INFORMAATIOJÄRJESTELMIEN (INFORMAATIOSYSTEEMIEN) KUVAAMINEN TIETOKONEJÄRJESTELMIEN SUUNNITTELUSSA

Vesa Surakka

16.9.2004

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

TIIVISTELMÄ

Tässä työssä käsitellään informaatiojärjestelmien kuvaamista tietokonejärjestelmien suunnittelussa. Työn tavoitteena on tuoda esiin käsitteellisen mallintamisen merkitys ja tarkoitus tietojenkäsittelyssä. Keskeisenä tutkimusongelmana on liiketoiminnan ja tietokonejärjestelmien yhteensovittaminen. Tutkimusstrategiana on ollut lähestyä aihetta kirjallisuudessa esitettyjä määritelmiä ja luonnehdintoja analysoiden. Tämä tarkoittaa käytännössä sitä, että kirjallisuuden asiasisältö on pyritty kokoamaan helpommin ymmärrettävään muotoon.

SISÄLLYSLUETTELO

1 JOHDANTO

Informaatiojärjestelmien kuvaamista on tutkittu jo vuosia ennen tietokoneita. Biologiassa, yhteiskuntatieteissä ja organisaatio-opeissa kehittyi 1900-vuosisadalla lähestymistapoja, joille oli tyypillistä systeemi-käsitteen käyttö ja sitä hyödyntävän metodologian kehittäminen. Norbert Wienerin kybernetiikka ja Claude Shannonin informaatioteoria ovat tunnetuimpia esimerkkejä. Nämä edellä mainitut lähestymistavat pohjautuivat antiikin ajan filosofiaan. (Benson & al. 1986)

Vasta 1960-luvulla systeemi-käsitteen käytöstä oli käytännöllistä hyötyä. Silloin systeemi-käsitettä ryhdyttiin ensin käyttämään yrityssuunnittelussa ja liikkeenjohdossa. Vähän myöhemmin systeemi-käsitettä ryhdyttiin käyttämään tietojenkäsittelyopissa (tietojenkäsittelytieteessä). Luonnossa itsessään ei ole systeemejä vaan ainoastaan ihmisen ajattelussa. Kun mietitään, mitä todellisuudessa tapahtuu, niin systeemit kertovat tapahtumat kuvien avulla. Näiden kuvien myötä tapahtumat on helpompi ymmärtää. Alkio voidaan erottaa toisesta alkiosta systeemiksi rajaamalla. Yksittäinen alkio voi olla esimerkiksi käsite. Kaaos voidaan käsitteellistää systeemiajattelun avulla. (Benson & al. 1986)

Osiin jakoon ja rajaukseen liittyy keskeisenä tarkastelukulman käsite. Systeemiajattelun päätarkastelukulmiksi on valittu merkkiopin eli semiotiikan mukaiset tiedon päätarkastelukulmat (Benson & al. 1986):

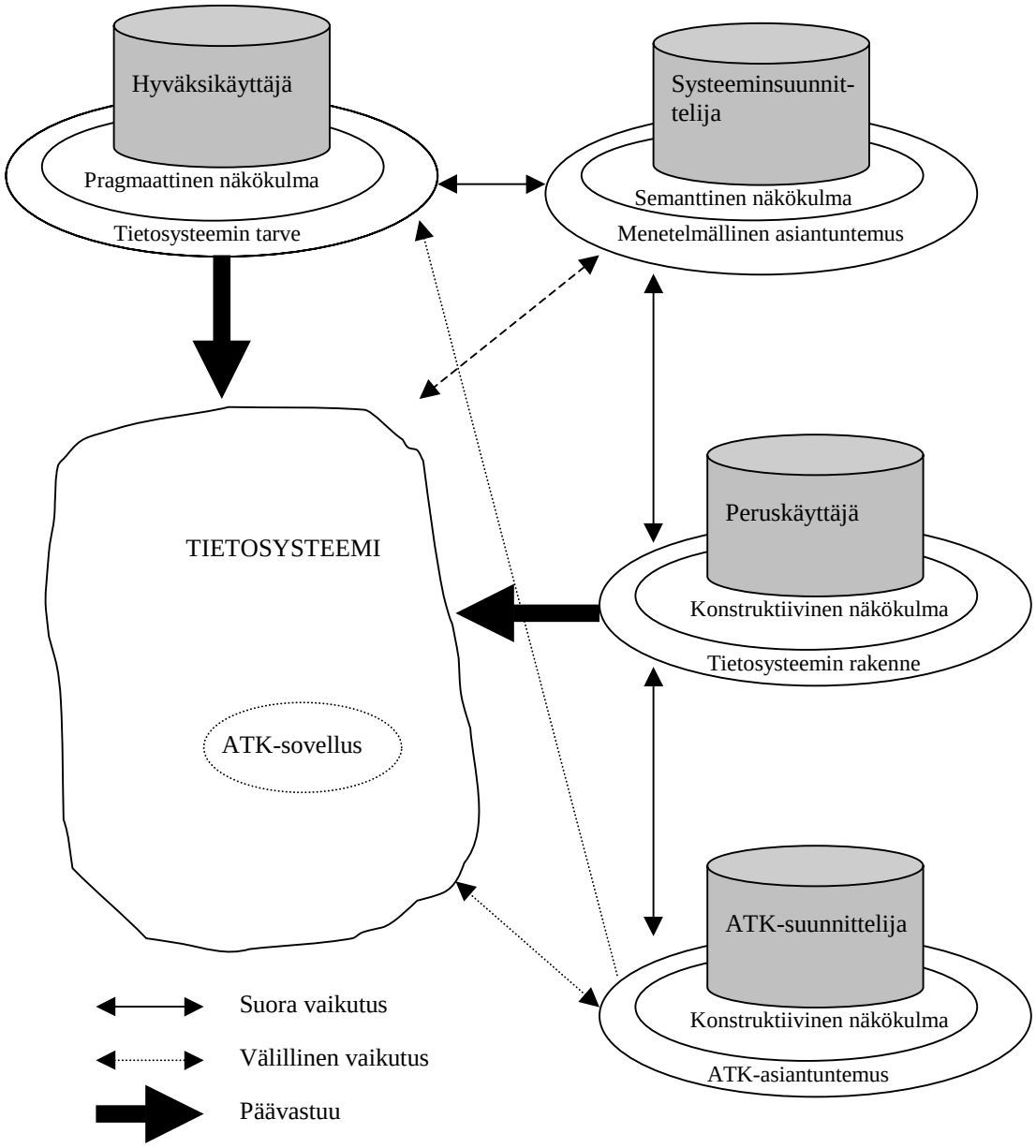
- rakenteellinen (eli konstruktiivinen)
- sisällöllinen (eli semanttinen)
- hyödyntävä (eli pragmaattinen)

Kun systeemiä tarkastellaan ikään kuin sisältä käsin puhutaan rakenteellisesta tarkastelukulmasta. Tarkoituksena on määritellä alkiot ja selvittää niiden väliset vaikutussuhteet. Tällöin saadaan tieto, miten systeemi yksityiskohtaisesti toimii. (Benson & al. 1986)

Kun kohdistetaan huomio systeemin ominaisuuksiin puhutaan sisällöllisestä tarkastelukulmasta. Tällöin systeemin tuottamat suoritteet ja niiden aikaansaamiseksi tarvittavat syötteet ovat tutkimuksen kohteena. Lisäksi tutkimuksen kohteena ovat säännöt, jotka toteuttamalla syötteistä saadaan suoritteet. (Benson & al. 1986)

Kun systeemiä tarkastellaan osana sen ympäristöä puhutaan hyödyntävästä tarkastelukulmasta. Tällöin kiinnostus kohdistuu systeemin vaikutuksiin. Tämä tarkoittaa, että selvitetään mihin, miten ja milloin systeemi vaikuttaa ympäristöönsä. Vastaavasti selvitetään mihin, miten ja milloin ympäristö vaikuttaa systeemiin. (Benson & al. 1986)

Kun tietosysteemin ja sen atk-sovelluksen valmistamistehtävä annetaan peruskäyttäjälle ja hyväksikäyttäjälle yhteisesti atk-suunnittelijan ja systeeminsuunnittelijan jäädessä taka-alalle neuvovan opastajan asemaan saadaan alla kuvattu tietojenkäsittelyn kehittämisen kommunikaatiomalli (kuva 1.1). (Benson & al. 1986)



Kuva 1.1: Tietojenkäsittelyn kehittämisen kommunikaatiomalli (Benson & al. 1986).

“Kuvassa 1.1 hyväksikäyttäjällä tarkoitetaan pääasiassa hyödyntävästä tarkastelukulmasta katsovaa käyttäjää, joka tietosysteemin tietojen käyttäjänä omaa tietoja myös sen sisällöstä. Kuvassa 1.1 peruskäyttäjällä tarkoitetaan henkilöä, joka päivittäisessä työssään joutuu tekemisiin tietosysteemin kanssa. Hän valmistelee tietosysteemin syöttötietoja, ylläpitää sen tietovarastoja, huolehtii sen tietojen tuottamisesta sovitun aikataulun mukaan tai viimeistellen ja jalostaen tuotettuja tietoja erilaisia omia tai muiden käyttötarkoituksia varten.” (Benson & al. 1986)

“Systeemi on eräänlainen suljettu rajapinta, jonka sisäpuolta kutsutaan varsinaiseksi systeemiksi ja ulkopuolta ympäristöksi”, Benson & al. (1986) esittävät. Tietokonejärjestelmistä puhuttaessa on tärkeätä ymmärtää, että järjestelmä ja systeemi ovat synonyymejä eli tarkoittavat samaa asiaa. Tällöin voidaan puhua atk-järjestelmästä tai atk-systeemistä. Atk-järjestelmän muodostavat laitteisto, ohjelmisto, käyttöliittymä sekä tietovarasto. Tietovarasto taas tarkoittaa yhtä nimettyä varastoitua kokonaisuutta. Tämä kokonaisuus on teknisesti yleensä joko tiedosto tai tietokanta. Mikäli kyseessä on tietokanta voidaan puhua *tietokantajärjestelmästä*. Tietokantajärjestelmä koostuu kahdesta osasta, *tietokannasta* (database) ja *tietokannan hallintajärjestelmästä* (DBMS). Tietokanta on joukko toisiinsa liittyvää talletettua tietoa. Tietokannan hallintajärjestelmä on ohjelmisto, jolla tietokantaa ylläpidetään ja jonka avulla käyttäjät pääsevät käsittelemään tietokantaa. Tietokannan hallintajärjestelmä prosessoi käyttäjiltä ja ohjelmista tulevia viestejä, jotka voivat olla 1.) kyselyjä tietokannasta, 2.) tietokannan tietojen määritysten muutoksia sekä 3.) tietokannan tietojen lisäyksiä, poistoja ja muutoksia. (Rajamäki 2004)

Tutkielma koostuu tästä johdantoluvusta, kolmesta muusta luvusta ja yhteenvetoluvusta. Ensimmäisessä tekstiluvussa määritellään, mitä on ohjelmistosuunnittelu. Kahdessa muussa tekstiluvussa esitellään informaatiojärjestelmien kuvaaminen ja johtopäätökset aiheesta. Kaiken esitetyn asiasisällön kokoaa yhteen yhteenveto.

1.1 Informaatiojärjestelmät (informaatiosysteemit)

Artikkelissa Avison & Nandhakumar (1995) on toistettu informaatiojärjestelmän määritelmä, jonka Buckingham & al. ovat julkaisseet jo aiemmin:

“Järjestelmä, joka kokoaa, varastoi, prosessoi ja jakaa sellaista informaatiota, joka on oleellista organisaatiolle (tai yhteiskunnalle) ja joka on niiden saatavilla ja hyödynnettävissä, jotka haluavat käyttää sitä, kuten johto, henkilökunta, asiakkaat ja kansalaiset. Informaatiojärjestelmä on sosiaalinen toimintojärjestelmä, joka saattaa sisältää tietokonejärjestelmän.”

Tämän määritelmän voidaan havaita käsittävän monia laajoja sovellusalueita, kuten esim. (Avison & Nandhakumar 1995):

- informaatioteoria (informaatio)
- semiologia (jaetaan informaatiota)
- organisaatioteoria ja sosiologia (organisaatio ja yhteiskunta)
- tietojenkäsittelytiede ja insinööritiede (tietokonejärjestelmät).

Malli on yksinkertaistettu kuva todellisuudesta. Tavallisesti toiminnan alussa on käsitteellinen malli. Myöhemmissä vaiheissa voidaan hyödyntää enemmän erikoistuneita malleja esim. matemaattisia tai rakenteellisia. Hyvin usein useita malleja tarvitaan eri vaiheissa toimintaa. Jotta kyetään vertailemaan tilanteita, kahta tai useampaa mallia voidaan tarvita samassa vaiheessa toimintaa. (Kangassalo & Jaakkola 1995, Benson & al. 1986)

Tietokoneet tekivät mahdolliseksi ei vain laskelmat vaan tietokone-elämykset ja simulaatiot monimutkaisten järjestelmien malleista, kuten biologiset, kielitieteelliset ja kognitiiviset. Kognitiota tutkivilla tieteenaloilla on tullut esille voimakas kiinnostus konnektioniseen tutkimusasetelmaan. Tästä merkkinä voidaan pitää niin kutsuttujen keinotekoisten hermoverkkojen tutkimusta. Konnektionismia voidaan pitää eräänlaisena vastareaktiona tekoälytutkimuksen piirissä harjoitetulle symboliselle informaationkäsittelylle. (Marjomaa 2002b)

Tietokonejärjestelmien kehittäminen on haastavaa puuhaa. Artikkelissa Avison & Nandhakumar (1995) Longworth identifoi yli 300 informaatiojärjestelmän kehittämismetodologiaa. Artikkelissa Avison & Nandhakumar (1995) Wood-Harper & Fitzgerald pohtivat kahta peruseroavaisuutta kehitystyössä: järjestelmät-paradigmaa, jota kuvaa ohjelmistojärjestelmät-metodi (soft systems method) (Checkland 1981); tieteellistä paradigmaa, jota kuvaa rakenteellinen analyysi ja suunnittelu (structured analysis and design) (DeMarco 1979; Gane & Sarson 1979).

Tiedonhallintafilosofia tarkoittaa periaatetta, jota koko organisaatio noudattaa. Tiedonhallintapolitiikka ja yksityiskohtaisemmat toimintaproseduurit ohjaavat toteuttamista. Tiedonhallinta on tapa toimia, kun tietokonejärjestelmät ovat käytössä. Jotta toteuttaminen mahdollistuisi tarvitaan organisaatio, ”joka suunnittelee, toteuttaa, seuraa ja ohjaa”, kertoo Kangassalo (1995) luentorungossaan. Kunnolliset työvälineet ovat organisaatiolle tarpeelliset. Organisaatio on yhteiskunnassa yleensä sosiaalinen organisaatio, jolle on ominaista mm. yhteinen päämäärä, toiminnan yhtenäinen järjestäminen ja jäsenten hierarkia sekä normien yhtäläisyys.

Kun muodostetaan organisaatiota, huomioon otettavia asioita ovat (Kangassalo 1995):

1. Ympäröivä yritys ja sen toiminta.
2. Tehtäväkokonaisuudet.
3. Henkilöresurssit.
4. Teknologian taso.
5. Osaamisen taso.

Tietohallinnon tavoitteet voi listata viidellä kohdalla (Kangassalo 1995):

1. Tietohallinnon pitää tukea yrityksen (liike)toiminnan tavoitteita.
2. Tietohallinnon strategian pitää palvella organisaation kaikkien tasojen tarpeita.
3. Tietohallinnon pitää antaa yhdenmukaista informaatiota kaikkialla yrityksessä.
4. Tietohallinnon on toimittava jatkuvasti huolimatta muutoksista organisaatiossa tai johtamisessa.
5. Tietohallinnon tulee toteuttaa järjestelmät niin, että ne sopivat kokonaisrakenteeseen.

Tietohallinnon on kyettävä vastaamaan tietoresurssin hoidosta käyttäen apuna tietojenkäsittelytoimintoa. Toisin sanoen vastaamaan seuraavista seikoista (Kangassalo 1995):

1. Hankinnasta,
2. Varastoinnista ja saatavuudesta,
3. Jakelusta,
4. Huollosta sekä
5. Näihin liittyvistä aputoiminnoista, kuten tiedon kuvaamisesta, koulutuksesta, jne.

ATK-pohjaisista johdon informaatiojärjestelmistä (Management Information Systems tai Management Support Systems, lyhyemmin MIS tai MSS) on ollut keskustelua korkeakoulumaailmassa siitä lähtien, kun hallinnon sovelluksia on ollut käytössä. Eri aikakausina MIS-käsitteen sisältö ja painotukset ovat vain hieman muuttuneet. Tarkastusviraston tarkastuskertomuksessa (2/1998) kerrotaan, että ”johdon informaatiojärjestelmä on ylimmän johdon, keskijohdon ja asiantuntijoiden käyttöön tarkoitettu elektroninen järjestelmä, joka tuottaa käyttäjälleen ajantasaista tietoa havainnollisessa muodossa. Järjestelmä raportoi oman organisaation sisäisestä tilasta sekä tuottaa tietoja toimintaympäristön tilasta ja sen muutoksista.” (Marjomaa 2002a)

MIS-järjestelmällä tarkoitettiin suuria taloudellis-hallinnollisen tiedon raportointijärjestelmiä 1960- ja 1970-luvuilla. ATK-keskuksissa tuotettiin raportteja käyttäen eräajoa. Tällöin raporttien sisällön muuttaminen oli hankalaa. Tämä oli syy, miksi raportit soveltuivat huonosti ylimmälle johdolle. (Marjomaa 2002a)

1980-luvulla (mikrotietokoneiden yleistyessä) MIS:n sijasta alettiin puhua päätöksenteon tukijärjestelmistä (DSS, Decision Support Systems). Nämä järjestelmät tarjosivat jo informaation analysointi- ja mallinnusvälineitä, mm. operatiiviseen suunnitteluun ja budjetointiin. (Marjomaa 2002a)

Tietoverkkojen aikakaudella 1980-luvun lopulla aloitettiin kehittää varsinaisia johdon informaatiojärjestelmiä (EIS, Executive Information Systems). Alkuaan EIS-järjestelmällä tarkoitettiin vain ylimmän johdon tarpeisiin tarkoitettua, integroitua kokonaisjärjestelmää.

