Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia
Syksy 2013

INFORMAATIOTIETEIDEN YKSIKKÖ
TAMPEREEN YLIOPISTO

INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 26/2014

TAMPERE 2014

TAMPEREEN YLIOPISTO
INFORMAATIOTIETEIDEN YKSIKKÖ
INFORMAATIOTIETEIDEN YKSIKÖN RAPORTTEJA 26/2014
TAMMIKUU 2014

Erkki Mäkinen (toim.)

Tietojenkäsittelytieteellisiä tutkielmia
Syksy 2013

INFORMAATIOTIETEIDEN YKSIKKÖ
33014 TAMPEREEN YLIOPISTO

ISBN 978-951-44- 9375-1 (pdf)

ISSN-L 1799-8158
ISSN 1799-8158

Sisällysluettelo

Elokuvien suosittelujärjestelmistä ……...……1

Panu Asikanius

Web-käytön tiedonlouhintaprosessi…...…..…….25

Joni Hämäläinen

Katsaus kolmiulotteisiin lääketieteellisiin sovelluksiin……..43

Sami Koivunen

Videopeliohjelmien ylläpidettävyyden parantaminen suunnittelumalleilla…..60

Miikka Kähkönen

Lääketieteellisten opetussimulaatioiden käytettävyydestä…..…...…….…….75

Timo Perälä

Lajittelualgoritmit ja rinnakkaislaskenta Javassa……...…..……..…….…….89

Antti Reunamo

Virtuaalitodellisuuden terapeuttisista mahdollisuuksista…..……..……...….103

Elias Roihuvuo

NoSQL-tietokannat……..……..…….……119

Ville Toni

Elokuvien suosittelujärjestelmistä

Panu Asikanius

Tiivistelmä.
Suosittelujärjestelmät auttavat käyttäjää löytämään itselleen sopivia tuotteita ja palveluita Internet-palveluiden ylitsepursuavista valikoimista käyttäen apuna käyttäjän omia mieltymyksiä ehdotuksissaan. Suosittelujärjestelmät ovatkin hyödyllisiä tilanteissa, joissa valikoima on todella suuri ja eri tuotteet ovat näennäisesti samankaltaisia, kuten Internetin elokuvapalveluissa. Tässä tutkielmassa käyn aihetta läpi alan kirjallisuuteen perustuen ja esittelen muutamia erilaisia elokuvien suosittelutekniikoita.

Avainsanat ja -sanonnat: Suosittelujärjestelmät, elokuvien suosittelu, tiedonlouhinta, koneoppiminen.
CR-luokat: H.2.8

1. Johdanto

Fyysistä tuotteiden jakelua rajoittaa resurssien puute. Kaupoilla on rajallinen määrä hyllytilaa ja kuluttajien nähtäville voidaan saattaa vain pieni osa mahdollisista vaihtoehdoista. Perinteisen videovuokraamon on helpohkoa arvioida, mitkä elokuvat saattaisivat kiinnostaa suurinta osaa käyttäjistä, ja suositella niitä. Käyttäjien ei myöskään ole kovin haastavaa löytää itseään kiinnostavia tuotteita verrattain pienestä valikoimasta. Internet-palvelut voivat tarjota käyttäjilleen periaatteessa koko olemassa olevan tuotevalikoiman. Niinpä perinteisessä videovuokraamossa voi olla satoja eri elokuvia, mutta Netflixin valikoima on kymmeniä tuhansia. Koko valikoimaa ei voida näyttää käyttäjälle kerralla, eikä voida olettaa käyttäjän kuulleen kaikista häntä mahdollisesti kiinnostavista tuotteista. Massiivista valikoimaa ei ole järkevää ylläpitää, jos käyttäjät eivät koskaan löydä mahdollisesti kiinnostavia, mutta vähemmän suosittuja elokuvia. Eräs ratkaisu ongelmaan on suosittelujärjestelmän käyttö. [Rajaraman and Ullman, 2012]

Suosittelujärjestelmät ovat tietokoneohjelmia, jotka antavat jonkin palvelun käyttäjälle ehdotuksia mahdollisesti kiinnostavista tai hyödyllisistä tuotteista tai palveluista. Suositukset liittyvät erilaisiin päätöksentekoprosesseihin, kuten ostettavan tuotteen, kuunneltavan musiikin tai katsottavan elokuvan valintaan. Suosittelujärjestelmät ovat pääosin suunnattuja käyttäjille, joilla ei ole tar-

peeksi henkilökohtaista kokemusta tai pätevyyttä arvioida ylitsepursuavasta valikoimasta itselle hyödyllisiä tuotteita.

Suosittelu on yleensä yksilöityä, joten eri käyttäjät ja käyttäjäryhmät saavat suosittelualgoritmilta erilaisia suosituksia. On olemassa myös yleisiä suositteluja, joita käytetään yleensä sanoma- ja aikakauslehdissä sekä muissa tilanteissa, joissa käyttäjille näytettävää sisältöä ei voida yksilöidä. Verrattavasta yksinkertaisuudestaan huolimatta ne voivat joissain tilanteissa olla tehokkaita, mutta niitä ei yleensä käsitellä suosittelujärjestelmien tutkimuksessa.

Yksinkertaisimmillaan yksilöidyt suositukset esitetään arvotettuna kiinnostavien asioiden listoina. Suosittelujärjestelmät yrittävät ennustaa käyttäjälle sopivia tuotteita tai palveluita käyttäjän mieltymyksiin ja järjestelmän rajoituksiin perustuen. Ennustuksien mahdollistamiseksi suosittelujärjestelmät keräävät käyttäjien mieltymyksiä joko eksplisiittisesti esimerkiksi tuotearvostelujen muodossa tai tekemällä johtopäätöksiä käyttäjän tekemien valintojen perusteella.

Suosittelujärjestelmissä käytetään monien tieteenalojen, kuten vuorovaikutteisen teknologian ja tiedonhaun, tekniikoita ja menetelmiä. Lähes kaikkien suosittelualgoritmien katsotaan kuitenkin olevan yksi tiedonlouhinnan osa-alueista. Suosittelujärjestelmät itsenäisenä tieteenalana nousi pinnalle 1990-luvun puolivälissä ja se on siis suhteellisen uusi tieteenala verrattuna joidenkin perinteisten tietojenkäsittelyn osa-alueiden, kuten tietokantojen, tutkimukseen. Tutkimuksen painopisteenä ovat olleet käytännön kaupalliset sovellutukset, sillä tarkoituksena on tavallisesti parantaa kaupallisten suosittelujärjestelmien toimintaa. [Ricci et al., 2010]

2. Hyödyllisyyden arviointi

Suosittelujärjestelmissä on kahdenlaisia entiteettejä: käyttäjiä ja tuotteita. Käyttäjillä on joistakin tuotteista mielipiteitä, jotka elokuvien osalta ilmaistaan tyypillisesti arvosanalla 1-5, mutta myös muunlaisia käyttäjän preferenssien esitystapoja käytetään. Käyttäjiltä saatu data tallennetaan *hyödyllisyysmatriisiin*, jollaisesta taulukko 1 on yksinkertaistettu esimerkki. Käyttäjät A-D ovat antaneet arvosanoja seitsemälle eri elokuvalle. Elokuvien nimet on lyhennetty muotoon HP1, HP2 ja HP3 elokuville Harry Potter I-III, TW elokuvalle Twilight ja SW1, SW2 ja SW3 elokuville Star Wars I-III. Suurin osa taulukon soluista on tyhjiä, eli käyttäjä ei ole arvostellut elokuvaa. On hyvä huomioida, että käytännön sovelluksissa taulukko olisi vielä paljon harvempi.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Taulukko 1. Hyödyllisyysmatriisi käyttäjien arvosanoista elokuville

Suosittelujärjestelmän tehtävänä on ennustaa arvosanat tyhjiin kohtiin, eli arvioida esimerkiksi. mitä mieltä käyttäjä A olisi SW2:sta. Monissa käytännön sovelluksissa ei ole tarpeellista tai järkevää ennustaa taulukon jokaista tyhjää kohtaa, vaan riittää, että kultakin riviltä löydetään jokin määrä kaikkein kiinnostavimmiksi arvioituja alkioita. [Rajaraman and Ullman, 2012] Erilaisia tekniikoita ennustuksien tekemiseen esitellään tutkielman luvussa 3.

Hyödyllisyysmatriisin populoiminen

Tuotteiden suosittelu ilman hyödyllisyysmatriisia on jokseenkin mahdotonta, mutta sellaisen muodostamiseen tarvittavan datan kerääminen on usein vaikeaa. Käyttäjien mieltymyksien kartoittamiseen käytetään yleisesti ottaen kahta erilaista tapaa:

I. **Tuotearvostelut**: Käyttäjiä voidaan pyytää arvostelemaan tuotteita. Tapaa käytetään yleisesti elokuvien arvostelussa. Arvosteluasteikko on tyypillisesti numeerinen, esimerkiksi 1-5, mutta myös binäärinen ”like/dislike” -asteikko on yleisesti käytössä. [Rajaraman and Ullman, 2012] Myös käyttäjien jättämien tekstikommenttien analysointia mielipiteen selvittämiseksi käytetään jossain määrin. [Ricci et al., 2010] Eksplisiittiseen arvosteluun perustuvan tavan tehokkuus on rajallinen, sillä monet käyttäjät ovat haluttomia arvostelemaan tuotteita, ja arvosanan antaneilta käyttäjiltä saatu informaatio saattaa olla vinoutunutta juuri siksi, että se on kerätty ihmisiltä, jotka ovat innokkaita arvostelijoita. [Rajaraman and Ullman, 2012]
II. **Päätelmät käyttäytymisen perusteella:** Koska useimmat käyttäjät ovat vuorovaikutuksessa itseään kiinnostavien tuotteiden kanssa, voidaan käyttäjän päätellä pitäneen jollain tasolla katsomastaan elokuvasta ilman, että käyttäjän tarvitsisi eksplisiittisesti arvostella sitä. [Ricci et al., 2010] Tällaisella ”arvostelulla” on siis vain kaksi mahdollista arvoa: *kiinnostava* tai *tuntematon*. [Rajaraman and Ullman, 2012]

3. Suosittelumenetelmistä

Pystyäkseen täyttämään tarkoituksensa, kiinnostavien tuotteisen suosittelun, suosittelujärjestelmän on pystyttävä ennustamaan yksittäisen tuotteen hyödyllisyys käyttäjälle. Sen toteuttamiseen on kehitetty erilaisia suosittelumenetelmiä, joilla on kullakin hyvät ja huonot puolensa. [Ricci et al., 2010] Menetelmät jaetaan toimintaperiaatteensa mukaan karkeasti kahteen eri ryhmään:

I. **Sisältöperustainen järjestelmä** yrittää yhdistää käyttäjät heitä kiinnostaviin tuotteisiin käyttäjien ja tuotteiden ominaisuuksien perusteella, ja suosittelemalla tuotteita, jotka ovat samankaltaisia niiden tuotteiden kanssa joista käyttäjä on aiemmin pitänyt [Takács et al., 2009]. Tuotteiden samankaltaisuutta arvioidaan vertailemalla tuotteiden ominaisuuksia, joita elokuvissa ovat esimerkiksi genre, julkaisuvuosi, ohjaaja ja näyttelijät [Rajaraman and Ullman, 2012].
II. **Yhteistoiminnallisen järjestelmän** suositukset perustuvat samankaltaisten käyttäjien mieltymyksiin. Käyttäjien samankaltaisuus määritetään käyttäjien arvosteluhistorioiden eli käyttäjien eri tuotteille antamien arvosanojen perusteella. Suosittelussa ei siis välitetä tuotteiden ominaisuuksista tai sisällöistä, vaan ainoastaan tuotteiden ja käyttäjien välisistä suhteista. [Takács et al., 2009] Yhteistoiminnalliseen suositteluun perustuva implementaatio on yleisisin suosittelujärjestelmien toteutustapa [Ricci et al., 2010].

Ricci ja muut [2010] listaavat näiden kahden kategorian lisäksi vielä muutamia menetelmiä, jotka voidaan nähdä kahden ylempänä mainitun menetelmän alakohtina tai tarkennuksina:

I. **Demograafiset** järjestelmät jakavat käyttäjät eri ryhmiin heidän ominaisuuksiinsa pohjautuen ja muodostavat suosituksia niihin perustuen. Menetelmä olettaa siis, että esimerkiksi eri ikäryhmille tai sukupuolille tulisi muodostaa erilaisia suosituksia.
II. **Tietoon perustuvat** järjestelmät arvioivat tapauskohtaisesti, kuinka paljon käyttäjä *tarvitsee* tuotetta perustuen käyttäjän ongelmakuvaukseen ja tuotteiden ominaisuuksiin. Menetelmä on siis puhtaasti sisältöperustainen. Lähestymistapa toimii hyvin uusille käyttäjille, mutta koska järjestelmä ei perustu varsinaisesti käyttäjän tuntemiseen, monet koneoppimiseen pohjautuvat menetelmät toimivat paremmin pitkäaikaisen käytön jälkeen.

III. **Yhteisölliset** järjestelmät antavat suosituksia käyttäjän ystävien mieltymysten perusteella. Sinha ja Swearingen [2001] havaitsivat, että käyttäjät luottavat enemmän ystäviensä kuin samanmielisten, mutta anonyymien ihmisten suosituksiin. Sosiaalisten medioiden suosion kasvun kautta myös kiinnostus yhteisöllisiin suosittelujärjestelmiin on noussut, mutta aiheeseen liittyvä tutkimus on vielä vähäistä ja tulokset menetelmään perustuvien järjestelmien tehokkuudesta ovat ristiriitaisia.

Monet käytössä olevat järjestelmät yhdistelevät tekniikoita eri menetelmistä paikatakseen yksittäisten menetelmien puutteita. Esimerkiksi puhtaasti yhteistoiminnallinen järjestelmä ei voi suositella uutta tuotetta kenellekään ennen kuin vähintään yksi käyttäjistä on arvostellut sen, ja luotettavien suositteluiden muodostamiseksi tarvitaan yleensä huomattavasti enemmän arvosteluita. Sisältöperustaisessa suosittelussa tällaista ongelmaa ei ole, koska tuotteen kiinnostavuus arvioidaan sen ominaisuuksien perusteella.

3.1. Sisältöperustainen suosittelu

Sisältöperustaista suosittelua käyttävät järjestelmät analysoivat tuotteita ja muodostavat niiden ominaisuuksien perusteella tuoteprofiilin, joka kuvastaa tuotteen olennaisia ja vertailtavissa olevia piirteitä. Elokuvien kohdalla tällaisia piirteitä ovat esimerkiksi näyttelijät, ohjaaja, ilmestymisvuosi ja genre. Jotkut pitävät elokuvista, joissa esiintyy tietty näyttelijä tai jonka on ohjannut tietty ohjaaja. Jotkut ihmiset katsovat esimerkiksi paljon uusia komedioita tai vanhoja kauhuelokuvia. [Rajaraman and Ullman, 2012]

Järjestelmä muodostaa käyttäjän arvostelemien elokuvien profiilien pohjalta käyttäjäprofiilin käyttäjän kiinnostuksen kohteista. Käyttäjän mielipiteet tuotteista tallennetaan palautetietokantaan käyttäjäprofiilin muodostamista ja päivitystä varten.

Suositteluprosessi pohjautuu käyttäjäprofiilin ominaisuuksien täsmäämiseen uusien tuotteiden profiilien kanssa. Järjestelmä ennustaa, mitkä analysoiduista tuotteista olisivat potentiaalisesti kaikkein kiinnostavimpia käyttäjälle ja järjestää ne listaan. Listan yläpäässä olevat tuotteet lisätään käyttäjälle näytettäviin suosituksiin ja käyttäjältä kerätään niistä lisää palautetta, jonka perusteella käyttäjäprofiilia päivitetään ja muodostetaan lisää suosituksia. Palautteeseen perustuva oppimissykli mahdollistaa käyttäjän mielipiteiden muutosvoimaisuuden huomioon ottamisen.

Puhtaasti sisältöperustaiset suosittelujärjestelmät kärsivät yleisesti tuotteiden ominaisuuksien analysointiin ja liialliseen erikoistumiseen liittyvistä ongelmista. Tuotteiden ominaisuuksien luotettava analysointi on hankalaa, jos

järjestelmällä on hyvin rajallinen määrä informaatiota kustakin tuotteesta. Pelkkä tuotteen kuvauksen analysointi on myös monesti melko epäluotettava tapa arvioida sen laatua. Sisältöperustaisilla suosittelujärjestelmillä on tapana suositella käyttäjän korkeasti arvioimien tuotteiden kanssa hyvin samankaltaisia tuotteita, koska niillä ei ole sisäänrakennettua tapaa suositella odottamattomia tuotteita. Tämä puute johtaa liialliseen erikoistumiseen. Järjestelmä voi suositella käyttäjälle esimerkiksi samaan genreen kuuluvia elokuvia tai elokuvia, joissa on samoja näyttelijöitä kuin käyttäjän jo katsomissa elokuvissa, mutta jättää suosittelematta elokuvia, jotka ovat ominaisuuksiltaan erilaisia, mutta silti käyttäjälle kiinnostavia. Ilmiötä kutsutaan *onnekkaan sattuman ongelmaksi*. [Ricci et al., 2010]

3.2. Yhteistoiminnallinen suosittelu

Yhteistoiminnallisessa suosittelussa ennustuksien muodostamiseen käytetään tuotteiden sisältöön perustuvan datan sijaan muiden käyttäjien antamia arvosanoja. Hieman yksinkertaistettuna perusidea on, että käyttäjän x uudelle tuotteelle antama arvosana olisi todennäköisesti lähellä käyttäjän y tuotteelle antamaa arvosanaa, jos x ja y ovat aikaisemmin arvostelleet tuotteita samankaltaisesti.

Yhteistoiminnalliset menetelmät paikkaavat joitain sisältöön perustuvien järjestelmien puutteita. Tuotteiden ominaisuuksia ei tarvitse analysoida, kun suosittelu perustuu muiden käyttäjien tekemiin arvosteluihin. Yhteistoiminnallinen suosittelu mahdollistaa myös hyvin erilaisten tuotteiden suosittelun, kunhan muut käyttäjät ovat jo osoittaneet kiinnostuksensa niitä kohtaan.

Yhteistoiminnalliset menetelmät voidaan jakaa toimintaperiaatteensa mukaan kahteen luokkaan:

I. **Naapurustoon perustuvassa** suosittelussa järjestelmään tallennettuja käyttäjän tuotearvosteluja käytetään suoraan ennustamaan uuden tuotteen arvosanaa. Naapureiksi kutsutaan joukkoa, jonka muodostavat ne käyttäjät, joilla on samankaltaiset arvostelutavat kuin käyttäjällä. Niin sanotut käyttäjäpohjaiset menetelmät arvioivat käyttäjän kiinnostusta tuotteeseen käyttäen suoraan naapureiden tuotteelle antamia arvosanoja. Tuotepohjaiset järjestelmät taas ennustavat tuotteen kiinnostavuutta käyttäjän samankaltaisille tuotteille antamien arvosanojen perusteella. Kaksi tuotetta katsotaan tässä yhteydessä samankaltaisiksi, jos muut käyttäjät ovat arvostelleet ne samalla tavalla.
II. **Malliin perustuvat** menetelmät eivät käytä arviointiin suoraan muiden käyttäjien arvosteluita vaan niiden pohjalta muodostetaan ennustusmal-

li. Ideana on käyttää käyttäjän ja tuotteiden välisiä vuorovaikutustilanteita käyttäjän ja tuotteiden piilevien ominaisuuksien kuvaamiseksi. Mallia opetetaan saatavilla olevan datan avulla ja käytetään sitten ennustamaan uusien tuotteiden kiinnostavuutta. Erilaisiin malleihin perustuvia lähestymistapoja tuotteiden suositteluun on lukuisia ja niihin kuuluu esimerkiksi bayesilaista klusterointia, Boltzmannin konetta ja pääakselihajotelmaa käyttäviä menetelmiä.

Kehittyneet malliin perustuvat menetelmät ovat osoittautuneet naapurustoon perustuvia menetelmiä paremmiksi arvosanojen ennustuksessa, mutta pelkkä tarkkuus ei takaa käyttäjille tyydyttävää kokemusta. Tarkkojen ennustuksien ohella miellyttävän käyttökokemuksen muodostumiseen vaikuttavat *onnekkaat sattumat*, eli käyttäjää kiinnostavat tuotteet, joita hän ei olisi ilman suosittelua löytänyt. Onnekkaat sattumat lisäävät käyttäjän kokemaa uutuudenviehätystä.

Malliin perustuvat menetelmät ovat tehokkaita löytämään käyttäjien piileviä mieltymyksiä. Ne voivat esimerkiksi oppia, että käyttäjä pitää elokuvista, jotka ovat ”hauskoja” ja ”romanttisia”, ilman että järjestelmän täytyy varsinaisesti määritellä käsitteitä, kuten ”hauska” tai ”romanttinen”. Tällainen järjestelmä voisi suositella käyttäjälle aikaisemmin tuntematonta romanttista komediaa, mutta jättää suosittelematta vaikkapa hauskaa kauhuelokuvaparodiaa. Naapurustoon perustuvat menetelmät havaitsevat datasta paikallisia yhteyksiä, minkä vuoksi niiden on mahdollista suositella käyttäjän tavallisesta elokuvamausta poikkeavaa elokuvaa, jos käyttäjän lähimmät naapurit ovat antaneet sille korkean arvosanan. Suositeltu elokuva ei ole välttämättä romanttisen komedian kaltainen takuuvarma menestys, mutta voi auttaa käyttäjää löytämään uuden suosikkigenren tai -ohjaajan. [Ricci et al., 2010]

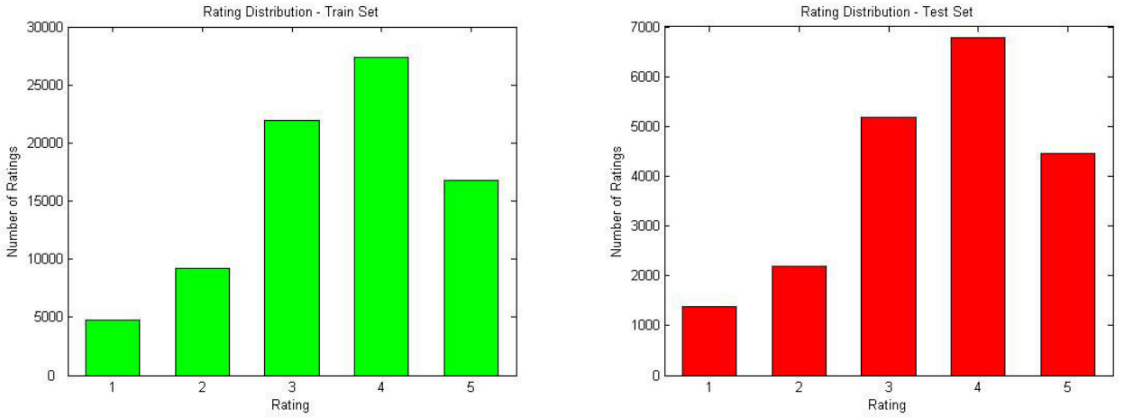
4. Muutamia erilaisia suosittelualgoritmin toteutuksia

Esittelen tässä luvussa muutamia erilaisia kirjallisuudessa esitettyjä suosittelualgoritmeja ja niiden testaukseen usein käytettävän aineiston.

