonaiskustannuksista, ks. [23]).

(4) Teoria on *yksisuuntainen*. Teoria olettaa, että tietotekniikan muutokset (syy) vaikuttavat yksisuuntaisesti tietojenkäsittelyn kohteisiin ja tietokoneistumisen muotoihin. Sensijaan esimerkiksi tietokoneistumisen muodoilla ei ole juurikaan vaikutusta tietokoneistumisen

kohteisiin ja sitä kautta tietokonelaitteiden kysyntään (tietokoneistumisen edellytykset).

(5) Teoria on *yksiulotteinen*. Nolan tarkastelee tietojenkäsittelytoiminnan muutosta vain teknisen muutoksen aikaansaamana. Sensijaan eivät esimerkiksi organisaatiomuutokset, organisaatiossa käyty kamppailu vaihtoehtoisista resursseista, yrityksen kilpailuasema tai tietojenkäsittelytoiminnon oma oppiminen vaikuta juuri lainkaan tietojenkäsittelytoiminnon muotoihin tai kohteisiin. Vielä vähemmän teoria selittää sitä *miksi* organisaatioiden ylipäänsä tarvitsee omaksua uutta tekniikkaa ja mikä niitä työntää oppimiskäyrällä ylöspäin? Teoria on malliesimerkki teknologisesta determinismistä: tekniikka on ulkoinen voima, joka kehittyy omien lakiensa mukaisesti luonnonvoimaisesti ja pakottaa organisaatiot tiettyyn käyttäytymiskaavaan. Nolania lukiessa lukija ei saa ylipäänsä vastausta kysymykseen miksi organisaatiot ovat valmiit haaskaamaan niukkoja resurssejaan tietojenkäsittelyyn. Hyvät veloitusmenettelyt tai referointi parempaan käyttäjien kustannustietoisuuteen, joilla päätösten rationaalisuutta organisaatioissa valvotaan, eivät selvästikään ole riittäviä selityksiä — ne ovat kuitenkin ainoat, joita Nolan lukijalleen tarjoaa.

Yhteenvetona voimme todeta, ettei Nolaninkaan vaiheteoria ole empiirisesti ja teoreettisesti kovin vakuuttava — eikä sen avulla voida ymmärtää tai selittää tietojenkäsittelytoiminnan kasvua organisaatioissa. Tämä ei kuitenkaan tarkoita, että Nolanin teoria olisi arvoton. Huonokin malli (joka auttaa meitä edes vähän) on parempi kuin ei mitään mallia. Erityisesti tietojenkäsittelytoiminnon johtamisen ja suunnittelun apuvälineenä Nolanin vaiheteorian kaltaisella yksinkertaisella ja havainnollisella mallilla on paikkansa.

6. Taloudelliset, organisatoriset ja poliittiset mallit

Kuten tähänastinen tarkastelumme osoittaa, ovat tietomme tietokoneistumisen ja organisaatioiden suhteista vähintäänkin puuttelliset. Valtaosa kehitetyistä malleista on yksinkertaisia kuvailevia periodisointeja. Merkittävin kehitysmalli – Nolanin vaiheteoria – on liian yksinkertainen ja

rajoitettu ja empiirisesti kyseenalainen. Tutkimus ei ole kyennyt tuottamaan selityksiä, joissa tietokoneistumisen muotoja, edellytyksiä ja kohteita ja niiden välisiä riippuvuuksia tarkasteltaisiin tasapainoisesti ja kokonaisvaltaisesti. Ilmiselvästi Nolanin vaiheteorian kaltaiset yksinkertaiset syy-seurausmallit eivät riitä, vaan tarvitaan dynaamisia tasapainomalleja, joissa eri komponenttien muutokset ovat keskenään riippuvia ja eri komponenttien aiheuttamat muutokset tietojenkäsittelyn tasapainotilassa johtavat uuteen tasapainotilaan (vaiheeseen). Täten jatkossa on paneuduttava tarkemmin *tietojenkäsittelytoiminnan taloudelliseen perustaan*, koska ainoastaan sitä kautta voidaan selittää muutokset tietojenkäsittelyn kokonaisekologiassa (tasapainossa). Toinen tutkimuksen kohde on kehittää parempia teorioita, joilla voidaan tapauskohtaisesti analysoida ja ymmärtää tietojenkäsittelytoiminnan kehitys jossakin organisatorisessa ympäristössä.

Eräs lupaava ja kehittelyn arvoinen lähestymistapa on *transaktiokustannusteoria* ([27], ks. myös [26]). Teoria on kehittynyt erityisesti 70- ja 80-luvun aikana ja sillä etsitään vastausta kysymykseen taloudellisesti optimaalisesta organisointitavasta. Sensijaan, että tietojenkäsittelyä tarkastellaan yksinomaan teknisenä muutosprosessina organisaatioissa kuten Nolan ehdottaa, transaktiokustannusteoriassa tarkastellaan *tietojenkäsittelypalvelujen* vaihdantaan liittyviä kustannuksia ja palvelujen optimaalisia hallintamekanismeja. Tutkimuksen kohteena ovat tällöin palvelujen käsite ja vaihtelut (muutos), niiden määrittely ja tuottaminen eri osapuolten kesken ja palvelujen vaihtoehtoiset hallintamekanismit ja niiden taloudelliset seuraukset. Transaktiokustannusteorian etu on siinä, että käyttäjien niinkuin tuottajienkin kannalta eivät tietokoneet ole itseisarvo: ne ovat vain keino tuottaa tietojenkäsittelypalveluja ja niiden avulla vaikutetaan lähinnä palvelun kustannuksiin ja sisältöön.

Transaktiokustannusteorian mukaan tietojenkäsittelypalvelut voidaan luokitella palvelun kompleksisuuden (performance ambiguity) ja opportunismin mukaan. Palvelun kompleksisuudella tarkoitamme eri osapuolten vaikeutta mitata tai määritellä palvelu ja arvottaa sitä. Opportunismilla tarkoitamme eri osapuolten kykyä ja halua ajaa omia etujaan esimerkiksi hävittämällä tai piilottamalla palveluja koskevia tietoja. Lisäksi transaktiokustannusteoria väittää, että kullekin kompleksisuus/opportunismi arvolle

löytyy taloudellisin vaihdannan hallintamekanismi (so. organisaatiomuoto jossa tietojenkäsittelypalveluja tuotetaan).

Palvelun kompleksisuutta voidaan eritellä neljän piirteen avulla: investointispesifisyys, ainutkertaisuus, monimutkaisuus ja epävarmuus. Investointispesifisyys tarkoittaa, että investointiin liittyvät varat/resurssit ovat sovellettavissa vain hyvin suppealla alalla. Esimerkiksi, jos tietojärjestelmähanke edellyttää tietyn merkkisen tietokoneen hankkimista, on kyse investointispesifisyydestä. Jos hankkeessa voidaan käyttää lukuisaa joukkoa erilaisia tietokoneita (avoimuus) laskee investointispesifisyys.

Transaktion ainutkertaisuus tarkoittaa, että vaihdannan kohde on uusi tai ainutkertainen. Jos esimerkiksi laaditaan aivan uusi räätälöity tietojärjestelmä on kyse ainutkertaisesta transaktiosta, jolloin sen arvoa on eri osapuolten hyvin vaikea määritellä. Jos kyse on standardiohjelmiston hankinnasta, esim. kirjanpito-ohjelmistosta, jota transaktion osapuolet voivat verrata aiempiin vastaaviin transaktioihin, on ainutkertaisuus alhainen. Transaktion monimutkaisuus syntyy vaikeudesta määritellä tarkasti transaktion kohde. Tietojärjestelmien kyseessä ollessa tulee yleensä kyseeseen korkea kompleksisuus, koska tietojärjestelmän ominaisuudet ovat vaikeasti määriteltävät ja ne riippuvat monen eri tekijän yhteisvaikutuksesta (laitteisto, ohjelmisto, käyttäjät jne.). Epävarmuus aiheutuu vaikeudesta määritellä tarkasti vaihdannan kohteen arvo eri osapuolille mm. mittaus- tai ajoitusongelmien vuoksi. Tietojärjestelmien kohdalla transaktion epävarmuus on korkea, joka näkyy mm. korkeina epäonnistumisasteina, eskaloituvina kehityskustannuksina jne.