Nykyään EIS-järjestelmän käsitettä on laajennettu (EIS, Enterprise or Everybody's Information Systems). Kuten uudistettu nimi sanoo, niin järjestelmää ei ole tarkoitettu pelkästään ylimmän johdon työvälineeksi, vaan yleensä koko organisaation hallintaan. (Marjomaa 2002a)

Informaation mallintaminen on keskeistä tietokonejärjestelmien suunnittelussa. Yleisesti voidaan sanoa, että mikä tahansa järjestelmä toimii jossain ympäristössä. Tällaista ympäristöä voidaan kutsua systeemiympäristöksi. Jos joku on kiinnostunut tietoperustasta, joka on sisällytetty informaationhallintaprosesseihin organisaatiossa (tai yrityksessä), hänen ei pitäisi miettiä organisaatiota pelkästään omana itsenään, vaan myös organisaation ympäristö pitäisi huomioida (esimerkiksi yhteiskunta). (Marjomaa 2002c, Kangassalo 1995)

1.2 Tutkimusongelmia

Avoimia tutkimusngelmia on yhä olemassa niin kuin yleensä tietojenkäsittelytieteissä ja tietojärjestelmätieteissä. Tietokonejärjestelmien suunnittelua varten on kehitetty erilaisia lähestymistapoja. Ei voi sanoa mikä on paras lähestymistapa, joten tutkimusongelmana on oikeanlaisen lähestymistavan löytäminen. Tunnetuin on ER-menetelmä, mutta muita vastaavia löytyy, kuten NIAM, SSADM ja MERISE. Tässä tutkielmassa keskitytään ER-menetelmään. Muut lähestymistavat jätetään vähemmälle tarkastelulle.

Tämän tutkielman keskeisin tutkimusongelma on liiketoiminnan ja tietokonejärjestelmien yhteensovittaminen. Tässä tutkielmassa ollaan siis kiinnostuneita ihmisten muodostamien organisaatioiden järjestelmistä, erityisesti organisaation johtamisen, tietojärjestelmien suunnittelun ja tietohallinnon kannalta. Tämä tarkoittaa sitä, että ollaan kiinnostuneita tietokonejärjestelmistä ja niihin sisältyvistä tietokannoista. Kiinnostus tietokantoja kohtaan johtuu siitä, että tietokantoihin kerätään tietoa liiketoiminnasta.

Yksi osaongelma on selvittää, mitä tarkoittaa systeemityö. Toinen osaongelma on selvittää mitä tarkoittaa systeemiympäristö. Systeemiympäristö esittää systeemille vaatimuksia ja rajoituksia, mutta se voi tarjota myös kehitysmahdollisuuksia. Systeemiympäristöt muodostavat monitasoisen rakenteen. Kolmas osaongelma on selvittää mitä tarkoittaa systeemi. Systeemi saadaan tarkastelemalla toimintakokonaisuutta annetusta

tarkastelukulmasta kiinnittäen huomio sen pysyväismuotoiseen rakenteeseen, toimintatapoihin (annettu sääntö tai muodostunut tapa) ja prosessinsuorittajiin (tehtävän tai osatehtävän suorittaja). (Kangassalo 1995)

2 TIETOKONEJÄRJESTELMIEN SUUNNITTELU

Tietohallinto ja tietojärjestelmäarkkitehtuuri voidaan määritellä seuraavasti (Kangassalo 1995):

1. Arkkitehtuurin merkitys ja sisältö.
2. Tietohallinto ja sen yhteys tietojärjestelmäarkkitehtuuriin.
3. Tietokantaympäristön kehitys.
4. Tietojärjestelmäarkkitehtuurin valintaan ja tietohallinnon muodostamiseen vaikuttavat tekijät.

Tietojenkäsittely-yksikkö / tietosysteemi on Kangassalon (1995) luentorungon mukaan mikä tahansa seuraavista:

- henkilö
- laite,
- teknologia,
- tietämysvaranto tai
- näiden yhdistelmä,

jolla on kyky käsitellä informaatiota. Informaation käsittelyn tulee olla tietojenkäsittely-yksiköllä sellaista, että informaatio on tunnistettavasti erilaista, mutta edelleen ymmärrettävää käsittelyn jälkeen. (Kangassalo 1995).

Ominaisuuksia, jotka ovat tietojenkäsittely-yksiköille ominaisia, ovat voima ja tehokkuus, joustavuus ja siitä aiheutuva monimutkaisuus, erikoistuneisuus sekä joustavuuden aste. Nämä ominaisuudet ovat suuressa määrin riippuvaisia toisistaan. Hitaus ja muutosvastus liittyvät ominaisuuden muutokseen. (Kangassalo 1995)

Perinteisesti organisaation rakenteena ovat 1.) erikoistuminen ja työnjako sekä 2.) integrointi ja koordinointi. Nämä ovat kaksi vastakkaista perusvaatimusta kaikessa organisoituneessa toiminnassa. Erään uudemman näkemyksen ovat esittäneet Knight & McDaniel Kangassalon luentorungossa (1995): Standardisoitujen tietovirtojen sisältö ja suunta, tehtävien

jakautuminen ja tietojenkäsittely-yksiköiden välinen vuorovaikutus muodostavat organisaatiorakenteen. Tästä johtuen tietojenkäsittely-yksiköiden valinta on organisaation suunnittelijoiden keskeinen tehtävä. (Kangassalo 1995)

Tietosysteemi koostuu neljästä funktionaalisesta osasta (Kangassalo 1995):

1. Tietojen hankinta tietokantaan
2. Tietokanta ja sen ylläpito
3. Laskenta ja päättely
4. Tulosten esittäminen

Edellä mainittuun listaan voi kuulua tiedonsiirtoon liittyviä funktioita. Käytännössä rakenne on monimutkaisempi. ”Osat eivät erotu näin selvästi toisistaan.” (Kangassalo 1995)

Kangassalo (1995) kertoo luentorungossaan, että ”tiedonhallintaan perustuvaan systeemien suunnitteluparadigman mukaan keskeinen suunnittelun kohde on tieto ja sen

- merkitys (käsitesisältö, yhteydet),
- rakenne,
- rajoitukset sekä
- sallitut operaatiot”

”Järjestelmien teossa ensin suunnitellaan tiedon pysyvät ominaisuudet, toisin sanoen mitä tietoja halutaan, sitten käsittelyssä tarvittavat ohjelmat ja systeemit.” (Kangassalo 1995)

Artikkelissa Avison & Nandhakumar (1995) Avison & Fitzgerald laajentavat näkemystä vertailtavuudesta ja kertovat, että tietokonejärjestelmien kehittämismetodeja voidaan vertailla perustuen filosofiaan (philosophy), malliin (model), tekniikoihin (techniques), työvälineisiin (tools), ulottuvuuksiin (scope), tuotoksiin (outputs), käytäntöön (practice) sekä tuotteeseen (product). Näiden perusteella voidaan luokitella lähestymistavat aiheittain eli
1. järjestelmät (systems), 2. strateginen (strategic), 3. osallistuva (participative), 4. protoilu (prototyping), 5. rakenteellinen (structured), 6. data (data) ja 7. oliosuuntautunut (object-oriented).

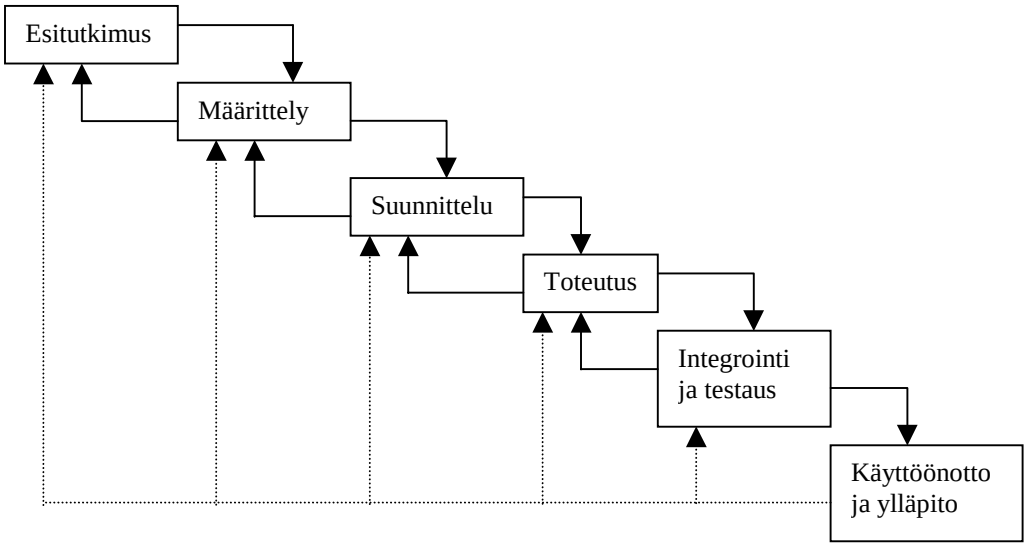
Ongelmia, joita näissä aiheissa voi olla

- Järjestelmät-lähestymistapa ei näytä relevantilta ammatinharjoittajan kannalta, joka haluaa nopeita vastauksia tiettyihin ongelmiin.
- Suunnittelu-lähestymistapa tavallisesti johtaa prioriteetteihin, joihin voimakas johto pystyy ja tällöin organisaationaalisia tarpeita ei oteta huomioon.
- Osallistuminen saattaa johtaa tehottomiin järjestelmiin, joita on ollut suunnittelemassa hyvät johtajat, virkailijat tai myyntihenkilöt. Nämä henkilöt ovat kehnoja ja haluttomia analysoijia.
- Protoilu keskittyy usein käyttöliittymään, mutta ei välttämättä osoita perustavaa laatua olevia tilanneongelmia.
- Monimutkaisen järjestelmän purkaminen johdettaviin yksiköihin, rakenteellinen analyysi, tarjoaa yksinkertaisen näkymän ja epäonnistuu kaikkien moduulien välillä olevien linkkimerkitysten nappaamisessa.
- Data-analyysi ei saata ratkaista olemassa olevia ongelmia, joita organisaatiolla on - se on saattanut napata olemassa olevat ongelmat mallista.
- Sekainen monimutkaisten organisaatioiden maailma, ihmisongelmat ja vastaavat eivät ole helposti esitettävissä objekteina.

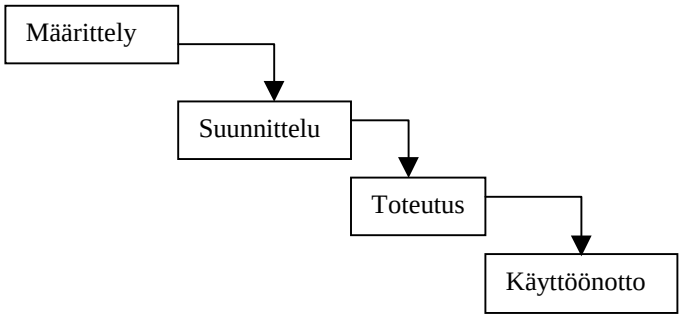
2.1 Vaihejakomallit

Ohjelmistotuotannossa käytetään yleisesti erilaisia vaihejakomalleja. Vaihejakomallit jäsentävät kehitystyön määrittely-, suunnittelu, toteutus- ja käyttöönottovaiheisiin. Kun ohjelmiston kehitystyö tai koko elinkaari (life cycle) jaetaan vaiheisiin, saadaan vaihejakomalli. Tällaisia malleja ovat mm. vesiputousmalli (waterfall model), EVO-malli (evolutionary delivery), sekä eri protoilumallit.

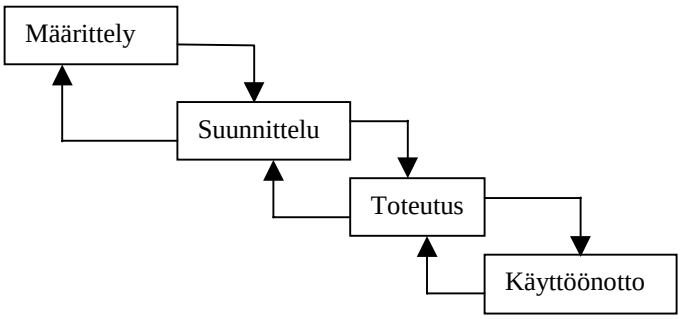
2.1.1 Vesiputousmallit (esimerkkejä)



Kuva 2.1: Vesiputousmalli (Haikala & Märijärvi 2002).



Kuva 2.2: Vesiputousmalli (Systeemityöyhdistys Sytyke ry 1992)

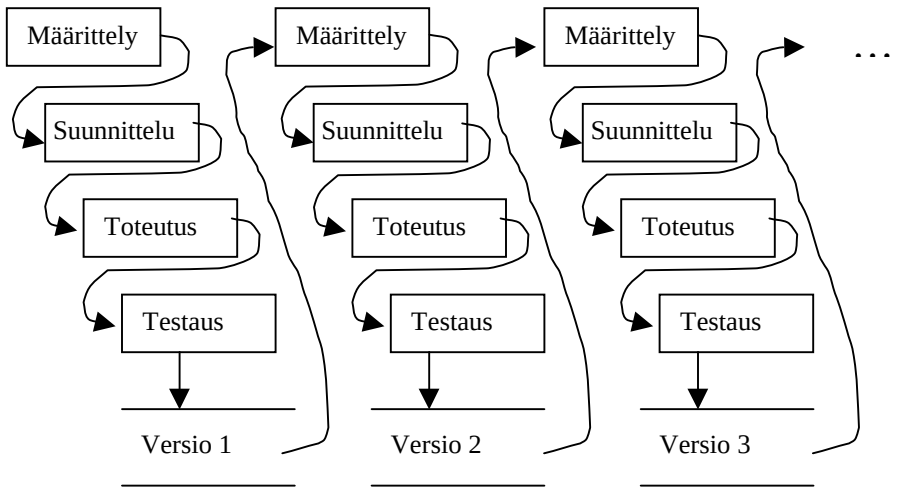


Kuva 2.3: Iteraation salliva vesiputousmalli (Systeemityöyhdistys Sytyke ry 1992).

Täsmälleen vesiputousmallin mukaisesti ei voida edetä käytännön ohjelmistokehityksessä sen vuoksi mm., että osa vaatimuksista selviää vasta projektin aikana ja vaatimukset lähes poikkeuksetta muuttuvat ajan mittaan. Vesiputousmallia voi kuitenkin pitää todellisen toiminnan mallina, jonka mukaisesti pyritään toimimaan mahdollisuuksien rajoissa.

2.1.2 EVO-malli (esimerkki)

EVO-mallin periaatteena on rakentaa ensimmäisessä projektissa ydinjärjestelmä. Tätä ydinjärjestelmää kehitetään seuraavissa projekteissa sitten edelleen. EVO-malli muodostuu sarjasta toistuvia vesiputouksia, joista jokaisen tuloksena on uusilla ominaisuuksilla kasvatettu järjestelmä.

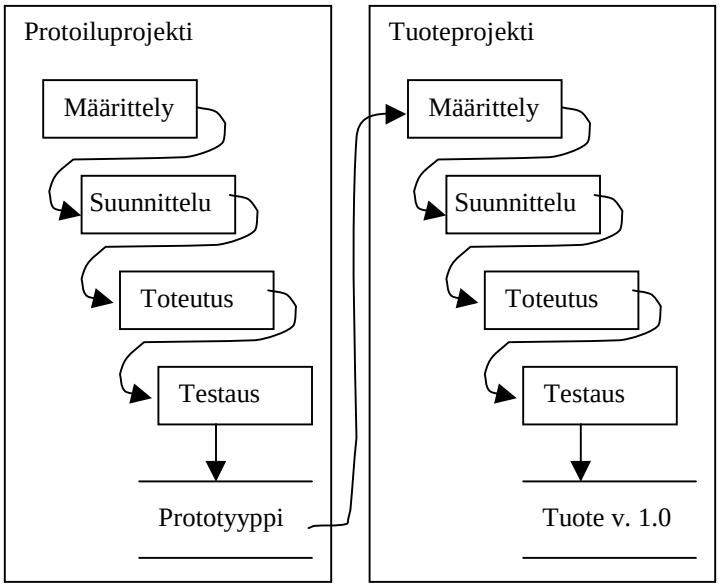


Kuva 2.4: EVO-malli (Haikala & Märijärvi 2002).

Useimpien tuotekehityshankkeiden läpivienti tapahtuu EVO-mallin mukaisesti. Tämä tarkoittaa sitä, että esimerkiksi kerran vuodessa järjestelmästä julkaistaan uusilla ominaisuuksilla täydennetty versio. (Haikala & Märijärvi 2002)

2.1.3 Protoilu (esimerkki)

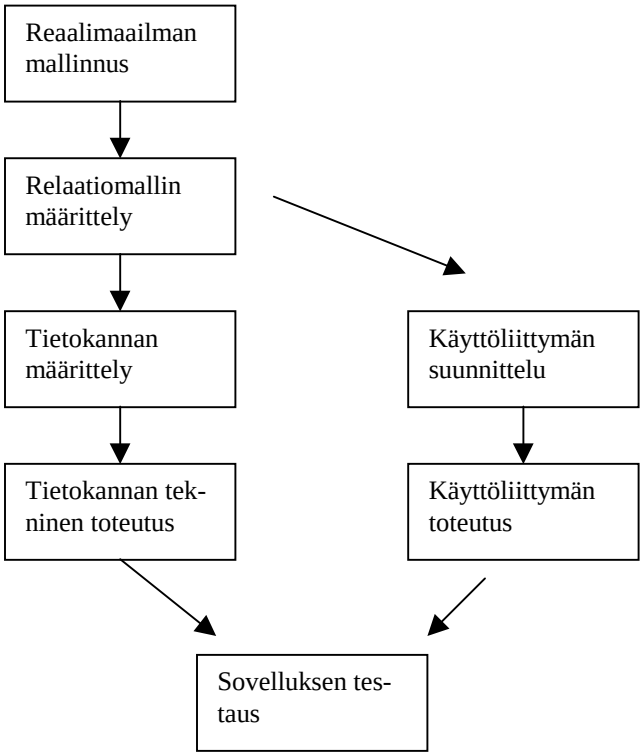
Protoilulähestymistavalla voidaan tehdä eri protoilumalleja. Tämä lähestymistapa tarkoittaa työskentelymallia, jossa jotain tuotteen piirrettä kokeillaan ennen varsinaisen tuotteen rakentamista.



Kuva 2.5: Esimerkki protoilumallista (Haikala & Märijärvi 2002).

2.1.4 Tietokantaprojektin vaihejakomalli

Vaihejakomalleissa useimmiten näkökulmana ovat käyttäjän toiminnot tai prosessit. Jos sovellusalue on voimakkaasti kortisto- tai rekisterityyppinen saattaa vaiheistus olla erilainen. Vaihejakomalli tietokantaprojektia varten voisi olla seuraavanlainen (huomaa tietokannan ja käyttöliittymän eriytyminen omiksi osaprojekteiksiinsa):



Kuva 2.6: Tietokantaprojektin vaihejakomalli (Rajamäki 2004).