4.1. MovieLens -aineistosta

Suosittelualgoritmien kehityksessä ja testauksessa käytetään yleisesti oikeilta käyttäjiltä kerättyjä elokuva-arvosteluja sisältävää MovieLens-aineistoa. Aineisto rakentuu Minnesotan yliopiston GroupLens Research -tutkimuslaboratorion ylläpitämältä MovieLens-sivustolta kerätysta datasta. Aineistosta on kolme eri versiota, joista pienin, MovieLens 100k, sisältää 100000 arvostelua, 1682 elokuvaa ja 943 käyttäjää. Aineistossa on mukana vain yli 20 elokuvaa arvostelleita käyttäjiä ja kukin elokuva on arvosteltu vähintään kerran asteikolla 1-5.

Ristiinvalidoinnin helpottamiseksi aineisto on ennalta jaettu viiteen osaan. Kukin osista sisältää opetusdatan, jota käytetään järjestelmän koulutukseen, ja testausdatan, johon järjestelmän tuottamia suosituksia verrataan. [GroupLens, 2013; Lam and Riedl., 2004; Pucci et al., 2007]



Kuva 1. Eri arvosanojen lukumäärät koulutus- (vasen) ja testausainestoissa (oikea). [Pucci et al., 2007]

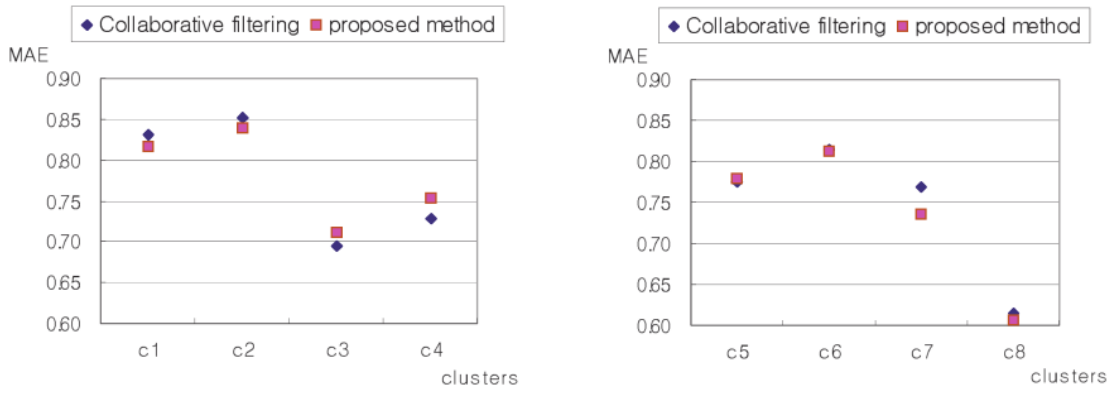
Kuvassa 1 näkyy MovieLens 100k -aineiston koulutus- ja testausdatan arvosanojen jakaumat. Huomionarvoinen seikka on, että eri arvosanojen lukumäärät eivät ole yhdenmukaiset, vaan jakaumat ovat oikealle vinoutuneita. Suurin osa arvosanoista on joko 3 tai 4 ja arvosanaa 1 on annettu vähiten. Arvosanojen keskiarvo on 3,53. [Pucci et al., 2007]

4.2. Itseorganisoituvaa karttaa hyödyntävä yhteistoiminnallinen järjestelmä

Lee ja muut [2002] esittelevät suosittelujärjestelmän, joka yhdistää yhteistoiminnalliseen suositteluun itseorganisoituvan kartan. Ensin käyttäjät segmentoidaan demograafisten ominaisuuksien perusteella ja kunkin segmentin käyttäjät ryhmitellään elokuvamieltymysten mukaan käyttäen itseorganisoituvaa karttaa. Tämän jälkeen elokuvasuositukset muodostetaan käyttäen naapurustoon perustuvaa yhteistoiminnallista algoritmia kuhunkin käyttäjäryhmään. Yksittäiselle käyttäjälle annettavat suositukset on siis muodostettu jo valmiiksi samankaltaisten käyttäjien joukosta. Tarkoituksena on vähentää tarvittavan laskennan määrää ja parantaa yhteistoiminnallisen suosittelualgoritmin suorituskykyä sekä skaalautuvuutta vähentämättä suositusten laatua.

Järjestelmää testattiin käyttäen MovieLens-aineistosta muodostettua 174 käyttäjää, 54 elokuvaa ja 5331 arvostelua sisältävää aineistoa. Kahdessa eri testissä käyttäjät jaettiin kahteen segmenttiin joko iän tai sukupuolen perusteella. Segmentit jaettiin edelleen kahteen ryhmään käyttäen itseorganisoituvaa karttaa. Elokuvat oli jaettu viiteen eri genreen ja kunkin käyttäjän genremieltymyk-

set määritettiin käyttäjän eri elokuville antamien arvosanojen perusteella. Itseorganisoituvan kartan syötevektoreina käytettiin käyttäjien genremieltymyksiä. Molemmissa testeissä muodostettiin siis neljä eri ryhmää genremieltymysten ja sukupuolen tai iän perusteella ja suosittelualgoritmi suoritettiin erikseen kullekin ryhmälle. Suositukset muodostettiin käyttäen GroupLens Researchin kehittämän avoimen lähdekoodin LensKit-kirjaston sisältämää yhteistoiminnallista suosittelualgoritmia. Suosittelu suoritettiin samalla algoritmilla myös ilman käyttäjien ryhmittelyä ja suositusten tarkkuutta vertailtiin ennustusten keskiarvoisien absoluuttisten virheiden (Mean Absolute Error, MAE) avulla.



Kuva 2. Testin tulokset, kun käyttäjät segmentoitiin sukupuolen (vasen) tai iän (oikea) perusteella. [Lee et al., 2002]

Kuvassa 2 on esitetty vertailun tulokset, joista voidaan havaita käyttäjien ryhmittelyyn perustuvan lähestymistavan olevan perinteistä yhteistoiminnallista suosittelua jopa hieman tarkempi arvosanojen ennustuksessa. Käyttäjän naapuruston rajaaminen jo valmiiksi samankaltaisiin käyttäjiin vähensi ennustusten muodostamiseen tarvittavan laskennan määrää, joten esitetty algoritmi myös suoriutui suosittelusta perinteistä yhteistoiminnallista algoritmia huomattavasti nopeammin. [Lee et al., 2002]

4.3. ItemRank - satunnaiskulkuun perustuva suosittelu

Pucci ja muut [2007] kuvaavat ItemRank-niminen malliin perustuvaa yhteistoiminnallista suosittelualgoritmia. Algoritmi pohjautuu elokuvien samankaltaisuuden perusteella luodun painotetun graafin kulkuun käyttäen muokattua PageRank-algoritmia.

4.3.1 Järjestelmän tietomalli

Lähestymistapa olettaa, että käyttäjillä on tapana katsoa samankaltaisia elokuvia, jolloin jos kaksi elokuvaa esiintyvät yhdessä useiden käyttäjien katsottujen elokuvien listassa, niiden voidaan olettaa olevan jokseenkin samankal-

taiset. Tämän oletuksen perusteella muodostetaan korrelaatiograafi, jota käytetään algoritmin tietomallina. Määritellään $U_{ij} \subseteq U$ niiden käyttäjien joukoksi, jotka ovat katsoneet elokuvat m_i ja m_j ja muodostetaan kunkin elokuvaparin arvostelleet käyttäjät käsittävä matriisi $\check{C}_{ij} = |U_{ij}|$, jossa $|\cdot|$ tarkoittaa joukon mahtavuutta. $\check{C}$ on siis symmetrinen neliömatriisi. Seuraavaksi matriisi normalisoidaan stokastiseksi korrelaatiomatriisiksi $C_{ij} = \frac{\check{C}_{ij}}{\omega_j}$, jossa ω_j on j:nnen sarakkeen alkioiden summa. Prosessin selventämiseksi esittelen alla yksinkertaistetun esimerkkitapauksen. Taulukossa 2 näytetään riveittäin, onko käyttäjä katsonut elokuvan (Y) vai ei (-). Järjestelmä olettaa, että käyttäjä on katsonut arvostelemansa elokuvat, mutta varsinaisista arvosanoista ei olla vielä tässä vaiheessa kiinnostuneita.

	m_1	m_2	m_3	m_4	m_5
u_1	Y	Y	-	-	-
u_2	-	Y	Y	Y	-
u_3	Y	-	Y	-	-
u_4	Y	-	-	Y	Y
u_5	Y	Y	Y	Y	-
u_6	Y	-	Y	-	-
u_7	Y	-	Y	Y	-
u_8	Y	Y	-	Y	-
u_9	Y	-	-	-	Y
u_{10}	Y	-	-	-	Y
u_{11}	-	Y	-	Y	-
u_{12}	-	-	Y	Y	-

Taulukko 2. Käyttäjien katsomat elokuvat

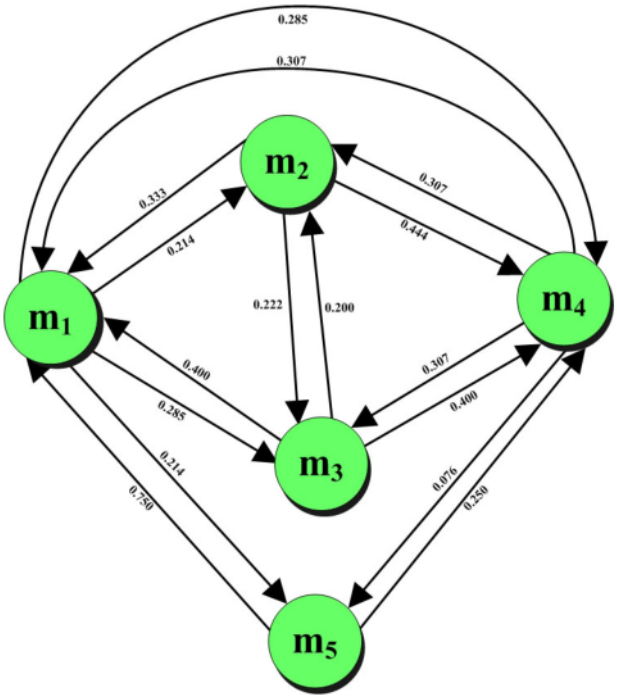
Taulukon esittämän datan perusteella muodostettu matriisi $\check{C}$ on

$$\check{C} = \begin{pmatrix} 0 & 3 & 4 & 4 & 3 \\ 3 & 0 & 2 & 4 & 0 \\ 4 & 2 & 0 & 4 & 0 \\ 4 & 4 & 4 & 0 & 1 \\ 3 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Siitä normalisoitu korrelaatiomatriisi C on

$$C = \begin{pmatrix} 0 & 0{,}333 & 0{,}400 & 0{,}307 & 0{,}750 \\ 0{,}214 & 0 & 0{,}200 & 0{,}307 & 0 \\ 0{,}285 & 0{,}222 & 0 & 0{,}307 & 0 \\ 0{,}285 & 0{,}444 & 0{,}400 & 0 & 0{,}250 \\ 0{,}214 & 0 & 0 & 0{,}076 & 0 \end{pmatrix}.$$

Korrelaatiomatriisin C voi ajatella myös painotettuna vierusmatriisina korrelaatiograafille G_c, jonka solmut ovat elokuvia, ja elokuvien m_i ja m_j välillä on kaari (m_i, m_j), jos $C_{ij} > 0$, ja kaaren painokerroin on tällöin C_{ij}. Kannattaa huomata, että C ei ole symmetrinen, joten kaaren (m_i, m_j) painokerroin voi saada eri arvon kuin kaaren (m_j, m_i). Esimerkkimatriisiin liittyvä korrelaatiograafi G_c on esitetty kuvassa 3. [Pucci et al., 2007]



Kuva 3. Yksinkertainen korrelaatiograafi G_c. [Pucci et al., 2007]

4.3.2 ItemRank-algoritmi

ItemRank-algoritmin ideana on, että korrelaatiograafin avulla voidaan ennustaa käyttäjien mieltymyksiä. Kun tiedetään käyttäjän joillekin elokuville antamat arvosanat, voidaan käyttäjän mieltymykset *levittää* läpi graafin. Sama prosessi toistetaan jokaisen käyttäjän kohdalla käyttäen samaa graafia ja kunkin käyttäjän omia arvosteluita.

Tärkeä tekijä mieltymysten leviämisessä on *eteneminen* graafissa. Perusoletuksena on, että jos elokuva m_k on vahvasti yhteydessä yhden tai useamman käyttäjän arvostaman elokuvan kanssa, niin käyttäjä todennäköisesti pitää myös elokuvasta m_k, ja nämä vuorovaikutussuhteet ja niiden voimakkuudet voidaan selvittää korrelaatiograafista. Toinen tärkeä tekijä mieltymysten leviämisessä on *vaimentumien*. Käyttäjän arvostamat elokuvat siirtävät positiivista vaikutustaan läpi graafin, mutta efektin voimakkuus laskee siirryttäessä kauemmas arvostetusta elokuvasta. Lisäksi, jos elokuva m_k on yhteydessä useam-

paan kuin yhteen muuhun elokuvaan, täytyy näiden elokuvien jakaa elokuvan m_k positiivinen vaikutus graafin kaarien painokertoimien mukaisessa suhteessa. Suositusten muodostamiseen käytettävällä algoritmilla täytyy siis olla tilanteeseen sopivat etenemis- ja vaimentumisominaisuudet.

Stanfordin yliopistossa kehitetty PageRank-algoritmi sopii hieman muokattuna graafin solmujen, eli eri elokuvien, hyödyllisyyden arviointiin. Tavallinen PageRank arvioi graafin solmujen tärkeyttä puolueettomasti niihin johtavien linkkien määrän ja laadun mukaan, mutta sen toimintaa voidaan suunnata tietynlaisilla syötevektoreilla. ItemRank on PageRankin pohjalta kehitetty suosittelualgoritmi, joka suuntaa hyödyllisyyden arviointia käyttäen syötteenä käyttäjän antamia arvosanoja.

ItemRank laskee korrelaatiograafin G_c jokaiselle elokuvasolmulle käyttäjäkohtaisen ItemRank-arvon IR_{u_i} iteratiivisesti käyttämällä korrelaatiomatriisia C kaavalla

$$IR_{u_i} = \alpha \cdot C \cdot IR_{u_i} + (1 - \alpha) \cdot d_{u_i},$$

jossa α on vaimentumiskerroin (0,85) ja d_{u_i} on käyttäjän antamista arvosanoista muodostettu jakaumavektori. Tämä dynaaminen systeemi pitää määritellä jokaisen käyttäjän kohdalla, mutta se vaatii keskimäärin vain noin 20 toistoa supetakseen. Tuloksena on käyttäjäkohtainen IR_{u_i}-vektori. Mitä korkeampi jonkin elokuvan ItemRank -arvo on, sitä todennäköisemmin käyttäjä pitää elokuvasta.

	m_1	m_2	m_3	m_4	m_5
u_1	4	2	-	-	-

Taulukko 3. Käyttäjän u_1 antamat arvosanat

Aiemmin käsitellyssä esimerkkitapauksessa käyttäjän u_1 antamat arvosanat on esitetty taulukossa 3. Niistä muodostettu jakaumavektori $d_{u_1} = (0{,}66; 0{,}33; 0; 0; 0)$, jolloin algoritmi tuottaa ItemRank-arvot $IR_{u_1} = (0{,}3175; 0{,}1952; 0{,}1723; 0{,}2245; 0{,}0723)$. [Pucci et al., 2007]

ItemRank-algoritmi on tehokas laskennallisesti ja muistinkäytöltään. Käytetty tietomalli skaalautuu hyvin käyttäjämäärän kasvaessa, koska graafin solmujen lukumäärä ei riipu käyttäjien lukumäärästä ja kaarien lukumäärä kasvaa hitaasti, kun käyttäjien lukumäärä on ylittänyt tietyn kynnysarvon.

ItemRank-algoritmin suorituskykyä arvioitiin suorittamalla erilaisia testejä MovieLens 100k -aineistolla. Aineisto jaettiin viiteen osaan ja ItemRank suoritettiin viisi kertaa niin, että kullakin kierroksella yhtä osista käytettiin testauk-

seen ja neljä muuta yhdistettiin opetusaineistoksi. Kullakin kierroksella opetusdata sisälsi 80000 arvostelua ja testidata 20000 arvostelua. Ennustusten tarkkuuden mittarina käytettiin *johdonmukaisuuden astetta* (Degree of Agreement, DOA), joka mittaa oikeaan järjestykseen sijoitettujen elokuvaparien prosenttiosuutta kaikista elokuvapareista. Satunnaisesti järjestettynä puolet elokuvapareista olisi oikeassa järjestyksessä, jolloin DOA-arvo on 50 %. DOA-arvon laskemiseksi yksittäiselle käyttäjälle u_i täytyy määrittää joukko $NW_{u_i} \subset M$, joka sisältää elokuvat, joita käyttäjä ei ole arvostellut opetus- eikä testiaineistoissa. Seuraavaksi määritellään totuusfunktio

$$järjestys_{u_i}(m_j, m_k) = \begin{cases} 1, jos\ IR_{u_i}^{m_j} \geq IR_{u_i}^{m_k} \\ 0, jos\ IR_{u_i}^{m_j} < IR_{u_i}^{m_k} \end{cases},$$

jota käytetään DOA-arvon laskemisessa käyttäjälle u_i kaavalla

$$DOA_{u_i} = \frac{\sum_{(m_j, m_k) \in T_{u_i} \times NW_{u_i}} järjestys_{u_i}(m_j, m_k)}{|T_{u_i}| \cdot |NW_{u_i}|},$$

jossa T_{u_i} tarkoittaa käyttäjän u_i testiaineistossa arvostelemia elokuvia. Koko järjestelmän tarkkuuden mittaamiseen käytetään kahta erilaista yksittäisten käyttäjien DOA-arvoista johdettua arvoa. Macro DOA on yksinkertaisesti kaikkien käyttäjien DOA-arvojen keskiarvo

$$Macro\ DOA = \frac{\sum_{u_i \in U} DOA_{u_i}}{|U|}.$$

Micro DOA taa on oikeassa järjestyksessä olevien elokuvaparien ja kaikkien tarkistettujen elokuvaparien välinen suhde, koskien kaikkia käyttäjiä. Se lasketaan kaavalla

$$micro\ DOA = \frac{\sum_{u_i \in U} \left(\sum_{(m_j, m_k) \in T_{u_i} \times NW_{u_i}} järjestys_{u_i}(m_j, m_k) \right)}{\sum_{u_i \in U} (|T_{u_i}| \cdot |NW_{u_i}|)}.$$

Micro DOA on siis hiukan kuin yksittäisten DOA-arvojen painotettu keskiarvo, sillä yksittäisen käyttäjän u_i vaikutus micro DOA -arvoon on sitä suurempi, mitä enemmän elokuvia T_{u_i} sisältää. Taulukossa 4 esitetään viidellä yllä esitetyllä MovieLens-aineiston jaolla suoritettujen ItemRank-algoritmin testien tulokset.

	1. jako	2. jako	3. jako	4. jako	5. jako	**Keskiarvo**
micro DOA	87,14	86,98	87,20	87,08	86,91	**87,06**
Macro DOA	87.73	87,61	87,69	87,47	88,28	**87,76**

Taulukko 4. ItemRank-algoritmin testauksen tulokset

ItemRank-algoritmia haluttiin vertailla etenkin muihin kirjallisuudessa esitettyihin graafiin pohjautuviin samankaltaisuuden arvioinnin menetelmiin. Monet aiemmista menetelmistä käyttävät tietomallinaan graafia, joka sisältää käyttäjä- ja elokuvasolmuja, ja solmujen välillä on kaari, jos käyttäjä on katsonut elokuvan. Vertailtaviksi menetelmiksi valittiin:

I. **Yksisuuntainen ensiohituksen aika** (Average first-passage time, **One-way**) merkitään $m(i,j)$ ja se mittaa graafin solmun i samankaltaisuutta solmun j kanssa satunnaiskulkijan ottamien askelten määrällä solmusta i solmuun j. Suunnatussa graafissa ensiohituksen aika ei ole symmetrinen.
II. **Palaava ensiohituksen aika** (Average first-passage time, **Return**) merkitään $m(j,i)$ ja se mittaa solmun i samankaltaisuutta solmun j kanssa satunnaiskulkijan ottamien askelten määrällä solmusta j solmuun i, eli käänteisesti yksisuuntaiseen aikaan nähden.
III. **Siirtymäaika** (Commute Time, **CT**) merkitään $n(i,j)$ ja se mittaa satunnaiskulkijan ottamien askelten määrää solmusta i solmuun j ja takaisin, eli $n(i,j) = m(i,j) + m(j,i)$.
IV. **Euklidiseen siirtymäaikaetäisyyteen pohjautuva pääkomponenttianalyysi** (Principal Component Analysis based on Euclidean Commute Time Distance, **PCA CT**). Graafia vastaavan Laplacen matriisin L pseudoinverssin L^+ ominaisvektorihajotelmalla solmut voidaan kuvata Euklidiseen avaruuteen, joka säilyttää Euklidisen siirtymäaikaetäisyyden. Pääkomponenttianalyysin avulla voidaan projisoida m-ulotteinen aliavaruus, josta laskettuja etäisyyksiä käytetään elokuvien järjestämiseen suosituslistaksi kullekin käyttäjälle.
V. **Laplacen matriisin pseudoinverssi** (Pseudoinverse of the Laplacian Matrix, **L**⁺). Matriisi L^+ sisältää solmuvektorien sisätulot Euklidisessa avaruudessa, jossa solmujen väli Euklidinen siirtymäaikaetäisyys, joten arvoa $l^+_{i,j}$ voidaan käyttää solmujen i ja j samankaltaisuuden mittana.
VI. **Katz** on tyypillisesti yhteiskuntatieteissä käytetty samankaltaisuuden mitta. Menetelmä arvioi solmujen samankaltaisuutta niitten välisten suorien ja epäsuorien yhteyksien lukumäärän perusteella. Katzin matriisi määritellään $K = \alpha A + \alpha^2 A^2 + \cdots + \alpha^n A^n + \cdots = (I - \alpha A)^{-1} - I$, jossa A on graafin vierusmatriisi ja α on solmujen välisten yhteyksien määrästä riippuva

vaimennuskerroin. Solmujen i ja j samankaltaisuus on $sim(i,j) = k_{i,j} = [K]_{i,j}$.

VII. **Dijkstra** on klassinen algoritmi lyhimmän polun etsimiseen suunnatussa ja painotetussa graafissa. Menetelmä ei ota huomioon useita kahden solmun välillä olevia yhteyksiä, minkä takia se soveltuu suositteluun huonosti.

VIII. **MaxF** on muiden menetelmien vertailukohtana käytetty menetelmä, jossa elokuvat laitetaan suosiojärjestykseen yksinkertaisesti elokuvan katsoneiden käyttäjien lukumäärän mukaan. Se tuottaa siis saman järjestyksen kaikille käyttäjille.

Kutakin menetelmää testattiin käyttäen MovieLens-aineistoa samalla tavoin jaettuna. Taulukossa 5 esitetään tuloksien keskiarvon menetelmäkohtaisesti. Kustakin menetelmästä kerrotaan sen Macro DOA -arvo ja erotus vertailukohtana käytettyyn MaxF-algoritmiin.