Edellä esitetyn valossa näyttää siltä, että tietojenkäsittelypalveluja karakterisoi korkea kompleksisuus. Niitä useimmiten luonnehtii myös korkea opportunismin aste, sillä palvelujen tuottamiseen/käyttämiseen osallistuvilla osapuolilla on usein alhainen tavoitekongruenssi ja korkea informaation epäsymmetria (toisen osapuolen on vaikea arvioida toisen osapuolen todellista panosta transaktiossa). Tällaisessa tilanteessa tietojenkäsittelytoiminta keskittyy transaktiokustannusteorian mukaan byrokraattiseksi hierarkiaksi (kuten myös Nolanin malli edellytti). Edellä esitetty tarkastelu pätee kuitenkin vain, jos kustannuskomponentit so. vaihdannan kohde ja sen tuottamistapa (kustannukset) säilyvät vakioina.

On kuitenkin kaksi seikkaa, jotka pitkällä aikavälillä muuttavat tilannetta (ja jota Nolanin teoria ei huomaa). Ensimmäinen liittyy *tietojenkäsittelypalvelujen kompleksisuuden laskuun* ja siten transaktiokustannusten laskuun palvelua kohti. Toinen liittyy tietojenkäsittelypalvelujen kokonaiskustannusrakenteen muutokseen siten, että kaikista kustannuksista *transaktiokustannukset muodostavat suhteellisesti kasvavan osan*. Tarkastelen seuraavassa lyhyesti kumpaakin trendiä ja niiden vaikutuksia tietojenkäsittelyn kehittymiseen.

Tietojenkäsittelypalvelujen kompleksisuuden lasku on seurausta pyrkimyksistä laskea palvelukohtaisia transaktiokustannuksia ja siten madaltaa tietokoneiden soveltamiskynnystä palvelukohtaisesti. Kyse on siten pitemmällä aikavälillä tapahtuvasta palveluja koskevien transaktioiden rationalisoinnista ja suoraviivaistamisesta. Vain tätä kautta voidaan kasvavalle tietokoneiden tuotannolle (ja kapasiteetille) löytää uusia markkinoita, koska palvelujen määrittelyyn liittyviin transaktiokustannuksiin eivät voi sitoutua kuin organisaatiot joilla on riittävästi resurssijoustoa (slack resources). Kompleksisuuden laskuun on päästy useilla eri keinoilla kuten investointispesifisyyden laskemisella (avoimet ja siirrettävät sovellukset, käyttäjäystävälliset sovellukset), ainutkertaisuuden vähentämisellä sovellusten standardoinnin kautta (avaimet käteen sovellukset, liitäntöjen standardointi), epävarmuuden vähentämisellä (määrittelymenetelmät, prototyyppien käyttö) jne. Pitemmällä aikavälillä tämä on merkinnyt entisen ainutkertaisen ja korkean transaktiokustannuksen aiheuttavan tietojenkäsittelypalvelun muuttumiseen toistuvaksi, standardoiduksi, suhteellisen hyvin määritellyksi ja alhaisemman investointispesifisyyden omaavaksi palveluksi. Samalla ovat muuttuneet taloudelliset edellytykset hallita palvelujen vaihdantaa. Byrokratian näkyvä käsi on muuttunut yhä enemmän markkinoiden tai kvasimarkkinoiden näkymättömäksi kädeksi. Tietojenkäsittelystä on kasvanut "business within business". Tämän seurauksena ovat tietojenkäsittelytoiminnon organisoinnin ja johtamisen edellytykset muuttuneet tavalla, jota Nolan ei kyennyt näkemään: tietojenkäsittelytoiminto erillisenä yksikkönä itseasiassa kuihtuu pois ellei muita vaikuttavia trendejä ole olemassa. Kypsyys Nolanin teoriassa muuttuisi pian vanhuudeksi, jollei jopa kuolemaksi. Tämän kaltaisen analyysin tuloksena

ympäröivä organisaatio säilyisi muuttumattomana, mutta tietojenkäsittelytoiminto erillisenä toimintona vähitellen sulautuisi siihen.

Kuitenkin tarkastelumme toinen trendi – *transaktiokustannusten kokonaissuhteen kasvu* – tuottaa lähes päinvastaisen lopputuloksen. Jos otamme tarkastelumme lähtökohdaksi sen, että tietojenkäsittelypalveluiden kokonaiskustannukset määräytyvät sekä palvelun hallinnasta ja määrittelystä aiheutuvista transaktiokustannuksista että palvelun tuottamisesta aiheutuvista tuotantokustannuksista, voimme helposti päätyä tähän johtopäätökseen. Tuotantokustannuksiin kuuluvat esimerkiksi laitteiden kustannukset ja/tai järjestelmien toteutuksesta (koodaus/installointi) aiheutuvat kustannukset. Jos järjestelmien toteutus/laitekustannuksia voidaan alentaa transaktiokustannusten vastaavasti kasvaessa voivat organisaatiot muuttaa tietojenkäsittelypalvelujensa tasoa, vaikka ne eivät kokonaisuudessaan investoisi juuri yhtään enempää tietojenkäsittelyynsä. Käytännössä tähän suuntaan näyttävät etenevän etupäässä vauraat yritykset, koska ne tietojenkäsittelyn suuren volyyminsä ansiosta kykenevät ensimmäisinä hankkimaan etuja tuotantokustannusten alenemisesta — seikka, joka on tullut esille Nolanin teorian empiirisissä selvityksissä. Tällöin tietojenkäsittelypalveluja koskevien transaktioden kompleksisuus kasvaa pitkällä tähäimellä jatkuvasti. Seurauksena tästä on, että tietojenkäsittelypalvelujen sisältö monimutkaistuu ja niiden taso kasvaa. Lyhyesti sanottuna sovellusten kompleksisuus kasvaa tavalla, jota karakterisoin edellä integroivan periodisoinnin yhteydessä.

Uudenlaisten palvelujen vaikutus organisaatioiden rakenteeseen ja toimintaan on kuitenkin usein radikaali (esim. ns. kilpailuedun tuottavat sovellukset) — ne parantavat organisaation toiminnan edellytyksiä ja sen kilpailukeinoja. Tämä taas puolestaan kiihdyttää organisaatoiden halua investoida tietotekniikkaan (Nolanin aloitus/kasvu -kausi). Yhä uusien tietojenkäsittelyn kohteiden valinnan tuloksena tietojenkäsittelypalvelujen kompleksisuus kasvaa — erityisesti palvelujen monimutkaisuus, ainutkertaisuus ja epävarmuus. Tämä monimutkaistuminen ei suinkaan vähennä – vaan päinvastoin korostaa – byrokraattisen transaktionhallintamekanismin asemaa tietojenkäsittelypalveluja määriteltäessä ja hallittaessa. Siten tämän trendin vallitessa ei myöskään erillisen tietojenkäsittelypalveluja määrittelevän ja valvovan yksikön asema heikkene, vaan päinvastoin korostuu.

Pitemmällä tähtäimellä tämä trendi näyttääkin vahvistavan Nolanin väitettä atk-osaston pysyvästä merkityksestä organisaatioissa — kuitenkin siten, että atk-osaston merkitys korostuu uusien sovelluskohteiden etsimisestä (so. aloitus- ja kasvukausien aikana), ei suinkaan palveluiden kypsyessä ja virtaviivastuessa.

Tällekin kehityssuunnalle löydämme todisteita, jos tarkastelemme joitakin atk-toiminnan viimeaikaisia muutoksia. Esimerkkinä tässä yhteydessä riittänee pyrkimys eriyttää varsinainen toteutus erikoistuneiden ohjelmistotalojen vastuulle, jolla tähdätään toteustuskustannusten alentamiseen. Tämä mahdollistaa palveluiden määrittelyyn käytettyjen transaktiokustannusten nousun siten, että enemmän aikaa ja rahaa pannaan sovellusten määrittelyyn ja kokonaisuunnitteluun.

Yhteenvetona voimme todeta, että transaktiokustannusten suhteellinen kasvu työntää organisaatioita kohti kasvavaa tietokoneistumista siinä määrin kuin toteutus- ja laitekustannusten suhteellinen osuus laskee ja uusien tietojenkäsittelypalveluiden merkitys organisaatioille paljastuu. Ensimmäinen trendi puolestaan rationalisoi ja virtaviivaistaa olemassa olevia palveluja ja muuttaa olennaisesti jo olemassaolevien palveluiden jakelukanavien suhteellista taloudellisuutta. On huomattava, että kaksi edellä mainittua trendiä ovat ristiriitaisia ja ne vaikuttavat samanaikaisesti tietojenkäsittelytoiminnan organisoinnin taloudellisuuteen. Tämä selittää mm. hyvinkin ristiriitaiset väitteet tietojenkäsittelytoiminnon tulevaisuudesta organisaatioissa: kun toiset soittavat kuolinkelloja näkevät toiset sen merkityksen vain kasvavan erityisesti strategisella tasolla.