Vertaa kuvaa 2.6 (tietokantaprojektin vaihejakomalli) kuvaan 3.7 (tietokannan suunnittelu). Reaalimaailman mallinnus on ensimmäinen vaihe. Tässä vaiheessa suunnittelija yhdessä käyttäjien kanssa pyrkii kuvaamaan sovellusalueen keskeiset objektit, niiden ominaisuudet ja objektien väliset suhteet (yhteydet). Itse mallinnus tehdään usein graafisilla menetelmillä (kuten ER-mallinnus), jotta käyttäjä pääsisi paremmin mukaan suunnitteluprosessiin. Mallinnuksen jälkeen saadaan aikaan relaatiomalli (tai oliomalli, mikäli tullaan käyttämään oliotietokantaa). Vaiheessa mietitään relaatiomallin loogista rakennetta eikä vielä mennä fyysiseen toteutukseen.

Relaatiomallin pohjalta määritellään tulevan tietokannan taulut, taulujen väliset suhteet (yhteydet) ja taulujen sisältämät tiedot (tietojen ominaisuudet). Tietokantaan voi jo tässä vaiheessa syöttää tietoja tietokannan hallintajärjestelmän omilla työvälineillä, vaikka tietokannan käyttöliittymiä ei vielä olisikaan olemassa. Tietokannan tekninen toteutus tehdään sitten valitulla tietokantajärjestelmällä. Kun suunnitellaan ja toteutetaan tietokantaa, niin tämän rinnalla voidaan tehdä tarvittaessa käyttöliittymän suunnittelua ja toteutusta. Tässä

käyttöliittymän suunnitteluvaiheessa ja toteutusvaiheessa tehdään ylläpitoon (lisäykset, poistot ja muutokset), tietojen kyselyihin ja tietojen raportointiin liittyvät ohjelmat. Viimeinen vaihe on sovelluksen testaus, jossa testataan toteutettuja osia eli tietokantajärjestelmää ja käyttöliittymää yhtenä kokonaisuutena.

2.2 Ohjelmistojärjestelmät-tuoteperheen käsitteellinen mallintaminen

Käsitteelliset mallit täytyy joskus rakentaa ohjelmistojärjestelmät-tuoteperhe -ajattelulle mieluummin kuin vain yhdelle ohjelmistojärjestelmälle. Tällaista tapahtuu kolmenlaisissa tilanteissa: 1.) evolutionistinen ohjelmistokehitys, 2.) ohjelmiston uudelleensuunnittelu ja 3.) ohjelmiston uudelleenkäyttö. (Jarzabek & Tiing 1995)

Esimerkiksi ohjelmiston uudelleenkäytössä etsimme jo olemassa olevia erityispiirteitä (tai konsepteja), kuten objekteja, liiketoimintasääntöjä, aliohjelmia, vaatimus/suunnittelumalleja ja koodimoduuleja, joita voidaan käyttää uudelleenmääritellyssä käyttötarkoituksessa. Käsitteelliset mallit on luotu yksittäisinä kuvina, jotka on organisoitu oliosuuntautuneella tavalla. (Jarzabek & Tiing 1995)

Käsitteellinen mallintaminen on olennainen osa missä tahansa systemaattisessa lähestymistavassa, kun ajatellaan ohjelmistokehitystä ja ylläpitoa. Käsitteellistä mallintamista pitäisi pitää määrittelyvaiheena, missä kaikki relevantti tieto tarkoituksenmukaisista näkökulmista sovelluksessa pitäisi muodollisesti ilmaista ja todentaa. Käsitteellinen mallintaminen tarjoaa korkean tason kaavamaisuuden sovelluksen informaatiorakenteen määrittämiseen - tämä tapahtuu riippumatta toteutukseen valitusta ohjelmistoympäristöstä. (Jarzabek & Tiing 1995)

2.2.1 Evolutionistinen ohjelmistokehitys (ohjelmiston elinkaari)

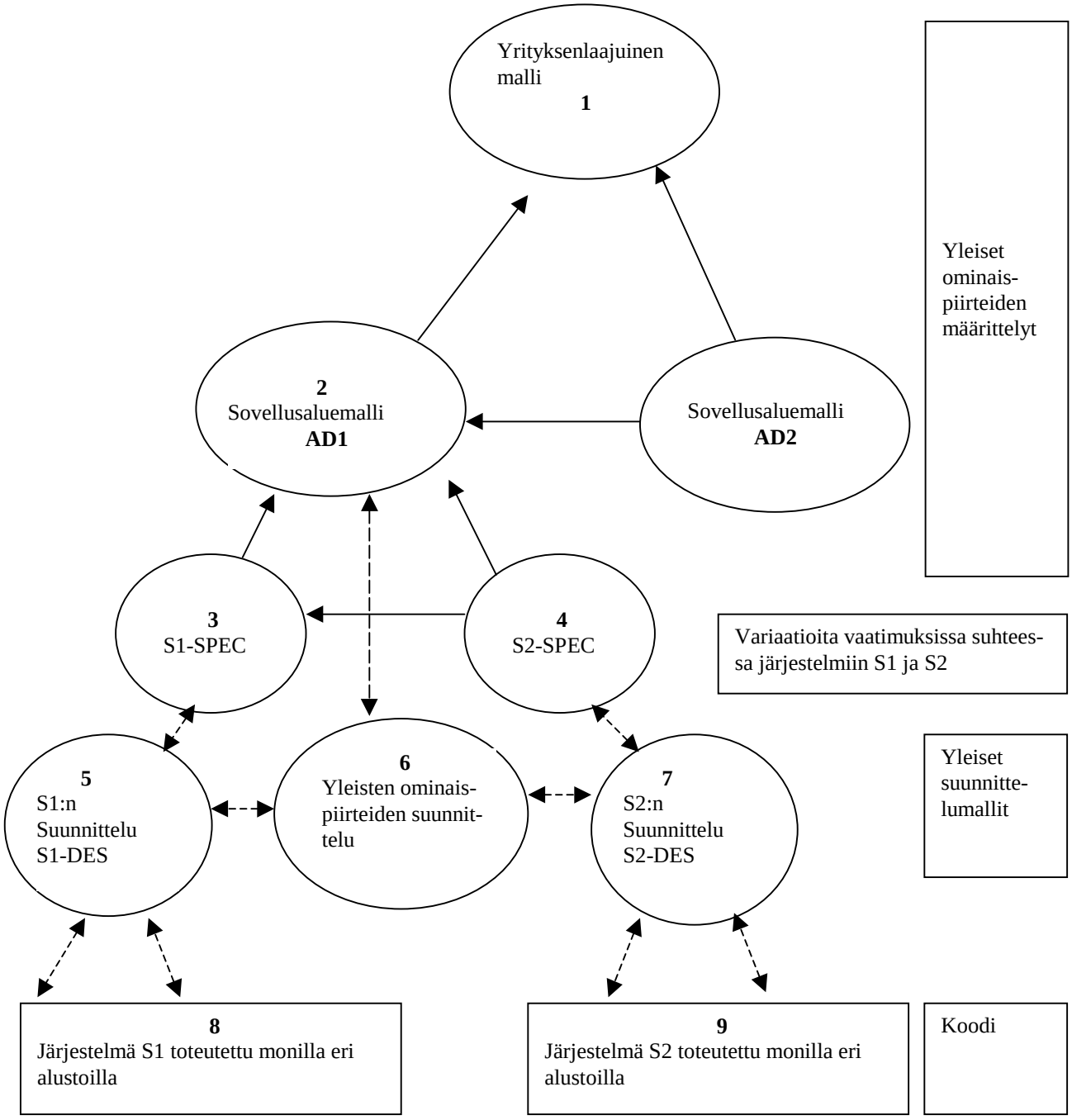
Ohjelmistotuote kehittyy, kun se läpikäy sarjan versioita, joista jokaisen osalta voidaan löytää joitakin uusia vaatimuksia. Nämä versiot muodostavat järjestelmien perheen, joilla on yleinen ydin, mutta eroavat yksityiskohdiltaan toisistaan. (Jarzabek & Tiing 1995)

2.2.2 Ohjelmiston uudelleensuunnittelu

Lähdejärjestelmä ennen uudelleensuunnittelua on S1 (ks. kuva 2) ja kohdejärjestelmä uudelleensuunnittelun jälkeen on S2 (ks. kuva 2). Nämä molemmat järjestelmät näyttävät yhtenevyyksiä ja eroja. Normaalisti perustoiminnallisuus säilytetään uudelleensuunnittelun aikana, mutta joitakin uusia vaatimuksia voidaan toteuttaa S2:ssa sekä muita vanhoja vaatimuksia voidaan muuttaa. Järjestelmät toimivat erilaisilla alustoilla ja voidaan toteuttaa eri tekniikoilla. Tehokkuuden takia on tärkeää olla olemassa yksinkertainen malli, josta voidaan jäljittää yhteneväisyydet ja erot käyttäjävaatimuksista ja toteutusyksityiskohdista molemmista järjestelmistä, sekä lähdejärjestelmästä että kohdejärjestelmästä.
(Jarzabek & Tiing, 1995)

2.2.3 Ohjelmiston uudelleenkäyttö

Kun ymmärrämme järjestelmät hyvin, tavalliset ominaispiirteet voidaan antaa ohjelmoijien käyttöön kirjastomuodossa (uudelleen käytettävät komponentit). Tavalliset ominaispiirteet voivat olla sisäänrakennettuna sovelluksen tuottamisjärjestelmässä. Jotkut vaatimukset, joita pidetään yhdelle järjestelmälle hyvinä ei välttämättä pidetä toiselle järjestelmälle samalla sovellusalueella. Kun otetaan ohjelmiston uudelleenkäyttö -lähestymistapa käyttöön, tarvitaan käsitteellinen malli ohjelmistojärjestelmät-tuoteperheelle sovellusalueella. Tämän mallin pitäisi yksilöidä ominaispiirteet, jotka ovat tavallisia kaikille järjestelmille annetulla sovellusalueella. (Jarzabek & Tiing 1995)



Kuva 2.7: Käsitteellinen ohjelmistomalli uudelleenkäyttöön (Jarzabek & Tiing 1995).

Käsitteellisen mallin arkkitehtuuri ohjelmistojärjestelmät-tuoteperhettä varten on kuvassa 2.7. Käsitteellisen mallin arkkitehtuuri koostuu yleisestä osasta (mallit 1 ja 2) ja järjestelmäominaisesta osasta (mallit 3 ja 4). Sovellusaluemalli käsittää ominaispiirteet, jotka ovat yhteisiä kaikissa järjestelmissä annetulla sovellusalueella (sovellusalueita kutsutaan myös liiketoiminta-alueiksi). Jotta ymmärrettäisiin hyvin sovellusalue, sovellusaluemalli voidaan luoda ennen kuin toteutetaan järjestelmiä. Uusille alueille mentäessä tulee toteuttaa yksi tai useampia järjestelmiä ennen kuin voidaan rakentaa sovellusaluemalli. (Jarzabek & Tiing 1995)

Kuvassa 2.7 uudelleenkäyttö voi tapahtua vaatimukset kohdassa (S1-SPEC & S2-SPEC eli mallit 3 ja 4) (erikoispiirremäärittely uudelleenkäyttöön), suunnittelukohdassa (S1-DES ja S2-DES eli mallit 5 ja 7)(erikoispiirresuunnittelu uudelleenkäyttöön) ja koodikohdassa (erikoispiirretoteutus uudelleenkäyttöön). (Jarzabek & Tiing 1995)

Käsitteelliseen malliin on menty siksi, että halutaan kuvata ymmärrettävästi uudestaan ominaispiirteet, jotka esiintyvät sovellusalueella. Käsitteellinen mallintaminen -metodimme ja notaatiomme on suunniteltu liiketoimintasovelluksiin. Kirjoittajat (Jarzabek & Tiing) mallintavat ohjelmistojärjestelmiä erikoispiirre muodossa. Erikoispiirteitä ovat esimerkiksi *objektit* (esim. kirja kirjastojärjestelmässä), *objektiattribuutit* (esim. kirjailija), *objektimetodit* (esim. kirjata kirja lainassa tilaan), *suhteet* objektien välillä (esim. asiakas *LAINASI* kirjan), *tapahtumat* (tilatun kirjan saapuminen), globaalit *aliohjelmat* ja *liiketoimintasäännöt* (esim. lainaussäännöt erityyppisille kirjastonkäyttäjille). “Nämä muodolliset linkit ohjelmistomallien ja liiketoimintamallien välillä ovat elintärkeitä, jotta ymmärretään ohjelmistojärjestelmien rooli liiketoimintaympäristössä.” (Jarzabek & Tiing 1995)

Keskeisiä käsitteellisen mallintamisen elementtejä ovat objektit, jotka käsittävät datan ja toiminnot suhteessa määrättyihin dataryhmiin. Monissa liiketoimintaohjelmissa voidaan sopivat kandidaatit objekteiksi johtaa käsitteellisestä datamallista.

3 INFORMAATIOJÄRJESTELMIEN KUVAAMINEN

Tietohallinnolla on oltava riittävä asiantuntemus seuraavista seikoista (Kangassalo 1995):

1. Tiedosta,
2. Tiedonhallinnan tavoitteista ja periaatteista,
3. Tiedonhallinnan menetelmistä ja välineistä, sekä
4. Organisaatiosta.

Tietohallinnon tehtäväkokonaisuudet käsittävät seuraavat neljä kohtaa (Kangassalo 1995):

1. Tiedonhallintaa koskevan teoreettisen ja käytännöllisen tietämyksen hankkiminen ja ylläpito yrityksen tietoresurssin hallintaa varten.
2. Tiedonhallintapolitiikan määritteleminen, toteuttaminen ja edelleen kehittäminen.
3. Tiedonhallintaa koskeva integrointi ja koordinointi.
4. Tiedottaminen ja koulutus.

”Tieto on jonkun asian säännönmukainen esitys viestitettävässä ja käsiteltävissä olevassa muodossa.” Tietämys (knowledge) taas voidaan jakaa viiteen eri muotoon (Kangassalo 1995):

1. Käsitteellistämättömät havaintotiedot (Non-conceptualised observation data)
2. Käsitteellinen tietämys (Conceptual knowledge)
3. Faktat (Facts)
4. Säännöt (Rules)
5. Päättelymekanismit (Inferencing mechanisms)

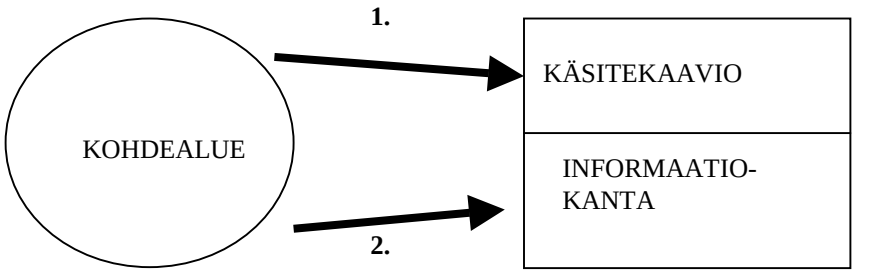
Ihmisen tietämys muodostuu monivaiheisen prosessin tuloksena. Tietämyksen ja tiedon muodostuminen tapahtuu havaintojen, abstrahoinnin ja päättelyn perusteella. Tasot, jotka tästä prosessista voidaan erottaa ovat fyysinen taso (physical level), käsitteellinen taso (conceptual level) ja kielellinen taso (linguistic level). (Kangassalo 1995)

Fyysinen taso, johon aistimukset perustuvat, on ensimmäinen taso. Käsitteellinen taso (, joka ei alkeellisessa muodossaan edellytä kielen olemassaoloa tai käyttämistä, mutta joka pystyy paljon monipuolisempaan toimintaan, kun kieli on käytettävissä), on toinen taso. Kielellinen taso, joka mahdollistaa kommunikoinnin ihmisten välillä, on kolmas taso. (Kangassalo 1995)

Tavallisesti reaalimaailman jokin osa herättää kiinnostuksemme. Tällöin pyrimme kuvaamaan tämän osan ja pyrimme ottamaan mukaan aikaelementin (menneisyys, nykytilanne ja tulevaisuus). Reaalimaailman kuvaaminen tapahtuu tietojen avulla. Reaalimaailmaa esittävästä kuvasta vain pieni osa on tietokannassa tietoina. (Kangassalo 1995)

Ajan kuluessa tapahtuu jatkuvasti kolmenlaisia muutoksia: kuvauksen kohteessa, kuvauksen rajauksessa sekä kuvattavien asioiden valinnassa. Nämä muutokset aiheuttavat tietosysteemin ja tietokannan suunnittelijalle ongelman: Kuinka vastaavuus tietojen ja kuvattavan maailman välillä voidaan säilyttää ajan kuluessa? Vastauksena Kangassalo (1995) esittää luentorungossaan seuraavaa:

- **Analysoimalla** kuvattavan maailman looginen rakenne,
- **Laatimalla** kuvattavan maailman loogista rakennetta vastaava käsitteellinen malli,
- **Huolehtimalla** siitä, että kuvattavaa maailmaa kuvaava tietojoukko noudattaa laadittua käsitteellistä mallia,
- **Huolehtimalla** siitä, että kuvattavan maailman muuttuessa myös laadittu käsitteellinen malli muutetaan,
- **Huolehtimalla** siitä, että myös tietojoukkoon kohdistuvat muutokset suoritetaan käsitteellisen mallin osoittamalla tavalla.



Kuva 3.1: Kohdealueen kuvaaminen (Kangassalo 1995)

Kuvassa 3.1 kohdealueen kuvaaminen (Kangassalo 1995):

1.) ”Kohdealuetta koskeva abstrahointi, luokittelu, yleistys, sääntöjen muodostaminen, yms. ja näiden esittäminen jollakin kielellä.”
2.) ”Faktojen ja tapahtumien kirjaaminen kohdealueesta.”

”Kohdealue jakaantuu fyysiseen ja abstraktiin osaan, mutta rajanveto näiden välillä on joissakin tapauksissa vaikeaa”, jatkaa Kangassalo (1995) luentorungossaan.