	Macro DOA	MaxF-erotus
MaxF	84,07	0
One-way	84,08	+0,01
Return	72,63	-11,43
CT	84,09	+0,02
PCA CT	84,04	-0,03
L^+	87,23	+3,16
Katz	85,83	+1,76
Dijksta	49,96	-34,11
ItemRank	**87,76**	**+3,69**

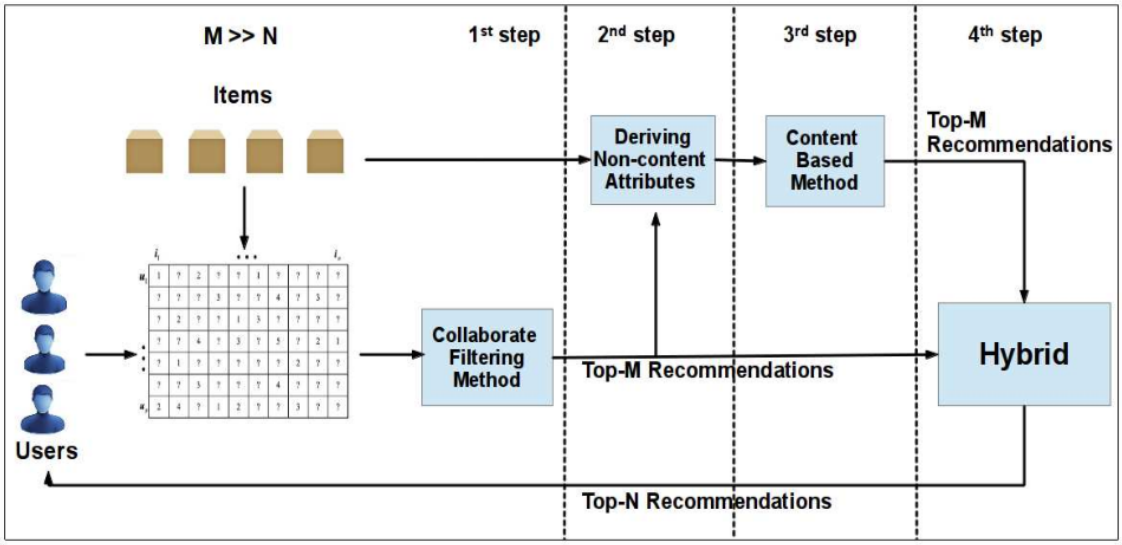
Taulukko 5. Eri menetelmien testien tulokset

ItemRank suoriutuu suositusten järjestämisestä paremmin kuin mikään vertailluista graafiin perustuvista menetelmistä. Lisäksi ItemRank on muita menetelmiä tehokkaampi laskennallisesti ja muistinkäytöltään. [Pucci et al., 2007] Huomionarvoinen seikka on, että pelkkä suosiojärjestys MaxF tuottaa parempia suosituksia, kuin osa monimutkaisista graafiin perustuvista samankaltaisuutta arvioivista menetelmistä.

4.4. Sisältöön liittymättömiin ominaisuuksiin perustuva hybridisuosittelu

Mourão ja muut [2013] esittävät menetelmän, joka käyttää sisältöperustaiseen suositteluun elokuviin liittyviä ominaisuuksia, jotka eivät liity elokuvan varsinaiseen sisältöön. Attribuutit mallinnetaan preferenssivektoreina, joita käytetään osana sisältöperustaista suosittelua. Tämä sisältöperustainen suosittelija taas voidaan yhdistää tulosten uudelleenpainotuksen avulla erilaisiin yhteistoiminnallisiin suosittelumenetelmiin.

Sisältöön liittymättömiksi ominaisuuksiksi valittiin *suosio, samankaltaisuus* ja *tuoreus*, koska aikaisemmissa tutkimuksissa on havaittu näihin attribuutteihin liittyviä trendejä käyttäjien mieltymyksissä. Ominaisuudet johdetaan kolmesta erilaisesta lähteestä: elokuvien kokonaiskulutuksesta (suosio), yksittäisten käyttäjien kulutustottumuksista (samankaltaisuus) ja elokuviin liittyvästä metadatasta (tuoreus). Elokuvan *suosiolla* tarkoitetaan yksinkertaisesti sen katsoneiden käyttäjien prosenttiosuutta kaikista käyttäjistä. *Samankaltaisuus* mittaa käyttäjien katsomien elokuvien samanlaisuutta käyttäen elokuvien *kulutusvektoreiden* kosinietäisyyttä. Käyttäjän taipumus samankaltaisten elokuvien katsomiseen määritetään kaikkien käyttäjän katsomien elokuvien samankaltaisuuspisteiden keskiarvona. Suosituksia muodostettaessa uuden tuotteen samankaltaisuutta vertaillaan käyttäjän katsomien elokuvien kanssa. *Tuoreutta* mitataan vertausajan ja elokuvan ensimmäisen katsomiskerran aikaleimojen erotuksella.



Kuva 4. Hybridimenetelmän toimintakaavio. [Mourão et al., 2013]

Kuva 4 esittää suositusten muodostuksen eri vaiheita. Ensimmäisessä vaiheessa muodostetaan suositeltavien elokuvien lista käyttämällä jotain yhteistoiminnallista menetelmää. Sen jälkeen tuotteille johdetaan sisältöön perustumattomat attribuutit, joita käytetään kolmannessa vaiheessa sisältöperustaisten suosittelujen muodostamisessa. Lopulta neljännessä vaiheessa sisältöperustaiset ja yhteistoiminnalliset suositukset yhdistetään tiettyjen painotusten mukaan. Tuloksena on järjestetty lista käyttäjälle kaikkein hyödyllisimmiksi arvioiduista elokuvista.

Yhteistoiminnallisen (YT) menetelmän tuottamat suositukset saattavat sellaisenaan heijastaa käyttäjien mieltymyksiä valittujen sisältöön liittymättömien ominaisuuksien suhteen, joten sisältöperustaisessa (SP) vaiheessa käytetään hieman tavallisesta poikkeavaa lähestymistapaa tuotteiden arvioinnissa.

Sen sijaan, että laskettaisiin suoraan SP-arvo, joka sitten yhdistetään YT-arvon kanssa, järjestelmä laskee SP-delta arvon, joka luontaisesti heijastaa astetta, jolla yhteistoiminnalliset suositukset kuvaavat mallinnettuja käyttäjäpreferenssejä. Näin ollen SP-delta ei muuta elokuvan sijoitusta listassa, jos se on jo valmiiksi sijalla, joka kuvastaa käyttäjän mieltymyksiä sisältöön perustumattomien attribuuttien suhteen.

Käytännössä suosituslistan muodostamiseen käytetyt arvot yhdistetään kaavalla

$$Arvo(u_k, m_j) = \alpha \cdot \frac{R_{u_k}(m_j)}{\max R_{u_k}(*)} + (1 - \alpha) \cdot \frac{G_{u_k}(m_j)}{\max G_{u_k}(*)},$$

jossa $R_{u_k}(m_j)$ tarkoittaa elokuvalle m_j laskettua YT-arvoa ja $G_{u_k}(m_j)$ sisältöön liittymättömistä attribuuteista laskettua SP-arvoa, koskien käyttäjää u_k. Molemmat normalisoidaan, koska arvot vaihtelevat erillisillä mittakaavoilla. Kaavassa α on painotuskerroin, jonka arvo on välillä]0, 1]. Painotuskerroin vaikuttaa sisältöön liittymättömien mieltymysten vaikutuksen suuruuteen suosittelussa. Kertoimen optimaalinen arvo luultavasti vaihtelee kullakin käyttäjällä, mutta sen käyttäjäkohtaisen määrityksen hankaluuden vuoksi Mourão ja muut [2013] käyttivät samaa painotuskerrointa kaikille käyttäjille. Huomionarvoinen asia on, että menetelmä voidaan helposti sisällyttää mihin tahansa yhteistoiminnalliseen suosittelujärjestelmään, kunhan suositeltavista tuotteista voidaan havaita joitain käyttäjän mieltymyksiin vaikuttavia sisältöön liittymättömiä ominaisuuksia.

Menetelmän suorituskykyä elokuvien suosittelussa testattiin kolmella eri aineistolla. Testaukseen käytettiin MovieLens 1M ja 10M aineistoja ja Netflix tilausvideopalvelun julkaisemaa sen omista käyttäjistä ja elokuvista muodostettua aineistoa. Taulukossa 6. esitellään eri aineistojen ominaisuuksia.

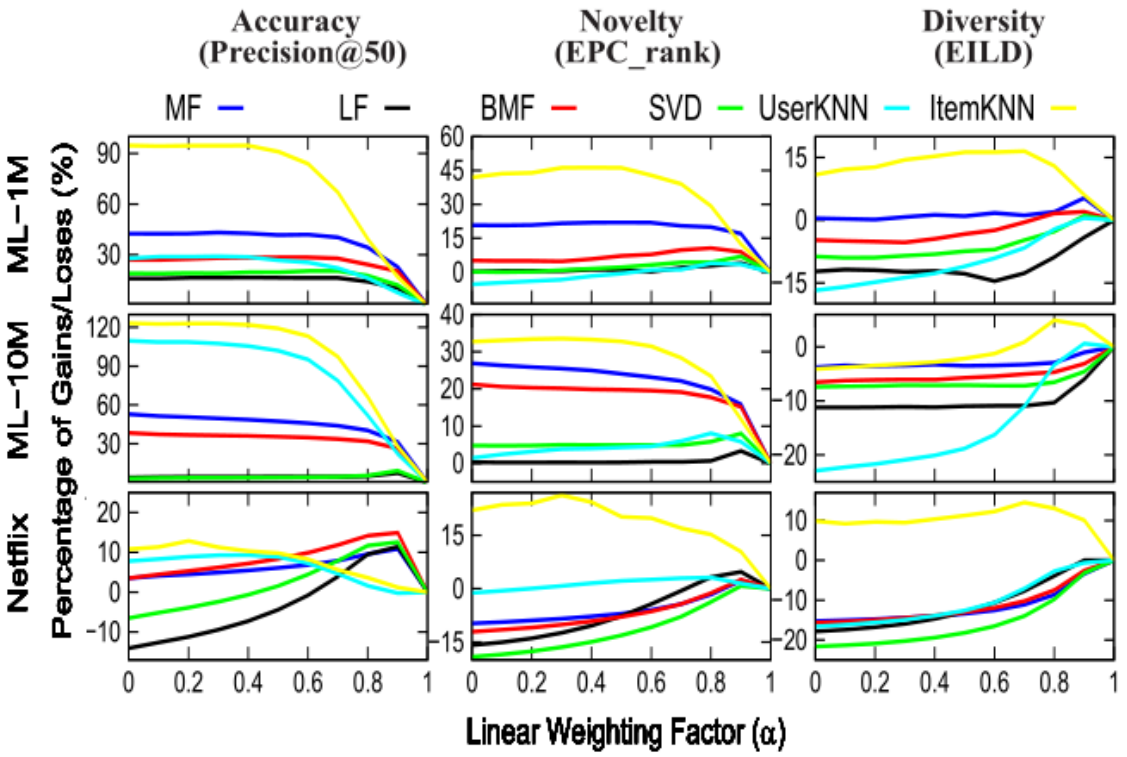
	MovieLens 1M	MovieLens 10M	Netflix
Käyttäjiä	6000	72000	480189
Elokuvia	4000	10000	17770
Arvosteluja	1 miljoona	10 miljoonaa	100 miljoonaa
Aika	149 viikkoa	671 viikkoa	310 viikkoa

Taulukko 6. Testaukseen käytetyt aineistot.

Järjestelmän yhteistoiminnallisena osana käytettiin kuutta eri avoimen lähdekoodin MyMediaLite-kirjaston tarjoamaa algoritmia. Aineistot jaettiin opetus- ja testausosioihin niin, että järjestelmän opetukseen käytettiin 70 % ja testauk-

seen 30 % datasta. Analyysissä otettiin huomioon kolme eri suositusten laatuun liittyvää ominaisuutta laajemman käsityksen saamiseksi. Testeissä arvioitiin suositusten *tarkkuutta*, *uutuudenviehätystä* ja *monipuolisuutta*. Tarkkuutta arvioitiin yksinkertaisesti laskemalla käyttäjälle muodostetun suosituslistan sisältämien elokuvien lukumäärä testiaineistossa. Uutuudenviehätyksen ja monipuolisuuden arviointiin käytettiin kirjallisuudessa aiemmin esitettyä analyysikehikkoa. Analyysi perustuu kunkin yhteistoiminnallisen algoritmin a_n tuottamien tulosten vertailuun esitetyn hybridimenetelmän tuottamien tulosten kanssa, kun se on yhdistetty algoritmiin a_n. Tarkoituksena ei ole kehitetyn menetelmän vertailu muihin suosittelumenetelmiin, vaan sisältöön liittymättömien ominaisuuksien hyödyllisyyden arviointi suosittelussa ylipäänsä.

Kuva 5. esittää sisältöön liittymättömien ominaisuuksien hyödyntämisen vaikutusta eri yhteistoiminnallisten algoritmien tuottamien suositusten laatuun kolmella eri osa-alueella eri painotuskertoimilla käytetyissä aineistoissa.

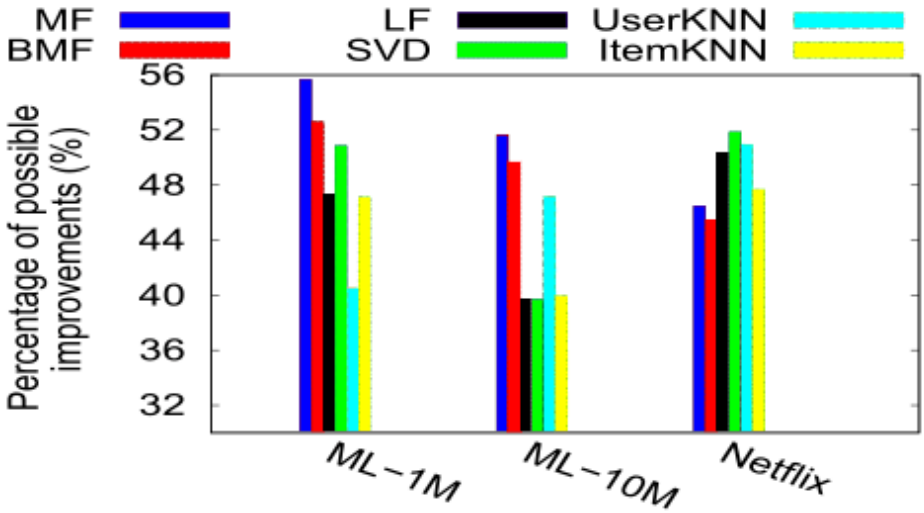


Kuva 5. Suositusten laadun analyysi. [Mourão et al., 2013]

Sisältöön liittymättömien ominaisuuksien hyödyntäminen tuotti tarkempia suosituksia lähes kaikilla testatuilla algoritmeilla sekä aineistoilla ja sillä oli vaikutusta myös suositusten uutuudenviehätykseen ja monipuolisuuteen. Yleisesti ottaen suhteessa suurin vaikutus havaittiin kunkin aineiston kohdalla huonoiten pärjänneiden yhteistoiminnallisten algoritmien kohdalla. Menetelmä riitti havaitsemaan käyttäjälle kiinnostavia seikkoja ja pelastamaan huonot suositukset.

Analyysissa oltiin kiinnostuneita myös siitä kuinka usein hybridimenetelmä tuottaa käyttäjälle parempia suosituksia verrattuna pelkkiin yhteistoiminnallisiin algoritmeihin. Kuva 6 esittää algoritmi- ja aineistokohtaisesti nii-

den käyttäjien osuuden kaikista käyttäjistä, joille muodostetut suositukset paranivat edes jonkin verran menetelmän ansiosta.



Kuva 6. Menetelmästä hyötyä saaneiden käyttäjien osuus. [Mourão et al., 2013]

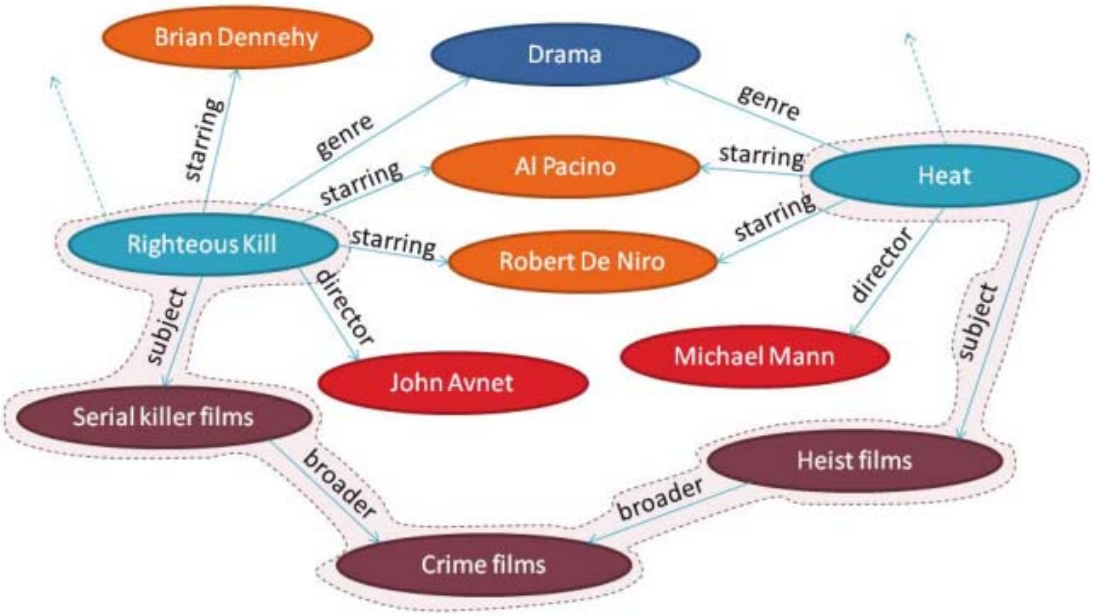
Sisältöön liittymättömien ominaisuuksien yhdistämisestä yhteistoiminnalliseen suositteluun oli suurimmassa osassa tapauksia hyötyä yli 40 prosentille käyttäjistä. Joissain tapauksissa menetelmä saattoi huonontaa joidenkin käyttäjien saamia suosituksia, johtuen käytetyn yhteistoiminnallisen algoritmin toimintatavasta, mutta huonompia suosituksia saaneiden käyttäjien osuus jäi alle kymmeneen prosenttiin. [Mourão et al., 2013]

4.5. Linkitettyä avointa dataa hyödyntävä sisältöperustainen suosittelujärjestelmä

Semanttisen Webin (Semantic Web) ja Linkitetyn avoimen datan (Linked Open Data, LOD) aloitteen ansiosta verkossa on valtavat määrät RDF-dataa (Resource Description Framework), mutta tietomäärää hyödyntävät sovellukset ovat harvassa. Di Noia ja muut [2012] kehittivät LOD-aineistoja hyödyntävän elokuvien suosittelujärjestelmän. Menetelmä on vektoriavaruusmalliin perustuva sisältöperustainen järjestelmä, jota opetetaan Dbpedia-, LinkedMDB- ja Freebase-nimisistä LOD-aineistoista peräisin olevalla datalla.

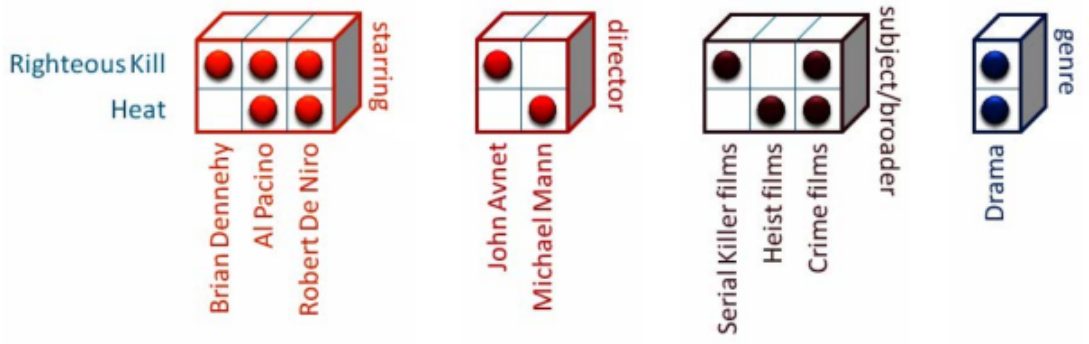
Sisältöperustaiseen suositteluun tarvittavat elokuvien väliset suhteet on melko yksinkertaista selvittää RDF-datasta käyttäen SPARQL-kyselykieltä (SPARQL Protocol and RDF Query Language). Ilmeisten yhdistävien ominaisuuksien, kuten saman ohjaajan tai näyttelijöiden lisäksi aineistosta löytyy myös muunlaisia suhteita elokuvien välillä. Esimerkiksi jatko-osat ovat suorassa yhteydessä niitä edeltäviin elokuviin. Data sisältää myös genreistä erillisen hierarkkisen järjestelmän elokuvien kategorisointiin *aiheen* mukaan, mikä mahdollistaa implisiittisten yhteyksien havaitsemisen elokuvien välillä. Esimerkki

tällaisesta yhteydestä on korostettu kuvassa 7. Datasta voidaan johtaa noin 20 erilaista huomioitavaa ominaisuutta, mutta esimerkissä näytetään selkeyden vuoksi vain neljä: näyttelijät, ohjaaja, aihe ja genre.



Kuva 7. Esimerkki elokuvien RDF-graafista. [Di Noia et al., 2012]

Elokuvien samankaltaisuuden arviointiin käytetään vektoriavaruusmallia (Vector Space Model, VSM), jota käytetään yleensä tekstidokumentteihin kohdistuviin hakuihin. Esitetyssä suosittelumenetelmässä perinteinen vektoriavaruusmalli semantisoidaan RDF-graafin käsittelyä varten. Graafi voidaan esittää vierusmatriiseina, joista kukin kuvaa jonkin ontologisen ominaisuuden (kuten genren) ja elokuvien väliset yhteydet. Kahden elokuvan samankaltaisuus määritetään ominaisuuskohtaisesti käyttäen elokuvia kuvaavien vektoreiden välisen kulman kosinia. Kuvassa 8 esitetään kuvan 7 graafi vierusmatriiseina.