Eri organisaatiot ovat eri trendien vaikutuksen suhteen hyvin erilaisessa asemassa riippuen mm. niiden strategisista valinnoista, kyvystä etsiä ja soveltaa tietotekniikkaa ja toimialan yleisestä rakenteesta. Siten kummankin trendin vaikutusta erillisen organisaation tasolla on tarkasteltava tapauskohtaisesti, eikä myöskään ole olemassa yleistä evoluutiomallia, joka pätisi kaikkien organisaatioiden tietojenkäsittelytoiminnan kehitykseen.

Tapauskohtaisissa tarkasteluissa korostuvat ratkaisevasti tietojenkäsittelyn palveluiden tuottamiseen/käyttöön osallistuvien osapuolten opportunismin ja rajoitetun rationaalisuuden vaikutus. Tämän vuoksi on käsittääkseni organisaatiokohtaisissa tarkasteluissa transaktiokustannusteorian

Superkone tulee - mikä muuttuu?

OLAVI NEVANLINNA, SUOMEN AKATEMIA

Kun edellisen kerran puhuin Porthaniassa, oli silloinkin muistaakseni Reino Kurki-Suonio yksi esitelmäsarjassa esiintynyt. Vajaat kaksikymmentä vuotta sitten kokoavana aiheena oli kybernetiikka, nyt tietojenkäsittelytiede. Silloin minua kiehtoi ajatus olla mukana ratkomassa aivojen salaisuutta. Nytkään en varsinaisesti esiinny matemaatikkona vaan jonkinlaisena tietojenkäsittelytieteen reuna-alueen edustajana. Ainakin minut on sijoitettu esiintymisvuorossa juuri "muiden tieteiden edustajien" edelle. Taustastani sen verran, että olen Teknillisen korkeakoulun teknillisen matematiikan linjan oppilaita, mutta toisin kuin Murkku, Shosta jne, jotka ovat Hassen kasvatteja, niin minun opettajinani olivat professori Laasonen, professori Lehti, professori Lokki jne. Aivofysiologian piiristä moottoripyörällään matematiikan pariin minut palautti nykyään Tampereen teknillisessä korkeakoulussa vaikuttava Armo Pohjavirta.

Jokaisen esitelmän kultaiseksi ohjeeksi voisi korottaa vaikkapa jäsentelyn:

1. Oman alan ylistys.
2. Olosuhteiden ruikutus.

Ajattelin poiketa tästä hieman. Voin piiloutua verhon taakse sanoen, etten itse asiassa paljoakaan puhu omimmasta alastani. Ylistelen olosuhteita Suomessa. Esitän pientä kritiikkiä aiheeseen liittyviä tieteenaloja kohtaan, ja lopuksi esittelen erästä tutkimusaihettamme, joka liittyy oikeastaan pikemminkin rinnakkaislaskentaan kuin varsinaisesti superkoneisiin.

1. Entäs jos

Olipa kerran kulttuurikoe, joka viisi minuuttia ennen loppuaan tuotti itselleen supertietokoneen. Siitä sanottiin, että sillä voi tehdä kokeita, vastata kysyksiin muotoa 'entäs jos'. Sillä olisi voinut pyrkiä rakentamaan tiedettä lopun tapahtumista, sen hidastamisesta. Näin ei tapahtunut. Sen sijaan tutkittiin, miten asiat olivat aikojen alussa, sekunnin biljoonasosien paikkeilla. Se oli tiedettä.

Tällä tasolla vastaus otsikon kysymykseen on siis valitettavan negatiivinen: ei muutosta. Tiedemaailma on siksi jähmeä, etten oikein jaksa uskoa ripeisiin painotusmuutoksiin. Hallitusten välillä, vaikkapa Euroopan mittakaavassa voidaan pystytellä – ja täytyykin – tutkimusohjelmia tulevaisuutemme laskemiseksi. Yliopistoissa tämä tulee näkymään lähinnä ja vain rahoituskanavana tai rahastuskohteena.

2. Reviirien merkitsemisestä

Tieteenalojen identiteetti ja sen varjeleminen selittää viime kädessä tiedeyhteisön edellä kuvaamani kykenemättömyyden nopeaan reagointiin.

Reviirien tai kauniimmin sanottuna identiteetin syntyminen on olennainen osa uuden tieteenalan rakentumista. Niitä tarvitaan rutiinien järjestämiseen. Aiemmissa esitelmissä on puututtu mm. matematiikan ja erikoisesti numeriikan sekä tietojenkäsittelytieteen reviirijakoon. Kurki-Suonio totesi mm. että Suomessa "uuden tieteenalan identiteetti pääsi kehittymään ilman liian vahvoja sidoksia numeeriseen laskentaan." Tuntuisi erikoiselta sivuuttaa asia kommenteitta.

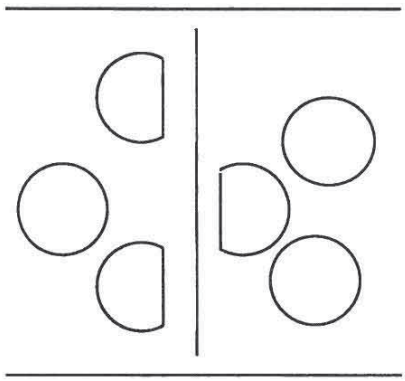
Oheisessa kuvassa 1 on kaksi naapurialan laitosta, ympyrät keskipisteessä olevien henkilöiden toimintaympäristöä. Reunan lähellä seistään selät vastakkain, katseet kohti laitosten keskuksia. Kun on ehdotettu, keskellä kahdeksankymmenlukua, että "Suomessa matemaatikot voivat hoitaa tämän numeriikan", niin se tarkoittaa ehdotusta raja-aidan vetämiseksi tietojenkäsittelytieteen ja matematiikan väliin eri yliopistoissa samalle

kohdalle. Koko maahan jää aidan viereen huonosti peittyvä kaistale. Kansanomaisesti sanottuna väärää politiikkaa.

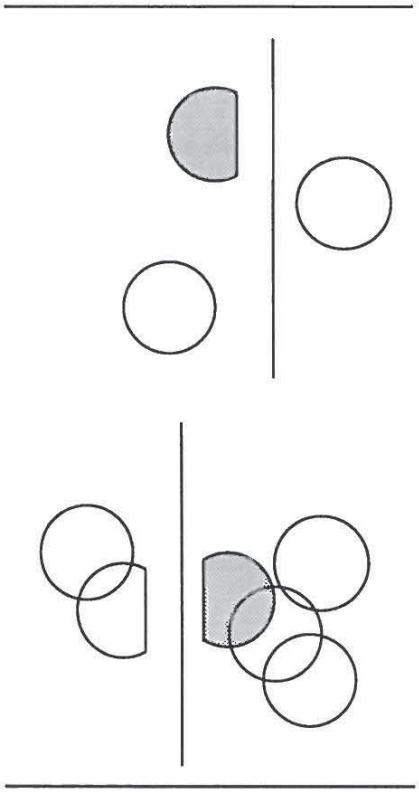
Kuvaan 2 olen merkinnyt kaksi eri yliopistoa, joissa raja-aidat sijaitsevat eri paikoissa. Samalla alalla työskentelevät eri suuntiin katselevat tutkijat tapaavat toisiaan kansallisissa alan katselmuksissa ja rikastuttavat toisiaan.

Nykyisin pääsen tästä osalliseksi vain ulkomaita matkatessani. Osa meistä suomalaisista matemaatikoista yrittää "hoitaa tämän numeriikan". Arvomaailmamme liittyy matematiikkaan. Suomessa tarvitaan tutkimusta numeerisen ja symbolisen laskennan alueella myös tietojenkäsittelytieteeseen liittyvistä lähtökohdista käsin. Tietojenkäsittelytiede on jo ajat sitten ohittanut vaiheen, jossa selkeän identiteetin puute uhkaisi sen tulevaisuutta.

Eri korkeakoulut voivat varsin perustellusti hieman leikitellä ja kilpailla erilaisilla ratkaisuillaan.



Kuva 1. Kaksi naapurialan laitosta.



Kuva 2. Kaksi eri yliopistoa.

3. Supertietokoneesta

Me suomalaiset olemme monesta asiasta ylpeitä — ja joskus syystäkin. Tieteellistekninen sanastomme on esimerkiksi varsin huolitellusti rakennettu. Superkone, superlaskenta, tai – kaikkein kamalimpana – superlaskentakulttuuri särähtävät. Super esiintyy tässä suurin piirtein samassa merkityksessä kuin superraskas sarja jossain voimaurheilussa. Tärkeintä ei ole itse painorajat. Sarja on se painavin.