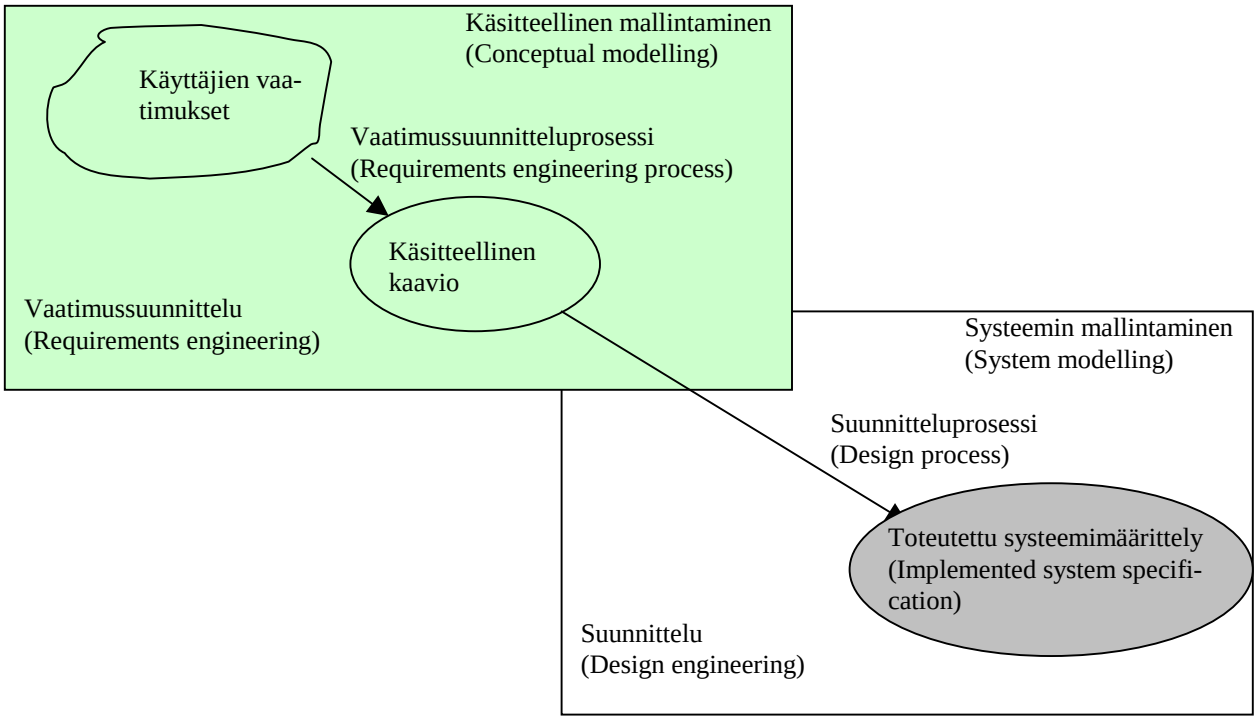
On olemassa monenlaisia malleja, joita voidaan käyttää sen mukaan, mitä näkökulmaa painotetaan. Esimerkiksi 1.) pienoismallit painottavat muotoa tai fyysistä rakennetta objektissa, 2.) käsitteelliset mallit painottavat käsitteitä ja rakennetta, jonka ne muodostavat, 3.) matemaattiset mallit painottavat matemaattisia ominaisuuksia järjestelmässä ja 4.) graafiset mallit painottavat kognitiivisia ja visuaalisia ominaisuuksia järjestelmässä. (Kangassalo & Jaakkola 1995)

Pystyäkseen käyttämään mallia henkilön pitää päättää työskentelytilanteessaan näkemys siitä, mitä pitää ottaa huomioon, mallin tyyppi, jota hän haluaa käyttää ja mitä hän haluaa saavuttaa mallin avulla, sekä mitä hän voi tehdä mallilla.
(Kangassalo & Jaakkola 1995)

3.1 Käsitteellinen mallintaminen

Käsitteellinen mallintaminen on tietojenkäsittelytieteen osa-alue. Se on kehitetty 1970-luvun loppupuolella yhdistämään sellaisia alueita kuten tekoäly, ohjelmointikielet, tietokantasuunnittelu ja käytettävyystekniikat. (Marjomaa 1997)

On tavallista nähdä tietosysteemi mallina jostakin aidosta organisaation osasta. Kuten kuvasta 3.2 voidaan nähdä näitä malleja kehitetään osana kahta pääkehitystoimintoa, nimeltään vaatimussuunnittelu (requirements engineering) ja suunnittelu (design engineering). (Rolland 1994)



Kuva 3.2: Tietojärjestelmän kehitysprosessi (Information system development process) (Rolland 1994)

Tietokannan suunnittelu muodostuu kolmesta osasta: käsitteellisestä, loogisesta ja fyysisestä osasta. Käsitteellisessä mallintamisessa tietokanta suunnitellaan riippumatta toteutusympäristöstä. Loogisessa osassa huomioidaan jo tulevan tietokannan tietomalli esim. relaatiotietokantamalli ja fyysisessä osassa toteutetaan tietokanta toteutusympäristöön.

Kangassalo (1999) määrittelee seuraavalla tavalla **käsitteellisen mallintamisen**: ”Sillä tarkoitetaan **tietojärjestelmän** tunnistamiseen, suunnittelemiseen ja määrittelemiseen tarvittavien, järjestelmän kohdealuetta ja siihen liittyvää tietosisältöä kuvaavien, toteutuksesta riippumattomien käsitteellisten mallien luomista.” Haluttuna järjestelmänä voidaan myös nähdä käsitteellinen malli eli käsitekaavio.

Tietokanta (database) tietosysteemin osana tarkoittaa tässä sitä, että tietokanta kuuluu toimivaan tietosysteemiin. ”Käsitekaavio määrittelee kaiken, mitä tietokannassa (informaatiokannassa) voi olla”, mainitsee Kangassalo (1995) luentorungossaan.

Tiedon mallintaminen on eri asia kuin käsitteellinen mallintaminen. Käsitteellisen mallintamisen päämäärä on rakentaa esitys UoD:stä eli yritetään saada selville käyttäjän

näkemys kohdemaailmasta. UoD:tä pidetään merkkien ja signaalien järjestelmänä. UoD:tä koskevan informaation tiivistäminen (condense) on kaiken merkittävän ilmaiseminen mahdollisimman lyhyesti, joka koskee UoD:n aineosia. (Kangassalo 1999, Viitanen 2002, Marjomaa 2002c)

Malli tai malleja voidaan laatia siitä kohdealueesta, josta halutaan tietämystä. Täten voidaan tehdä ’näkyväksi’ se UoD (Universe of Discourse), josta halutaan tietoja. Näin voidaan esittää, mitä tietoja halutaan ja miksi. Jotta systeemi olisi paras mahdollinen, mallien avulla voidaan suunnitella, mitä tietoja pitäisi haluta. Kun mietitään mikä kuvaamisen tarkoitus on, löytyy viisi motiivia: Havainnollistamis-, pelkistämis-, jäsentely-, päättely- sekä ymmärtämismotiivi.

Käsitteellisiin malleihin ja käsitteelliseen mallintamiseen on olemassa kaksi lähestymistapaa:

1.) Puhdas abstrahointi kohteesta
- Käsitteitä ei luokitella.
- Käsitteet ja niiden muodostama rakenne perustuvat pelkästään suoritettuun abstrahointiin.

2.) Mallikäsitteitä käyttävä mallintaminen
- Käytetään ’tyhjää’ mallikäsitteistöä tukemaan abstrahointia.
- Mallikäsitteistöön kuuluvat käsitetyypit määräävät muodostettavien käsitteiden luokittelun.
- Mallikäsitteistön rakenne määrää syntyvän käsitteellisen mallin rakenteen.

Käsitteellisen mallintamisen välineillä voidaan tarkastella informaatiojärjestelmää, joka koostuu tällöin maailmaa kuvailevista objekteista. Tällaiset objektit kuuluvat erilaisiin luokkiin. Objekteilla on omanlaisiaan ominaisuuksia ja ne voivat olla toisiinsa liittyneitä monin erilaisin tavoin. Maailman tarkasteleminen objektien ja liitoksien avulla on keino kuvata informaatiojärjestelmiä strukturoidulla tavalla. Yrityksen rakentamisessa käsitteellistä mallintamista on voitu hyödyntää esimerkiksi selvitettäessä ja kehitettäessä yrityksen tehtävää ja päämääriä. (Boman & al. 1997)

Kun käsitteellisen mallintamisen avulla analysoidaan laajaa objektijärjestelmää, tehtävästä voi tulla liian monimutkainen yhdelle ihmiselle tai jopa pienelle ryhmälle. Tällaista

monimutkaista objektijärjestelmää varten on tavallista soveltaa ”hajoita ja hallitse” –strategiaa. Strategiassa porukka jaetaan pieniin ryhmiin, joista jokainen keskittyy objektijärjestelmän tiettyyn osaan ja konstruoi käsitekaavion, jota kutsutaan tuon osan *lokaaliseksi kaavioksi*. Kun lokaaliset kaaviot on saatu valmiiksi, ne yhdistetään *globaaliksi kaavioksi*, joka kuvailee koko järjestelmää. Lukuisten lokaalisten kaavioiden integroimista yhdeksi globaaliksi kaavioksi kutsutaan *näkökulmien integroinniksi (view integration)*.

Samanlaista toimintaa, *tietokantaintegraatiota (database integration)*, on hajautettujen tietokantojen suunnittelussa (distributed database design), missä jo olemassa oleville informaatiojärjestelmille oleva kaaviojoukko on yhdistetty globaaliksi kaavioksi. Jotta voidaan viitata sekä näkökulmaintegrointiin että tietokantaintegrointiin käytetään yleisempää *termiä kaaviointegrointi (schema integration).*

”Tietojärjestelmän suunnittelun aikana tehtävä mallintaminen sisältää kaksi vaihetta: **ensin** ihmisen kognitioon perustuvien ja käyttäjän tarpeiden mukaisesti muotoiltujen käsitteiden tunnistaminen, nimeäminen, kuvaaminen, analysointi, tarkentaminen, määrittely, mahdollinen uudelleenmäärittely ja synteesi, sekä näihin käsitteisiin perustuvan käsitteellisen mallin muodostaminen ***ja sitten*** tämän käsitteellisen mallin muuntaminen tietojärjestelmän toteuttamista varten sopivaan muotoon, esim. UML-kielen (Unified Modeling Language) mukaiseksi.” (Kangassalo 1999)

1.) **Hyvä käsitteellinen malli** ja 2.) **näkyvyys** ovat perusperiaatteet, kun suunnitellaan ihmisille tarkoitettuja esineitä. Tämä hyvä käsitteellinen malli takaa sen, että voimme ennustaa toimintamme seuraukset. Jos meiltä puuttuu hyvä käsitteellinen malli, emme voi täysin ymmärtää :

”mitä teemme, miksi teemme, mitä seurauksia on odotettavissa tai mitä pitäisi tehdä, jos jokin menee vikaan.”

Näkyvyys taas toimii muistutuksena siitä; mitä pystytään tekemään ja kuinka toiminto on suoritettava. Hyvä yhteys säätimen sijainnin ja säätimen toiminnan välillä takaa sen, että käyttäjän on helppo löytää määrättyä toimintoa vastaava säädin. (Hesso 2002)

Mallinnustyö on iteratiivista, koska käyttäjän ja suunnittelijan tietämyksen kehitysprosessi käynnistyy mallinnustyössä erittäin usein. Käsitekaavioita joudutaan tekemään useita. Tietokone voi olla apuna mallinnustyössä. Ns. **esittävät käsitekaaviot** ja **määrittelevät käsitekaaviot** voidaan erottaa toisistaan. ”Käsitteitä käytetään esittävässä käsitekaaviossa pelkästään nimien ilmaisemiseen tietyssä asiayhteydessä. Määrittelevässä käsitekaaviossa tarkkuus lisääntyy. Tällöin määrittelevässä käsitekaaviossa esitetään nimien ja asiayhteyksien lisäksi kunkin käsitteen määritelmä tai useampia määritelmiä. Näistä käyttäjä voi halutessaan tutkia käsitteen sisällön haluamalleen tarkkuustasolle asti.” Käsitejärjestelmät ja käsitekirjastot voivat helpottaa käsitteiden muotoilua ja määrittelemistä sekä koko sovellusten suunnittelua, jos niitä käytetään tietoisesti sovellusten määrittelyn perustana. (Kangassalo 1999)

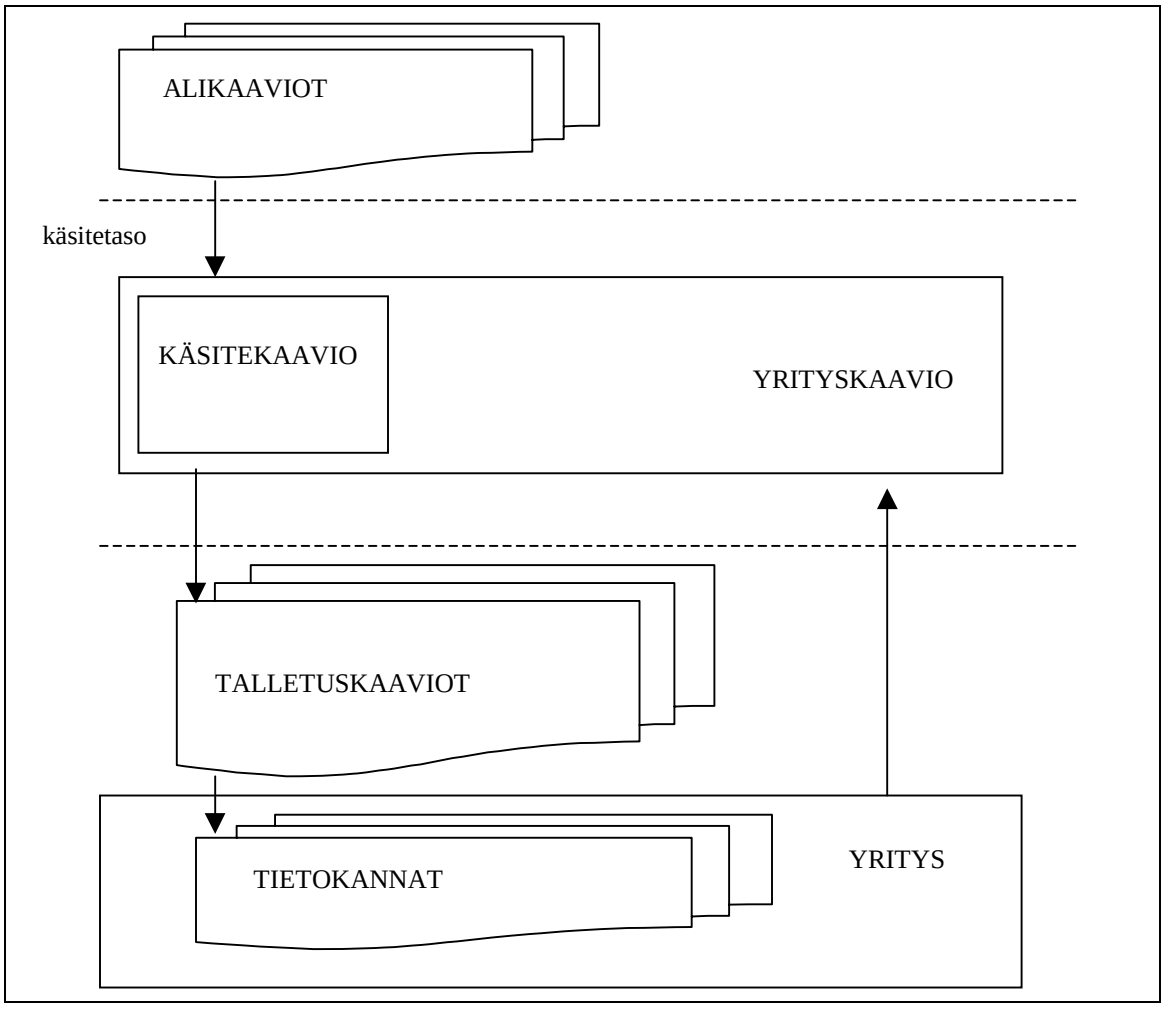
Yksinkertainen malli on se, mikä luodaan ensin, kun käsitellään asiaa tieteellisesti. Tätä olemassa olevaa yksinkertaista mallia sitten kehitetään ja monimutkaistetaan jos asiaa halutaan käsitellä tieteellisesti. Malli, tai teoreettinen skematisaatio, on käsitteellinen järjestelmä, joka yrittää edustaa joitakin todellisten järjestelmien toisiinsa liittyviä puolia. (Hölttä 2002)

Käsitteellisen mallin luovaa suunnittelua kutsutaan käsitteelliseksi mallintamiseksi (conceptual modelling) ja mallin altistamista kuvitteellisille muutoksille kutsutaan simulaatioksi (simulation). Mallintamis-simulaatioparia taas kutsutaan ajatuskokeeksi (thought experiment) (Hölttä 2002)

Mallintamistyö voidaan aloittaa mahdollisesti jo olemassa olevan järjestelmän käsitteellisestä kuvaamisesta. Uuden järjestelmän määrittely voidaan tehdä joko muuttamalla jo olemassa olevan järjestelmän käsitteellistä kuvausta asteittain (iterointi) tai tekemällä kokonaan uusi kuvaus. Mallintajan (mallin tekijä) on osattava ainakin yksi systemaattinen työmenetelmä. Työmenetelmät ovat tapoja soveltaa kuvaustekniikoita eli notaatioita. Kuvaustekniikat taas ovat “kieliä” erilaisten asioiden ilmaisemiseksi, esim. tietovirtakakaaviot eli tietovuokaaviot (data flow diagram, DFD) ja ER-kaaviot (Entity-Relationship diagram, ERD). Suunnittelija joutuu mallinnustyössään miettimään myös omaa metakognitiotaan, toisin sanoen sitä, kuinka hän ajattelee ja ratkaisee olemassa olevia ongelmia. ”Metakognitio on henkilön tietoisuutta

omista kognitiivisista prosesseistaan ja kykyä valita, ohjata, säädellä ja evaluoida omaa ajatteluaan.” (Kangassalo 1999, Haikala & Märijärvi 1998)

Kun informaatiojärjestelmää suunnitellaan, on hyvä tietää minkälaisia kuvaamisen tasoja yrityksen tulee käyttää. Näitä kuvaamisen tasoja ovat yrityskaavio, käsitekaavio, alikaavio (eli ulkoinen kaavio) sekä talletuskaavio (eli sisäinen kaavio). Yrityskaavio (enterprise schema) on käsitteellinen kuvaus yrityksestä, yrityksen toiminnasta sekä ympäristöstä. Käsitekaavio (conceptual schema) on käsitteellinen sisällön kuvaus. Käsitekaaviolla kuvataan myös, kuinka tietokanta liittyy yrityksen toimintaan. Alikaavio (subschema) on kuvaus tietokannasta sellaisena kuin sovellusohjelma sen tuntee. Talletuskaavio (storage schema) on tietokannan fyysisen rakenteen kuvaus. Kuvaamisen tasot on esitetty kuvan 3.3 avulla. (Kangassalo 1995)