Kuva 8. Esimerkkigraafin esitys vierusmatriiseina. [Di Noia et al., 2012]

Järjestelmässä päätettiin käyttää binääristä ”tykkäysasteikkoa” käyttäjien mieltymysten kartoitukseen. Käyttäjän u profiili muodostetaan kaavan

$$profiili(u) = \{(m_j, v_j) \mid \begin{cases} v_j = 1, kun\ u\ tykkää\ m_j\text{:stä}, \\ v_j = -1\ muulloin \end{cases}\} \quad \#$$

mukaan. Käytännössä siis käyttäjän elokuvalle antaman arvostelun v_j arvo on joko *tykkää* (1) tai *tuntematon* (-1). Jotta voidaan arvioida uuden elokuvan m_i hyödyllisyyttä käyttäjälle u, täytyy elokuvan m_i kaikkien ominaisuuksien samankaltaisuusarvot yhdistää. Järjestelmää testattiin kahdella eri yhdistämistavalla, joista parempiin tuloksiin johti

$$r(u, m_i) = \frac{\sum_{m_j \in profiili(u)} \frac{v_j \cdot \sum_p a_p \cdot sim^p(m_j, m_i)}{P}}{|profiili(u)|}.$$

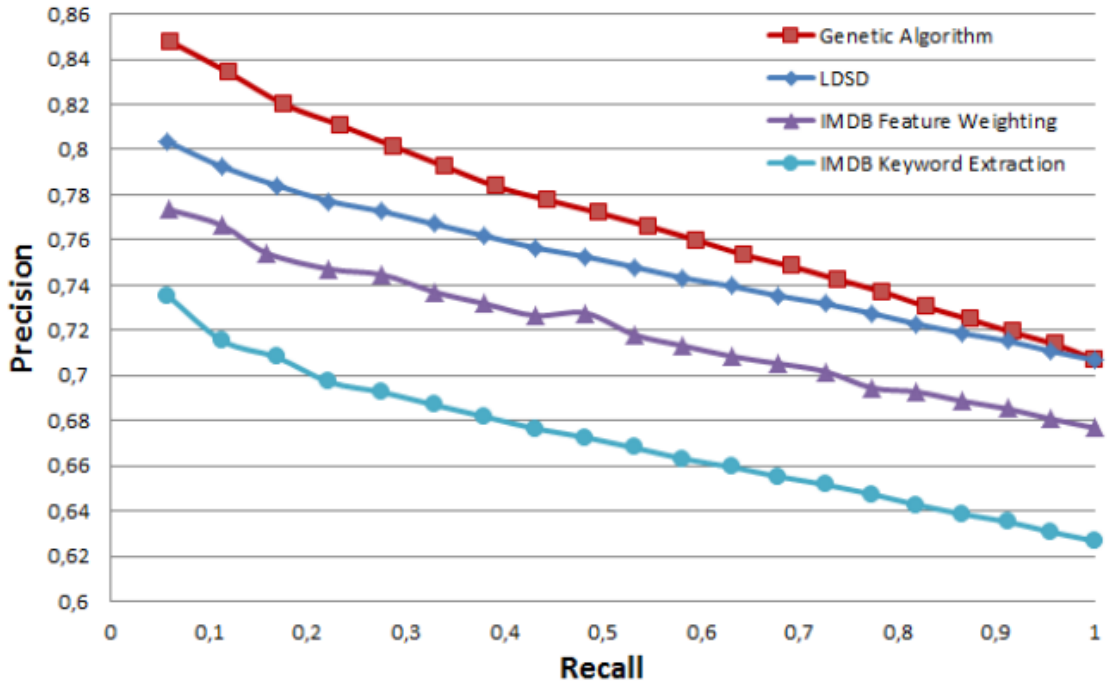
Kaavassa $sim^p(m_j, m_i)$ tarkoittaa elokuvien m_j ja m_i samankaltaisuutta ominaisuuden p suhteen. P on huomioitavien ominaisuuksien lukumäärä. Ominaisuuksilla on painotuskerroin a_p, joka lasketaan käyttäjäkohtaisesti kullekin ominaisuudelle geneettisen opetusalgoritmin avulla. Tarkoituksena on, että järjestelmä oppii syyt käyttäjän mieltymysten takana. Jos käyttäjä esimerkiksi katsoo paljon elokuvia samoilta ohjaajilta genrestä riippumatta, järjestelmä antaa *ohjaaja*-ominaisuudelle suuremman arvon kuin *genre*-ominaisuudelle. Uuden käyttäjän kohdalla painotuskertoimet saavat datasta tilastollisesti johdetut keskimääräiset arvot. Yksi sisältöperustaisen suosittelun vahvuuksista yhteistoiminnalliseen suositteluun verrattuna on, että käyttäjille voidaan kertoa miksi juuri tätä elokuvaa suositellaan heille, mikä auttaa kehittämään luottamusta järjestelmää kohtaan.

Järjestelmää testattiin MovieLens 1M -aineistolla. LOD-aineistoista peräisin olevat elokuvat liitettiin yhteen MovieLens-aineiston elokuvien kanssa vertailemalla julkaisuvuosia ja nimiä käyttäen Levenšteinin etäisyyttä. MovieLens-aineisto sisälsi noin 300 elokuvaa, joille ei löytynyt vastinetta LOD-aineistoista ja manuaalisessa tarkistuksessa havaittiin 60 tapausta, joissa automaattinen yhdistäminen oli tuottanut virheellisen tuloksen ja vaadittiin käsin tehtävää korjaamista. MovieLens-aineisto käyttää käyttäjäprofiileissaan viisiportaista arvosteluasteikkoa, joka piti kuvata menetelmän käyttämäksi tykkäysasteikoksi. Se tehtiin laskemalla kunkin käyttäjän u antamien arvosanojen keskiarvo $\bar{r}(u)$ ja muodostamalla profiili kaavan

$$profiili(u) = \{(m_j, v_j) \mid \begin{cases} v_j = 1, kun\ r(u, m_j) \geq \bar{r}(u), \\ v_j = -1\ muulloin \end{cases}\}$$

mukaan, kun $r(u, m_j)$ tarkoittaa MovieLens-aineiston käyttäjän u arvosanaa elokuvalle m_j.

Testausta varten MovieLens-aineisto jaettiin opetus- ja testausosioihin ja suositusten arviointiin käytettiin kahta eri mittaria, jotka olivat *tarkkuus* ja *takaisinotto* (recall). Tarkkuudella tarkoitetaan käyttäjälle relevanttien elokuvien osuutta suosituslistan sisältämistä elokuvista ja takaisinotolla suosituslistan sisältämien relevanttien elokuvien osuutta kaikista käyttäjälle relevanteista elokuvista. Järjestelmää vertailtiin muutamiin muihin sisältöperustaisiin menetelmiin. Vertailuun valittiin toinen LOD-aineistoja hyödyntävä menetelmä Linked Data Semantic Distance (LDSD) ja kaksi Internet Movie Database (IMDb) -palvelusta kerättyä dataa käyttävää algoritmia. LDSD on kehitetty alun perin musiikin suositteluun, mutta vertailua varten sitä muokattiin toimimaan elokuvien yhteydessä.



Kuva 9. Esitetyn menetelmän vertailu muihin sisältöperustaisiin menetelmiin. [Di Noia et al., 2012]

Kuva 9 esittää testien tuloksia eri menetelmillä tarkkuuden ja takaisinoton osalta. Ylin, neliöillä merkitty kaari liittyy esitettyyn menetelmään kun käytettiin geneettisellä algoritmilla laskettuja painotuskertoimia. Algoritmi suoriutui siis vertailluista menetelmistä parhaiten tarkkojen suositusten muodostamisessa. Alimpaan, palloilla merkittyyn kaareen liittyvä menetelmä käyttää samankaltaisuuden arviointiin vain tekstistä poimittuja avainsanoja, eikä dataa ole siis millään tavalla strukturoitu. Kaikki muut vertaillut menetelmät käyttävät jollain tavalla järjestettyä dataa ja suoriutuvat testeistä sen takia huomattavasti paremmin. [Di Noia et al., 2012]

5. Yhteenveto

Suosittelujärjestelmät ovat tärkeässä asemassa tuotteiden saattamisessa asiakkaiden saataville etenkin suuren valikoiman sisältävillä sovellusalueilla. Tässä tutkielmassa luotiin katsaus erilaisiin suosittelumenetelmiin elokuvien toimialueelta. Aiheeseen tutustuttiin ensin yleisellä tasolla esitellen erilaisia ongelman lähestymistapoja ja niiden hyviä ja huonoja puolia. Lopulta käytiin läpi muutaman kirjallisuudessa esitetyn elokuvien suosittelujärjestelmän toimintaa tarkemmin ja esiteltiin suosittelujärjestelmien toiminnan arviointiin yleisesti käytetty MovieLens-aineisto.

Viiteluettelo

[Di Noia et al., 2012] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker, Linked open data to support content-based recommender systems. In: *Proc. of the 8th International Conference on Semantic Systems.* 1-8.

[GroupLens, 2013] GroupLens research, MovieLens datasets. http://grouplens.org/datasets/movielens/. Checked 2.12.2013.

[Lam and Riedl, 2004] Shyong K. Lam and John Riedl, Shilling recommender systems for fun and profit. In: *Proc of the 13th International Conference on World Wide Web*. 393-402.

[Lee et al., 2002] Meehee Lee, Pyungseok Choi, and Yongtae Woo, A hybrid recommender system combining collaborative filtering with neural network. In: Paul De Bra, Peter Brusilovsky, and Ricardo Conejo (eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 2002, 531-534.

[Mourão et al., 2013] Fernando Mourão, Leonardo Rocha, Joseph A. Konstan, and Wagner Meira, Jr., Exploiting non-content preference attributes through hybrid recommendation method. In: *Proc. of the 7th ACM Conference on Recommender Systems.* 177-184.

[Pucci et al., 2007] Augusto Pucci, Marco Gori, and Marco Maggini, A Random-Walk Based Scoring Algorithm Applied to Recommender Engines. In: Olfa Nasraoui, Myra Spiliopoulou, Jaideep Srivastava, Bamshad Mobasher, and Brij Masand (eds.), *Advances in Web Mining and Web Usage Analysis*. Springer, 2007, 127-146.

[Rajaraman and Ullman, 2012] Anand Rajamaran and Jeffrey David Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2012.

[Ricci et al., 2010] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, *Recommender Systems Handbook*. Springer, 2011.

[Sinha and Swearingen, 2001] Rashmi Sinha and Kirsten Swearingen, Comparing recommendations made by online systems and friends. In: *Proc of the*

2nd DELOS Network of Excellence Workshop on Personalization and Recommender Systems in Digital Libraries (June 2001), 18-20.

[Takács et al., 2009] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk, Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research* **10** (2009), 623-656.

Web-käytön tiedonlouhintaprosessi

Joni Hämäläinen

Tiivistelmä.
Web-käytön tiedonlouhinnassa tutkitaan käyttäjän käyttäytymistä web-sivustolla. Ongelmana on löytää merkitsevää tietoa alun perin testauskäyttöön suunnitelluista web-palvelinlokeista. Haasteena on löytää oikeat tiedonlouhintamenetelmät, joilla haluttu tieto löydetään tehokkaasti. Tutkielmassa käydään läpi web-käytön tiedonlouhinnan prosessi ja esitellään yleisimpiä tiedonlouhintamenetelmiä. Lopputuloksena on lyhyt katsaus useisiin eri menetelmiin. Samoja tiedonlouhintamenetelmiä voidaan hyödyntää useiden eri ongelmien ratkaisuissa.

Avainsanat ja -sanonnat: web-tiedonlouhinta, palvelinlokit, WUM, louhintamenetelmät.
CR-luokat: I.5.0, H.2.8, H.3.3

1. Johdanto

Tiedon ja sen käyttäjien määrä Internetissä lisääntyy huomattavaa vauhtia. Yhden päivän aikana useilla eri sivustoilla tehdään lukemattomia sivukäyntejä. Näistä käynneistä kertyy runsaasti tietoa, jota ei välttämättä hyödynnetä. Esimerkiksi suurilla yrityksillä on usein verkkokauppa, jonka käytöstä kertyy päivittäin paljon tietoa, mutta tuota tietoa ei käytetä hyödyksi.

Web-louhintaa on tutkittu runsaasti jo 1990-luvun puolivälistä alkaen, jolloin Internet oli uutta ja käyttäjiä vielä vähän, mutta nykyisin web-louhintaa tutkitaan hyvin laaja-alaisesti. Heterogeenisyydestä johtuen alaa kuvaavaa yleiskuvaa on hankala muodostaa. Siten selkeästi asian jaottelevat artikkelit ovatkin peräisin 2000-luvun alun tienoilta asti.

Tutkielman tavoitteena on havainnollistaa, miten arkipäiväisestä suuresta tietomassasta on mahdollista louhia merkittävää tietoa, ja millaisia tiedonlouhintamenetelmiä se vaatii.

Webin tiedonlouhinta jaetaan kolmeen osa-alueeseen. Tutkielma keskittyy niistä yhteen, web-käytön tiedonlouhintaan, jonka tavoitteena on tutkia käyttäjän suhdetta web-sivustoon. Tutkielmassa keskitytään web-palvelimilta louhittavaan tietoon, koska suurin osa alan tutkimuksista pohjautuu kyseiseen tietoon.

Web-palvelimilta louhittua tietoa voidaan hyödyntää useisiin eri tarkoituksiin. Esimerkiksi on mahdollista ennustaa tulevaisuuden trendejä, personoida sivujen sisältöjä käyttäjäkohtaisesti sekä parantaa sivuston rakennetta. Tiedosta on hyötyä niin liiketoiminnan suunnittelijoille kuin verkkosivuston ylläpitäjille.

Tutkielma rakentuu seuraavasti: Luvussa 2 esitellään web-tiedonlouhinta yleisellä tasolla. Web-louhinnan kaikki kolme osa-aluetta esitellään lyhyesti, jolloin web-käytönlouhinnan sijoittuminen web-louhinnan tutkimusalalla tulee ilmi. Tämän jälkeen käydään läpi, mistä web-louhintaprosessissa louhittava tieto voidaan hankkia. Sen jälkeen luvussa 3 aloitetaan web-käytön tiedonlouhinnan käsittely. Esitellään tiedon ominaisuuksia sekä paneudutaan web-palvelinlokien merkitykseen koko prosessille. Myös web-käytön tiedonlouhintaprosessin vaiheet esitellään lyhyesti. Luvussa 4 käydään läpi louhintaprosessin ensimmäinen vaihe, tiedon esikäsittely. Luvussa 5 esitellään erilaisia web-käytön louhinnassa hyödynnettäviä tiedonlouhintamenetelmiä. Luvussa 6 käydään läpi tiedon analysoinnin merkitys. Luku 7 on yhteenveto.

2. Web-louhinta

Webin tiedonlouhinta (web mining) on yksi *tiedonlouhinnan* (data mining) osa-alue. Siinä hyödynnetään *tietämyksen muodostamisen* (knowledge discovery) tekniikoita, joiden avulla suurista tietomassoista löydetään kiinnostavaa tai piilevää tietoa [Patil and Patil, 2012].

Webin tiedonlouhinta jaetaan kolmeen alueeseen [Srivastava *et al.*, 2000]:

1. *Sisällön louhinta* (web content mining)
2. *Rakenteen louhinta* (web structure mining)
3. *Käytön louhinta* (web usage mining).

Sisällön louhinnan osa-alue keskittyy sisällöstä löytyvän tiedon, erityisesti tekstin, louhimiseen. Tyypillisiä sovelluksia ovat *sisältöpohjainen kategorisointi* (content-based categorization) sekä *sisältöperustainen arvojärjestys* (content-based ranking) [Facca and Lanzi, 2005].

Rakenteen louhinnassa kiinnostus kohdistuu web-sivuston rakenteeseen: joko HTML-merkintäkielen muodostamaan sivuston *puurakenteeseen* (intra-page) tai sivustolla oleviin hyperlinkkeihin, jotka linkittävät sivuston muut sivut *yhdeksi rakenteeksi* (inter-page) [Srivastava *et al.*, 2000]. Sovelluksia ovat esimerkiksi *linkkiperustainen kategorisointi* (link-based categorization) ja *web-sivuston takaisinmallinnus* (reverse engineering of Web site models) [Facca and Lanzi, 2005].

Käytön louhinnassa tavoitteena on löytää mielenkiintoista tietoa sivuston käyttäjistä. Louhittua tietoa voidaan hyödyntää esimerkiksi sivuston rakenteen

suunnittelussa, sivuston personoinnissa käyttäjille sekä esimerkiksi verkkokaupoissa tulevaisuuden trendien ennustamiseen [Cooley *et al.,* 1997]. Tässä tutkielmassa keskitytään käsittelemään käytön louhinnan sovellusalaa.

Web-tiedonlouhinnassa tietoa voidaan kerätä joko asiakkaalta, palvelimelta, välityspalvelimelta tai web-tiedonkeruuseen tarkoitetuista tietokannoista [Srivastava *et al.,* 2000]. *Asiakkaalla* (client) tarkoitetaan käytännössä Internet-selainta. *Palvelimella* (server) tarkoitetaan web-palvelinta, jossa web-sivusto sijaitsee. *Välityspalvelimella* (proxy) tarkoitetaan palvelinta, joka välittää tietoa asiakkaan ja web-palvelimen välillä. Tietokannoilla tarkoitetaan tietosäilöjä, joiden tehtävä on säilöä web-tietoa organisaatioiden sisällä. Valitut tiedonlouhintamenetelmät riippuvat tiedon sijainnista ja tyypistä.

Web-käyttötietoa on perinteisesti louhittu kolmesta eri lähteestä:

1. Web-palvelimilta
2. Välityspalvelimilta
3. Asiakkailta (selaimista).

Jokaisella lähteellä on omat vahvuutensa ja heikkoutensa. Näistä suosituin ja luotettavin tiedonlähde on kuitenkin web-palvelimilta kerätty tieto, jolloin louhintaprosessin syötteenä on sivukäynneistä kerätty lokitiedosto [Hussain *et al.,* 2010].

Tiedon kerääminen asiakkaasta on hankalaa, koska jokainen selain on muokattava tiedonkeruuta varten, ja siten vaatii käyttäjän myötävaikutusta. Asiakkaalta kerätty tieto on kattavinta, sillä se kuvaa yksiselitteisesti asiakkaan käyttäytymistä sivustolla [Hussain *et al.,* 2010]. Tämä on selkeä etu verrattuna palvelinlokien käyttöön, joissa asiakkaat pitää tunnistaa ja eritellä suuresta tietomassasta.

Välityspalvelimissa tiedon keruun tekee haastavaksi asiakkaiden ja välitettävien palvelimien määrä: välityspalvelin välittää useiden palvelimien tietoja useille eri asiakkaille [Hussain *et al.,* 2010]. Yksittäisen käyttäjän vierailuja yhdellä sivustolla on hankala koostaa tiedon määrän takia [Facca and Lanzi, 2005]. Web-palvelimilla on etu välityspalvelimiin nähden, sillä tiedon keruu keskittyy vain kyseisen palvelimen liikenteeseen.

3. Web-käytön tiedonlouhinta

Web-käytön louhinnan (web usage mining, *WUM*) tavoitteena on ymmärtää käyttäjän käyttäytymistä web-sivustolla [Patil and Patil, 2012]. Kun käyttäytymisestä on kerätty riittävästi tietoa, voidaan tiedonlouhinnan menetelmillä löytää kiinnostavaa ja merkityksellistä tietoa, joka muuten jäisi piiloon. Tätä tietoa voidaan käyttää hyödyksi esimerkiksi sivuston rakenteen suunnittelussa

tai sivuston personoinnissa käyttäjille. Web-käytön louhinnasta saatavaa tietoa voidaan myös käyttää kaupallisesti hyödyksi, esimerkiksi tulevaisuuden trendien ennustamisessa, mainostusten kohdentamisessa tietyille asiakasryhmille ja yhteensopivien tuotteiden myynnin lisäämisessä suositusten avulla [Cooley *et al.*, 1997].

3.1. Tiedon abstraktiot

Tiedonlähteistä saatavista tiedoista voidaan identifioida eri abstraktioita. Srivastava ja muut [2000] erottelevat seuraavat abstraktiot: *käyttäjät (*users*), istunnot* (server sessions), *tapahtumat* (episodes), *klikkausvirrat* (click streams) ja *sivunäkymät* (page views).

Käyttäjä on yksi web-selainta käyttävä henkilö. Srivastava ja muut [2000] huomauttavat, että yksittäisiä käyttäjiä on vaikea tunnistaa, koska käyttäjä voi käyttää sivustoa usealla eri koneella tai selaimella. *Sivunäkymä* koostuu tekstistä, grafiikasta ja skripteistä. Sivunäkymä on vastaus käyttäjän pyyntöön näyttää sivu. *Klikkausvirta* on sarja sivunäkymäpyyntöjä, jota kutsutaan myös navigointipoluksi (*navigational path)*.

Istunto on joukko yhden käyttäjän sivunäkymäpyyntöjä, toisin sanoen yksi istunto kuvaa käyttäjän vierailua verkkosivustolla. Usein yhden istunnon rajaamiseen käytetään 30 minuutin aikakatkaisua. Jos käyttäjä ei ole tehnyt sivulatauksia 30 minuuttiin, katsotaan istunto päättyneeksi [Cooley *et al.*, 1997]. Tapahtumaksi kutsutaan istunnon alijoukkoa, josta voidaan erottaa jokin semanttinen merkitys. Esimerkki tapahtumasta voisi olla käyttäjän ostoprosessi verkkokaupassa: tuotteiden valinta, ostoskorin vahvistus ja maksutapahtuma.

Abstraktiot ovat tärkeitä, jotta merkityksettömästä datasta voidaan tunnistaa merkityksellisiä kokonaisuuksia. Useimmiten tärkeimpiä abstraktioita web-käytönlouhinnan kannalta ovat käyttäjät ja istunnot, koska ollaan kiinnostuneita käyttäjistä ja heidän liikkeistään sivustolla.

3.2. Web-palvelinlokit

Web-käytön louhintaa kutsutaan myös englanninkielisellä termillä web log mining, joka viittaa lokien keskeiseen asemaan web-käytön louhinnassa. Vaikka käyttötietoa voidaan louhia myös asiakkaalta ja välityspalvelimelta, Hussain ja muut [2010] toteavat, että suurin osa web-käytön louhinnan tutkijoista hyödyntää palvelinlokeja tutkimustensa louhintaprosesseissa. Tässä tutkielmassa sivuutetaan asiakkailta ja välityspalvelimilta saatava tieto ja keskitytään web-lokeista louhittavaan tietoon.

Web-palvelimen loki on tärkeä tiedonlähde, koska se tallentaa kaikkien web-sivustolla vierailevien kävijöiden liikkeet (sivulataukset, ts. klikkaukset)

[Srivastava *et al.*, 2000]. Tällaista lokia kutsutaan englanninkielisellä termillä access log eli käyttöloki. Usein lokit ovat tekstitiedostoja, jotka vaihtelevat muodoltaan. On olemassa useita standardoituja lokiformaatteja, esimerkiksi Common Log Format ja Extended Log Format [Facca and Lanzi, 2005]. Myös useilla järjestelmillä voi olla oma lokiformaattinsa, esimerkiksi Microsoft IIS. Yleisesti tallennettavia tietoja lokeissa ovat mm. käyttäjän IP-osoite, selain sekä aikaleima, jolloin sivupyyntö on suoritettu. Web-lokit eivät välttämättä tallenna käyttäjän jokaista sivulla käyntiä, sillä välimuistista tulevat sivunlataukset eivät jätä lokiin jälkeä, vaan sivunäkymä käyttäjän selaimeen tulee joko selaimen tai välityspalvelimen välimuistista [Suneetha and Krishnamoorti, 2009]. Myöskään selaimen Takaisin-painikkeen käyttö ei tallennu palvelinlokiin. Näiden seikkojen takia lokeista louhittavasta tiedosta on hankala muodostaa absoluuttista totuutta.