Matemaatikkona tekee mieleni määritellä superkone, ja juuri tuohon tapaan se määrittyy: suurin ja kaunein tehokkaaseen laskentaan saatavissa oleva tietokone. Se on tutkimusväline. Tutkimusvälineillä on rajalliset markkinat ja niinpä suurimman ja kauneimman hinta pysyy reaaliarvoltaan vakiona, tai erittäin lievässä nousussa, suuruusluokan ollessa 10-20 miljoonaa dollaria. Näin on ollut jo noin neljännesvuosisadan ajan. Muiden laitteiden kohdallahan olemme tottuneet jatkuvaan halpenemiseen.

Edeltä seuraa, että mikä tahansa yksittäinen superkone menettää asemansa varsin pian, putoaa raskaaseen tai keskiraskaaseen sarjaan. Juuri tällä hetkellä Suomeen ostettu Cray X-MP EA/416 täyttää kaikki superkoneen määritelmän vaatimukset (tässä se lupaamani ylistys). Jokainen superkonetta tarvitseva yksittäinen tutkija osaa esittää vaihtoehtoisen ratkaisun, joka on kokonaisuudessaan halvempi ja joka sopii hänelle paremmin. Siksi superkoneen osto on vaikeaa ja jatkuvuus alituisessa vaarassa.

Jätän tahallani suoritusnopeutta kuvaavat teholuvut sivuun. Siksikin, että niissä voi sotkeutua, mutta erityisesti siksi että eivät ne sinällään ole tarpeen superkoneen luonteen ymmärtämiseksi. Uusin ja ajanmukaisin tutkimusväline on uusin ja ajanmukaisin. Ilman sitä meiltä puuttuisi sekä väline sinänsä että syy keskustella eri tieteenalojen yli sillä harjoitettavasta tutkimustoiminnasta.

Tässä kohdin vastaus otsikon kysymykseen kuuluu: pääsemme ainakin hetkeksi tutkimusvälineen osalta samalle lähtöviivalle muiden kanssa.

4. Tieteellinen laskenta

Supertietokoneella harjoitetaan tieteellistä laskentaa, uuden tieteellisesti merkittävän tiedon tuottamista laskien. Olen äskettäin, superkoneteemapäivien avajaisissa viime syyskuussa, pohtinut tieteellisen laskennan tilaa, menneisyyttä ja tulevaisuutta Suomessa. Esitys on luettavissa Arkhimedes-lehdestä [1], joten olen tässä siksi melko lyhytsanainen.

Keskeistä on taata ensinnä välineistön jatkuva uusiutuminen. Vain siten jää tilaa ja aikaa yhteisen tieteellisen laskennan kulttuurin kehittymiselle. Toisaalta jatkuva uusiutuminen edellyttää nykyistä parempaa kulttuuria. Päätin esitykseni seuraavasti: "iso mahtava elefantti on onnistuttu ujuttamaan lasikauppaan ilman, että oikeastaan mitään on mennyt rikki. Siinä se voi nyt nostella jalkojaan. Meidän kaikkien on yritettävä edelleen laajentaa sen liikkumatilaa, jotta se voisi hengittää rauhassa, leikkiä ja lisääntyä, onnistua ja epäonnistua ja tehdä virheitä. Ja samalla miettiä, onko tuo kaikki koskaan mahdollista lasikaupassa."

Lasikauppa tuossa viittaa ratkaisumallin puitteisiin: koneen monimutkaisista omistussuhteista viime kädessä johtuu, että tieteellisen laskennan kulttuurin kasvusta vastaava ympäristö ei toistaiseksi tule sijoittumaan opetusministeriön tontille. Tieteellisen laskennan palvelu on Valtion tietokonekeskuksessa majaansa pitävä yksikkö. Silti sitä tulee kehittää ja siltä on edellytettävä vastuunottoa. Konkreettinen sivukuvamuutos on sitä paitsi syntymässä: superkoneen myötä saamme "superlaskentakulttuurin" vaalimisesta ja kehittämisestä vastaavan tieteellisen johtajan.

Tieteellisen laskennan kulttuuri on Suomessa ymmärretty kapeahkosti ja teknisluonteisesti: on osattava käyttää konetta oikein, saatava siitä teho irti. Ehkä tässä on ollut siksi paljon puutteita, ettei paljoa huomiota ole jäänyt itse toimintaan: tieteentekotavan tieteellisyyden kehittymiseen. Sen mitä tieteelliseksi mielletään tulee olla olennaisella tavalla samaa kaikilla niillä tieteenaloilla, joilla tieteellistä laskentaa harrastetaan. Tältä osin yhteinen kulttuuri tuskin voi kehittyä ilman yhteistä foorumia.

Minulle on yritetty ehdottaa pyrkimistä vielä "enempään": perustetaan tieteellisen laskennan professuureja joihinkin korkeakouluihin. En

oikein osaa vastata tähän muuta kuin, että onpas outo ajatus. Ei meillä ole kokeellisen tieteenkään professuuria, jolle laskennallisen tieteen professuuri olisi rinnasteinen. Laskennallinen fysiikka on vastakin ennen kaikkea fysiikkaa. Tarvitaan yhteistä foorumia, joka kokoaa eri tieteiden piirissä tehtävän työn yhteen. Tarvitaan kansallista "superkomputointi-instituuttia" kuten niitä muualla nimitetään. Suoraan opetusministeriön alle. Sellaista instituuttia tarvitaan vielä pitkään. Siihen asti kunnes laskennallisen tieteen pelisäännöt ovat yhtä itsestään selvät kuin sanokaamme kokeellisen tieteen ja havaintomateriaalin tilastollisen käsittelyn.

5. Kirjanpito, rahastus ja koneen käyttö

Edellä mainitussa avajaistilaisuudessa puhui myös ministeri Taxell. Kun tässä sarjassa aiemmin Pertti Järvinen on lukenut kirjeitään, niin minäkin luen tässä yhden.

Hra Ministeri Taxell

Eilen 14.9.1988 juhlimme Superkoneteemapäivän merkeissä Suomen siirtymistä superkoneaikakauteen. Minäkin toivon, että sanastomme vaikkapa humanistien avulta tältä osin siistiytyisi. Kirjeeni ei kuitenkaan koske kielenkäyttöä vaan vielä vakavampaa asiaa.

Mainitsitte avajaissanoissanne suunnitelmista jakaa superkoneiden kustannuksien peittämiseksi tarkoitetut varat osittain tai kokonaan korkeakouluille. Tämä on sinänsä johdonmukaista päätösvallan hajauttamista, vastuun antamista ja sen kantamisen edellyttämistä.

Superkoneen kohdalla sen tieteellisestä käytöstä tulee vastata asiantuntevan tieteellisen johtokunnan, joka jakaa koneaikaa projekteille. Kiintiönsä puitteissa tutkija tai tutkimusryhmä voi käyttää superkonetta kuin tieteellistä kirjastoa. Suomessa on jo pitkään ollut käytössä tämän tapainen toimintapolitiikka, eikä meidän ole sitä syytä hävetä, päinvastoin. Muualla, kuten Yhdysvaltain kansallisissa superlaskentakeskuksissa, on äskettäin siirrytty tai ollaan siirtymässä kirjastomaiseen "ilmaiseen" superkoneen käyttöön.

Ajatus astua meillä päinvastaiseen suuntaan juuri nyt, kun suuri kansallinen tutkimusväline tekee tuloaan,

vaikuttaa erehdykseltä. Kansallisen resurssin käytön kriteerien tulee perustua tutkimuksen tieteelliseen odotusarvoon. Siten tutkimushankkeen sijaintipaikalla eikä ao. korkeakoulun omalla atk-politiikalla tulisi olla tässä mitään tekemistä.

Tieteellistä laskentaa harjoittava yhteisö Suomessa on ohut, korkeatasoinen ja tieteenaloittain viipaloitunut. Superkonehankkeen yhteydessä yhteisöltä kaikuneet viestit ovat siten saattaneet olla epäselviä ja keskenään ristiriitaisia. Tämän nyt esiin tulleen ajatuksen kohdalla uskon yhteisön kuitenkin ajattelevan varsin samansuuntaisesti.

Omassa esitelmässäni pyrin valottamaan suomalaisen tieteellisen laskennan kulttuurin kehittymisen muita edellytyksiä. Kirjoitin sen viime viikolla, joten en koskettele siinä käyttöpolitiikkaa lainkaan. Vaikka esitykseni sävy onkin hieman kepeähkö, rohkenen liittää sen tähän kirjeeseni eräänlaisena tilannearviona.

Otaniemessä kunnioittaen 15.9.1988

Olavi Nevanlinna
Suomen Akatemian tutkijaprofessori

Toisin kuin Järvinen minä en saanut kirjeeseeni vastausta.