Kuva 3.3: Kuvaamisen tasot. (Kangassalo 1995)

3.2 Käsitekaaviokielet

“Käsitekaaviokieli on käsitekaavion esittämiseen tarkoitettu, systemaattisesti kehitetty kuvauskieli“ (Kangassalo 2004). Tällainen kieli voi olla tekstuaalinen, graafinen tai matemaattinen tai niiden yhdistelmä. Useimmat käsitekaaviokielet perustuvat joukko-oppiin. (Kangassalo 2004)

Ohjelmistokehityksestä kattaa suuren osan tietokantojen suunnittelu ja toteutus. 1950- ja 1960-luku olivat tiedostojen aikaa. Tällöin tiedostot olivat ns. peräkkäistiedostoja. 1960-luvn puolivälissä rumpu- ja levylaitteiden läpimurto mahdollisti suorahaun käytön tiedostoja käsiteltäessä. 1970-luvulla alkoivat tietokannat yleistyä. Aluksi tietokannat olivat verkkotietokantoja, joiden esikuva oli 1965 General Electricin julkistama IDS (Integrated Data Store) –ohjelmisto. Codasyl (Conference on Data Systems Languages) eli käyttäjien ja laitteistovalmistajien yhteisö julkisti standardin vuonna 1971. Tämän seurauksena 1970-luvun alkupuolella syntyi muutamia hyviä ratkaisuja ajatellen tulevaisuutta. Näitä olivat tiedon määrittelykieli (data definition language), tiedon käsittelykieli (data manipulation language) sekä perusideat eli fyysinen tietoriippumattomuus ja looginen tietoriippumattomuus. Standardia täydennettiin vuosina 1978 ja 1981. (Sainio 2001 & Rajamäki 2004)

Standardoinneista seurasi se, että lähes jokaiselle 1970- ja 1980-luvun tietokonelaitteistolle on tuotettu Codasyl-standardien mukainen tietokannan hallintajärjestelmä. Relaatiotietokantojen vuoro tuli 1980-luvulla. Mutta relaatiotietokannan idean esitti E. F. Codd jo vuonna 1970 julkaisussaan “A relational model for large shared data banks”. Relaatiotietokannan keskeiset piirteet ovat:

- tietokannan näkyminen käyttäjille taulukoina.
- talletustapa (sisäinen) ei näy sovellusohjelmille eikä siis myöskään käyttäjille
- sql-kielen (structured query language) kehittäminen
- perustuu relaatioalgebraan eli on mateemaattisesti perusteltu

Syy, miksi tietokantojen suunnittelu ja toteutus ovat mukana tässä tutkielmassa johtuu siitä, että tämän tutkielman keskeisin tutkimusongelma on yleisesti liiketoiminnan ja tietokonejärjestelmien yhteensovittaminen. Tietokantoihin kerätään keskeistä tietoa

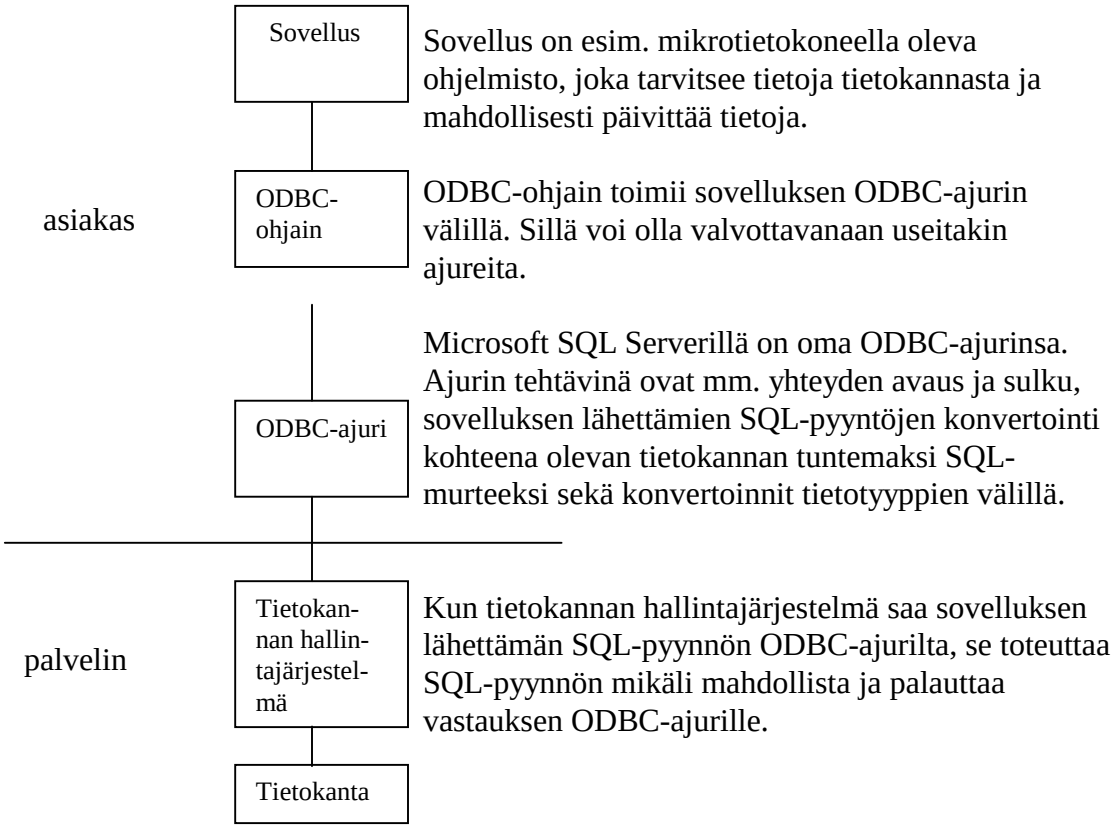
liiketoiminnasta. Ja kuten tutkielman otsikkoon viitatakseni, tietokantojen kuvaamisessa käytetään apuna käsitteellistä mallintamista yms.

3.2.1 SQL

Vuosina 1974-1979 olleen projektin lopputulos oli System R, jossa käytettiin relationaalisena kielenä SEQUEL-kieltä, josta nykyinen SQL (Structured Query Language = rakenteellinen kyselykieli) on kehittynyt. SQL kehitettiin IBM:n San Josen tutkimuslaboratorioissa. Alkuperäinen nimi oli SEQUEL. SQL oli menestys ja niinpä se on käytössä lukuisien toimittajien sadoissa eri järjestelmissä. Vuonna 1979 Relational Software Inc. (nykyinen Oracle Corporation) julkisti ensimmäisen SQL-pohjaisen tuotteen. Oraclea seurasi IBM:n julkistukset nimeltään SQL/DS (vuonna 1981) ja DB2 (vuonna 1983). (Laiho 1992 & Rajamäki 2004)

Kuitenkaan kaikki relaatiotietokannat (esim. Ingress) eivät tue SQL-kieltä. Koska erilaisia tietokantaohjelmistoja on useita, ohjelmistovalmistajat ovat sopineeet yhteysmenetelmistä. Tämä tarkoittaa sitä, että on standardoitu tapa, jolla kaksi eri tietokantajärjestelmää keskustelee keskenään. Yksi tällainen standardeja muodostava yhteisö on SAG (SQL Access Group), jossa ovat mukana useimmat ohjelmistovalmistajat. SAG:n ehdostusten pohjalta on syntynyt useita ratkaisuja. Tietokantajärjestelmiä varten kehitettiin ODBC-rajapinta (Open DataBase Connectivity) tukemaan SQL-kielen käyttöä erilaisissa tietokantajärjestelmissä. Tämän ODBC-rajapinnan ideana on, että sovellusohjelma, joka käyttää SQL-kieltä voi käyttää mitä tahansa tietokantaa, johon on ODBC-ohjain sekä ODBC-ajuri (driver). (Sainio 2000, Sainio 2001 & Rajamäki 2004)

Microsoft Access –ohjelman mukana tulee eräs ODBC-ohjain, Microsoft SQL Server. Sovellusohjelman ja tietokannan hallintajärjestelmän välissä on ohjain sekä ajuri, jotka muuttavat sovelluksesta tulevat SQL-käskyt sellaiseen muotoon, että ne kelpaavat tietokannan hallintajärjestelmälle. Vastaavasti tietokannan hallintajärjestelmästä tulevat viestit muutetaan standardi-SQL-muotoisiksi, jotta sovellus kykenisi ottamaan viestit vastaan. SQL on kyselykielen lisäksi tietokannan määrittely- tai kuvauskieli (DDL, Data Definition Language tai Data Description Language), sekä tietokannan käsittelykieli (DML, Data Manipulation Language), jolla voi tehdä tietokantaan tietojen lisäyksiä, poistoja ja muutoksia. Rakennekuvana edellä kuvattu ODBC-rajapintaratkaisu on seuraavanlainen (kuva 3.4):



Kuva 3.4: ODBC-rajapintaratkaisu (Sainio 2001).

Edellä olevasta kuvasta puuttuu tarkoituksella tietoliikenneyhteys, jota varten tarvitaan oma ohjelmistonsa. (Sainio 2000 & Sainio 2001)

Hyvillä SQL-tietokannan hallintaohjelmilla on kolme osaa. Niillä on SQL-syntaksi, joukko funktioita ja joukko proseduraalisia komentoja. On olemassa sekä *proseduraalisia* että *ei-proseduraalisia* kieliä. Proseduraaliset kielet ovat sellaisia, että niitä voi pitää perinteisinä ohjelmointikielinä. Ne sallivat sellaisten ohjelmien luomisen, jotka on laadittu proseduureilla . Toisin sanoen tällaisella kielellä jono komentoja voidaan tallettaa suoritustiedostoon tai ohjelmaan. Proseduraalisilla kielillä on tekemistä kahdenlaisen ohjelmahallinnan kanssa: ohjelmavuokontrollin sekä datavuokontrollin. Proseduraalinen kieli eroaa ei-proseduraalisesta kielestä siinä, että proseduraalisella kielellä on sanastossaan komentoja tai lauseita, jotka hoitavat *ohjelmavuokontrollia*. (Stephenson & Hartwig 1992)

Ohjelmavuokomennot ovat niitä, jotka kontrolloivat ohjelmaprosessia. Ne siis käskevät PC:n käyttöjärjestelmää suorittamaan jonon komentoja määrätyssä järjestyksessä.

Datavuokomennot ovat niitä joilla on tekemistä ainoastaan lisäyksen, poiston tai datan poiminnan kanssa joko yhdestä tai useammasta tietokannasta tai datatauluista. Datavuokomennot ovat olemassa vain datan manipulointiin ja kyselyn tuloksien palauttamiseen. Täten on olemassa kieli, joka sallii vain datan manipuloinnin. Tällaisella kielellä ei ole ohjelmavuokontrollikomentoja sanastossaan ja tällaista kieltä kutsutaan ei-proseduraaliseksi kieleksi. Sellaisia kieliä ovat kyselykielet eli esim. SQL. Ohjelmaa ei voi kirjoittaa SQL:llä. Mutta SQL-kieltä voi sisällyttää (sulauttaa) ohjelmaan, joka on kirjoitettu proseduraalisella kielellä. (Stephenson & Hartwig 1992)

SQL-kieltä voi käyttää 1.) itsenäisesti eli erikseen, 2.) upotettuna (jollakin ohjelmointikielellä esim. C-kieli tai COBOL), 3.) staattisesti (eli SQL-käskyt ovat kiinteä osa ohjelmaa) tai 4.) dynaamisesti (ohjelman suoritusvaiheessa annetaan SQL-käskyjä). (Stephenson & Hartwig 1992, Sainio 2000 & Rajamäki 2004)

Relaatioalgebran muutamat perusoperaatiot, joita käytetään relaatiotietokantojen yhteydessä ovat *yhdiste* (union), *liitos* (join), *valinta* (selection) sekä *projektio* (projection). Yhdiste tarkoittaa sitä, että kaksi samanlaista taulua (siis samanlaiset sarakkeet) yhdistetään uudeksi tauluksi. Liitos taas tarkoittaa kahden taulun yhdistämistä uudeksi tauluksi jonkin avaimen perusteella. Valinta tarkoittaa tilannetta kun jostain taulusta halutaan valita uuteen tauluun vain tietyt rivit. Projektio tarkoittaa tilannetta missä jostain taulusta valitaan uuteen tauluun vain tietyt sarakkeet. (Rajamäki 2004)

Taulu voidaan määritellä SQL-kielellä esim.

```
CREATE TABLE viinikellari
( viinimerkki          CHAR(20)    NOT NULL,
  vuosikerta           SMALLINT,
  pulloja              INTEGER );
```

Tämä edellä kuvattu muodostaa taulun rakenteen. Tiedot (testiaineisto) voidaan viedä sitten tauluun jolloin saadaan esim.:

VIINIKELLARI

VIINIMERKKI	VUOSIKERTA	PULLOJA
Zinfandel	1977	90
Chardonnay	1898	1
Fume Blanc	1985	207
Côtes du Rhône	1985	160
Côtes du Rhône	1986	304

Taulun rakenteen määrittelyssä CHAR(20) tarkoittaa, että taulun riveille sarakkeeseen viinimerkki voidaan tallettaa tiedon arvoksi korkeintaan 20 merkkiä pitkiä merkkijonoja. Määrityksellä “NOT NULL” asetetaan rajoitus. Viinimerkillä on oltava joku arvo jokaisella taulun rivillä. Toisin sanoen viinimerkki ei saa puuttua. Vuosikerta-sarakkeen tietotyypppi SMALLINT tarkoittaa pieniä käytännössä korkeintaan 4 numeroisia kokonaislukuja. Pulloja-sarakkeen tietotyyppi INTEGER tarkoittaa korkeintaan 9 numeroisia kokonaisulukuja. (Laiho 1992)

Relaatiokaavana taulu voisi olla: VIINIKELLARI (viinimerkki, vuosikerta, pulloja). Alleviivaus tarkoittaa, että kyseessä on perusavain. Eli perusavaimen muodostaisivat “viinimerkki” sekä “vuosikerta” yhdessä. (Laiho 1992)

Esimerkkitaulusta (viinikellari) voidaan tehdä SQL-kysely seuraavilla valintaehdoilla (Laiho 1992):

```
SELECT viinimerkki, vuosikerta, pulloja
FROM viinikellari
WHERE vuosikerta = 1985;
```

SQL-kysely tuottaa seuraavanlaisen taulun:

(TULOSTAULU)

VIINIMERKKI	VUOSIKERTA	PULLOJA
Fume Blanc	1985	207
Côtes du Rhône	1985	160

Kun käytetään valintaa (selection) niin valitaan taulusta tietyn ehdon täyttävät rivit (Laiho 1992):

SELECT * FROM viinikellari
WHERE vuosikerta BETWEEN 1985 AND 1986;

(TULOSTAULU)

VIINIMERKKI	VUOSIKERTA	PULLOJA
Fume Blanc	1985	207
Côtes du Rhône	1985	160
Côtes du Rhône	1986	304

Kun käytetään projektiota (projection) niin taulusta valitaan vain tietyt sarakkeet (Laiho 1992):

SELECT viinimerkki, vuosikerta FROM viinikellari;

(TULOSTAULU)

VIINIMERKKI	VUOSIKERTA	
Zinfandel	1977	
Chardonnay	1898	
Fume Blanc	1985	
Côtes du Rhône	1985	
Côtes du Rhône	1986	

Projektion tuloksena voi tulla eteen tilanne, jossa esiintyy sarakearvoiltaan samoja rivejä eli kaksosrivejä. Esim. (Laiho 1992):

SELECT viinimerkki FROM viinikellari;

(TULOSTAULU)

VIINIMERKKI
Zinfandel
Chardonnay
Fume Blanc
Côtes du Rhône
Côtes du Rhône

Projektion voi saada sellaiseen muotoon, että kaksosrivit poistuvat. Tämä tapahtuu lisäämällä SELECT sanan jälkeen tarkenne “DISTINCT”. Eli seuraavasti (Laiho 1992):

SELECT DISTINCT viinimerkki FROM viinikellari;

(TULOSTAULU)

VIINIMERKKI
Zinfandel
Chardonnay
Fume Blanc
Côtes du Rhône

Kun esimerkkiä laajennetaan niin luodaan valmistajat-taulu. Eli relaatiokaavana (Laiho 1992):

VALMISTAJAT (<u>valtun</u>, valnimi, alue)

Viinikellari-taulun ominaisuus “valmistaja” on perusavaimen osa ja samalla viiteavain tauluun valmistajat. Tiedot (testiaineisto) voidaan viedä sitten tauluun (valmistajat), jolloin saadaan esim. (Laiho 1992):

VALMISTAJAT

VALTUN	VALNIMI	ALUE
A1067	Mirassou	California
A1345	Buena Vista	California
A3105	ALKO (pull.)	Ranska
B1092	Ch.St. Jean	California

Kun käytetään liitosta (join) niin kootaan uusi taulu kahdesta tai useammasta taulusta jonkin avaimen perusteella (Laiho 1992 & Rajamäki 2004):

SELECT valmistaja, valnimi, viinimerkki, vuosikerta FROM viinikellari, valmistajat
WHERE valmistaja = valtun;

(TULOSTAULU)

VALTUN	VALNIMI	VIINIMERKKI	VUOSIKERTA
A1067	Mirassou	Zinfandel	1977
A1345	Buena Vista	Chardonnay	1898
A3105	Ch.St. Jean	Fume Blanc	1985
B3105	ALKO (pull.)	Côtes du Rhône	1985
A3105	ALKO (pull.)	Côtes du Rhône	1986

Perustaulujen (base table) lisäksi on olemassa ns. näkymiä (view). Näkymiä ei käytetä samalla tavoin, kuin perustauluja. Näkymät eivät ole todellisia, tallentuvia tauluja kuten perustaulut, vaan näkymä muodostuu perustaulujen tiedoista, kun näkymää käytetään. Esimerkki näkymän käytöstä on alun perin englanninkielisen taulun muunto suomenkielelle. Tämä tapahtuu seuraavasti (Laiho 1992):

CREATE VIEW varasto (artikkeli, maara, hinta, arvo)
AS SELECT article, qty, price, price*qty FROM inventory;

Relaatiotietokannat hallitsivat myös 1990-lukua. Relaatiotietokannat ovat edelleenkin käytetyin tietokantatekniikka. Kaupallisesti tunnetuimpia relaatiotietokantoja ovat Microsoft Access, Paradox, IBM:n DB2, Oracle, Ingress, Informix, Sql Base, Interbase, Xbase, dBaseIV.