Käyttölokin lisäksi usein kerätään tietoa myös viittauksista. *Viittausloki (referrer log)* on lokitietoa, jolla pystytään tunnistamaan mikä oli käyttäjän vierailema edellinen sivu. Lokiin merkitään viittaus sivusta, jolta käyttäjä saapui nykyiselle sivulle [Hussain *et al.*, 2010]. Viittausloki voi olla oma tiedostonsa tai viittaukset voivat sisältyä käyttölokiin. Viittauslokien avulla voidaan jäljittää käyttäjän liikkumista sivuston sisällä. Esimerkiksi välimuistista ladatut sivut voidaan tunnistaa viittauslokin avulla.

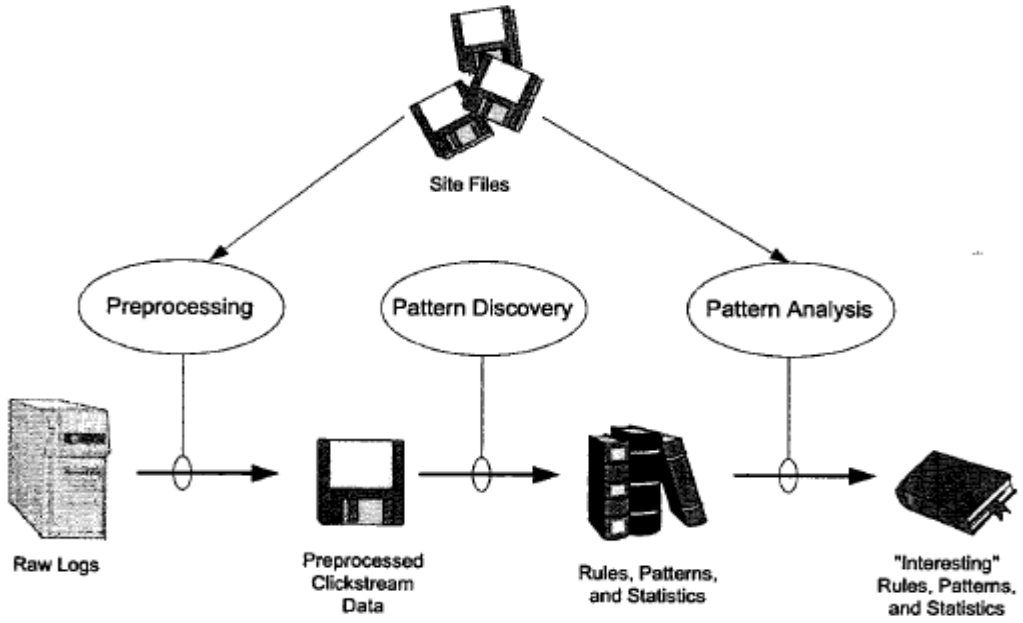
Web-lokit ovat suuria sekä tiedonmäärältään että moniulotteisuudeltaan. Useimmiten lokeissa käyttötieto on niukkaa, sillä yhden istunnon aikana käyttäjä vierailee yleensä vain pienessä osassa kaikista mahdollisista sivustolla sijaitsevista web-sivuista [Hasan *et al.*, 2009]. Hasanin ja muiden [2009] mukaan nämä lokien ominaisuudet tuovat esiin kaksi ongelmaa. Ensimmäisenä tiedon määrä ja moniulotteisuus lisäävät tiedonlouhintaprosessin laskennallista vaativuutta. Toiseksi em. käyttötiedon niukkuus heikentää louhinta-algoritmien mahdollisuuksia löytää merkitsevää ja mielenkiintoista tietoa sivustolla vierailevien käyttäytymisestä.

Palvelinlokit on alun perin suunniteltu vain testausta varten. Hussain ja muut [2010] toteavatkin, että alkuperäistä lokitiedostoa ei voida käyttää suoraan web-käytön louhintaprosessissa, koska se sisältää paljon epäolennaista tietoa. Siksi lokien *esikäsittely* on web-käytön louhintaprosessin ensimmäinen vaihe.

3.3. Web-käytön tiedonlouhintaprosessi

Web-käytön tiedonlouhintaprosessi jaetaan yleensä kolmeen vaiheeseen (Kuva 1). Prosessin ensimmäinen vaihe on raa'an lokidatan *esikäsittely* (preprocessing), jossa lokista pyritään löytämään vain merkitsevä tieto seuraavia vaiheita var-

ten. Esikäsittelyn lopputuloksena on sopivaan muotoon muokattu, tietorakenteessa sijaitseva tieto, jota *tiedonlouhintavaiheessa* (pattern discovery) louhitaan. Louhinnan jälkeen tieto ei yleensä ole esitettävässä muodossa, joten sitä täytyy *analysoida* (pattern analysis), joka on web-käytön tiedonlouhintaprosessin viimeinen vaihe. Analysoinnin tuloksena on usein visuaalinen esitys louhintatuloksesta, eli mielenkiintoisesta tiedosta.



Kuva 1: Web-käytön louhintaprosessin vaiheet [Srivastava et al., 2000].

Luvuissa 4-6 käydään läpi prosessin vaiheet tarkemmin.

4. Esikäsittely

Esikäsittelyn tavoitteena on parantaa saatavilla olevan tiedon laatua ja siten edistää seuraavien vaiheiden tehokkuutta [Hussain *et al.,* 2010]. Esikäsittely on web-käytön louhintaprosessin haastavin osuus. Haasteena on löytää raa'asta lokitiedostosta merkitsevää tietoa. Esikäsittelyyn kuluu suurin osa koko louhintaprosessiin käytettävästä ajasta, jopa 80 % [Pabarskaite, 2002]. Esikäsittelyn jälkeen lokitiedostosta on saatu eroteltua käyttäjät ja istunnot, mutta myös muunlaista tietoa on mahdollista erottaa. Lopputulos riippuu aina tutkittavan tiedon vaatimuksista.

Palvelinlokeista on saatavilla vain tietoa käyttäjien IP-osoitteista, selaimista sekä klikkausvirroista. Käyttäjän tunnistamisen apuna voidaan käyttää myös selaimen evästeitä, mutta evästeisiin ei voi täysin luottaa, koska käyttäjällä on mahdollisuus poistaa evästeet. Näiden tietojen avulla esikäsittelyssä pyritään tunnistamaan käyttäjät. Srivastava ja muut [2000] luettelevat muutamia tunnistamiseen liittyviä yleisiä ongelmia:

- Yksi IP-osoite / Useita istuntoja: Internet-palveluntarjoajat (ISP) käyttävät usein välityspalvelimia, jolloin yhden IP-osoitteen takaa sivulla voi olla todellisuudessa monta eri käyttäjää.
- Useita IP-osoitteita / Yksi istunto: Jotkin ISP:t antavat käyttäjälle satunnaisesti IP-osoitteen jokaiselle sivupyynnölle, jolloin palvelimen on mahdoton tunnistaa yhtä käyttäjää.
- Useita IP-osoitteita / Yksi käyttäjä: Jos käyttäjä vierailee sivuilla eri päätteillä, IP-osoite vaihtuu usein ja käyttäjää on vaikea identifioida.
- Useita selaimia / Yksi käyttäjä: Jos käyttäjä käyttää useita selaimia, näkyy hän palvelinlokeissa useina eri käyttäjinä.

Esikäsittely voidaan jakaa useisiin eri vaiheisiin. Vaiheiden määrä vaihtelee suuresti eri tutkimusten välillä ja on aina riippuvainen tutkimuksen tavoitteesta. Alla esitetään joitakin eri tutkimuksissa esiintyviä esikäsittelyn vaiheita, joita Hussain ja muut [2010] ovat arvioineet.

4.1. Tiedon puhdistus

Esikäsittelyn ensimmäinen vaihe on *tiedon puhdistus* (data cleaning). Puhdistusvaiheessa kaikki epärelevantti tieto karsitaan pois. Epärelevanttia tietoa ovat esimerkiksi graafisten tiedostojen (kuvat, videot) lataukset sekä erilaisten robottien sivukäynnit (esimerkiksi hakukonerobotit) [Facca and Lanzi, 2005]. Robottien käynnit voidaan tunnistaa heuristisesti: on esimerkiksi huomattu joidenkin robottien skannaavan sivuston leveyshakua käyttäen, jolloin muodostuneet navigointipolut ovat tunnistettavissa palvelinlokeista [Facca and Lanzi, 2005]. Myös mahdolliset järjestelmän ylläpitäjän käynneistä jääneet jäljet [Pabarskaite, 2002] sekä virheelliset sivupyynnöt [Suneetha and Krishnamoorti, 2009] täytyy puhdistaa aineistosta. Puhdistuksen jälkeen käsiteltävän lokin vaatima levytila vähenee. Suneethan ja Krishnamoortin [2009] tutkiessa NASA:n palvelinlokeja lokin tiedostokoko puolittui esikäsittelyn ansiosta.

4.2. Käyttäjien ja istuntojen tunnistus

Kuten aiemmin on jo todettu, käyttäjän tunnistaminen yksiselitteisesti on erittäin vaikeaa. Yleensä palvelinlokien tiedoista käytetään käyttäjien tunnistamiseen IP-osoitteen, käyttöjärjestelmän ja selainversion yhdistelmää [Hussain *et al.*, 2010]. Yuan ja muut [2003] esittävät sääntöjä käyttäjien tunnistamiseen: Jos IP-osoite on uusi, kyseessä on uusi käyttäjä. Jos IP-osoite on sama, mutta käyttöjärjestelmä tai selain eri, oletetaan, että kyseessä on eri käyttäjä.

Usein käyttäjät vierailevat tietyllä aikavälillä samalla sivustolla useita kertoja. Tällöin on kiinnostavaa erottaa tietyn käyttäjän klikkausvirrat eri istuntoihin [Yuan *et al.*, 2003].

Kuten todettu, istuntojen erotteluun on usein käytetty 30 minuutin aikakatkaisua. Jos samalla (IP-osoite, käyttöjärjestelmä, selainversio) -yhdistelmällä (käyttäjällä) on edellisestä sivupyynnöstä aikaa yli 30 minuuttia, voidaan todeta uuden istunnon alkaneen. Suurimmat ongelmat istuntojen tunnistamiseen liittyvät välimuistista ladattaviin sivunäkymiin. Selaimen (tai välityspalvelimen) välimuistista ladatut sivunlataukset eivät tallennu palvelinlokeihin, jolloin lokiperspektiivistä käyttäjän klikkausvirtaan tulee aukkoja. Samoin käy selaimen Takaisin-painikkeen kanssa, jolloin edellinen sivu ladataan selaimen välimuistista, eikä palvelimelta. Tätä ongelmaa kutsutaan *polun täydentämiseksi* (path completion) [Cooley *et al.*, 1999]. Ongelman ratkaisemiseksi voidaan käyttää viittauslokia.

Esimerkiksi jos käyttäjä pyytää sivua, jota ei ole linkitetty käyttäjän edelliseen vierailemaan sivuun, voidaan tarkastaa viittauslokista, miltä sivulta käyttäjä tuli. Jos viittauslokissa oleva sivu löytyy jo käyttäjän aikaisemmin vierailtujen sivujen joukosta, voidaan olettaa käyttäjän käyttäneen Takaisin-painiketta. Tämä päätelty tieto lisätään istuntotietoihin ja arvioitu vierailuaika lasketaan [Cooley *et al.*, 1999].

4.3. Tiedon muotoilu

Kun lokitiedosto on esikäsitelty, tieto täytyy saattaa formaattiin, jossa sitä voidaan seuraavissa vaiheissa louhia. Mahdollisia formaatteja on lukuisia. Käytetty formaatti määräytyy aina tiedontarpeesta – mitä tietoa halutaan. Facca ja Lanzi [2003] mainitsevat muutamia tietorakenteita, joita on käytetty eri tutkimuksissa: relaatiotietokantoja, puu-rakenteita sekä kuutioita.

5. Tiedonlouhinta

Web-käytön tiedonlouhinnassa ollaan kiinnostuneita käyttäjän toimista websivustolla [Patil and Patil, 2012]. Tiedonlouhinnan (tai tietämyksen muodostamisen) tekniikoilla esikäsitellystä tiedosta voidaan löytää kiinnostavaa tietoa ja tavoitteena onkin jollain tapaa hyötyä tästä tiedosta. Web-käytön louhinnassa pyritään esimerkiksi tarjoamaan suosituksia käyttäjälle, profiloimaan ja luokittelemaan käyttäjiä sekä kehittämään sivuston rakennetta.

Web-käytön louhinnassa erilaisia louhinta-algoritmeja esiintyy tutkimuksissa runsaasti. Usein tutkimukset keskittyvät yhden tietyn ongelman ratkaisuun, jolloin tekijä on päätynyt käyttämään tiettyä menetelmää ja algoritmia. Usein web-käytön louhinnan ongelmia ei ratkaista vain yhdellä menetelmällä,

vaan tutkijat päätyvät usein käyttämään useita menetelmiä parhaan lopputuloksen saamiseksi. Samanlaisia ongelmia voidaan myös ratkaista eri menetelmillä. Facca ja Lanzi [2003] toteavatkin, että valitun menetelmän ja ongelman sovellusalueen välillä ei välttämättä ole korrelaatiota, vaan lähes kaikkia menetelmiä voidaan käyttää lähes kaikilla sovellusalueilla. Valitut tiedonlouhintamenetelmät ja algoritmit riippuvat aina halutusta lopputuloksesta. Web-käytön louhintaan liittyvissä tutkimuksissa on useimmiten käytetty joitakin seuraavista menetelmistä**.**

5.1. Assosiaatiosäännöt

Suuresta määrästä tietoa voidaan tunnistaa assosiaatiosääntöjä, jotka kuvaavat aineiston tietoalkioiden välisiä suhteita [Tan *et al.,* 2005]. Alkiojoukoksi voidaan ajatella yhden istunnon aikana vieraillut sivut: esimerkiksi käyttäjä vieraili istuntonsa aikana sivuilla {A.html, B.html, C.html}. Esimerkiksi voidaan löytää assosiaatiosääntö {A.html, B.html} ⊚ {C.html}, joka toteaa: jos käyttäjä vierailee sivulla *A.html* sekä sivulla *B.html,* on todennäköistä, että saman istunnon aikana käyttäjä vierailee myös sivulla *C.html* [Facca and Lanzi, 2005].

Assosiaatiosääntöjen mielenkiintoisuuden mittareina toimivat *tuki* (support) ja *tarkkuus* (confidence) [Tan *et al.,* 2005]. Tuki kertoo, kuinka usein assosiaatiosääntö esiintyy aineistossa [Tan *et al.,* 2005]. Tarkkuus kertoo prosentuaalisen osuuden siitä, kuinka useassa tapauksessa jossa esiintyy A, esiintyy myös B [Tan *et al.,* 2005]. Esimerkiksi taulukossa 1 esiintyy assosiaatiosääntö {A} ⊚ {B}, jonka tuki on 50 % (alkiojoukot {A} ja {B} esiintyvät kahdessa istunnossa neljästä). Assosiaatiosäännön tarkkuus on 66,7%, koska alkiojoukko {A} esiintyy aineistossa kolme kertaa, ja näistä kahdessa tapauksessa esiintyy myös {B}.

Istunto	Sivut
1	A, B, C
2	A, B
3	A, C
4	B, C

Taulukko 1: Esimerkki-istunnot ja -sivunlataukset.

Louhittaessa tuelle ja tarkkuudelle asetetaan alarajat, jotka assosiaatio-

sääntöjen tulee täyttää. Alarajoilla pyritään eliminoimaan sattumalta esiintyvien sääntöjen päätymistä analysointivaiheeseen, sillä löydetty assosiaatiosääntö, jonka tuki on alhainen ei yleensä ole kiinnostava. [Tan *et al.*, 2005]

Cho ja muut [2002] hyödynsivät assosiaatiosääntöjä rakentaessaan suosittelujärjestelmää, joka suosittelee käyttäjälle seuraavaksi ostettavia tuotteita. Perustuen ostohistoriaan, ostoskorikäyttäytymiseen ja käyttäjien klikkausvirtoihin Cho ja muut [2002] kehittivät assosiaatiosääntöjä tuoteryhmittelyille, jonka avulla tuoteryhmien väliset suhteet saatiin selville. Kun käyttäjän ostohistoriasta löydetään jo ostettuja tuotteita, käyttäjälle voidaan tarjota tarkkoja suosituksia jostain toisesta tuoteryhmästä.

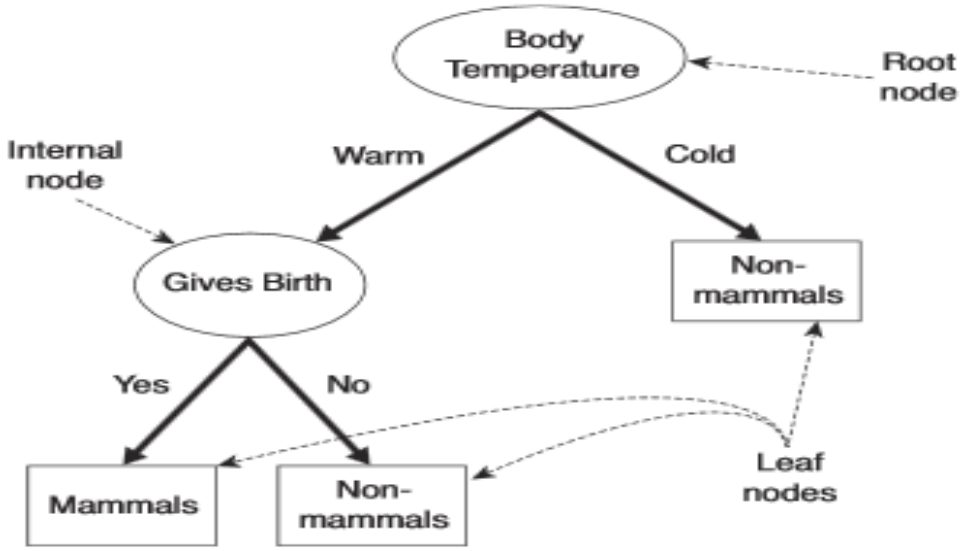
Mobasher ja muut [2001] kehittivät suosittelujärjestelmän, joka muodosti assosiaatiosääntöjä perustuen sivuston klikkausvirtoihin. Käyttämällä Apriori-algoritmia he muodostivat sivunäkymien toistuvat alkiojoukot. Erikoisuutena he käyttivät useita tuen eri alarajoja, tarkoituksenaan löytää myös syvällä sivustohierarkiassa sijaitsevat, mahdollisesti käyttäjiä kiinnostavat, sivut. Järjestelmän suosittelumoottori käytti näitä taustalla olevia assosiaatiosääntöjä hyödykseen suositellessaan käyttäjälle sivuja.

5.2. Luokittelu

Luokittelun tavoitteena on luokitella uudet, ennestään tuntemattomat, tapaukset ennalta määrättyihin kategorioihin. Tapaus koostuu joukosta attribuutteja (*attribute set*) sekä sille asetetusta luokkaleimasta (*class label*). Luokittelu tehdään *luokittelumallin* (classification model) avulla, joka luokittelee sille syötteenä tulevat tapaukset. Luokittelumallin toteutuksessa käytetään *luokittelijaa* (classifier), joka puolestaan käyttää hyväkseen *oppimisalgoritmia* (learning algorithm). Mallia toteutettaessa luodaan ensin *opetusaineisto* (training set). Opetusaineisto on luokiteltu valmiiksi käsin. Opetusaineisto toimii syötteenä oppimisalgoritmille, joka opettelee ja muodostaa annetusta syötteestä luokittelumallin. Luokittelumallia testataan testiaineistolla, jonka tapausten luokkaleimat ovat tuntemattomia. Testatun luokittelumallin tuottama luokittelu tarkastetaan ja jos havaitaan virheellisiä arvoja, algoritmia tai opetusaineistoa muokataan ja testataan uudelleen. Tavoitteena on, että luokittelumalli onnistuu luokittelemaan uudet tuntemattomat tapaukset oikein. [Tan *et al.*, 2005.]

Tiedonlouhinnassa yleisiä luokittelumenetelmiä ovat mm. päätöspuut, neuroverkot ja naiivi Bayes-luokittelu [Tan *et al.*, 2005]. Web-käytön louhinnassa usein käytetty luokittelumenetelmä on päätöspuu [Cho *et al.*, 2002]. Esimerkki päätöspuusta on kuvassa 2. Päätöspuun syötteenä on tapaus sen attribuutteineen. Päätöspuu koostuu *solmuista* (node): *juurisolmusta* (root node), *sisäsolmuista* (internal node) sekä *lehtisolmuista* (leaf node) [Tan *et al.*, 2005].

Juuri- ja sisäsolmut ovat testejä, joissa verrataan tapauksen tietyn attribuutin arvoja. Solmuista toisiin solmuihin lähtevät haarat ovat edellisen testin tulos. Jos seuraava solmu on sisäinen solmu, seuraa uusi testi. Kun päätöspuussa saavutaan lehtisolmuun, lehtisolmuun liittyvä luokkaleima asetetaan tapauksen luokaksi [Tan *et al.,* 2005].



Kuva 2: Päätöspuuesimerkki: nisäkkään luokittelu [Tan *et al.,* 2005].

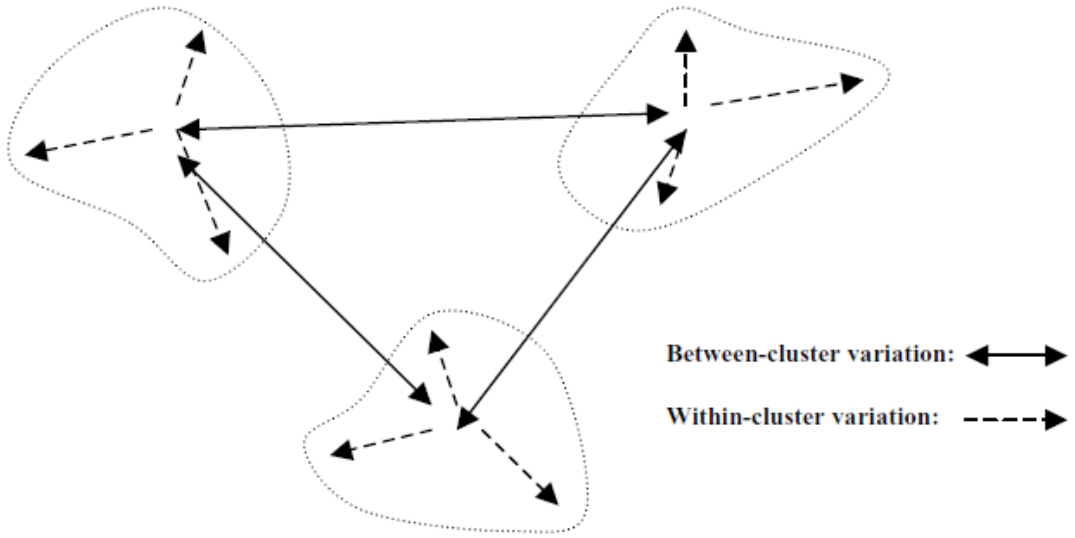
Cho ja muut [2002] hyödynsivät suosittelujärjestelmässään myös päätöspuuta. Välttääkseen turhia suosituksia, he käyttivät päätöspuuta luokittelemaan potentiaaliset uudelleenostajat viimeaikaisten ostojen perusteella. Esimerkiksi Yu ja muut [2005] kehittivät luokittelijan, jolla sivuston vierailijat luokiteltiin kolmeen luokkaan: ostohaluiset vierailijat, tavalliset vierailijat (ei ostoaikeita), sekä robotit. Luokittelussa käytettiin hyväksi yhdeksää eri lokeista saatavaa attribuuttia, joiden perusteella pystyttiin kategorisoimaan sivustolla vierailleita käyttäjiä.