Sitten viime syyskuun asia ei ole selkiintynyt, vaan mennyt jos mahdollista aina vaan oudommin solmuun. Kun ministeriö toimi oleellisesti toisin, kuin mitä sen asettama työryhmä oli viisaaksi katsonut, politiikan muutos jäi osalta korkeakoulumaailmaa huomaamatta. Pitemmällä aikavälillä ministeriön politiikka asettaa korkeakoulut viime kädessä tarkastelemaan supertietokonetta minä tahansa vaihtoehtoisena "atk-laitteena" ja silittää siten edellä kuvaamiani kulttuurin kehityksen edellytyksiä vastakarvaan.

Superkoneen käyttöaikaa tulisi jakaa tieteellisen toimikunnan toimesta projekteille, ei korkeakouluille. Projektien kirjanpidossa aika tulisi kirjata kuluihin selvinä markkoina, mutta projektin tai edes korkeakoulun kannalta niillä ei olisi lainkaan vaihtoarvoa, markkoja ei voisi käyttää mihinkään muuhun (jolloin itse rahaliikennettäkään ei tietenkään tarvittaisi).

6. Supertietokone sovellusten kimpussa

Supertietokoneella on kaupallisesti merkittäviä sovellusalueita, kuten elokuvateollisuus , hävittäjälentokoneet tai supertietokoneet itse. Suomen mittakaavassa nuo alueet eivät ole keskeisiä. Teollisuuden varovaisuus heittäytyä superkoneen maksajaksi taisi osaltaan viivyttää koneen ostopäätöstä.

Suomen teollisuudelle on tärkeätä, että osaamista löytyy. Superkoneen tärkein merkitys teollisuudelle onkin laskennan osaamisen kehittyminen.

7. Rinnakkaisuutta kohti

Ajateltaessa superkoneiden arkkitehtuurin todennäköistä kehityssuuntaa helpointa on ennustaa sen kehittyvän rinnakkaislaskennan suuntaan. Tällä hetkellä rinnakkaissuorittimet ovat tehokkaita erityistehtävissä, mutta superkoneen tasolla vaadittava yleiskäyttöisyys asettaa vielä vakavia rajoituksia. Toistaiseksi superkoneiden eri "sukupolvet" on ollut helppo kuvata piiriteknologian edistysaskeleiden avulla ilman, että koneen toimintatapoihin on tehty mullistavia muutoksia. Vaikka nykyisinkin superkoneissa on jo monissa useita suorittimia, niin varsinaisista rinnakkaisuuteen perustuvista superkoneista voisimme puhua esimerkiksi silloin, kun pienehkökin koneelle lähetetty työ tulee ositetuksi monelle suorittimelle.

Tämä ositus on erittäin vaikea tehtävä. Äskettäin oli suurimmassa päivälehdessämme kirjoitus Suomessa rakennetusta rinnakkaiskoneesta. Kirjoittajat totesivat jotenkin siihen tapaan, että insinöörit eivät vielä ole keksineet automaattia, joka suorittaa tehtävien osituksen. Selvästikään se ei kuulunut heidän tontillaan olleisiin ongelmiin. Flyygelin pakkaus pieniin helposti siirrettäviin kenkälaatikoihin tapahtuu muualla.

Englannin kielestä sananmukaisesti kääntäen rinnakkaislaskennassa puhutaan kuvausongelmasta, jolla tarkoitetaan annetun laskennallisen tehtävän saattamista annetun arkkitehtuurin mukaisesti toteutettavaksi siten että jokin tehokkuuskriteeri maksimoituu. Viime vuonna ilmestyneessä

kokoomakirjassa [2] jaotellaan kuvausongelman lähestymistavat kuuteen eri tyyppiin. Tietojenkäsittelytieteilijät puhuvat usein algoritmien rinnakkaistumisesta, rinnakkaisuuden havaitsemisesta yms. Minä puhun tässä nyt lähinnä tehtävien, en algoritmien osituksesta.

Ison laskennallisen tehtävän ositus melko usein ja luonnollisella tavalla vastaa alkuperäisessä tehtävässä avaruus-aika alueen jakoa osiin ja osatehtävien kytkemistä, liimaamista alueiden reunojen yli yhteen. Yksi perusvaikeus voitaisiin ehkä yksinkertaistaa seuraavasti. Kun tavanomaisessa laskennassa ei matkan varrella saa tulla nollalla jakoa vastaan, niin tässä osatehtävien tulee kaikkien koko ajan pysyä hyvin asetettuina. Nollalla jakamisen kiertäminen ja sen havaitseminen ovat paljon helpompia asioita kuin matriisien ei-singulaarisuuden kiertäminen tai havaitseminen.

On olennaista ymmärtää, että rinnakkaiseen laskentaan liittyy usein algoritmien koko idean uusiminen. Esimerkiksi suuressa osassa luontoa pätee tosiasia, että lähivaikutukset ovat tärkeämpiä kuin kaukovaikutukset. Tehokkaasti rinnakkaistettu laskenta suorittaa lähivaikutusten laskennan samalla suorittimella ja pyrkii minimoimaan kaukovaikutusten merkityksen kuljettamalla suorittimien välillä mahdollisimman vähän tietoa ja sitäkin tiivistetyssä muodossa. Mitä tuolla sitten tarkoittanenkin, niin lienee selvää, että pelkästä yksinkertaisesta automaatista ei ole kysymys silloin, kun tuo kaikki on kunnossa. Siihen on tarvittu monenlaista kädenlyöntiä erilaisten reviirirajojen yli.

8. Suurten dynaamisten systeemien rinnakkaistaminen

Omassa tutkimusryhmässä olemme viitisen vuotta tutkineet suurten ajasta riippuvien systeemien tilan laskemista rinnakkaisesti. Menettelyn juuret ovat erittäin suurten integroitujen piirien toiminnan simuloinnoissa ja ajoittuvat vuosikymmenen alkuun. Matemaatikko kun olen, niin lähestymistapamme on sarja idealisointeja ja sen jälkeen yksityiskohtaista tarkastelua mahdollisuuksien ja mahdottomuuksien välillä.

Perinteisesti suurten systeemien käyttäytyminen on laskettu annetusta tilasta pienen hetken kuluttua olevaan tilaan koko systeemin osalta alusta loppuun saakka valmiiksi ennen kuin laskenta siirretään

Ihmisestä kone vai koneesta ihminen?

JUHANI PIETARINEN, TURUN YLIOPISTO

1. Ihminen – kaiken keskus?

Tieteen kehitys osoittaa kouriintuntuvasti, mikä ihminen pohjimmiltaan on: parantumattomasti itsekeskeinen narsisti.

Kun kreikkalaiset luonnonfilosofit 600- ja 500-luvuilla eaa hylkäsivät myyttiset selitykset maailman alkuperästä ja toiminnasta, he samalla valoivat ihmiskeskeisen ajattelun perustan. Thales väitti, että vesi on kaiken alkuperä. Toisin sanoen: emme tarvitse jumalia, jotta ymmärtäisimme todellisuutta. Ihmisjärki riittää. Niin ihmisen älystä alkoi kehittyä maailmankaikkeuden keskus.

Toisaalta, tavallaan paradoksaalisesti, tiede on sitten historiansa kuluessa useasti asettanut ihmiskeskeisyyden kyseenalaiseksi. Ajatellaan vaikkapa kuvaa, jota tähtitiede Kopernikuksesta lähtien on rakentanut maailmankaikkeudesta. Ihmiskunnan asemahan on siinä häviävän pieni, lähes olematon. Darwin puolestaan palautti ihmisen maapallon muiden elollisten olentojen yhteisöön — mitään poikkeuksellista, ainutlaatuista, muuhun luomakuntaan nähden korkeampaa tai arvokkaampaa alkuperää meillä ei ole. Ja Freud horjutti pahasti ihmisen uskoa rationaalisiin kykyihinsä paljastamalla, että me elämmekin mitä suurimmassa määrin salaperäisten tiedostamattomien ja alitajuisten voimien mahdin alaisina.

Monipuolinen tiedemies Michael Polanyi on huomauttanutkin, että jos suhtautuisimme todellisuuteen tieteen piirtämän objektiivisen kuvan

mukaisesti, ihmiskeskeiselle egoismille ei jäisi sijaa. Mutta, Polanyi sanoo, "kukaan – tiedemiehet mukaanlukien – ei suhtaudu todellisuuteen tuolla tavalla, huolimatta objektiivisuuden näennäisestä tunnustamisesta. Eikä sen pitäisi meitä hämmästyttää. Sillä ihmisolentoina meidän tulee väistämättä nähdä universumi itsessämme sijaitsevasta keskuksesta käsin." Tuolla keskuksella Polanyi tarkoittaa ihmismieltä, ajattelua, järkeä. Meidän on siis eräänlainen pakko pitää ihmismieltä maailmankaikkeuden älyllisenä keskuksena.