Joihinkin relaatiotietokantoihin on viime aikoina lisätty olioita tukevia piirteitä. Oliotietokannat ovat uusin idea. Ensimmäiset kaupalliset toteutukset on jo kehitetty (esim. Jasmine, Objectivity sekä Versant). Oliotietokannat ovat tulleet tukemaan oliopohjaista systeemityötä. Ongelmana oliotietokantojen yleistymiselle on ollut standardoinnin puute. Tällä hetkellä relaatiotietokantamaiset ratkaisut ovat yleisimpiä, mutta tietyissä ympäristöissä tiedostoilla on yhä käyttöä. (Sainio 2001 & Rajamäki 2004)

Microsoft Access –tietokannan luomisessa voidaan erottaa seuraavat vaiheet:
- tietokannan nimeäminen ja sijoituspaikan valinta,
- taulujen määritys sekä
- tietokannan rakenteen kuvaaminen.

Microsoft Access-tietokanta on nimetty taulujen joukko. Kaikki tietokannan taulut tallentuvat yhteen tietokantatiedostoon, jonka nimeksi tulee käyttäjän antama nimi lisättynä tarkenteella MDB. Samaan tietokantatiedostoon tallennetaan myös muut tietokantaan liittyvät tietokantaobjektit. Nämä mahdolliset tietokantaobjektit ovat kyselyt, lomakkeet (näytöt), raportit, makrot, moduulit (ohjelmat). Samalla kun Microsoft Access luo MDB-tarkenteisen tiedoston, Microsoft Access perustaa toisen tiedoston, jolla on sama nimi kuin tietokantatiedostollakin, mutta nimen tarkenne onkin LDB. Yksittäisellä työasemalla työskenneltäessä tiedostolla ei ole merkitystä, mutta tietokannan yhteiskäytössä sillä on merkitystä. Nimittäin tässä tapauksessa siihen tallennetaan tietokannan lukituksiin liittyvää tietoutta. (Sainio 2000)

Microsoft Access –ohjelmassa taulut määritellään valitsemalla vasemmanpuoleisin välilehti nimeltään Taulukot (Tables) ja painamalla oikealla olevaa Uusi (New) –painiketta. Uuden taulun määritystavassa on valittavissa Taulukkonäkymä ja Rakennenäkymä, joista valitaan Rakennenäkymä (Design View). Nyt näkyvillä on ikkuna, jossa yksittäisen taulun rakenne on kuvattavissa. Ensimmäisenä kirjoitetaan kentän nimi (sarakkeen nimi), tietotyyppi ja kuvaus. Vastaavalla tavalla tehdään seuraavalla rivillä jne. Taulu nimetään vasta aivan lopussa. Tämän vuoksi otsikossa on oletusnimi. (Sainio 2000)

Taulun perusavain määritetään taulun rakenneikkunassa. Se tapahtuu valitsemalla perusavaimen määritysrivi. Jos halutaan yhdistetty avain, painetaan CTRL-näppäintä ja

napsautetaan kutakin tarvittavaa sarakeriviä. Valitaan seuraavaksi Muokkaa Perusavain (Edit Primary key) tai painetaan avaimen kuvalla varustettua painiketta. (Sainio 2000)

Taulun viiteavain taas määritellään taulun rakenneikkunassa Indeksoitu-kohdassa (tarkoittaa hakemistoa eli indeksiä). Mikäli yhteyden tyyppi on 1:1, niin Indeksoitu-kohtaan pitää valita Kyllä (ei kaksoisarvoja). Mikäli yhteyden tyyppi on 1:n, niin Indeksoitu-kohtaan valitaan Kyllä (kaksoisarvot sallittuja) tai Ei. (Sainio 2000)

Yhteyksien määritys tapahtuu valitsemalla Työkalut Yhteydet (Tools Relationships), jonka jälkeen päästään Yhteydet-ikkunaan. Tässä Yhteydet-ikkunassa voidaan valita yhteydet eri taulujen välillä. Yhteydet perustuvat taulun perusavain ja taulun viiteavain määrityksiin. Taulua edustavia laatikoita voi siirrellä Yhteydet-ikkunassa uusiin paikkoihin tarttumalla laatikkoon taulun nimen kohdalta. Kun halutaan piirtää taulujen välille yhteys, niin tehdään seuraavalla tavalla. Viedään kohdistin yhteyteen kuuluvan perusavaimen kohdalle (tummennettu). Sen jälkeen hiiren vasen näppäin alhaalla vedetään yhteys vastaavaan viiteavaimeen. Tällä tavoin luodaan halutut yhteydet. Tuplaklikkaamalla hiirellä yhteysviivaa, joka on laatikoiden välillä (Yhteydet-ikkunassa) päästään määrittelemään yhteystyyppiin ja viite-eheyden valvontaan liittyviä asioita. (Sainio 2000)

Relaatiotietokannoissa tiedot talletetaan tauluina, joiden väliset yhteyden muodostetaan tauluissa olevien tietojen avulla. Taulut eivät siis ole toisistaan irrallaan, vaan niiden välillä on yhteyksiä (relationships). Kun tiedot jaetaan tauluihin, niin pyritään välttämään saman tiedon toistamista eli päällekkäisyyttä (redundanssia). Tällöin kukin tieto esiintyy vain yhdessä paikassa, jolloin tietojen päivittäminen on helpompaa. Tiedon jakamiseen eri tauluihin on olemassa ns. normalisointisäännöt. Normalisointisääntöjä on olemassa neljä kappaletta. Kun yksi sääntö on läpikäyty, niin saadaan normaalimuoto. Eli ensimmäisen säännön läpikäynnin jälkeen tietokanta on ensimmäisessä normaalimuodossa. Tietokanta on siis tietyssä normaalimuodossa, kun täyttää tietyt matemaattiset säännöt. Normalisoinnit voidaan esittää ilman matematiikkaa esimerkkien avulla.

Ensimmäinen normaalimuoto:

- taulukossa (taulun sarakkeissa) ei saa olla tietotoistoja
- taulukossa (taulun sarakkeissa) ei saa olla tietokoosteita

Toinen normaalimuoto:

- taulukko on toisessa normaalimuodossa, jos taulukon kaikki tiedot (ominaisuudet eli taulun sarakkeet) ovat täysin riippuvaisia taulun koko avaimesta

Kolmas normaalimuoto:

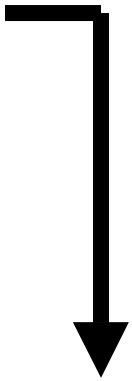
- taulussa ei saa olla transitiivisia riippuvuuksia. Eli perusavaimen ulkopuolisten sarakkeiden tulee olla keskenään riippumattomia

Neljäs normaalimuoto:

- joillakin ominaisuuksilla saattaa olla monia arvoja samanaikaisesti. Tällaisia ominaisuuksia on esim. kielitaito, tutkinto ja harrastukset. Esimerkkinä kielitaito:

henk	
hetu	nimi
11	Jussi
12	Maija

henktaulu		
hetu	nimi	kielitaito
11	Jussi	englanti
11	Jussi	saksa
11	Jussi	ruotsi
12	Maija	saksa
12	Maija	englanti



kielitaito	
hetu	kieli
11	englanti
11	saksa
11	ruotsi
12	saksa
12	englanti

Kuva 3.5: Kielitaito (Rajamäki 2004).

Taulut koostuvat riveistä (row) ja sarakkeista (column). Sarakkeita kutsutaan kentiksi (field) ja rivejä tietueiksi (record). Ylimmällä rivillä on kenttien nimet. Tietokantateoriassa nimityksen kenttä sijaan käytetään usein nimitystä attribuutti (attribute) ja nimityksen tietue sijaan termiä monikko (tuple).

Relaatiotietokannan tapauksessa kahden taulun välinen yhteys on aina toteutettava perusavain - viiteavain -yhdistelmällä. Kun tietokannan tietueita käsitellään, niin on pystyttävä tunnistamaan tietty tietue yksikäsitteisesti. Tämä tapahtuu perusavaimen (primary key) avulla. Avain muodostuu yhdestä tai useammasta sarakkeesta. Avainehdokkaana voi olla sarake tai sarak-

keiden yhdistelmä. Avaimen arvo ei voi olla tyhjä (Null). Taulut yhdistetään toisiinsa niin, että toisen taulun kenttä viittaa toisen taulun perusavaimeen. Viittaavaa kenttää sanotaan viiteavaimeksi (foreign key). Viiteavaimen ei tarvitse olla oman taulunsa perusavain.

Yhteystyypin ollessa 1:1, viiteavain voi olla kummassa taulussa tahansa. Yhteystyypin ollessa 1:n, viiteavain on “n-pään” taulussa. Yhteystyyppi n:m taas toteutetaan aputaulun avulla, johon kummastakin alkuperäisestä taulusta on 1:n yhteys. Tämä tarkoittaa sitä, että aputaulussa on viittaukset alkuperäisiin tauluihin. Yhteys voi olla joko pakollinen tai sitten ehdollinen. Tämä pakollisuus tarkoittaa sitä, että viiteavaimen arvo on löydyttävä vastaavan perusavaimen arvoista. Ehdollisuus taas lisää edelliseen mahdollisuuden käyttää myös tyhjäarvoa. (Sainio 2001)

3.2.2 UML ja OMT

Oliomallinnusmetodit, kuten Rational-nimisen firman UML (Universal Modelling Language) sekä Rumbaughin ja kumppaneiden OMT (Object Modeling Technique) kehitettiin pääasiallisesti ohjelmistokehitykseen.

OMT koostuu seuraavista vaiheista: analyysi (Analysis), atk-tekninen suunnittelu (System Design), oliosuunnittelu (Object Design) ja toteutus (Implementation). Analyysivaiheen malli määrittää, mitä systeemin pitää tehdä. Mallin oliot ovat kohdealueen käsitteitä, mutta eivät tietokonetoteutukseen liittyviä. Analyysivaiheessa on seuraavat tehtävät (Systeemityöyhdistys Sytyke ry 1992):

- Ongelma-alueen vaatimukset (mitä systeemiltä halutaan, systeemin yhteydet, oletukset ja suoritusvaatimukset).

- *Olioiden mallintaminen (Object Model*). Olioluokat määritetään, lisätään luokkien väliset yhteydet sekä olioiden ja linkkien ominaisuudet. Olioluokat organisoidaan ja pyritään yksinkertaistamaan käyttämällä periytymistä. Hakupolut testataan käyttämällä tapahtumien käsikirjoituksia (scenario). Toisiaan seuraavien tapahtumien joukkoa voidaan nimittää käsikirjoitukseksi. Luokat ryhmitellään moduuleiksi.

- *Dynaaminen mallintaminen (Dynamic Model).* Vuorovaikutustilanteista laaditaan käsikirjoituksia. Olioiden väliset tapahtumat määritetään ja jokaisesta käsikirjoituksesta laaditaan tapahtumapolku. Tapahtumavirran laadinta systeemistä. Tilakaavion suunnittelu jokaiselle luokalle, jossa on merkittävää dynaamista käyttäytymistä.

- *Toiminnallinen mallintaminen (Functional Model).* Kohdealueelle sisään tulevan ja sieltä ulos lähtevän tiedon määrittäminen. Tietovirtakaaviot otetaan tarvittaessa käyttöön määrittämään toiminnallista riippuvuutta. Kunkin toiminnallisen osan tehtävän kuvaaminen. Rajoituksien ja optimoitavien asioiden määrittäminen.

- Palveluiden lisääminen. Toiminnallisen mallintamisen aikana löydetyt tärkeimmät palvelut lisätään oliomalliin. Luokkien yhteyksien, ominaisuuksien ja palveluiden johdonmukaisuuden ja täydellisyyden tarkistaminen. Kolmen mallin vertaaminen asetettuihin vaatimuksiin ja testataan malli käyttämällä käsikirjoituksia.

Yleisarkkitehtuurista tehdään päätöksiä atk-teknisessä suunnittelussa. Osasysteemeihin jako, joka pohjautuu määrittelyn malliin ja ehdotettuun yleisrakenteeseen. Seuraavat tehtävät ovat atk-teknisessä suunnittelussa (Systeemityöyhdistys Sytyke ry 1992):

- Osasysteemeihin jako
- Selvitetään samanaikaisuudet
- Jaetaan suorittimille osasysteemit
- Kokonaisresurssien hallinta
- Toteutuksen valinta ohjelmiston valvontaan
- Käsitellään rajaehdot
- Suuntaviivat yleisrakenteelle.

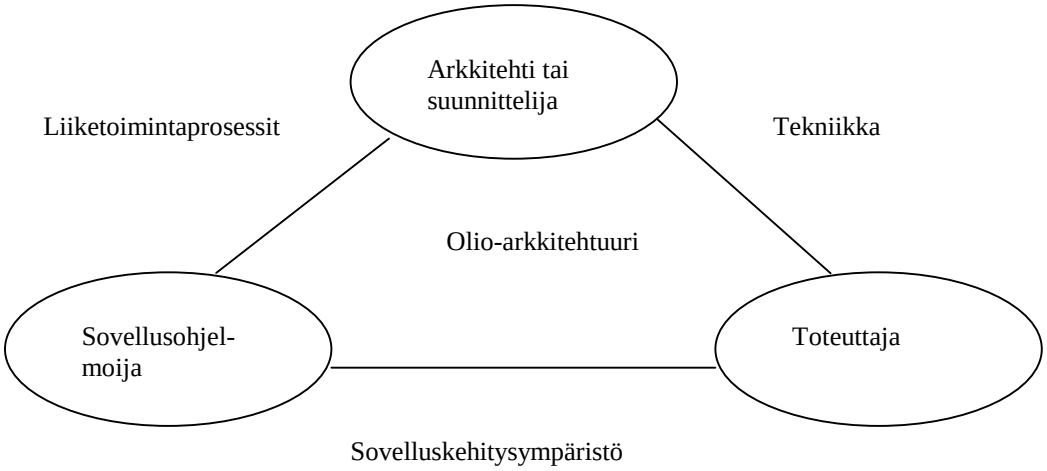
Suunnittelumalli rakennetaan määrittelymallin pohjalta oliosuunnitteluvaiheessa sellaiseksi, että se sisältää toteutuksen vaatimat yksityiskohdat. Luokkien toteutuksessa tarvitaan tietorakenteita ja algoritmeja. Mukaan otetaan sekä kohdealueen oliot että tietokoneläheiset oliot, jotka molemmat kuvataan käyttämällä samoja oliolähestymistavan käsitteitä ja

merkintätapoja. Seuraavat tehtävät ovat oliosuunnittelussa (Systeemityöyhdistys Sytyke ry 1992):

- Yhdistetään mallit
- Suunnitellaan algoritmit
- Optimoidaan suunnitelma
- Toteutetaan valvonta
- Tarkistetaan periytyminen
- Suunnitellaan yhteydet
- Esitetään oliot
- Fyysinen paketointi
- Kuvataan tekniset päätökset.

Olioluokat ja niiden riippuvuudet käännetään valitun ohjelmointikielen, tietokannan ja laitteiston mukaiseksi toteutusvaiheessa. Valittu ohjelmointikieli vaikuttaa jonkin verran suunnittelupäätöksiin, mutta suunnittelu ei saisi koskaan riippua kielen yksityiskohdista. (Systeemityöyhdistys Sytyke ry 1992)

Oliosysteemityöprosessin osapuolia ovat tietojärjestelmäarkkitehdit, luokkien ja oliosysteemien suunnittelijat, luokkien käyttäjät eli sovellusohjelmoijat ja luokkien toteuttajat (ks. kuva).



Kuva 3.6: Systeemityön osapuolet (Systeemityöyhdistys Sytyke ry 1992).

Tämä kuvassa oleva työnjako ei perustu samaan hierakkiseen tehtäväjakoon, johon perinteisessä systeemityössä on totuttu. “Työnjako tarjoaa systeemityön osapuolille mahdollisuuden siirtyä taipumusten mukaan tehtävästä toiseen ja mahdollistaa entistä paremmin liiketoiminnan asiantuntijoiden osallistumisen systeemityöprosessiin joko suunnittelijan tai luokan käyttäjän ominaisuudessa“ (Systeemityöyhdistys Sytyke ry 1992).

Oliolähestymistapaan perustuvassa systeemityöstä kaikki nykyisen systeemityön osapuolet löytävät kyllä paikkansa. “Se vaatii kuitenkin atk-ammattilaisilta entistä laajempaa näkemystä liiketoiminnasta ja sen tietojärjestelmien yhteistyöstä” (Systeemityöyhdistys Sytyke ry 1992).

Suuri osa ohjelmistokehityksestä on tietokannan suunnittelulla, jota ohjelmamoduulit käyttävät. Keskeinen osa näitä metodeja ovat luokkakaaviot (luokka vastaa suunnilleen kohdetyyppiä, mutta sille on määritelty operaatiot). Luokat ja aliluokat ovat esillä etenkin oliomallinnuksessa, mutta myös ER-mallinnuksessa voidaan muodostaa luokkia ja aliluokkia.