Myös muita luokittelumenetelmiä on käytetty web-käytön louhinnan alalla. Esimerkiksi Khosravi ja Tarokh [2010] käyttävät naiivia Bayes-luokittelijaa ennustaakseen käyttäjän seuraavan sivupyynnön. Ennustamisella voidaan esimerkiksi nopeuttaa seuraavaa sivunlatausta välimuistin avulla tai käyttäjälle voidaan tarjota linkkejä mahdollisesti mielenkiintoiselle sivulle.

5.3. Klusterointi

Klusteroinnissa (*clustering*) tavoitteena on ryhmitellä yhteen joukko tapauksia, joilla on keskenään yhtenäisiä piirteitä [Srivastava *et al.*, 2000] ja jotka lisäksi eroavat toisen ryhmän tapauksista [Tan *et al.,* 2005]. Näitä ryhmiä kutsutaan *klustereiksi*. Kuva 3 havainnollistaa klusteroitujen tapausten suhteita: klusterin

sisällä tapaukset ovat toisiinsa nähden hyvin samanlaisia, kun taas erot toisen klusterin tapauksiin ovat suuret.



Kuva 3: Klusterin sisällä tapausten erot ovat pienempiä ja klustereiden väliset tapausten erot suurempia [Markov and Larose, 2007].

Markovin ja Larosen [2007] mukaan klusterointi on usein alustava tiedonlouhintavaihe, jossa löydetyt klusterit toimivat syötteenä jollekin muulle tiedonlouhintamenetelmälle, kuten neuroverkoille.

Web-käytön louhinnassa klusteroidut tapaukset ovat usein käyttäjiä tai istuntoja. Klusteroinnin avulla voidaan tunnistaa ja ryhmitellä käyttäjät, joilla on keskenään samankaltaisia selailutapoja [Srivastava *et al.*, 2000]. Esimerkiksi Aghabozorgi ja Wah [2009] kehittivät suosittelujärjestelmän, jossa esiprosessoidusta istuntotiedosta klusteroitiin käyttäjäprofiileja käyttäen tunnettua k:n keskiarvon klusterointimenetelmää. Klusteroidut profiilit muodostivat pohjan suosittelujärjestelmälle. Klustereiden ja sivuston kategorioiden välille muodostettiin suhteet, joiden perusteella voitiin ennustaa, mikä kategoria kiinnostaa tiettyjä käyttäjäklustereita. Uuden käyttäjän tullessa sivustolle käyttäjä profiloitiin klusteriin, minkä jälkeen käyttäjälle tarjottiin kyseiselle klusterille kohdennettuja suosituksia. Aghabozorgin ja Wahin [2009] menetelmä otti huomioon myös käyttäjien kiinnostuksen muutokset, sillä jos uusi käyttäjä ei ollut tarpeeksi lähellä nykyisiä klustereita, järjestelmä loi uuden klusterin, tai jos vanhan käyttäjäprofiilin navigointitottumukset muuttuivat, siirrettiin käyttäjä toiseen klusteriin.

Sudhamathyn ja Venkateswaran [2011] vertailivat kolmea web-lokien klusterointimenetelmää. Heidän esittelemänsä kolme menetelmää olivat: Fuzzy Clustering -algoritmi, Temporal Clustering Migration Matrices (TCMM) ja

Particle Swarm Optimization (PSO) -algoritmi. Heidän mukaansa TCMM:ää voi käyttää vain käyttäjien kategorisointiin ja käyttäytymisen seuraamiseen pitkällä aikavälillä. Fuzzy-menetelmää he kehuivat yksinkertaiseksi ja joustavaksi. PSO tarjoaa tietoa käyttäjien käyttäytymisestä ja sivuston sivujen mielenkiintoisuudesta, mutta Sudhamathyn ja Venkateswaran mukaan verrattuna Fuzzy-menetelmään PSO on monimutkaisempi ja raskaampi toteuttaa. Sudhamathy ja Venkateswaran päätyvät testien ja analysoinnin jälkeen lopputulokseen, jossa Fuzzy-menetelmä on tehokkain web-lokien klusterointiin. [Sudhamathy and Venkateswara, 2011]

Etminani ja muut [2009] puolestaan käyttivät klusterointimenetelmänä Kohosen itseorganisoituvia karttoja (SOM, Self-Organizing Map). He klusteroivat yliopiston web-palvelinlokeista käyttäjien navigointipolkuja, tuloksenaan paljon tietoa siitä, miten opiskelijat navigoivat yliopiston sivustolla. Etminani ja muut [2009] toteavat menetelmänsä sopivan sivuston ylläpitäjille, jotka haluavat lisätä sivuston kiinnostavuutta, tai markkinoijille, jotka haluavat kampanjoista tuottavampia.

5.4. **Sekvenssihahmot**

Sekvenssihahmojen (*sequential patterns*) louhinnalla tarkoitetaan toistuvien alisekvenssien löytämistä suuresta määrästä aikajärjestettyä tietoa. Aikajärjestetty tieto voi olla esimerkiksi verkkokaupan ostotapahtuma-tietokanta tai web-palvelinloki. Ensimmäisenä ongelman sekvenssihahmojen löytämisestä transaktioiden välillä esittelivät Agrawal ja Srikant [1995] artikkelissaan ”Mining Sequential Patterns”. Sekvenssihahmot muistuttavat assosiaatiosääntöjä, mutta sekvenssihahmojen tapauksessa tapahtumien aikajärjestyksellä on merkitystä. Etsitään siis usein toistuvia hahmoja. Esitellään esimerkki sekvenssihahmosta [Agrawal and Srikant, 1995]: asiakas vuokraa ensin elokuvan ensimmäisen version, ja myöhemmin ensimmäisen jatko-osan ja vielä tätä myöhemmin toisen jatko-osan. Vuokrauksen ei tarvitse seurata heti edellisen jälkeen, vaan välissä voi olla vuokrauksia muistakin elokuvista [Agrawal and Srikant, 1995].

Web-käytön louhinnassa kiinnostavia sekvenssihahmoja ovat käyttäjien navigointipolut (tai hahmot) [Mabroukeh and Ezeife, 2010]. Esimerkiksi olkoon web-lokista tunnistettu seuraavat istunnot (S) ja sivukäyntisekvenssit (<a-f>): [S1,<abdac>]; [S2, <eaebcac>]; [S3, <babfaec>]; [S4, <abfac>]. Sekvenssihahmon tunnistusalgoritmi löytäisi toistuvan sekvenssin *abac*, joka viittaa yli 90 % kävijöistä vierailevan ensin sivulla *a*, seuraavaksi sivulla *b*, jonka jälkeen palataan takaisin sivulle *a*, josta palataan edelleen sivulle *c*. Esimerkiksi verkkokaupassa sivulle *a* voitaisiin lisätä kohdennettua mainontaa, joka potentiaalisesti lisäisi myyntiä. [Mabroukeh and Ezeife, 2010]

Sekvenssihahmojen löytämiseen web-lokeista on alalla kehitetty kymmeniä tekniikoita. Mabroukeh ja Ezeife [2010] ovat pyrkineet kehittämään sekvenssihahmojen louhinnalle taksonomiaa, sillä eri menetelmien toteutustavat ovat melko heterogeenisiä. He luokittelevat eri algoritmeja kolmeen eri luokkaan: Apriori-perustaiset algoritmit, Pattern-Growth -algoritmit sekä Early-Pruning -algoritmit. Apriori-perustaiset algoritmit olivat ensimmäisiä sekvensshahmojen tunnistamiseen kehitettyjä algoritmeja (ensimmäiset esitetty Agrawalin ja Srikantin (1995) toimesta), mutta niiden heikkoutena on kuitenkin vaativa laskennallisesti raskas kandidaattien muodostamisvaihe. Kandidaattien muodostamista on pyritty välttämään 2000-luvun aikana kehittämällä Pattern-Growth -algoritmejä. Uusimpia menetelmiä ovat Early-Pruning -algoritmit, joissa kandidaattien muodostamisen lisäksi pyritään välttämään turhia tukiarvojen laskemisia. [Mabroukeh and Ezeife, 2010]

Rajanveto eri menetelmien välille ei ole yksinkertaista. Esimerkiksi Ezeifen ja Lun [2003] kehittämä suosittu algoritmi PLWAP käyttää hyödykseen ominaisuuksia, joista osan Mabroukeh ja Ezeife [2010] luokittelevat Pattern-Growth -algoritmien ominaisuuksiksi ja osan Early-Pruning -kategoriaan.

PLWAP on yksi tehokkaimmista menetelmistä sekvenssihahmojen louhimiseen [Nguyen, 2009], sillä se käyttää hyödykseen paranneltua WAP-puurakennetta. Nguyen [2009] esittelee tutkimuksessaan menetelmän, jossa ensin esiprosessoidusta lokista louhitaan PLWAP-algoritmin avulla usein toistuvat sekvenssihahmot. Tämän jälkeen Nguyen käyttää sekvenssihahmoja syötteenä dynaamisesti klusteroivalle Markov-mallille, joka osaa ennustaa käyttäjän seuraavia liikkeitä sivustolla. Nguyen [2009] esittääkin menetelmänsä pystyvän tarkasti ennakoimaan tai suosittelemaan mielenkiintoisia sivuja ja siten Nguyen uskoo sen olevan hyödyllinen suosittelujärjestelmille.

6. Tiedon analysointi

Tiedonlouhintamenetelmät eivät olisi hyödyllisiä, jos ei olisi olemassa työkaluja ja mekanismeja, jotka auttavat analyytikkoja paremmin ymmärtämään louhittua tietoa [Cooley *et al.*, 1997]. Analysointivaiheessa louhitusta tiedosta suodatetaan analysoitavaksi vain mielenkiintoiset tiedot ja hahmot. Usein hyödyksi käytetään visualisointia: tulokset esitetään esimerkiksi taulukoina ja diagrammeina, joista voidaan vertailla ja luokitella kiinnostavaa tietoa [Varnagar *et al.*, 2013]. Esimerkiksi klusteroinnin tuloksia on usein mielekästä esittää graafisina kuvina, kuten myös navigointipolkuja. Cooley ja muut [1997] toteavatkin visualisoinnin olevan luonnollinen valinta web-käyttäjien ymmärtämiseen.

Markov ja Larose [2007] esittelevät joitakin alustavia aineistoanalyyseja (*exploratory data analysis*) esiprosessoidulle lokille. He esimerkiksi etsivät keskimääräisiä sivullakäyntiaikoja ja mittaavat istuntojen pituuksia. He visualisoivat löydettyjä tietoja esimerkiksi pylväsdiagrammin avulla.

Pabarskaite ja Raudys [2006] mainitsevat web-palvelinlokitiedon syöttämisen suoraan tietokantaan olevan hyvä analysointimenetelmä, mutta vähemmän tehokas, sillä menetelmä vaatii paljon käsin tehtyä työtä ja lisäksi analyytikon täytyy suunnitella ja ajaa tietokantakyselyitä. Pabarskaiten ja Raudysin [2006] mukaan OLAP (Online analytical processing) on arvokas väline liiketoimintatiedon hallinnassa, jota voidaan hyödyntää myös web-käytön louhinnassa, sillä useat tiedonlouhinta- ja tiedonvarastointityökalut tarjoavat OLAP- toiminnallisuuksia.

7. Yhteenveto

Web-käytön tiedonlouhinnan päämenetelmistä ei ole yksiselitteistä listausta, mutta tutkielmassa käsitellyt neljä tiedonlouhintamenetelmää esiintyvät suuressa osassa alan julkaisuista. Esiteltyjä tiedonlouhintamenetelmiä ei ole myöskään rajattu vain yhden ongelman ratkaisemiseen, vaan kuten osoitettu samoja menetelmiä voidaan käyttää erilaisten ongelmien ratkaisuun. Useat tutkijat ovatkin yhdistelleet eri menetelmiä päästäkseen haluamaansa lopputulokseen. Erilaisten menetelmien kirjo tulee varmasti lähivuosina kasvamaan. Myös web-käytön tiedonlouhinnan merkitys tulee kasvamaan vuosi vuodelta, erityisesti sähköisen kaupankäynnin osalta, jossa taloudellisesti haastavissa tilanteissa olevat yritykset pyrkivät löytämään keinoja liikevaihdon lisäämiseen.

Web-käytön tiedonlouhinta on erittäin laaja tutkimusala, jonka kaikkia tutkimussuuntia on mahdotonta kattaa yhdessä kandidaatin tutkielmassa. Alan laajuudesta kertoo myös erilaisten katsausten suppeus, sillä kaiken kattavia katsauksia ei viime vuosilta löydy. Lähes jokainen tutkimus poikkeaa sisällöltään ja käytetyiltä menetelmiltään jollain tavalla muista tutkimuksista, mikä lisää katsauksen muodostamisen haastavuutta.

Tämän tutkimuksen tarkoitus oli esitellä lyhyesti web-käytön tiedonlouhintaa, minkä vuoksi tarkempi syventyminen tiedonlouhintaprosessin eri vaiheisiin ei ollut mahdollista. Tutkielma tarjoaa lyhyen katsauksen yleisimpiin web-käytön louhinnan eri vaiheisiin, joten sen avulla saa käsityksen prosessin pääpiirteistä. Laajemmassa tutkielmassa olisi voinut käydä eri menetelmät sekä algoritmien pääpiirteet tarkemmin läpi, ja kattaa monipuolisemmin eri louhintamenetelmien käyttöä web-käytön louhinnassa. Toinen mahdollinen tutkiel-

man lähestymistapa olisi ollut keskittyä vain yhteen käytännön sovellusalueeseen.

Viiteluettelo

[Aghabozorgi and Wah, 2009] S. R. Aghabozorgi and T. Y. Wah, Recommender systems: Incremental clustering on web log data. In: *Proc. of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (2009), 812-818.

[Agrawal and Srikant, 1995] R. Agrawal and R. Srikant, Mining sequential patterns. In: *Proc. of the 11th International Conference on Data Engineering* (1995), 3-14.

[Cho *et al.,* 2002] Y. H. Cho, J. K. Kim and S. H. Kim, A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications* **23,** 3 (2002), 329-342.

[Cooley *et al.,* 1997] R. Cooley, B. Mobasher and J. Srivastava, Web mining: Information and pattern discovery on the World Wide Web. In: *Proc. of the International Conference on Tools with Artificial Intelligence* (1997), 558-567.

[Cooley *et al.,* 1999] R. Cooley, B. Mobasher and J. Srivastava, Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems* **1** (1999), 5-32.

[Etminani *et al.,* 2009] K. Etminani, A. R. Delui, N. R. Yanehsari and M. Rouhani, Web usage mining: Discovery of the users' navigational patterns using SOM. In: *Proc. of the First International Conference on Networked Digital Technologies* (2009), 224-249.

[Ezeife and Lu, 2003] C. I. Ezeife and Y. Lu, Position coded pre-order linked WAP-tree for web log sequential pattern mining. In: *Proc. of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (2003), 337-349.

[Facca and Lanzi, 2005] F. M. Facca and P. L. Lanzi, Mining interesting knowledge from weblogs: A survey. *Data and Knowledge Engineering* **53** (2005), 225-241.

[Hasan *et al.,* 2009] T. Hasan, S. Mudur and N. Shiri, A session generalization technique for improved web usage mining. In: *Proc. of the 11th International Workshop on Web Information and Data Management* (2009), 23-30.

[Hussain *et al.,* 2010] T. Hussain, S. Asghar and N. Masood, Web usage mining: A survey on preprocessing of web log file. In: *Proc. of the International Conference on Information and Emerging Technologies* (2010), 1-6.

[Khosravi and Tarokh, 2010] M. Khosravi and M. J. Tarokh, Dynamic mining of users interest navigation patterns using naive Bayesian method. *Intelligent Computer Communication and Processing* (2010), 119-122.

[Mabroukeh and Ezeife, 2010] N. R. Mabroukeh and C. I Ezeife, A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys* **43,** 1 (2010), article 3.

[Markov and Larose, 2007] Z. Markov and D. T. Larose, *Data Mining the Web: Uncovering Patterns in Web Content, Structure and Usage*. John Wiley & Sons, 2007.

[Mobasher *et al.*, 2001] B. Mobasher, H. Dai, T. Luo and M. Nakagawa, Effective personalization based on association rule discovery from web usage data. In: *Proc. of the 3rd International Workshop on Web Information and Data Management* (2001), 9-15.

[Nguyen, 2009] S. T. T. Nguyen, Efficient web usage mining process for sequential patterns. In: *Proc. of the 11th International Conference on Information Integration and Web-based Applications and Services* (2009), 465-469.

[Pabarskaite, 2002] Z. Pabarskaite, Implementing advanced cleaning and end-user interpretability technologies in web log mining. In: *Proc. of the 24th Int. Conf. Information Technology Interfaces ITI* (2002), 109-113.

[Pabarskaite and Raudys, 2006] Z. Pabarskaite and A. Raudys, A process of knowledge discovery from web log data: Systematization and critical review. *Journal of Intelligent Information Systems* **28**, 1 (2007), 79-104.

[Patil and Patil, 2012] U. M. Patil and J. B. Patil, Web data mining trends and techniques. In: *Proc. of the International Conference on Advances in Computing, Communications and Informatics* (2012), 961-965.

[Srivastava *et al.*, 2000] J. Srivastava, R. Cooley, M. Deshpande and P.-N. Tan, Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explorations* **1**:2 (2000), 12-23.

[Sudhamathy and Venkateswaran, 2011] G. Sudhamathy and C. J. Venkateswaran, Web log clustering approach: a survey. *International Journal on Computer Science and Engineering* **3**:7 (2011), 2896-2903.

[Suneetha and Krishnamoorti, 2009] K. R. Suneetha and R. Krishnamoorthi, Identifying user behaviour by analyzing web server access log file. *IJCSNS International Journal of Computer Science and Network Security* **9,** 4 (2009), 324-332.

[Tan *et al.*, 2005] P-N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining.* Addison-Wesley, 2005.

[Varnagar *et al.,* 2013] C. R. Varnagar, N. N. Madhak, T. M. Kodinariya and J. N. Rathod, Web usage mining: A review on process, methods and techniques. In: *Proc. of the International Conference on Information Communication and Embedded Systems* (2013), 40-46.

[Yu *et al.,* 2005] J. X. Yu, Y. Ou, C. Zhang and S. Zhang, Identifying interesting visitors through web log classification. *Intelligent Systems* **20**, 3 (2005), 55-59.

[Yuan *et al.,* 2003] F. Yuan, L.-J. Wang and G. Yu, Study on data preprocessing algorithm in web log mining. In: *Proc. of the 2nd International Conference on Machine Learning and Cybernetics* (2003), 28-32.

Katsaus kolmiulotteisiin lääketieteellisiin sovelluksiin

Sami Koivunen

Tiivistelmä.
Tämä tutkielma on katsaus kolmiulotteisuutta hyödyntäviin lääketieteellisiin sovellutuksiin. Jaan alan tutkimusaihetta silmälläpitäen erilaisiin osa-alueisiin ja annan esimerkkejä kolmiulotteisista käytännön sovelluksista. Esittelen myös kolmiulotteisuuden viihdekäytön nykysovelluksia ja pohdin kolmiulotteisen tekniikan mahdollisia ongelmakohtia.

Avainsanat ja -sanonnat: 3D-näyttö, 3D-tekniikka, stereoskooppinen kuva, lääketiede

CR-luokat: I.2.10., I.3.1., I.4.8., J.3

1. Johdanto

Kolmiulotteisen kuvan esittämiseen tarvittava tekniikka on ollut olemassa jo pitkään, mutta vasta viime vuosina siitä on tullut varteenotettava vaihtoehto esimerkiksi elokuvateattereihin tai televisiomarkkinoiden myyntivaltiksi. Tarinankerronnan osana kolmiulotteisuus on vielä lapsenkengissään, mutta kehitys on ollut nopeaa [Atkinson, 2011]. Kolmiulotteisuus on tullut käyttöön myös lääketieteen alalla [Magalhães et al., 2011]. Tässä tutkimuksessa esittelen, millaisia 3D-teknologioita on olemassa ja missä lääketieteellisissä käyttötarkoituksissa niitä voidaan hyödyntää. Sivuan myös viihdekäytöllisiä mahdollisuuksia.

Ensimmäiset 3D-televisiot tulivat markkinoille vuoden 2010 alkupuolella. Useat asiantuntijat ovat verranneet kolmiulotteisuuden saapumista elokuviin äänen saapumiseen 1920-luvulla, värien saapumiseen 1930-luvulla ja laajakuvan saapumiseen 1950-luvulla. Kaksiulotteista kuvaa (2D) voidaan kutsua ”monoksi” ja kolmiulotteista (S3D) ”stereoskooppiseksi”. Ilman silmälaseja katsottavaa kolmiulotteista kuvaa voidaan puolestaan kutsua ”autostereoskooppiseksi”. Silmälasien lisäksi kolmiulotteisuuteen liittyviä tekniikoita voidaan erotella sen mukaan, voiko kolmiulotteista kuvaa katsella useampi henkilö yhtä aikaa ja voiko kolmiulotteiseksi luotua objektia katsella eri kulmista vaihtamalla katselijan katselukulmaa (volumetrinen tai holografinen näyttö).

Kolmiulotteisuus ei ole uusi ajatus varsinkaan elokuvakerronnassa. Jo 1950- ja 1980-luvuilla tehtiin kokeiluja tekniikalla, mutta yleiseksi standardiksi 3D on muuttunut vasta vähän ennen 2010-lukua. Ensimmäinen merkittävä kolmiulotteinen kaupallinen elokuva oli Eric Brevigin vuonna 2008 julkaistu Journey to the Centre of the Earth [Sandler, 2011]. Läpimurtona S3D-tekniikalle voidaan pitää puolestaan James Cameronin menestyselokuvaa Avatar (2010). Yleisesti kolmiulotteisuuden voi sanoa olevan jo hyväksytty ja standardoitu muoto katsella elokuvia elokuvateatterissa tai kotona tv:tä katsellessa. Viihdekäytössä kolmiulotteisuus sopii myös esimerkiksi mainostamiseen, pelaamiseen, teemapuistoihin ja televisioon. Televisiokanavia voidaan käyttää 3D-tekniikkaa hyödyntäen ja siten näyttää esimerkiksi kolmiulotteisesti kuvattuja suoria urheilulähetyksiä [Ohta et al., 2006]. Suomalaisissa kaupoissa myydään 3D Blu-ray -levyjä, joita on mahdollista katsoa 3D-tekniikkaa tukevilla televisiolaitteilla.

Lääketieteen alalla kolmiulotteisuuden käyttö on lisääntynyt tämän vuosituhannen aikana. Esimerkiksi diagnosoinnin, tähystyskirurgian ja lääketieteellisen harjoittelun osa-alueilla stereoskooppisilla näytöillä on piirteitä, jotka potentiaalisesti mahdollistavat kehittyneemmän toiminnan. Aloitan tutkielman esittelemällä yleisellä tasolla kolmiulotteisuutta ja jatkan lääketieteellisillä sovelluksilla. Tutkielman tavoitteena on edetä kolmiulotteisuuden hypoteettisista hyödyistä käytännön sovelluksiin käyttäen hyväksi aiheesta tehtyjä tutkimuksia. Kolmiulotteisuus on tuonut mukanaan myös haasteita ja varsinkin käytännön sovellusten hyödyllisyys on monilla lääketieteen aloilla kiistanalaista. Tutkielman lopussa käynkin vielä lyhyesti läpi kolmiulotteisuuden mahdollisia ongelmakohtia.