Niinhän teemmekin. Vaikka universumin mittasuhteissa maapallo kutistuukin häviäväksi hiukkaseksi, pidämme itseämme ainoina älyllisinä olentoina. Darwinin aiheuttama järkytys väheni, kun huomattiin, että ihminen voi käsittää itsensä evoluution korkeimmaksi saavutukseksi — siis pysyä luomakunnan kruunuksi. Myös kauhukuvat irrationaalisten voimien vallasta ihmisessä ovat haalistuneet kun nuo voimat on opittu tuntemaan ja hallitsemaan.

Nyt on edessä uusi uhka ihmiskeskeisyydelle: koneet, jotka näyttävät pystyvän älyllisiin suorituksiin. Eikö älyllisyys olekaan enää vain ihmiselle kuuluva ominaisuus? Mihin sitten meidän ainutlaatuisuutemme perustuu?

Mielenkiintoinen uusi tulokas on ilmestynyt kulttuuriimme. Miten siihen tulisi suhtautua?

2. Onko tietokone elävä?

Sherry Turkle, Massachusettsin teknillisen tutkimuslaitoksen (MIT:n) sosiologian professori, tutki 4 - 14-vuotiaiden lasten suhtautumista tietokoneisiin (tietokoneleluihin ja -peleihin).

Lasta askarruttaa kysymys, onko tietokone elävä. Turkle tutki, millaisten kriteerien perusteella lapset luokittelevat erilaisia kohteita eläviksi tai elottomiksi. Hän jakoi kriteerit kolmeen luokkaan, fysikaalisiin ('se liikkuu', 'se kiehuu', 'sen latva huojuu'), biologisiin ('se hengittää', 'se syntyy', 'se kasvaa', 'se syö') ja psykologisiin ('se ajattelee', 'se ymmärtää', 'se aikoo', 'se haluaa').

Tulos: Perinteisten kohteiden (kuten aurinko, pilvi, puu, koira, puhelin, televisio) osalta 68 % lapsista käytti fysikaalisia tai biologisia kriteerejä luokitellessaan ne eläviksi, ja 11 % psykologisia kriteerejä. Tietokonetyyppisten kohteiden kohdalla tilanne oli päinvastainen — 67 % käytti psykologisia kriteerejä, 17 % fysikaalisia. Psykologisten kriteerien käyttö kasvoi siirryttäessä alle 8-vuotiaista vanhempiin.

Lapsille siis tietokoneilla on selvästi mieli. Lisäksi juuri tuo mieli tekee ne heidän mielestään eläviksi — se, etteivät ne liiku eivätkä syö ja kasva, ei ole tärkeää. Vielä sittenkin kun lapsi perinteisten kohteiden osalta käyttää vain biologisia elävyyden kriteerejä, siis pitää kissaa elävänä muttei pilveä, hän luokittelee tietokoneen psykologisin kriteerein eläväksi. Lasten maailmaan on ilmestynyt uuden tyyppinen elävä olento, jolla on mieli kuten ihmiselläkin.

Turklen tutkimuksissa useille lapsille heräsi välittömästi toinen kysymys: miten ihminen eroaa koneesta? Alle 8-vuotiaista useimmat näkivät eron fysikaalisessa, aineellisessa, puolessa tai alkuperässä ("koneessa on johtoja, ihmisessä verta ja luuta"; "koneet on tehty tehtaissa, ihmiset on Jumala tehnyt"). Mutta yli 8-vuotiaista lähes 90 % sanoi eroksi sen, ettei koneilla ole tunteita kuten ihmisillä. Äly ja tunteet erotetaan jyrkästi. Koneiden mieli ei ole yhtä monipuolinen kuin ihmismieli.

Kysymys tietokoneiden elävyydestä menettää tietysti kiinnostavuuttaan lapsen iän mukana, samoin laitteiden yleistyessä. Mutta tietokoneen ja ihmismielen samankaltaisuuden ja eroavuuden ongelma jää silti askarruttamaan. Turklen huomioiden mukaan tästä ongelmasta ovat erityisen kiinnostuneita lapset ja nuoret, jotka ovat jatkuvasti tekemisissä tietokoneiden kanssa, tuntevat ne hyvin.

3. Tietokone – lähimmäiseni?

Uusien koneiden kiinnostavin piirre on epäilemättä niiden määrätynlainen älykkyys. Ne näyttävät ymmärtävän ja muistavan asioita. Ne päättelevät, ratkaisevat ongelmia, oppivat kehittämään itselleen uusia toimintoja, luovat jazz-improvisointeja, pelaavat shakkia ja tekevät ikäänkuin harkinnanvaraisia päätöksiä.

Otan pari esimerkkiä, kuinka läheiseksi tietokoneen ja ihmismielen analogia voidaan kokea.

Sherry Turkle totesi tutkimuksissaan, että tietokoneiden ohjelmointi saattaa auttaa nuoria ihmisiä selvittämään henkilökohtaisia ongelmiaan. 13-vuotias Deborah oli arka ja aloitekyvytön tyttö. Hän ei luottanut itseensä. Hän oppi yksinkertaisen ohjelman, ja sen avulla hän sai tietokoneen luomaan sellaisia kuvioita ja hahmoja kuin hän itse halusi. Hän oppi näin rakentamaan pienen maailman, joka toimi hänen haluamallaan tavalla. Samalla tyttö tunnisti keskeisen ongelmansa: En hallitse itseäni. Mutta minä hallitsen koneen, osaan luoda sille sääntöjä. Miksi en tekisi niin itselleni? Voinhan luoda itsellenikin sääntöjä ja harjoitella niitä!

Monet nuoret, jotka oppivat ohjelmoimaan tietokonetta, ajattelevat luontevasti olevansa itsekin ohjelmoituja ja edelleen ohjelmoitavissa. Miksen itse ohjelmoi itseäni? Näin syntyy itsehallinnan, autonomian kokemus: minä itse pystyn määräämään, mitä olen ja mitä teen.

1970-luvun alussa Joseph Weizenbaum laati maineikkaan ELIZA-ohjelman, joka sai tietokoneen toimimaan psykoterapeuttina. ELIZA toteutti yksinkertaista peilausmenettelyä eli toisti myötäelävällä (empaattisella) tavalla, tietenkin mekaanisesti, mitä potilas sanoi. Potilas: "Minulla on ongelmia tyttöystäväni suhteen." ELIZA: "Miksi kerrot minulle, että sinulla on ongelmia tyttöystäväsi suhteen?", tai: "Ikävä kuulla, että sinulla on ongelmia tyttöystäväsi suhteen." Potilas: "Olen siitä hyvin masentunut." ELIZA: "Ymmärrän, että olet siitä hyvin masentunut.", tai: "Miksi kerrot minulle, että olet siitä hyvin masentunut?".

Hyvin monet Weizenbaumin työtoverit ja oppilaat kokivat tilanteen niin, että ELIZA todella kykenee ymmärtämään heidän huoliaan. Ohjelma toimi terapeuttisesti. Monet halusivat säilyttää vaikutelman, että ELIZA kykenee vastaamaan heille, vielä sittenkin kun ohjelman juju paljastui.

Tuollaiset hyvin inhimillisiksi koetut piirteet tietokoneiden toiminnassa tekevät seurustelusta uusien laitteiden kanssa kiinnostavan ja merkityksellisen. 'Peeseestä' tulee helposti hyvä kaveri, jopa lähimmäinen. Sherry Turkle huomauttaakin, kuinka ajallemme tunnusomaisin neuroosi on 'läheisyyden pelko' — varotaan antautumista läheisiin suhteisiin toisen ihmisen kanssa, koska sellainen tuo mukanaan vaikeasti hallittavia emotionaalisia ongelmia. Mutta yksinkin on paha olla. Tietokone tarjoaa

silloin lähes ihanteellisen ratkaisun: siinä ystävä, jonka kanssa voi seurustella vailla vaikeiden tunnesiteiden ja ristiriitojen riskiä.

4. Tietokoneesta ihminen?

Uusilla laitteilla on monia ihmismäisiä ominaisuuksia yksinkertaisesti sen vuoksi, että ne rakennetaan jäljittelemään ihmisen kykyjä ja suorituksia. Ne seuraavat kielellisiä ohjeita, tallettavat informaatiota muistiin, suorittavat matemaattisia ja loogisia operaatioita, vastaavat kysymyksiin ja tekevät ilmoituksia omista sisäisistä tiloistaan — paljastavat sisintään.

(Toisaalta tietokoneet rakennetaan tarkoituksella niin, että ne nimenomaan eivät jäljittele ihmismieltä. Mihin tarvittaisiin mikroa, jolla on muistikatkoja tai joka tekee päättelyvirheitä tai jonka hermot pettävät tiukassa tilanteessa?)