3.2.3 MR, ER, EER, NIAM, SSADM ja MERISE

On olemassa systemaattisia selvityksiä moniyhteyden (MR, multiple relationship) käsitteestä. Hyvin tunnettuja käsitteellisiä malleja on käytetty perustana MR-mallinnuksessa. Tunnetuin käsitekaaviokieli on ER-menetelmä (Entity-Relationship -method), jonka kehittäjä on Peter P. Chen. ER-menetelmä on graafinen kuvaustapa. Se sopii hyvin tietokannan suunnittelijan ja käyttäjän yhteistyössä suorittamaan reaalimaailman mallintamiseen. Muita menetelmiä ovat EER-menetelmä (Extended Entity-Relationship -method), NIAM (Nijssen ja Halpin 1989), SSADM (SSADM 1990) ja MERISE (Rochfeld 1987). SSADM on kansallinen metodi järjestelmien kehittämiseen (Iso-Britannian ja Pohjois-Irlannin) Yhdistyneessä kuningaskunnassa. Vastaavasti MERISE on kansallinen metodi järjestelmien kehittämiseen Ranskassa. (Flynn & al. 1995, Rajamäki 2004)

Kaksi tavallisesti esiintyvää rajoitetyyppiä, jotka koskevat entiteettiä suhteessa (yhteydessä), ovat kardinaliteettirajoite ja osallistumisrajoite. Molempien rajoitteiden täytyy olla totta mihin aikaan tahansa, kun kyseessä on entiteetin elinaika. Kardinaliteettirajoite määrittää tapauksien lukumäärän (joko yksi tai monta) entiteetille, mikä saattaa liittyä toisen entiteetin yhteen (joko yksi tai monta) tapaukseen. Kaksoisyhteys annetussa suunnassa käsittää neljä mahdollisuutta

kardinaliteettisuhteessa: yksi yhteen, yksi moneen, moni yhteen ja moni moneen. Osallistumisrajoite määrittää pitäisikö entiteetin tapauksien osallistua (pakollinen osallistuminen) tai vain olla tilassa saattaa osallistua (valinnainen osallistuminen). (Flynn & al. 1995)

ER (entity relationship) -malli koostuu **kohteista** eli yksilötyypeistä (entity), **ominaisuuksista** eli ominaisuustyypeistä (attribute) sekä **suhteista** eli yhteystyypeistä (relationship). Tietokannan suunnittelussa ja yleisesti järjestelmän suunnittelussa malli on varsin yleinen. Suorakaide kuvaa kohdetta, ellipsi ominaisuutta sekä vinoneliö suhdetta. (Rajamäki 2004)

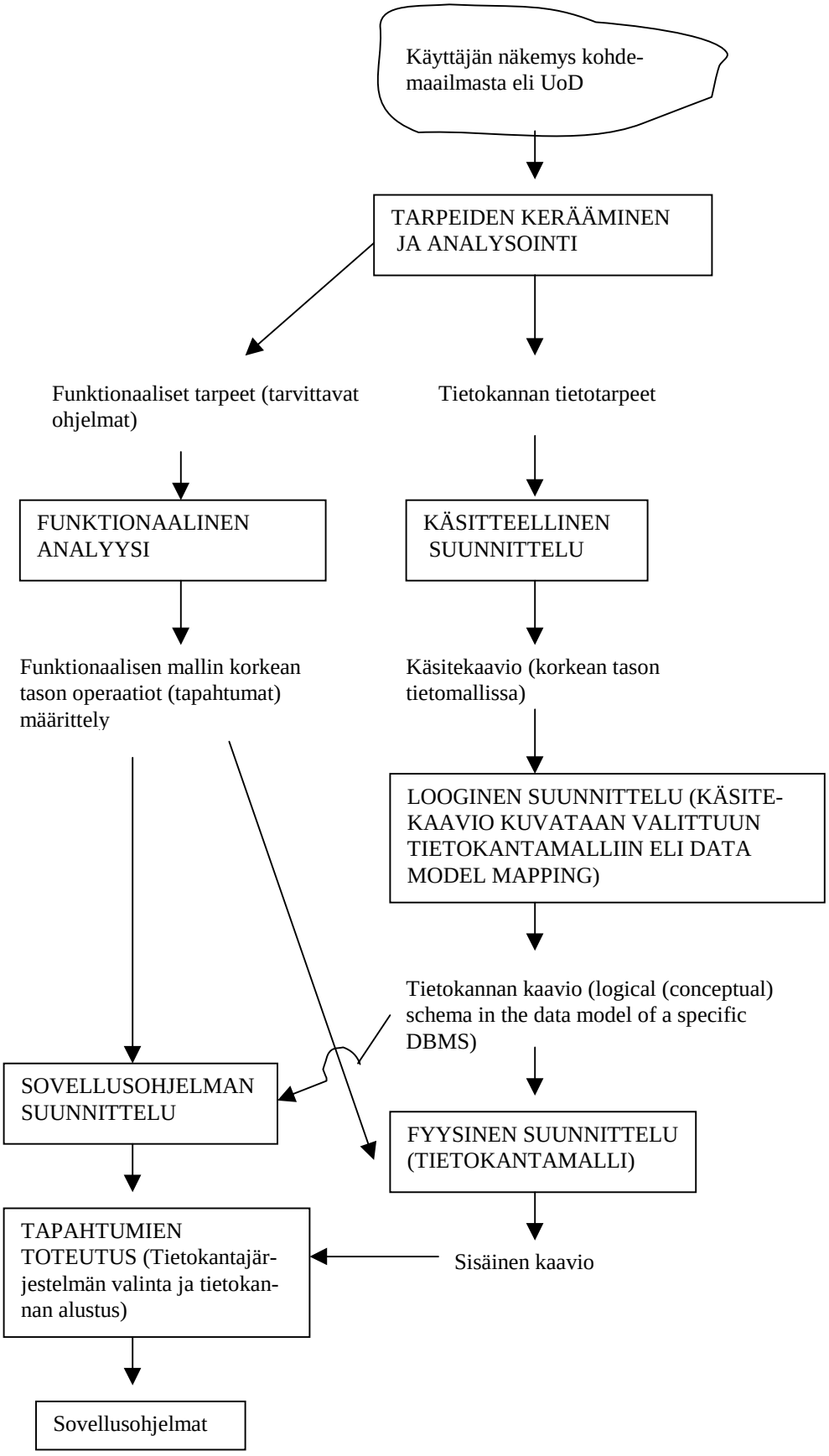
Kohde on kohdealueen olio, jolla on muista riippumaton olemassaolo. Kohteella on olemassa tiettyjä ominaisuuksia, jotka kuvailevat oliota. Nämä ominaisuudet erottavat kohteen muista kohteista. Esimerkiksi työntekijä-kohde voidaan kuvailla seuraavilla ominaisuuksilla: työntekijän nimi, ikä, osoite, palkka ja työ. Kohteen ominaisuuksilla on arvo kullekin ominaisuudelleen. Ominaisuuksilla voi olla yksi arvo; esimerkiksi Henkilö-kohteella on yksi arvo ominaisuudelle ikä. Toisissa tapauksissa ominaisuuden arvot muodostavat joukon; esimerkiksi oppiarvot-kohteelle työntekijän nimi. Moniarvoisuus tarkoittaa tällaista tapausta. Moniarvoiselle ominaisuudelle voi olla ylä- sekä alarajat.

ER-mallin mukaan tietokannan suunnittelu etenee seuraavasti. Ensimmäinen vaihe on **tarpeiden kerääminen ja analysointi** (requirements collection and analysis), jolloin tietokannan suunnittelijat haastattelevat tulevia tietokannan käyttäjiä. Voidaan sanoa, että yritetään saada selville **käyttäjän näkemys kohdemaailmasta** (UoD). Haastattelun tuloksena saadaan selville **tietokannan tietotarpeet** (database requirements) ja rinnakkain tietojärjestelmän **funktionaaliset tarpeet** (functional requirements) eli tarvittavat ohjelmat. Funktionaalisen analyysin tuloksena ovat dokumentit, joissa kuvataan funktionaaliset tarpeet (esim. tietovuokaavioina, skenaarioina jne.) eli funktionaalisen mallin korkean tason operaatiot (tapahtumat) määrittely. Tietokanta-analyysin tuloksena on **käsitekaavio**. (Viitanen 2002)

Käsitekaavio esitetään korkean tason käsitekielellä (esim. ER), joka ei sisällä toteutuksen yksityiskohtia. Tätä käsitekaaviota, voidaan käyttää esittämään analyysin tulos tuleville

käyttäjille. Tällöin käyttäjät voivat parantaa käsitekaaviota. Käsitekaaviota voidaan käyttää määrittelemään funktionaalisen mallin korkean tason operaatioita. (Viitanen 2002)

Käsitekaavion ja sen analyysin jälkeinen vaihe on **looginen suunnittelu**, missä käsitekaavio kuvataan valittuun tietokantamalliin eli data model mapping (relaatiomalli, oliomalli, tekstitietokanta tms.). Tuloksena on **tietokannan kaavio** (logical (conceptual) schema in the data model of a specific DBMS), jota käytetään vaiheessa **sovellusohjelmasuunnittelu**. Tästä seuraa fyysisen suunnittelun jälkeen **tietokantamalli** (internal schema), jota käytetään, kun ollaan vaiheessa **tapahtumatoteutus**. Tässä vaiheessa tietokantamalli toteutetaan valitulla tietokantajärjestelmällä ja tietokanta alustetaan perustiedolla. Tarpeiden analysoinnin eli tarveanalyysin yhteydessä on saatu selville mahdollisesti luontevimmat tiedostotyypit, joilla tietokanta kannattaa toteuttaa. Lisäksi on saatu selville tiedostojen koot (levytilan tarve) sekä tapahtumien määrä aikayksikköä kohti (prosessointitarve). Lopputuloksena tästä tietokannan suunnittelusta seuraavat **sovellusohjelmat**. Käyttäjien näkemykset voidaan toteuttaa osatietokantoina, niin sanottuina näkyminä. Tällöin käyttäjä näkee tietokannasta (joka on kohdemaailman kuva) vain häntä koskevan osan. (Viitanen 2002)



Kuva 3.7: Tietokannan suunnittelu. (Viitanen 1992)

Tietokannan suunnittelussa käsitekaavion laatimisen jälkeen seuraava askel on käsitekaavion kuvaaminen tietokannan kaavioksi. Tietokantamallin valinta tulee tässä vaiheessa eteen, koska tietokannan toteutus vaatii sen valinnan. Todennäköisesti pitää valita myös käytettävä tietokantajärjestelmä.

ER-kaavio kuvataan relaatiokaavioksi toteuttamalla kaavion elementit relaatioina, attribuutteina ja vierasavaimina. Koska ER-malli sallii, että kohteella on useampia tunnisteita, tulee niistä yksi valita avaimeksi.

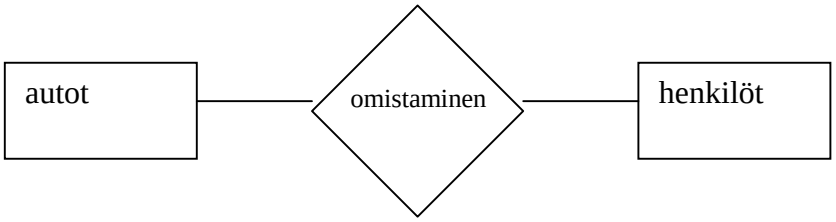
Säännöt, joilla ER-kaaviosta saadaan relaatio ovat:

- **yksilötyypistä** (kohde) tulee relaatio eli taulu ja kaikista yksilötyypin **ominaisuustyypeistä** (ominaisuus) tulee relaation sarakkeita.
- **yhteystyypistä** (suhde) tulee myös relaatio, jonka tietoja yhteystyypin omat ominaisuudet ja yhteysviivojen päissä olevat yksilöiden avaimet ovat.

Relaatiotietokannan mallia (schema) ei ole pakko kuvata taulukoina, vaan lyhyemmin seuraavasti relaatiokaavoina.

HENK(<u>hetu</u>, sukunimi, etunimi, lähiosoite, postinro, postitoimipaikka)
AUTO(<u>rekno</u>, merkki, malli, kottovuosi)
OMISTAJA(<u>hetu</u>, <u>rekno</u>)

Relaatiokaavat ER-kaaviona olisi:



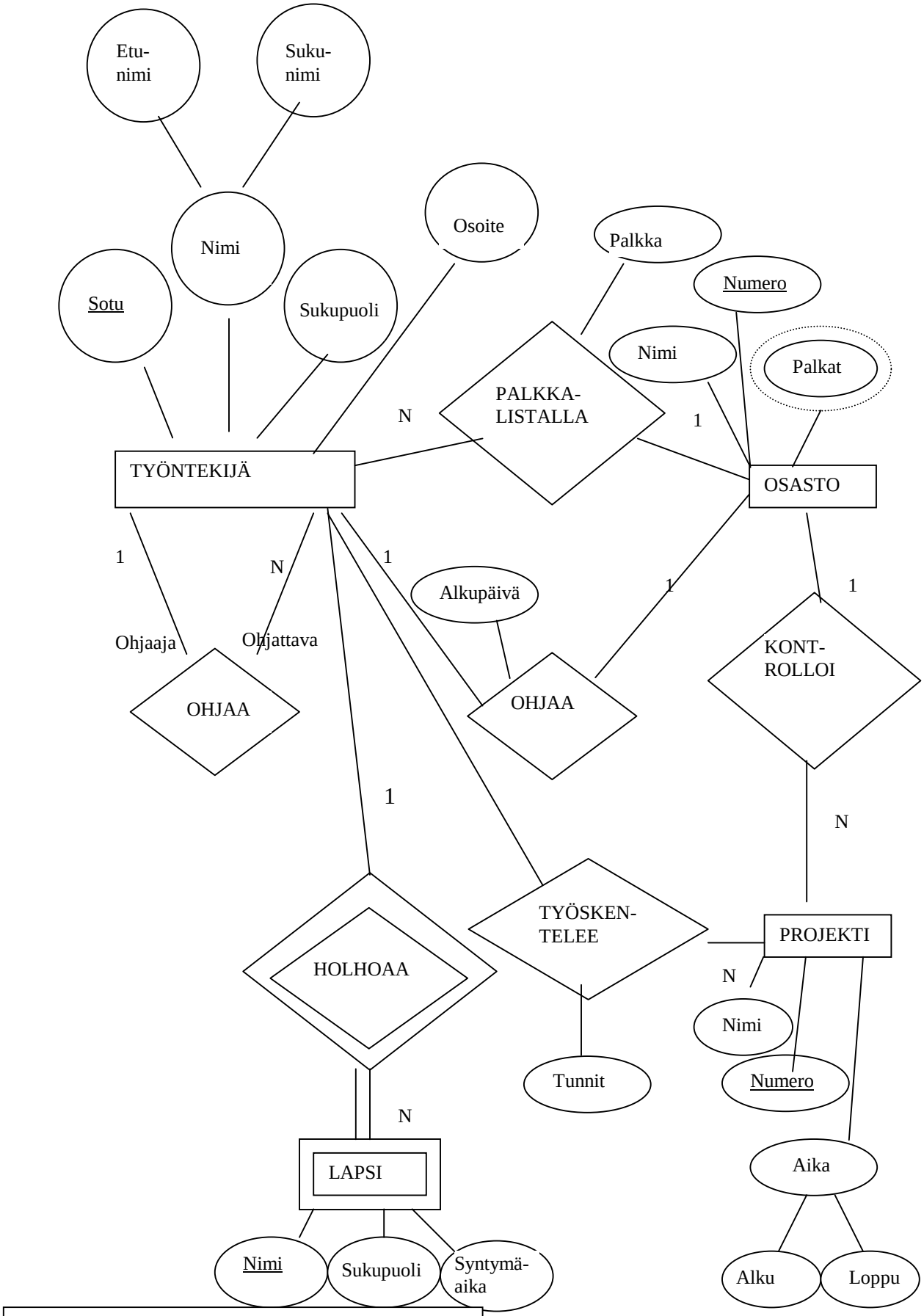
Kuva 3.8: ER-kaavio (Rajamäki 2004).

Yksittäinen relaatiokaava (taulu HENK) voidaan kuvata myös yksityiskohtaisemmin:

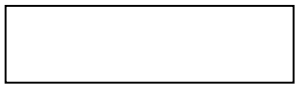
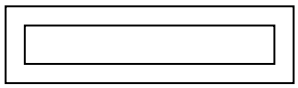
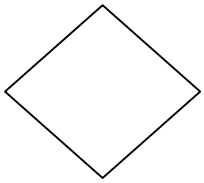
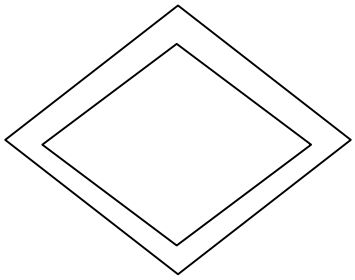
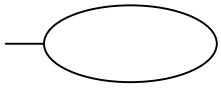
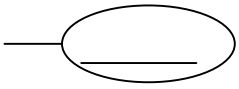
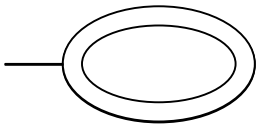
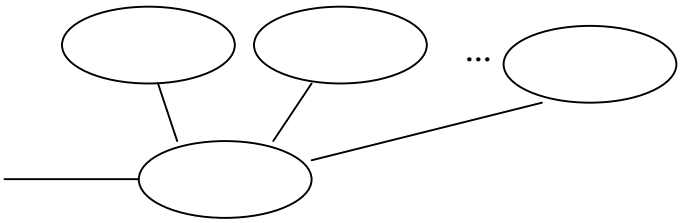
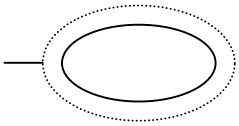
HENK (

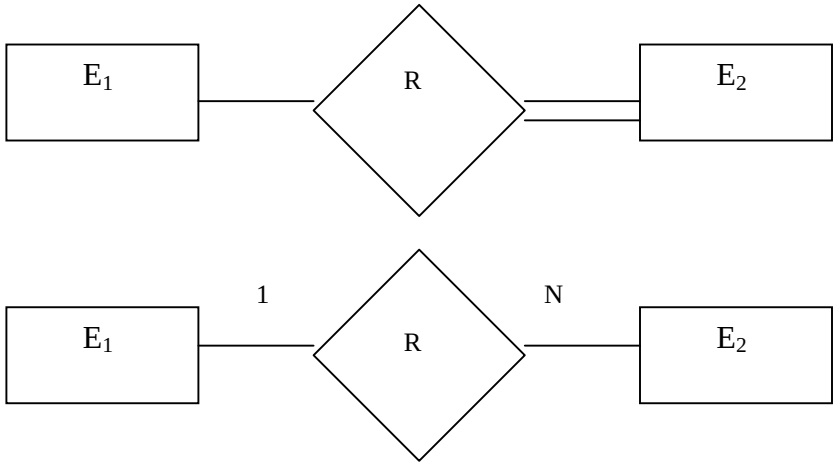
hetu, string
sukunimi, string
etunimi, string
lähiosoite, string
postinro, integer
postitoimipaikka, integer)

Vaihe 1: kohde. Jokaista (vahvaa) kohdetyyppiä *E* vastaten luodaan relaatiokaavioon relaatio *R*, joka sisältää kaikki *E*:n yksinkertaiset ominaisuudet. Rakenteisista ominaisuuksista mukaan tulee ainoastaan yksinkertaiset ominaisuudet. Yksi tunnisteominaisuuksista valitaan pääavaimeksi. Jos valittu tunniste on rakenteinen, pääavaimeksi tulevat yksinkertaiset ominaisuudet. Jos tarkastellaan käsitekaaviota kuvasta 3.9, tämä askel koskee kohteita Työntekijä, Osasto ja Projekti.