2. Tekniikka

Tärkein yksittäinen syy käyttää kolmiulotteisuutta on saavuttaa syvyysvaikutelma. Kolmiulotteisuuden saavuttamisen mahdollistavia teknologioita on useita eikä ole todennäköistä, että yksittäinen teknologia saavuttaisi valta-asemaa ainakaan lähitulevaisuudessa. Perusmallissa stereoskooppisella kuvalla tarkoitetaan prosessia, jossa luodaan ja esitetään vasemmalle ja oikealle silmälle stereokuvapari. Ihmiselle luonnollinen stereonäkö yhdistää katselijan aivoissa kaksi kuvaa yhdeksi kolmiulotteiseksi kuvaksi [Pank, 2008]. Tämän toteuttamiseen voidaan käyttää joko laseja, jotka erottavat kuvaa molemmille silmille, tai käyttää valonlähdettä, joka erottaa kuvat suoraan käyttäjän silmille.

Eräs toinen tapa toteuttaa kolmiulotteinen kuva on hologrammien käyttö. Hologrammien ongelmina ovat kuitenkin yleensä rajattu katselukulma ja vähäinen valaistus. Tästä huolimatta vertikaalisesti hajottava hologramminen näyttö (vertically dispersive holographic screen) on eräs uusimmista ja mielenkiintoisimmista tavoista saavuttaa kolmiulotteisuus ilman laseja. Silmälaseja käyttämätön 3D-tekniikka on erityisen hyödyllistä lääketieteen alalla, jossa mahdollisuus katsoa magneetti- ja tietokonekerroskuvia kolmiulotteisena on tervetullut [Magalhães et al., 2011].

Autostereotyyppiset näyttöteknologiat käyttävät optisia komponentteja, jotka hajottavat kaksi kuvaa suoraan käyttäjän silmiin luoden kolmiulotteisen vaikutelman. Tämä kuitenkin rajoittaa päänliikkeitä vaatien katsojalta tiettyä katselukohtaa. Multiautostereotyyppiset näytöt mahdollistavat useamman näkymän samasta tilanteesta. Tällöin käyttäjä voi liikkua ja nähdä oikean näkymän sijainnista riippumatta.

Yleisimmin käytettävät lasilliset 3D-tekniikat näytöissä ovat aktiivinen 3D, passiivinen 3D ja anaglyfinen 3D. Passiivisella 3D-tekniikalla tarkoitetaan, että käytetään laseja, jotka puolittavat esimerkiksi Full HD -resoluution eli 1080p:n tarkkuuden molemmille silmille. Tällöin kumpikin silmä näkee kuvan 540p:n tarkkuudella. Aktiivisilla 3D-laseilla on puolestaan mahdollista luoda 1080p:n kuva molemmille silmille.

Käytännössä aktiivisessa 3D:ssa näytetään eri kuvaa molemmille silmille ja aivot yhdistävät erilliset kuvat yhdeksi kolmiulotteiseksi kuvaksi. Passiiviteknologiaa käyttävä 3D-kuva koostuu leveyssuuntaisista viivoista, joista parittomia näytetään vasemmalle silmälle ja parillisia oikealle silmälle. Silmälasien linssit ovat polarisoitua lasia, joka tuottaa oman kuvan kummallekin silmälle. Aivot yhdistävät myös tässä tapauksessa molempien silmien näkemän kuvan yhteiseksi kolmiulotteiseksi kuvaksi.

On myös mahdollista muuttaa perinteinen kaksiulotteinen kuva kolmiulotteiseksi. Yksinkertaisesti muutos voidaan tehdä mittaamalla syvyyttä vihjeiden, kuten kuvan hämäryyden, objektien koon ja liikkeen perusteella, mutta lopputulos saattaa olla liian sattumanvaraista todellisuuteen verrattuna [Cheng et al., 2010]. Tämän ongelman ratkaisemiseksi on kehitetty järjestelmiä, jotka tunnistavat kuvan syvyyden käyttämällä esimerkiksi hyväksi todennäköisyyslaskentaa.

3. Lääketiede ja 3D

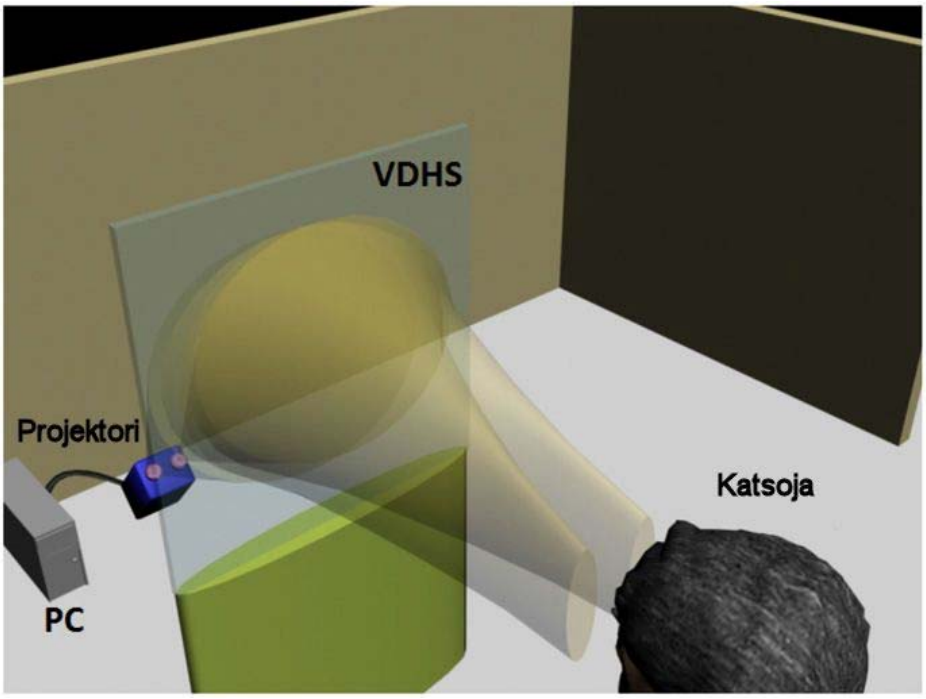
Suurin käytännön tarve kolmiulotteisuudelle on tieteen parissa, erityisesti lääketieteessä, kun pyritään havainnollistamaan monimutkaisia 3D-raken-

teita. Myös roboteissa käytetään kolmiulotteisuutta ja se on samoin tarpeellista pyrittäessä erittäin tarkkaan kauko-ohjaukseen.

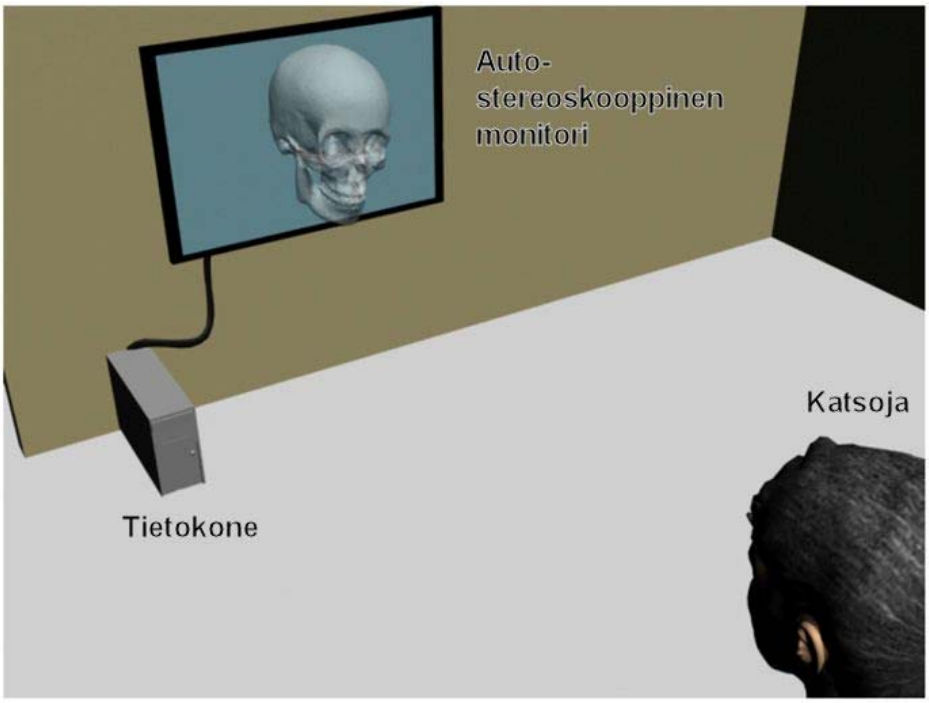
Lääketieteen alalla kolmiulotteista kuvaa voidaan hyödyntää esimerkiksi radiologiassa, tietyntyyppisissä leikkauksissa ja harjoittelussa. Kolmiulotteisuus auttaa antamaan paremman avaruudellisen näkemyksen kehosta, paremman näkökulman epäselviin tilanteisiin ja paremman suoritustason näppäryyttä vaativissa toimenpiteissä [Magalhães et al., 2011]. Se voi myös tehostaa oppimista ja kehittää lääkärin ja potilaan välistä kommunikointia.

On olemassa ainakin kaksi tapaa esittää lääketieteellisiä kuvauksia (kuten magneettikuvia ja CT-kuvia) kolmiulotteisesti ilman laseja. Pystysuunnassa hajottava holografinen näyttö (vertically dispersive holographic screen) ja multiautostereoskooppinen järjestelmä on muiden muassa esitetty vaihtoehdoiksi [Magalhães et al., 2011]. Van Beurden ja muut [2011] mainitsevat kuitenkin magneettikuvien ja CT-kuvien kohdalla, että kolmiulotteisen syvyysvaikutelman tuomaa lisäarvoa ei ole toistaiseksi luotettavasti demonstroitu.

Magalhães ja muut [2010] esittelevät pystysuunnassa hajottavan holografisen näytön (VDHS) luotaessa kolmiulotteista kuvaa magneettikuvista ja CT-kuvista. Esitelty kokoonpano koostuu holografisesta näytöstä, projektorista ja tietokoneesta. VDHS:n toimintaperiaatteena on tuottaa katselukulmia spatiaalisesti. Tämä edellyttää katselijan sijoittumista tietylle katselualueelle, mutta eliminoi toisaalta tarpeen käyttää laseja. Käytännössä tekniikka edellyttää hajottavan holografisen näytön käyttöä, jotta tapahtuu pystysuuntainen diffraktiivinen dispersio. Valaistuksen on suuntauduttava joko ylhäältä tai alhaalta, että pystysuuntainen dispersio tapahtuu diffraktiivisessa valaistuksessa. Tekniikassa kumpikin silmä näkee kuvan eri kulmasta ja luo illuusion kolmiulotteisuudesta. Kuva 1 esittää, kuinka tietokone lähettää kuvaparin projektorille, josta VDHS vastaanottaa kuvat ja näyttää ne katselijalle. Kuvaparin muodostamiseen 3D-kerroskuvista ja 3D-magneettikuvista vaaditaan sovellus, joka lukee ja renderöi kuvat.



Kuva 1. Pystysuunnassa hajottava holografinen näyttö (VDHS) näyttää stereoskooppisen projektorin tuottaman stereoskooppisen kuvaparin katsojan silmille [Magalhães et al., 2011].



Kuva 2. Autostereoskooppinen järjestelmä, joka koostuu tietokoneesta, monitorista ja katsojasta [Magalhães et al., 2011].

Autostereotyyppisillä näytöillä on Magalhãesin ja muiden [2011] mukaan mahdollista saavuttaa nykytekniikalla 1680 x 1050 -resoluutio. Kuva on tarkka, eikä kolmiulotteisuus aiheuta ongelmia esimerkiksi tarkennuksessa. Autostereotyyppiset 3D-näytöt auttavat lääkäreitä havaitsemaan esimerkiksi aneurysmia ja muita verisuoniin liittyviä ongelmia. Kolmiulotteinen kuva tarjoaa paremman spatiaalisen ymmärryksen anatomisista rakenteista. Kuvaa katsoessaan tarkkailija pystyy esimerkiksi havaitsemaan toisen laskimon olevan syvyyssuunnassa toisen laskimon edessä, mikä helpottaa työskentelyä merkittävästi verrattuna kaksiulotteiseen kuvaan. [Magalhães et al., 2011].

Lääketieteen alueella kolmiulotteisuutta voidaan hyödyntää useilla eri tavoilla ja panostukset teknisiin kehitysaskeleisiin ovat rahallisesti suuria, mutta käytännön sovelluksiin tekniikat päätyvät vain, jos niiden hyöty on osoitettu joko parannettuna potilashoitona tai merkittävinä säästöinä [van Beurden et al., 2011]. Esimerkkejä hoidon ja säästämisen kehittämisestä ovat koulutukseen liittyvän ajan väheneminen, leikkauksiin liittyvän ajan väheneminen, operaatiotilan käyttökulujen väheneminen, potilaan toipumisajan lyheneminen ja inhimillisten virheiden määrän väheneminen. Van Beurden ja muut [2011] esittävät myös, että kolmiulotteisuuden mahdollisuuksia lääketieteessä voidaan jaotella esimerkiksi diagnosointiin, operaatiosuunnitteluun, mini-invasiiviseen kirurgiaan sekä opetus- ja harjoitustarkoitukseen. Käyn jaottelun avulla lyhyesti läpi esimerkkejä alalla tehdyn tutkimuksen pohjalta.

3.1 Diagnosointi

Lääketieteellisessä visualisoinnissa voidaan erottaa kaksi eri tapaa renderöidä materiaalia: pintarenderöinti (surface rendering) ja volumetrinen renderöinti. Pintarenderöinnissä esimerkiksi elimet ja luut ovat selvästi näkyvissä, kun taas alla olevat rakenteet eivät. Tässä tapauksessa kolmiulotteinen vaikutelma luodaan monokulaarisilla syvyysvihjeillä, kuten perspektiivin, okkluusion, tekstuurin, varjojen, värien ja liikkeen avulla. Kickuth ja muut [2002] vertailivat koehenkilön tarkkuutta lonkan alueen murtumien diagnosoinnissa käyttämällä apuna tavallista CT-kuvaa ja stereoskooppista CT-kuvaa. Tuloksissa ei havaittu eroja parempaan tai huonompaan suuntaan, mutta on mahdollista, että jommassakummassa tapauksessa oli selvempiä ja helpommin diagnosoitavia tapauksia.

Volumetrisessä kuvauksessa tieto on esitetty läpikuultavassa muodossa, jossa näkyvät sekä päällä että alla olevat rakenteet. Toisaalta syvyysvihjeitä on vähemmän ja kolmiulotteisen rakenteen muodostaminen on hankalampaa. Wang ja muut [2010] tutkivat stereoskooppisten näyttöjen kliinistä hyödyl-

lisyyttä keuhkonodulien havaitsemisessa CT-kuvien avulla. Tavalliseen perspektiiviin ja viipaloituun CT-kuvaan verrattuna stereoskooppinen kuva ei tarjonnut merkittävästi parempia tuloksia, mutta selvästi havaittavaa trendiä parempiin tuloksiin oli kuitenkin olemassa. Käytännössä volumetristä kuvausta käytetään esimerkiksi verisuonten visualisointiin. Päällekkäisiä verisuonirakenteita on hankala tulkita ja volumetrinen kuvaus tarjoaa avaruudellista näkökulmaa verisuonten välisiin yhteyksiin ja muotoihin.

Abildgaard ja muut [2010] vertasivat kaksiulotteista ja kolmiulotteista autostereoskooppista kuvaa näyttämällä kolmelle neuroradiologille sekä 2D- että 3D-kuvia. Heidän tehtävänään oli tunnistaa merkittyjä valtimoita kliinisestä aineistosta kerätyistä kuvista. Tulokset osoittivat, että radiologit tunnistivat kolmiulotteisista kuvista paremmin yksittäisiä kallonsisäisiä valtimonosia. Vaikka tämä vahvistaa entisestään kolmiulotteisten kuvien hyödyllisyyttä erityisesti verisuonia tutkittaessa, niin Abildgaard ja muut toteavat, että autostereoskooppisen visualisoinnin hyödyntämisestä radiologiassa tarvitaan vielä lisätutkimusta.

Ultraäänestä voidaan myös luoda sekä kaksiulotteisia että kolmiulotteisia malleja. Kolmiulotteisilla malleilla voidaan parantaa kuvanlaatua vähentämällä kaksiulotteisissa kuvissa esiintyvää kohinaa. Myös ultraäänikuvissa päällekkäisten rakenteiden havainnollistaminen on helpompaa stereoskooppisista kuvista kuin kaksiulotteisista kuvista. Syvyysvaikutelmasta saadun hyödyn ansiosta esimerkiksi rintasyövän havaitseminen on helpompaa [Hernandez et al., 1998].

Nelson ja muut [2008] tutkivat stereoskooppisen ultraäänikuvan ja tavallisen ultraäänikuvan eroja sikiön luustoa tutkittaessa. Vertailussa tutkittiin lähes 50 ihmissikiötä esittämällä stereoskooppista ultraäänikuvaa visuaalisesti tarkoitusta varten rakennetussa tilassa. Tutkimuksen visuaalisten parametrien optimointi suoritettiin interaktiivisesti. Testin avulla havaittiin, että sekä tavallisella että stereoskooppisella kuvauksella pystytään havaitsemaan sikiön luuston rakenne ainakin pääpiirteittäin. Kuitenkin havaittiin tilastollisesti merkittävä parannus monimutkaisten rakenteiden havainnoinnissa ja havaintotarkkuudessa, kun käytössä oli kolmiulotteinen stereoskooppinen ultraäänikuva. Tulosten perusteella voidaan sanoa, että kolmiulotteisuus toi arvokasta lisätietoa sikiön luuston diagnosointiin ja arviointiin.

Cherniy ja muut [2007] puolestaan korostavat, kuinka kaksiulotteinen röntgenkuva ei toisinaan anna lääkäreille edes mahdollisuutta havaita keuhkoissa sijaitsevien leesioiden koon ja muodon pieniä muutoksia, jonka kolmiulotteisuutta hyväksikäyttävä AMTsR_1–röntgenlaite puolestaan mahdollis-

taisi. AMTsR_1–röntgenlaitteen toiminta perustuu useista eri kulmista otettuihin röntgenkuviin, joiden ottamiseen vaaditaan vähäisesti säteilyä ja ainoastaan yksi säteilylähde, mutta tuloksena on silti korkean resoluution kuvia. Vähäinen säteily mahdollistaa kolmiulotteisten kuvien saamisen myös lapsipotilailta. Erityisesti kolmiulotteisuus tarjoaisi mahdollisuuden parempaan seurantaan potilaan hoitovaiheessa. Siinä missä perinteinen malli toteutetaan yleensä ottamalla kuva ennen hoitoa ja hoidon jälkeen, kolmiulotteinen malli tarjoaisi mahdollisuuden pienienkin muutosten havainnointiin ja tarvittaessa mahdollistaisi lääkityksen korjaamisen hoitoprosessin aikana.

Hyvä esimerkki kolmiulotteisesta diagnosointitavasta on silmätautiopissa käytettävä stereoskooppinen kuvaus. Stereoskooppinen ONH-kuvaus (optic nerve head eli näköhermonpää) on yksinkertainen ja halpa tapa tuottaa värillinen kolmiulotteinen kuva näköhermonpäästä glaukoomatutkimuksessa. Glaukooma on sairaus, jossa kohonnut silmänpaine vahingoittaa hitaasti ja huomaamatta näköhermoa. Sairaus tunnetaan myös kansanomaisemmin nimillä silmänpainetauti ja viherkaihi. Sharma ja muut [2008] mainitsevat, että monokulaariset vihjeet eivät mahdollista herkkien kudoskerrosten tarkkaa havainnointia ja muotojen yksityiskohtaista kuvaamista yhtä hyvin kuin stereoskooppiset näytöt silmän rakennetta tutkittaessa. Van Beurdenin ja muiden [2011] mukaan stereoskooppisesta tekniikasta voidaankin sanoa olevan merkittävää hyötyä silmätautiopin alalla nimenomaan koska se helpottaa glaukooman havaitsemista.

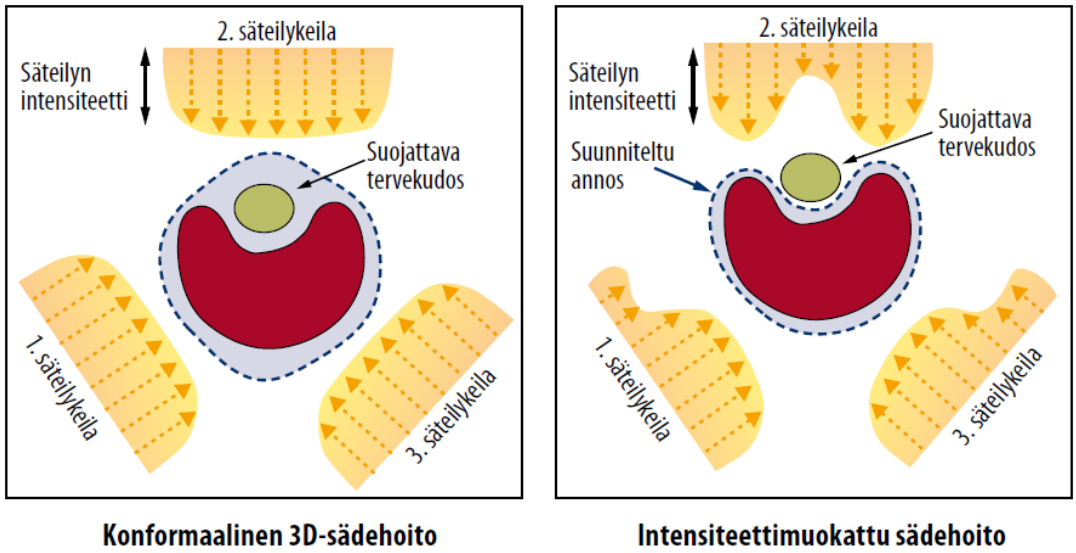
3.2 3D operaatiosuunnittelun tukena

Kirurgisten operaatioiden valmistelu alkaa perinteisesti preoperatiivisella suunnittelulla, jonka tarkoituksena on löytää optimaalinen tapa suorittaa operaatio. Tarkka suunnitelma lisää operaation tarkkuutta, vähentää siihen kuluvaa aikaa ja siihen liittyviä komplikaatioita. Tietokonesimulaation, eli virtuaalisen todellisuuden, käyttäminen suunnitteluvaiheessa mahdollistaa erilaisten strategioiden testaamisen samalla tavalla kuin teollisuuden aloilla [Satava and Jones, 2002]. Preoperatiivisessa suunnittelussa korostetaan kvantitatiivista tietoa etäisyyksistä, volyymeistä ja kulmista, jotka ovat operaation kannalta keskeisiä tietoja.