Mutta saadaanko koneesta ihminen?

Eniten on pohdittu kysymystä, ajatteleeko tietokone. Jos looginen ja matemaattinen päättely on ajattelua, silloin tietokone epäilemättä ajattelee – ja on älykäs. Mutta varsinaisesti ihmiselle tyypillistä ajattelua on vasta *tietoinen* ajattelu, siis sellainen missä ajattelija itse on selvillä siitä, *että* hän ajattelee ja *mitä* hän ajattelee. Onko tietokoneen ajattelu tuolla tavalla 'refleksiivistä'? Voiko koneella olla tietoisuutta?

Kysymykseen on vaikea antaa yksioikoista vastausta, sillä emme tiedä, miten tietoisuus tulisi ymmärtää. Tietoisuus on aivojen tuote. Ihmisaivot pystyvät hyvin monimutkaisella tavalla valvomaan ja ohjaamaan omia toimintojaan, ja juuri tuohon kykyyn tietoisuus perustuu. Mutta sama on mahdollista myös automaateille, sillä omien toimintojen ohjailuun riittää mekaaninen *feedback*-järjestelmä. Voi sanoa, että mitä paremmin tietoisuuden luonnetta opitaan tuntemaan, sitä tarkemmin sitä pystytään jäljittelemään. Mielenkiintoinen tosiasia on, että tietoisuuden tutkimus käyttää hyväkseen tekoälytutkimuksen kehittämiä malleja — siis tietokoneiden toiminnan teoriaa. On perusteltua odottaa, että ihmisaivojen toimintoja yhä tarkemmin jäljittelevät koneet pystyvät jäljittelemään erehdyttävällä tavalla myös ihmisen tietoista ajattelua ja tietoisia kokemuksia. Silloin ihmisen ja

koneen ratkaiseva ero ei löytyisikään tietoisen refleksiivisen ajattelun alueelta.

Löytyisikö ero ehkä tunteiden alueelta? Sherry Turklen tutkimuksissahan useimmat lapset iän karttuessa päätyivät tunnekriteeriin. Samoin valtaosa aikuisista, jotka pitivät ihmistä selvästi tietokoneesta poikkeavana, löysivät ratkaisevan eron ihmisen kyvystä kokea tunteita. Koneet katsottiin tunteettomiksi mekaanisiksi laitteiksi.

Tuollainen tulos selittyy pitkälle sillä, ettei koneita ole rakennettu ilmaisemaan tunteita. Sitä ei ole pidetty tarpeellisena — tunteita ilmaisevista koneista ei ilmeisesti koidu mitään käsin kosketeltavaa hyötyä. Mutta tietokoneet voidaan tietysti rakentaa ja ohjelmoida jäljittelemään tunteita siinä missä ajatteluakin. ELIZA jäljitteli tavallaan empatiaa. Raivonpurkauksia ja riemunkiljahduksia kone voisi päästellä vallan helposti. Tietysti jälleen voi kysyä, *kokeeko* kone todella, siis tietoisesti, tunteita. Mutta edessä on täsmälleen sama tietoisuuden ongelma kuin ajattelun kohdalla, eikä siitä, kuten totesin, voi sanoa mitään ehdottoman varmaa puoleen eikä toiseen.

Olisiko ehkä rakennettava myös tuntevia koneita? Jos nimittäin sellainen ajattelu yleistyy, että älyllisyys ei ole erityinen ihmiselle tunnusomainen ominaisuus vaan tunteet, ihmiskuva uhkaa pelottavalla tavalla vääristyä. Äly ja tunteet erotetaan jyrkästi ja aidosti inhimillinen löydetään epä-älyllisten, 'puhtaiden' tunteiden alueelta. Jotenkin panee aavistelemaan, että tuohon suuntaan ollaan menossa, että ajattelu ja älyllisyys jätetään lisääntyvässä määrin koneille ja ihminen etsii tunteisiin vetoavaa viihdettä, mystiikkaa tai vastaavia elämysten lähteitä.

Ihmisen ja koneen eroa voi tietenkin etsiä muualtakin kuin ajattelun ja tunteiden alueelta, siis muualta kuin mielen toiminnoista. Eihän koneilla ole samanlaista ruumista kuin meillä. On melko ilmeistä, ettei esimerkiksi rakkautta henkilökohtaiseen tietokoneeseen voi ilmentää psykofyysisesti yhtä monipuolisella tavalla kuin henkilökohtaiseen rakastettuun, ei ainakaan ennenkuin koneiden materiaalia, muotoilua ja liikuntajärjestelmiä kehitetään reippaasti nykyisestä. — Jätän kuitenkin somaattisen puolen ongelmat tässä sivuun.

5. Ihmisestä tietokone?

Ihmismieli, ihmisen psyykkinen toiminta, on tavattoman rikas ja monivivahteinen. Vaikka sen jäljitteleminen olisi periaatteessa mahdollista, varmaa kuitenkin on, etteivät tietokoneet koskaan todellisuudessa tule saavuttamaan samanlaista monipuolisuutta. Yleiseen käyttöön tulevat koneet jäävät rakenteeltaan suhteellisen yksinkertaisiksi ja noudattavat melko rajoitettua ohjelmoinnin logiikkaa.

Mitä siitä seuraa? Alkavatko ihmiset jäljitellä koneita?

Jos perehtyy hiukankin tietokoneiden toimintaan, huomaa, kuinka helposti samojen periaatteiden soveltaminen käy ihmiseen. Tietokonetermien käyttö ihmisen toiminnan kuvaamisessa yleistyykin kaiken aikaa paitsi tieteessä myös arkielämässä. Ihminen alkaa näyttäytyä meille tietokoneena. Samalla monet käsitteet, jotka alunperin kuvasivat ihmismielen toimintaa ja joilla sitten ruvettiin kuvaamaan tietokoneen toimintoja, saavat ihmiseen uudelleen sovellettuina uuden 'teknisen' merkityksen. Sellaisia käsitteitä ovat 'muisti' ja 'ajattelu'. Jos ajattelulla aletaan ymmärtää tietokoneen ajattelua, kiista ihmisen ja koneen ajattelun erosta menettää merkityksensä. Ajattelun jäljittely on silloin ajattelua, niinkuin monet tekoälyn tutkijat väittävätkin. Uudella tekniikalla on siis myös puhtaasti semanttisia vaikutuksia.

Mutta haluaisin tässä kiinnittää huomiota toisenlaisiin vaikutuksiin. Lapset jäljittelevat herkästi tietokoneita, matkivat niiden ääniä ja eläytyvät niihin muutenkin, tavallaan elävät niissä. Sherry Turkle totesi, että monilla lapsilla, jotka pelasivat paljon tietokonepelejä, oli vaikeuksia sopeutua normaaliin arkimaailmaan. Pelaaja oppii hallitsemaan pelimaailman tapahtumat, pääsee niihin 'sisälle', mutta samalla tavallisen elämän asiat ristiriitoineen käyvät ahdistavalla tavalla monimutkaisiksi ja vaikeasti käsiteltäviksi, kun arkielämä ei noudatakaan pelien logiikkaa.

Eräässä amerikkalaisessa koulussa tehty tutkimus kertoo, kuinka tyypillinen menestyvä ohjelmoija käsittää tehtävänsä niin, etta asiat pitää saada toimimaan selvän suunnitelman mukaan. Maailma on hänelle jotain, mitä pitää voida ohjailla, mikä pitää saada hallintaan. Sellaiset lapset ja nuoret

suhtautuivat toisiin ihmisiin jännittyneesti ja kireästi elleivät päässeet heitä selvästi ohjailemaan ja *hallitsemaan*.

Toisin sanoen tietokoneen toiminta luo mallin, jonka mukaisesti suhtaudutaan arkielämään. Silloin syntyy tavallisesti ristiriitoja ja jännityksiä, kun ihmisten kanssakäyminen ei sujukaan ohjelmoinnin periaatteiden mukaisesti. Syntyy paineita ohjata kanssakäymisen kuvioita mallin mukaisiin uomiin. Helpoimmin se käy tietysti, kun ohjelmoidaan säännöllisiä prosesseja — liikennettä, työtehtävien kulkua, urheilukilpailuja jne. Vaikeinta on saattaa vapaamuotoinen, spontaani yhteiselo ohjelmoitavan mallin mukaiseksi. Mutta paine siihen suuntaan kasvaa, kun asioiden ohjailu kaikkialla lisääntyy.

6. Menettääkö ihminen arvokkuutensa?

Vastaukseksi otsikon kysymykseon "Ihmisesta kone vai koneesta ihminen?" saadaan: sekä että. Näin on, jos tarkoituksena on ennustaa tulevaa kehitystä. Koneita kehitetään jäljittelemään yhä monipuolisemmin ihmistä, ja toisaalta ihmisen toimintoja ymmärretään ja ohjaillaan yhä yleisemmin koneen tarjoamien mallien mukaisesti.