Kuva 3.9: Liike. (Viitanen 2002)

SYMBOLI	TARKOITUS
	Kohdetyyppi (entity type)
	Heikko kohdetyyppi (weak entity type)
	Suhdetyyppi (relationship type)
	Tunnistava suhdetyyppi (identifying relationship type)
	Ominaisuus (attribute)
	Avainattribuutti (key attribute)
	Moniarvoinen attribuutti (multivalued attribute)
	Koottu ominaisuus (composite attribute)
	Johdettu attribuutti (derived attribute)



Täysi osallistuminen E_2:lta suhteessa R (total participation of E_2 in R)

Kardinaliteettisuhde 1:N E_1:n ja E_2:n välillä (cardinality ratio 1:N for E_1: E_2 in R)

Kohteita vastaamaan luodaan relaatiot Tyontekija, Osasto ja Projekti. Attribuuteiksi valitaan

Tyontekija:

- Sotu (avain)
- Etunimi
- Sukunimi
- Sukupuoli

Osasto:

- Numero (avain)
- Nimi

Projekti:

- Numero (avain)
- Nimi
- Alkuaika
- Loppuaika

Käsitteistä puhuttaessa käytetään termiä ominaisuus, mutta tietokannan toteutuksessa käytetään termiä attribuutti. Tässä tulee huomata että **Osaston Palkat** –ominaisuus on moniarvoinen, eikä sitä tämän vuoksi oteta vielä mukaan

Vaihe 2: heikko kohde. Jokaiselle heikolle kohteelle *W*, jolla on omistaja *E* luodaan relaatio *R*, jossa on kaikki yksinkertaiset ominaisuudet (tai rakenteisten ominaisuuksien yksinkertaiset

komponentit). Lisäksi mukaan otetaan vierasavain attribuutteina, *R*:ään *E*:tä vastaavan relaation pääavainattribuutit. Vierasavaimet vastaavat kohteen *W* tunnistavaa suhdetyyppiä. *R*:n pääavaimeksi tulee vierasavaimen ja heikon kohdetyypin osittainen tunniste. On tärkeää huomata, että heikko kohdetyyppi voi toimia toisen heikon kohdetyypin omistajana, joten relaatioiden teko pitää aloittaa polun alusta.

Esimerkkikaaviossa (kuva 3.9) heikko kohdetyyppi on Lapsi, sen omistajana on Työntekijä ja tunnistavana suhdetyyppinä on Holhoaa. Lapsen attribuutit ovat Nimi, Sukupuoli ja Syntyma_aika. Työntekijä kohdetyyppiä vastaa Tyontekija-relaatio, jonka avain on Sotu. Niinpä Sotu tulee myös Lapsi-relaatioon; se on vierasavain joka osoittaa Tyontekija-relaatioon. Lapsen avaimeksi tulee <VanhSotu, Nimi> kombinaatio (”Sotu” olisi huono nimi, koska se voitaisiin tulkita lapsen sosiaaliturvatunnukseksi). Kaikkiaan attribuutteina on täten VanhSotu, Nimi, Sukupuoli, Syntyma_aika.

On tavallista valita Cascade tyyppinen eheysrajoitus vierasavaimelle. Tämä tarkoittaa sitä, että kun Työntekijä hävitetään, kaikki hänen Lapsensa hävitetään.

Vaihe 3: 1 : 1 **suhdetyyppi.**

Vaihe 4: 1 : *n* **suhdetyyppi**

Vaihe 5: *m* : *n* **suhdetyyppi**

Vaihe 6: moniarvoiset ominaisuudet

Vaihe 7: *n*-**paikkaiset suhteet**

Yhteenveto vaiheista (1 - 7). Esimerkkitietokanta. Esimerkkinä käytetystä ER-kaaviosta (kuvasta 10 syntyi seuraavanlainen relaatiotietokanta:

- Tyontekija(**Sotu**, Etunimi, Sukunimi, Sukupuoli, OsastonNumero*fk*, Palkka, OhjaajanSotu*fk*)
- Osasto(**Numero**, Nimi, OsastoJohtaja*fk*, JohtajaAlkoi)

- Projekti(**Numero**, Nimi, Alkuaika, Loppuaika)
- Lapsi(**VanhSotu**^{fk}, **Nimi**, Sukupuoli, Syntyma_aika)
- OsastoProjekti(**ProjektiNo**^{fk}, OsastoNo^{fk})
- ProjektiTunnit(**Tsotu**^{fk}, **Pnumero**^{fk}, Tunnit)

Vierasavaimet on merkitty ”attribuutti^{fk}” –merkinnällä. Eheysrajoitukset voidaan esittää SQL:n FOREIGN KEY –lauseen lisämääreillä.

- OsastoNumero: viittaa Osastoon. Eheysrajoitukset ON DELETE SET NULL, ON UPDATE CASCADE
- OhjaajanSotu viittaa Tyontekijaan. Eheysrajoitukset ON DELETE SET NULL, ON UPDATE CASCADE
- OsastoJohtaja viittaa Tyontekijaan. Eheysrajoitukset ON DELETE SET NULL, ON UPDATE CASCADE
- VanhSotu viittaa Tyontekijaan. Eheysrajoitukset ON DELETE CASCADE, ON UPDATE CASCADE
- ProjektiNo viittaa Projektiin. Eheysrajoitukset ON DELETE SET CASCADE, ON UPDATE CASCADE
- OsastoNo viittaa Osastoon. Eheysrajoitukset ON DELETE SET CASCADE, ON UPDATE CASCADE
- TSotu viittaa Tyontekijaan. Eheysrajoitukset ON DELETE SET CASCADE, ON UPDATE CASCADE
- PNumero viittaa Projektiin. Eheysrajoitukset ON DELETE SET CASCADE, ON UPDATE CASCADE

Suhteiden toteutus: Relaatiomallista tulee huomata, että suhdetyyppejä ei esitetä eksplisiittisesti, vaan ne esitetään kahtena attribuuttina *A* ja *B*, joista toinen on pääavain ja toinen vierasavain kahdessa eri relaatiossa *S* ja *T*.

Vierasavaimia käytetään ER-kaavion läpikäyntiin. Jos haluamme esimerkiksi saada selville henkilöt, jotka työskentelevät osaston kontrolloimissa projekteissa, mutta eivät ole osaston palkkalistoilla, tulee kyseeseen kohdetyypit Osasto, Projekti ja Työntekijä.

Suhdetyyppeinä ovat **Kontrolloi**, **Työskentelee** ja **Palkkalistalla**. **Kontrolloi**-suhdetyypin toteutti OsastoProjekti-relaatio ja sen vierasavaimet Osasto ja Projekti –relaatioihin. **Työskentelee**-suhdetyypin toteutti relaatio Tyoskentelee ja sen vierasavaimet Tyontekija ja Projekti –relaatioihin. **Palkkalistalla**-suhdetyypin toteutti Tyontekija-relaation vierasavain Osasto-relaatioon.

4 JOHTOPÄÄTÖKSET JA KRITIIKKIÄ

Tutkielmassa on kuvattu johdantoluvussa informaatiojärjestelmät ja tutkimusongelma. Sen jälkeen käsiteltiin ohjelmistosuunnittelua omana lukunaan ja informaatiojärjestelmien kuvaamista omana lukunaan (mm. käsitteellistä mallintamista). Työssä luotiin myös katsaus tietohallintoon ja sen erilaisiin tehtäviin. Jos halutaan kehittää aina vain nopeampia työprosesseja, niin niiden läpiviemiseen tarvitaan myös uusia näkökulmia ja vasta tämän jälkeen voidaan hyödyntää olemassa olevia tekniikoita. Minun mielestäni nopeutta tulee myös lisää jos kehitystä tapahtuu ohjelmistosuunnittelussa ja informaatiojärjestelmien kuvaamisessa. Mallit ja mallintaminen perustuvat osin standardeihin ja jotkut ovat vasta matkalla standardeiksi. Olen sitä mieltä, että esim. miellekartan laatiminen on hyvä lähtökohta varsinaiselle mallintamiselle.

Tietokonealan markkinoilla vallitsee odottava tunnelma, mitkä mallit ja mallinnustekniikat saavat valta-aseman jatkossa. Yleisesti voidaan määritellä kahdenlaisia malleja, staattisia malleja sekä dynaamisia malleja. Jos hyväksytään ajatus, että tietojärjestelmä on malli reaalimaailmasta niin tämä malli on tyypiltään dynaaminen. Staattinen malli ei voi olla kyseessä, koska se pysyy samanlaisena kuvana kohteesta valmistumishetkestään lähtien.

Olen sitä mieltä, että ensimmäisissä malleissa tavoitteena on selvittää kehittämiskohteena olevan liiketoiminnan tai vastaavan nykytila sekä kehittämistarpeet. Malleja voidaan sitten luoda kehitystyön aikana lisää ja voidaan myös laatia malleja tulevaisuudesta. Yksi mahdollinen keino kuvata yrityksen tilaa on laatia SWOT-analyysi. Yritysanalyysi (nykyisyys) eli vahvuudet ja heikkoudet sekä ympäristöanalyysi (tulevaisuus) eli mahdollisuudet ja uhat.

Kun tietokonejärjestelmää suunnitellaan niin lähdetään liikkeelle tietojenkäsittelyn kokonaissuunnittelusta. Tämä kokonaissuunnittelu kohdistuu koko organisaatioon tai toimeksiannossa rajattuun osaan. Kehittämisen painopistealueet määritetään, selvitetään toiminnan ja sitä tukevan tietojenkäsittelyn kehittämistarpeet. Pitkän aikavälin kehittämistyö jaetaan osiin: hankkeiksi ja projekteiksi. Hanke tässä tarkoittaa kehittämiskokonaisuutta, jossa tavoitteena olevaan lopputulokseen kohdistetaan kannattavuuden tarkastelu. Projekti taas tarkoittaa kertaluontoista ajallisesti rajattua työsuoritusta, jolle on kiinnitetty resurssipuitteet ja

tehtävä. “Toiminta ja sen kehittämistavoitteet kuvataan melko karkealla tasolla lähtökohdaksi tietojärjestelmäkohtaiselle ja huomattavasti tarkemmalle kohteen käsittelylle ja kuvaamiselle.” (Virkki & Somermeri 1993)

Systeemityön vaiheistamisen tavoitteena on kaikilla karkeustasoilla, että työstä syntyisi tuloksen kannalta selkeä lopputuloksen osa tai välitulos, jotta työ etenisi määrätietoisesti kohti tavoitteena olevaa tietojärjestelmää. Vaiheet ovat Virkin & Somermeren (1993) mukaan määritys, suunnittelu, toteutus ja käyttöönotto. Mielestäni paremmat vaiheet olisivat määrittely, suunnittelu, toteutus ja testaus (ks. kuva 2.4, 2.5 ja 2.6). Vaiheiden sisällä on sitten erilaisia tehtäviä. Tehtävät on usein vielä esitetty kronologisessa, mielekkäässä suoritusjärjestyksessä. Tehtävätasolla eri rakentamismallien sisällöt voivat poiketa toisistaan. (Virkki & Somermeri 1993)

5 YHTEENVETO

Tutkimusalueella (informaatiojärjestelmien kuvaaminen tietokonejärjestelmien suunnittelussa) on tehty kehitystyötä mm. pitämällä konferensseja. Tällaisia konferensseja ovat olleet esim. IFIP (The International Federation for Information Processing) international working conference –konferenssi. Lisäksi alalla on julkaistu artikkeleita kirjasarjassa nimeltään Frontiers in Artificial Intelligence and Applications. Tässä tutkielmassa on esitelty informaatiojärjestelmien kuvaamista tietokonejärjestelmien suunnittelussa. Se mitä pitäisi vielä tehdä on kehittää käsitteellistä mallintamista, muita mallintamislähestymistapoja ja tiedon esittämistekniikoita. Nämä edellä mainitut ovat jo nykyään laajalti käytettyjä kommunikaatiovälineenä suunnittelijoiden ja käyttäjien välillä monimutkaisten informaatiojärjestelmien suunnittelussa.

Jatkossa kiinnostavia tutkimuskohteita voisivat olla EER-menetelmä (Extended Entity-Relationship -method), NIAM (Nijssen ja Halpin 1989), SSADM (SSADM 1990) ja MERISE (Rochfeld 1987). Näiden käsittelyn yhteydessä voitaisiin pyrkiä antamaan tietoa oikeanlaisen lähestymistavan valintaan.

VIITELUETTELO

Painetut lähteet:

Avison, D. E. & Nandhakumar, J.: The discipline of information systems: Let many flowers bloom!. *Information System Concepts – Towards a consolidation of views* (Ed. Falkenberg, E. D., & al.), Chapman & Hall, London, 1995, 1-17.

Boman, M., Bubenko, J., Johannesson, P. & Wangler, B.: *Conceptual Modelling.* Prentice-Hall, Inc., Upper Saddle River, NJ, 1997.

Benson, Y., Koskinen, J., Peltoniemi, J. & Stenlund, H.: *Ymmärrä oikein systeemityö*. Tietokruunu Oy, Gummerus Oy:n kirjapaino, Jyväskylä, 1986.

Checkland, P. B.: *Systems Thinking, Systems Practice*. John Wiley & Sons, Chichester, 1981.

DeMarco, T.: *Structured Analysis and Systems Specification*. Prentice Hall, New York, 1979.

Flynn, D. J., Knight, D. R. & Laender, A. H. F.: Multiple relationships: An analysis of their semantics and their modelling. *Information System Concepts – Towards a consolidation of views* (Ed. Falkenberg, E. D., & al.), Chapman & Hall, London, 1995, 36-51.

Gane, C. & Sarson T.: *Structured Systems Analysis: Tools and Techniques*. Prentice Hall, New York, 1979.

Haikala, I. & Märijärvi, J.: *Ohjelmistotuotanto.* Suomen Atk-kustannus Oy, Gummerus Kirjapaino Oy, Jyväskylä, 1998.

Haikala, I. & Märijärvi, J.: *Ohjelmistotuotanto.* Suomen Atk-kustannus Oy - Kauppakaari, RT-Print, Pieksämäki, 2002.

Jarzabek, S. & Tiing, T. S.: Conceptual modeling of families of Software System. *Information Modelling and Knowledge Bases VI* (Ed. Kangassalo, H., & al.), IOS Press, Amsterdam, 1995, 299-312.

Kangassalo, H. & Jaakkola, H.: European-Japanese Research in Information Modelling and Knowledge Bases: A Survey. *Information Modelling and Knowledge Bases VI* (Ed. Kangassalo, H., & al.), IOS Press, Amsterdam, 1995, 469-486.

Laiho, M.: *SQL*. VAPK-kustannus, Valtion painatuskeskus, Helsinki, 1992.

Nijssen, G. M**.** & Halpin, T. A.: *Conceptual Schema and Relational Database Design: A Fact Oriented Approach.* Prentice-Hall, Sydney, 1989.

Marjomaa, E.: *Aspects of Relevance in Information Modelling*. Tampereen yliopisto, Jäljennepalvelu, Tampere, 1997.

Marjomaa, E., 2002b: ”Mallintavan ihmismielen mallintamisesta”. *Ihmisen tiedonkäsittely, symbolien manipulointi ja konnektionismi* (Ed. Marjomaa, E. & Váden, T.), Filosofisia tutkimuksia Tampereen yliopistosta XXII, Tampere, 1991, 133-153.

Rochfeld, A.: MERISE, an information system design and development methodology (Ed. Spaccapietra, S.), *Entity-Relationship Approach*, Elsevier Science Publishers, Amsterdam, 1987.

Rolland, C.: Modeling the Requirements Engineering Process. *Information Modelling and Knowledge Bases V* (Ed. Jaakkola, H., & al.), IOS Press, Amsterdam, 1994, 85-96.

Sainio, A.: *Relaatiotietokannan kehittämisen perusteet – Access*. Yliopistopaino, Helsinki, 2000.

Sainio, A.: *Tietovarastotekniikan perusteet.* Edita Oyj, Helsinki, 2001.

SSADM (Structured Systems Analysis and Design Method) Version 4 Reference Manual, 4 vols. NCC (National Computing Centre) Blackwell, Oxford, 1990.

Stephenson, P. & Hartwig, G.: *SQL: self-teaching guide.* John Wiley & Sons, USA, 1992.

Systeemityöyhdistys Sytyke Ry: *Oliot systeemityössä.* Suomen Atk-kustannus Oy, Anson Oy, Myllykoski, 1992.

Virkki, P. & Somermeri, A.: *Systeemityö tutuksi.* Painatuskeskus Oy, Helsinki, 1993.

Julkaisemattomat käsikirjoitukset:

Kangassalo, H.: *Tietojärjestelmien ja tietokantojen suunnittelu, luentorunko I.* Tampereen yliopisto, tietojenkäsittelyopin laitos, 1995.

Kangassalo, H., 1999: ”Introduction to concepts and Conceptual modelling - Johdanto käsitteisiin ja käsitteelliseen mallintamiseen.” Julkaisematon käsikirjoitus.

Marjomaa, E., 2002c: ”Concepts Are Legisigns”. Julkaisematon käsikirjoitus.

WWW-julkaisut:

Hesso, V. *Miten avata mahdottomia ovia?* Internet WWW-sivu, URL: http://www.hut.fi/~vhesso/wwwrefe.htm (31.1.2002).

Hölttä, T. *Tiede, teknologia ja malli – referointia Mario Bungen klassikosta Scientific Research, osat I ja II*. Internet WWW-sivu, URL:http://www.helsinki.fi/hum/kognitiotiede/writings/bungereh.html (9.4.2002).

Kangassalo, H. *Käsitteelliset mallit ja ontologiat II.* Internet WWW-sivu, URL: http://www.uta.fi/~hk60418/kasmal_2luento.pdf (8.2.2004).

Marjomaa, E., 2002a: Käsitteellisen mallintamisen luentomoniste. Internet WWW-sivut, URL:
http://cs.joensuu.fi/~marjomaa/CM/luento1.htm (8.9.2004),
http://cs.joensuu.fi/~marjomaa/CM/luento2.htm (8.9.2004),
http://cs.joensuu.fi/~marjomaa/CM/luento3.htm (8.9.2004) &
http://cs.joensuu.fi/pages/marjomaa/CMkurssi/CM.html (8.9.2004).

Rajamäki, J. *Tiedonhallinnan perusteet*. Internet WWW-sivu, URL:
http://cs.stadia.fi/~lehtonen/TiTe/Tietokannat/thperusteet.doc (15.4.2004).

Viitanen, A. V. *Entity Relationship -malli*. Internet WWW-sivu, URL:
http://www.cs.uta.fi/~av/tietokantaohjelmointi/01/luennot/
er.pdf (30.1.2002).