Mihin jää silloin ihmisen ainutlaatuisuus? Onko lopultakin pakko luopua puhtaasti ihmiskeskeisestä ajattelusta?

Niin ei käy. Ihminen voi pitää kiinni omasta erityisluonteestaan periaatteessa kahdella tavalla.

Ensimmäinen on se, että aina löydetään kuitenkin jokin ratkaiseva ihmisen ja koneen toimintojen ero. Tärkeimpiä mahdollisuuksia olen jo käsitellyt. Koneen älykkyys voidaan kieltää sen perusteella, ettei kone ymmärrä mitä se tekee; sen äly ei ole tietoista kuten ihmisellä. Tai voidaan ajatella, ettei kone pysty kokemaan tunteita niinkuin ihminen. Tai osoitetaan, ettei kone pysty yhtä hienovaraisiin ja monimutkaisiin toimintoihin kuin ihmisen ruumis ja mieli. Yritetään siis etsiä sellainen ihmisen 'ydin', jota ei ole mahdollista edes periaatteessa esittää ohjelmoitavana informaationkäsittelynä. Ihmisen elämällä olisi siten aivan toisenlainen merkityssisaltö hänelle itselleen kuin koneen toiminnalla koneelle.

Jos tuollaiset selitykset hyväksytään, tietokoneiden tuleminen ei merkitse uhkaa ihmisyydelle. Voimme säilyttää ainutlaatuisuutemme ns. 'tietoyhteiskunnassakin'. Paradoksaalista kyllä, tämä ratkaisu saattaa johtaa siihen, että ihmisen ytimestä, tosi ihmisyydestä, tulee uudessa uljaassa konekulttuurissa selittämätön, mystinen, älyllisesti luoksepääsemätön asia. Tekniikan kehitys lisäisi mystiikkaa.

Toinen ratkaisu on, että hyväksytään ihmisen ja koneen periaatteellinen samanlaisuus, siis se, ettei meissä ole sen erityisempää eikä selittämättömämpää kuin monimutkaisissa koneissakaan. Tämäkään näkemys ei välttämättä hävitä ihmisen arvokkuutta, mikäli hän säilyttää aloitteen tekijän arvovallan itsellään — siis kokee olevansa koneiden keksijä ja niiden käytön hallitsija. Käy samalla tavalla kuin uuden ajan alun suuren tieteellisen murroksen kohdalla. Ihmisen omanarvon tunto kasvaa, kun hän huomaa kehitäneensä uudenlaisen, tehokkaan välineen maailman ja itsensä tutkimiseksi ja informaation saattamiseksi palvelemaan hänen omia päämääriään.

On kuitenkin muistettava, että vaikka olisikin vaikea loytää ihmisen ja koneen toimintojen periaatteellista eroa, aina on mahdollista vedota ihmismielen ja ihmisaivojen 'aitouteen'. Meillä on takanamme ihmeellinen biologinen evoluutio, jota mikään kone ei voi toistaa. Ihminen on luonnollinen olento, kone keinotekoinen. Tämän perusteella vain ihmisen ajattelu ja tunteet ovat aitoja, ja kone täydellisimmilläänkin pystyy pelkkään jäljittelyyn. Oma ongelmansa tietysti on, annetaanko tulevassa konekulttuurissa aitoudelle mitään erityistä arvoa.

Koneiden ihmismäisen luonteen tunnustaminen tuskin hävittää ihmisen omaa arvokkuutta. Paljon pahempikin uhka on olemassa: se, että käytämme uusia välineitä kelvottomasti — luonnon tuhoamiseen, perinteen hävittämiseen, elämän tärkeiden arvojen kadottamiseen, pahimmassa tapauksessa ihmis- tai peräti koko luomakunnan hävittämiseen. Tieteen ja tekniikan kehitys kasaa meille moraalista vastuuta. Emme voi siirtää sitä koneille. Siksi ihmisen arvokkuuden kannalta on oleellista, ettei tapahdu sellaista ihmiskäsityksen muutosta, joka tuon vastuullisuuden poistaisi.

Kirjallisuus

1. Seymour Papert, *Lapset, tietokoneet, ajattelemisen taito*. Kirjayhtymä, Helsinki, 1985.
2. Juhani Pietarinen, Tietokone, kaltaiseni? *Tiede 2000*, 1/1986.
3. Juhani Pietarinen, Tiede ja ihmiskäsityksen muutos. *Teknorama-näyttelykirja*. Insinöörilehdet Oy, Tampere 1987.
4. Sherry Turkle, The Second Self. *Computers and the Human Spirit*. Granada, London, 1984.

PROFESSORI JUHANI PIETARINEN
TURUN YLIOPISTO
FILOSOFIAN LAITOS
20500 TURKU

Kirjoittajien esittelyt

KURKI-SUONIO, REINO (1937)

filosofian tohtori, Tampereen teknillisen korkeakoulun tietojenkäsittelytekniikan professori, aikaisemmin Tampereen yliopiston tietojenkäsittelyopin professori. Toiminut vierailevana tutkijana ja professorina Carnegie-Mellon- ja Stanfordin yliopistoissa. Julkaissut tutkimuksia ja oppikirjoja tietojenkäsittelytieteestä. Nykyisenä tutkimuskohteena hajautettujen ja reaktiivisten ohjelmien spesifiointi.

TIENARI, MARTTI (1935)

filosofian tohtori, Helsingin yliopiston tietojenkäsittelyopin professori. Toiminut vuodesta 1967 alkaen Helsingin yliopiston Tietojenkäsittelyopin laitoksen esimiehenä, sekä vuosina 1971-78 Matemaattis-luonnontieteellisen osaston dekaanina. Julkaisuja matematiikan, numeerisen analyysin, ohjelmointikielten kääntäjien sekä hajautettujen järjestelmien ja tietokoneverkkojen aloilta. Tietojenkäsittelyalan maailmanjärjestön International Federation for Information Processing hallituksen jäsen vuodesta 1989 alkaen.

MANNILA, HEIKKI (1960)

filosofian tohtori, Helsingin yliopiston tietojenkäsittelyopin professori. Julkaissut tutkielmia tietojenkäsittelyopin alalta, erityisesti tiedonhallinnasta ja algoritmianalyysistä.

UKKONEN, ESKO (1950)

filosofian tohtori, Helsingin yliopiston tietojenkäsittelyopin professori, Teknillisen korkeakoulun dosentti. Julkaissut artikkeleita tietojenkäsittelyopista, erityisesti algoritmitutkimuksen alalta.

JÄRVINEN, PERTTI (1940)

filosofian tohtori, Tampereen yliopiston tietojenkäsittelyopin professori. Toiminut valtion luonnontieteellisessä toimikunnassa yli 9 vuotta, Tampereen yliopiston taloudellis-hallinnollisen tiedekunnan dekaanina 4 kertaa, Suomen edustajana IFIPin TC9:ssä (Computers and Society) vuodesta 1985. Julkaissut tutkimuksia ja oppikirjoja systeeminsuunnittelun ja atk:n yhteiskunnallisten vaikutusten alalta.

LYYTINEN, KALLE (1953)

kauppatieteiden tohtori, Jyväskylän yliopiston tietojenkäsittelyopin professori. Vieraileva professori Kööpenhaminan kauppakorkeakoulussa vuosina 1989-1990. TEKES FINSOFT Osa-alue I (Tieto- ja toimistojärjestelmät) johtoryhmän jäsen. TEKES FINSOFT -tutkimusprojektin Systeemityöympäristöt (SYTI) vastuullinen johtaja. IFIP WG 8.:n (Information Systems and Organizations) sihteeri. Toiminut tutkijana Tukholman yliopistossa ja London School of Economicsissa. Julkaissut yli 30 kansainvälistä julkaisua tietojärjestelmien suunnittelun, teorian, toteutuksen ja johtamisen alueelta. Konsultoinut sekä yksityisissä yrityksissä että julkisissa yksiköissä.

NEVANLINNA, OLAVI (1948)

tekniikan tohtori, Suomen Akatemian tutkijaprofessori, Teknillisen korkeakoulun matematiikan professori, Rolf Nevanlinna -instituutin esimies. Julkaissut tutkielmia numeerisen analyysin alalta.

PIETARINEN, JUHANI (1938)

valtiotieteen tohtori, Turun yliopiston käytännöllisen filosofian professori, Helsingin yliopiston käytännöllisen filosofian dosentti. Julkaissut tutkielmia tieteenfilosofian, yhteiskuntafilosofian ja etiikan aloilta.