Sädehoito on eräs tärkeimmistä syövän hoitomuodoista. Menetelmä perustuu säteilyn ohjaamiseen useammalta eri suunnalta. Hubbold ja muut [1999] tutkivat mahdollisuutta käyttää stereoskooppisia näyttöjä sädehoidon suunnittelussa. Kouri ja muut [2009] kertovat kasvaimen kolmiulotteisen määrityksen ja annoslaskennan olleen jo pitkään sädehoidon standardi.

Sädehoito suunnitellaan tietokonetomografian (TT) tai ultraäänikuvauksen avulla. Tällöin muodostetaan kolmiulotteinen kuvapakka, jossa määritellään hoidon kohdetilavuus ja suojattavat kudokset.

Kolmiulotteiseksi kohteenmukaiseksi sädehoidoksi kutsuttava menetelmä toteutetaan käyttämällä useita eri suunnista annettavia säteilykeiloja. Ennen kolmiulotteisia sädehoitokäytäntöjä hoidot perustuivat kaksiulotteisiin mallikenttiin, joissa säteilyä saatettiin kohdistaa hyvin laajalle alueelle, kuten koko aivojen alueelle, ja siten pahimmillaan vahingoittaa merkittävissä määrin tervettä kudosta. Kuvassa 3 esitetty intensiteettimuokattu sädehoito on sädehoidon merkittävimpiä kehitysaskelia ja se tuli käyttöön Suomessa vuosituhannen vaihteessa [Kouri ja muut, 2009]. Intensiteettimuokattu hoitomenetelmä mahdollistaa vähäisemmän säteilytyksen suojeltavalle tervekudokselle, kun nykyisessä tavanomaisessa 3D-sädehoidossa intensiteetit ovat tasaiset koko säteilytettävän kentän alueella [Kouri ja muut, 2009]. Menetelmässä käytetään annoslaskentaohjelmaa, joka laskee hoitokenttien annosintensiteetit geometrisesti. Käytännössä teknisesti vaativassa prosessissa annosintensiteettiä lasketaan niistä suunnista, joissa on suojeltavaa kudosta ja prosessissa syntyvä aliannos kohdennetaan kasvaimeen. Vaikka sädehoidossa ei suoranaisesti hyödynnetä kolmiulotteisuutta näyttöjen ja kuvien avulla, on se alan havainnollistavimpia esimerkkejä kolmiulotteisuuden hyödyntämisestä.



Kuva 3. Havainnollistus tavanomaisen konformaalisen sädehoidon ja intensiteettimuokatun sädehoidon erosta.

3.3 Laparoskooppinen ja mini-invasiivinen kirurgia

Laparoskooppinen ja mini-invasiivinen kirurgia ovat tarkentaneet kirurgisia operaatioita, joissa on perinteisesti operoitu elimistöä suurien viiltojen ja suoran näköyhteyden avulla. Mini-invasiivinen operaatio suoritetaan pienten viiltojen kautta laitettavilla kameroilla, valonlähteillä ja kirurgisilla välineillä. Operaation aikana kirurgi luottaa videokuvaan, joka esittää anatomiset rakenteet kaksiulotteisesti ja usein myös epänormaaleista kulmista [van Beurden et al., 2012].

Kongin ja muiden [2010] tutkimuksessa verrattiin kolmiulotteisen laparoskooppisen leikkauksen suoritusta perinteiseen kaksiulotteiseen leikkaussuoritukseen. Tutkimuksen aikana yhteensä 27 eri kokemustaustaista kirurgia suoritti kahta erilaista tehtävää neljänä perättäisenä päivänä. Kirurgien suoritusaikaa ja virheitä seurattiin yhdessä joka suorituksen jälkeen kirjattujen subjektiivisten näkemysten, kuten syvyysnäkökokemusten ja visuaalisten epämukavuuskokemusten lisäksi. Tuloksista ilmeni, että kolmiulotteinen menetelmä tuotti selvästi paremman syvyysvaikutelman kuin perinteinen kaksiulotteinen näkymä. Huimauksen ei havaittu olevan ongelma ja laparoskooppisen leikkaussuorituksen havaittiin olevan tarkempi. Uusi menetelmä oli erityisesti kokeneiden kirurgien mielestä perinteistä leikkaustapaa parempi. Tulosten havaittiin lisäävän selvästi esimerkiksi vasemman käden käyttöä operaation aikana ja oikean käden käytön vastaavasti vähenevän. Tämän voisi ajatella tarkoittavan useiden kirurgien kohdalla siirtymistä kohti enemmän molempia käsiä hyödyntävää leikkaussuoritusta.

Havaitsin kuitenkin, että leikkauksista saadut tulokset vaihtelevat tutkimuksesta riippuen. Hofmeister ja muut [2001] tarkastelivat 15 tutkimusta, jotka vertailivat monoskooppista ja stereoskooppista mini-invasiivista kirurgiaa vuosina 1992-1999. Erikoisesti kuusi tutkimusta havaitsi merkittävää hyötyä stereoskooppisten näyttöjen käyttämisestä ja muut tutkimukset eivät havainneet tilastollista hyötyä kumpaankaan suuntaan. Van Beurden ja muut [2010] ovat lisäksi käyneet läpi uudempia tutkimuksia, joiden tulokset olivat vaihtelevia ja kolmiulotteisuuden hyödyn kyseenalaistavia. Syyksi tähän epäiltiin osittain puutteellista 3D-kuvanlaatua, oppimiskäyrän huomioitta jättämistä ja joissain tutkimuksissa alhaista koehenkilömäärää. Lisäksi on huomioitava, että erityisesti kokeneet kirurgit ovat oppineet käyttämään monokulaarisia syvyysvihjeitä optimaalisesti hyväksi kaksiulotteisella näytöllä. Joka tapauksessa mini-invasiivisessa kirurgiassa ei voida laajankaan tutkimusaineiston perusteella sanoa kolmiulotteisuuden tuovan yksiselitteisesti merkittävää lisäarvoa kirurgiselle operaatiolle.

3.4 Kolmiulotteisuus opetuksen ja harjoituksen tukena

Simulaatioita käytetään laparoskooppisten leikkaustaitojen opetteluun ennen pääsyä varsinaiseen leikkaussaliin. Votanopoulos ja muut [2008] vertailivat, parantaisiko kolmiulotteisen harjoitusmallin käyttäminen laparoskooppisen leikkauksen suoritusta aloittelijoilla, joille kaksiulotteinen harjoittelu oli tuttua verrattuna aloittelijoihin, joille edes kaksiulotteinen harjoittelu ei ollut vielä tuttua. Tutkimus suoritettiin testaamalla kuutta laparoskooppiseen leikkaamiseen liittyvää taitoa joko 2D- tai 3D-näyttötekniikalla toteutetulla simulaatioharjoituksella ja kolmen kuukauden jälkeen testihenkilöiltä testattiin taitoja uudelleen, mutta käänteisesti toisella näyttötekniikalla. Suoritusta arvioitiin tehtävään kuluneen ajan ja virhemäärän perusteella. Tuloksena kokeneemmat osallistujat suoriutuivat tehtävistä kokemattomia paremmin harjoitustavasta huolimatta. Mielenkiintoista oli kuitenkin, että kaksiulotteisella näyttötekniikalla jo ennen tutkimusta harjoitelleet eivät merkittävästi hyötyneet kummankaan harjoitustavan käyttämisestä verrattuna toiseen. Kokemattomat osallistujat suoriutuvat sen sijaan merkittävästi paremmin harjoiteltuaan ensin kolmiulotteisella mallilla ja sitten siirryttyään kaksiulotteiseen malliin verrattuna päinvastaisesti harjoitelleeseen ryhmään. Tutkimustuloksista pääteltiin, että kolmiulotteinen harjoittelutapa tarjosi merkittävästi hyötyä laparoskooppisten taitojen harjoittelussa kokemattomilla henkilöillä.

4. Kolmiulotteisuus elokuva- ja kotikäytössä

Lääketieteessä hyödynnettävän kolmiulotteisuuden lisäksi kolmiulotteisuuden näkyvimpiä muotoja ovat erityisesti elokuva- ja kotiteatterikäyttö. 3D-tekniikkaa hyödynnetään esimerkiksi suorissa urheilulähetyksissä ja lukuisissa elokuvajulkaisuissa. Vaikka tutkielmani pääpaino on lääketieteellisissä sovelluksissa, on tarkoituksena tehdä pikainen katsaus myös kolmiulotteisuuden viihdekäyttöpuoleen.

Kolmiulotteinen elokuva eroaa kaksiulotteisesta elokuvasta teknisesti merkittävästi ja syvyysvaikutelma tuo haasteita muun muassa tekniselle toteutukselle ja tarinankerronnalle. Kaksiulotteisessa elokuvassa kuvaa rajaavat kankaan tai näytön reunat, jolloin mielessä rakentuva kolmiulotteinen hahmotus perustuu syvyysvihjeisiin, jotka voivat olla esimerkiksi esineiden välisiä suhteellisia kokoeroja ja nopeuksia.

Atkinson [2011] kuvailee, kuinka kolmiulotteinen videointi vaatii uudenlaista ajattelutapaa ohjaajalta ja kameraryhmältä. Toisin kuin kaksiulotteisessa videossa, tyhjä tila näyttelijän ja kameran välillä korostuu ja videon kuvaus-

vaiheessa on ajateltava, kuinka syvyysvaikutelma siirtyy valkokankaalle tai vaihtoehtoisesti näytölle.

Silmälasitonta kokemusta tavoitellaan myös kolmiulotteisiin kotiteatte-reihin. Tämän tekniikan ongelmana on toistaiseksi liian heikko resoluutio. Jos näytöllä olevaa kuvaa jaetaan keinotekoisesti molemmille silmille, on yhden osanäkymän resoluutio liian pieni mukavan kokemuksen saavuttamiseksi. Ongelmaan saadaan ratkaisu, kun markkinoille tulevat uudet ultratarkat resoluutiot.

Eräs vaihtoehto kolmiulotteisuuden aikaansaamiseksi elokuvakäytössä on toteuttaa täydellinen holografinen kolmiulotteisuus (H3D). Erona kaksiulot-teisuuteen teknologia mahdollistaisi useamman ihmisen keskittymisen eri kohteisiin näytöksen samassa kohdassa. Kokemusta voisi verrata teatteriin. Jo vuonna 2010 kehitteillä olevan H3D-tekniikan arveltiin olevan noin 8-10 vuoden päässä kaupallisesta toteutuksesta.

Atkinsonin [2011] mukaan elokuvateollisuuden sisäpiirilähteet ovat esittäneet kaikkien elokuvien muuttuvan lähitulevaisuudessa kolmiulotteisiksi siten, että 3D-etuliitettä ei käytettäisi enää ollenkaan elokuvien nimen yhteydessä. Tällöin kaksiulotteisuudesta tulisi historiallinen ja vanhanaikainen tyyli tuottaa elokuvia.

Käytännössä esimerkiksi S3D-tekniikassa on jo huomioon otettavana seikkana tarinan sisällön ja juonen hukkuminen visuaalisuuden taakse. H3D-tekniikalla perinteinen elokuvakerronta olisi jo hyvin haasteellista. Kolmiulot-teisuuden myötä ohjauksen merkitys on elokuvallisessa kolmiulotteisuudessa korostunut ja tekniikka vaatii osaavaa tulkintaa, jotta saadaan haluttu reaktio katsojassa. Haasteena on katsojan mielikuvituksen ja emotionaalisen tunnesi-teen rakentaminen kuvan ollessa enemmän reaalimaailmaa vastaava.

5. Ongelmat

Hologrammeilla pystytään korjaamaan myös nykyiseen stereoskooppiseen kolmiulotteisuuteen liittyviä ongelmia. Yksi suurimmista ongelmista kolmi-ulotteisuudessa on syvyysvaikutelman vääristyminen [Atkinson, 2011]. S3D-tekniikassa syvyysvaikutelma syntyy käytännössä vain yhdessä ”kohdassa” taustojen esimerkiksi sumentuessa.

Ongelmana on myös silmien ja aivojen tottuminen syvyysvaikutelmaan. Kolmiulotteisissa elokuvissa kohtausten kestolla on merkitystä enemmän verrattuna kaksiulotteisuuteen. 3D-elokuvien kohtauksista on mukavuus-syistä tehtävä pitkäkestoisempia ja jatkuva kohtausten vaihto, jossa syvyys-vaikutelma on kuvassa eri alueella, ei ole mahdollista.

Lambooijin ja muiden [2012] mukaan nopea liikkuminen kohtauksen sisällä voi aiheuttaa visuaalista epämukavuutta. Lisäksi erityisesti tekstitysten asettelu syvyyssuunnassa ja erilaisuus taustaan nähden saattaa aiheuttaa epämukavuuden tunnetta katselijassa. Joillakin ihmisillä tiedetään olevan herkkyyksiä tai puutteita, joiden takia kolmiulotteisen sisällön katseleminen on mahdotonta. Lisäksi Atkinsonin [2011] mukaan on yleisesti tiedettyä, että pitkäkestoinen kolmiulotteinen elokuva voi jatkuvilla kohdennusmuutoksilla ja epätarkkuuksilla aiheuttaa päänsärkyä ja pahoinvointia kaikilla katsojilla.

Kooi ja Toet [2004] tutkivat, että vähäinenkin altistuminen kolmiulotteisuuteen liittyviin ominaispiirteisiin, kuten vertikaalisiin eroavaisuuksiin, haamukuviin, kuvan vuotamiseen väärälle silmälle ja sumeudelle voi aiheuttaa merkittävää visuaalista epämukavuutta ja esimerkiksi rasitusta silmissä. Kuvan vuotamisella väärälle silmälle tarkoitetaan passiivisten lasien epätäydellistä optista erotusta vasemmalle ja oikealle silmälle, tai synkronisointivirhettä aktiivisilla laseilla, kun lasien pitäisi estää toiselle silmälle tarkoitetun näkymän näkyminen [Aflaki et al., 2013]. Pölönen ja muut [2012] mainitsevat kuitenkin esimerkiksi jo pitkäaikaisen tietokonekäytön lisäävän riskiä silmään kohdistuville rasitusvammoille, kuten sumentuvaa näköä, kuivuutta silmissä ja päänsärkyjä.

Kolmiulotteisuutta hyödyntävät sovellutukset ovat usein lapsille suunnattuja, mutta aiheeseen perehtynyttä tutkimusta on tehty melko vähän [Pölönen et al., 2012]. Pölösen ja muiden [2012] havaintojen perusteella S3D-sisällön katseleminen kohtuullisessa määrin ei aiheuta yleisesti ottaen ongelmia aikuisille tai lapsille. Tutkimus koottiin rakentamalla seitsemän kokeellista ympäristöä, jossa eri-ikäiset koehenkilöt katselivat televisionäytöltä erilaisia 3D-mediasovelluksia, kuten pelejä ja elokuvia. Vertailuryhmänä yksi aikuisryhmä pelasi peliä kaksiulotteisesti. Tutkijat havaitsivat laitteiston, tehtävän ja koehenkilön iän vaikuttavan kokemukseen, kuten oli odotettavaakin. Yleisesti ottaen useimmat visuaalisten toimintojen muutoksista olivat suhteellisen vähäisiä ja verrattavissa kaksiulotteisesta katselukokemuksesta syntyviin muutoksiin. Tutkijat kuitenkin varoittavat 130 koehenkilöä kattavan tutkimuksen olevan liian pieni, jotta voisiin tehdä laajempia yleistyksiä. Tästä huolimatta tulokset olivat linjassa tutkimuksessa vertailtuihin aikaisempiin tutkimuksiin. Päätelmänä Pölönen ja muut korostavat, että lyhyiden taukojen pitäminen silloin tällöin on suositeltavaa käyttäjän omien kokemusten mukaisesti.

6. Yhteenveto

Tässä tutkielmassa tutkin erityisesti kolmiulotteisen tekniikan hyödyntämismahdollisuuksia lääketieteen alalla. Aineistona ovat olleet erilaiset empiiriset tutkimukset, joista olen yrittänyt löytää paitsi puoltavia tuloksia kolmiulotteisten sovellusten hyödyllisyydestä, myös kriittisiä näkökulmia ja parannusehdotuksia.

Tutkielmaa kirjoittaessani huomasin stereoskooppisilla näytöillä usein havaituiksi hyödyiksi paremman syvyysnäkemyksen ja paremman kyvyn havaita erityisesti helposti ympäristöönsä hukkuvia objekteja. Erilaisia sovellusaloja rajoittaakseni käytin van Beurdenin ja muiden [2011] käyttämää lääketieteellistä rajausta diagnosointiin, preoperatiiviseen suunnitteluun, laparoskooppiseen/mini-invasiiviseen kirurgiaan ja kolmiulotteisuuden käyttöön harjoittelu- ja opetuskäytössä.

Vaikka varsinkin joillain lääketieteen osa-alueilla askel kaksiulotteisista kuvauskeinoista kolmiulotteisiin kuvauskeinoihin vaikuttaa ilmiselvästi hyödylliseltä, niin muun muassa diagnosoinnissa perspektiivistä kuvaustapaa käytetään pääasiasiallisesti edelleen [van Beurden et al., 2011]. Eräs syy tähän saattaa olla se, että korkeatarkkuuksisen kaksiulotteisen kuvan ottaminen on ollut helpompaa ja pitkään myös tehokkaampaa kuin korkeatarkkuuksisen kolmiulotteisen kuvan ottaminen. Stereoskooppiset näytöt voivat kärsiä myös crosstalk-efektistä, eli kuvan vuotamisesta väärälle silmälle, tai haamukuvista. Kolmiulotteisilla toteutuksilla tehdyt tutkimukset keskittyvät usein monimutkaisiin tapauksiin, joissa esimerkiksi leesiot on edullista havaita juuri kolmiulotteisella menetelmällä. Yleisemmällä tasolla ajateltuna kaksiulotteinen kuva kykenee edelleen MRI- ja CT-kuvauksessa sellaiseen kuvanlaatuun, joka jää kolmiulotteiselta kuvauskeinolta saavuttamatta [van Beurden et al., 2011].

Kolmiulotteisuutta voidaan yleisesti ottaen pitää turvallisena muotona katsoa kuvaa ja videota. Vertailussa kaksiulotteiseen katselukokemukseen ei ole yleisesti havaittu merkittävää muutosta visuaalisissa toiminnoissa. On kuitenkin turvallista pitää mielessä pitkäaikaisen altistumisen lisäävän erityisesti silmään kohdistuvia rasitusoireita. Lisäksi joidenkin ihmisten tiedetään yleisesti kärsivän sellaisista oireista, jotka tekevät keinotekoisen kolmiulotteisen kuvan katselemisesta visuaalisesti epämiellyttävää tai mahdotonta.

Viiteluettelo

[Abildgaard *et al.*, 2010] Andreas Abildgaard, Alaa Kasid Witwit, Jørn Skaarud Karlsen, Eva Astrid Jacobsen, Bjørn Tennøe, Geir Ringstad and Paulina Due-Tønnessen. An autostreoscopic 3D display can improve

visualization of 3D models from intracranial MR angiography. *International Journal of Computer Assisted Radiology and Surgery* **5**(5) (2010), 549-554.

[Aflaki *et al*., 2013] Payman Aflaki, Miska M. Hannuksela, Hamed Sarbolandi and Moncef Gabbouj. Simultaneous 2D and 3D perception for stereoscopic displays based on polarized or active shutter glasses. To appear in *Journal of Visual Communication and Image Representation*.

[Atkinson, 2011] Sarah Atkinson, Stereoscopic-3D storytelling – Rethinking the conventions, grammar and aesthetics of a new medium, *Journal of Media Practice* **12**:2 (2011), 139–156.

[van Beurden *et al*., 2011] M.H.P.H. van Beurden, W.A. Ijsselsteijn and J.F. Juola. Effectiveness of stereoscopic displays in medicine: a review. *3D Research* **3**:3 (2012).

[Cheng *et al*., 2010] Chao-Chung Cheng, Chung-Te Li and Liang-Gee Chen. A novel 2D-to-3D conversion system using edge information. *IEEE Transactions on Consumer Electronics*, **56**(3) (2010), 1739-1745.

[Cherniy *et al*., 2007] A. N. Cherniy, B. M. Kanter, E. V. Serova and G. V. Ratobyl'skii. Use of stereoscopic vision for analysis of digital x-ray images of lungs. *Biomedical Engineering* **41**(5) (2007), 214-217.

[Hernandez *et al*., 1998] A. Hernandez, O. Basset, I. Dautraix, I. Magnin, C. Favre and G. Gimenez. Stereoscopic Visualization of 3D Ultrasonic Data for the Diagnosis Improvement of Breast Tumors. *European Journal of Ultrasound* **8**(1) (1998), 51-65.

[Hofmeister *et al*., 2001] Jörg Hofmeister, Timothy Frank, Alfred Cuschieri and Nicholas Wade. Perceptual aspects of two-dimensional and stereoscopic display techniques in endoscopic surgery: review and current problems. *Seminars of Laparoscopic Surgery* **8**(1) (2001), 12-24.

[Hubbold *et al*., 1999] Roger J. Hubbold, David J. Hancock and Christopher J. Moore. In: *Proc. SPIE 3012, Stereoscopic Displays and Virtual Reality Systems IV* (1999), 16-27.

[Kickuth *et al*., 2002] R. Kickuth, G. Hartung, U. Laufer, C. Gruening, C. Stueckle, D. Liermann and J. Kirchner. Stereoscopic 3D CT vs standard 3D CT in the classification of acetabular fractures: an experimental study. *The British Journal of Radiology* **75** (2002), 422-427.

[Kong *et al*., 2010] Seong-Ho Kong, Byung-Mo Oh, Hongman Yoon, Hye Seong Ahn, Hyuk-Joon Lee, Sun Geun Chung, Norio Shiraishi, Seigo Kitano, Han-Kwang Yang. Comparison of two- and three-dimensional

camera systems in laparoscopic performance: a novel 3D system with one camera. *Surgical Endoscopy* **25**(5) (2010), 1132-1143.

[Kooi and Toet, 2004] Frank L. Kooi and Alexander Toet. Visual comfort of binocular and 3D displays. *Displays* **25** (2004), 99-108.

[Kouri ja Kangasniemi, 2009] Mauri Kouri ja Aki Kangasmäki. Moderni sädehoito. *Duodecim* **125**(9) (2009), 947-958.

[Lambooij *et al*., 2012] M. Lambooij, M.J. Murdoch, W.A. Ijsselsteijn and I. Heynderickx, The impact of video characteristics and subtitles on visual comfort of 3D TV. *Displays* **34** (2013), 8-16.

[Magalhães *et al*., 2011] Daniel S.F.Magalhães, Rolando L. Serra, André L. Vannucci, Alfredo B. Moreno and Li M. Li, Glasses-free 3D viewing systems for medical imaging. *Optics & Laser Technology* **44** (2012), 650-655.

[Nelson *et al*., 2008] Thomas R. Nelson, Eun K. Ji, Jong H. Lee, Michael J. Bailey and Dolores H. Preterius. Stereoscopic Evaluation of Fetal Bony Structures. *Journal of Ultrasound in Medicine* **27**(1) (2008), 15-24.

[Ohta *et al*., 2006] Yuichi Ohta, Itaru Kitahara, Yoshinari Kameda, Hiroyuki Ishikawa and Takayoshi Koyama. Live 3D Video in Soccer Stadium. *International Journal of Computer Vision* **75**(1) (2007), 173-187.

[Oliveira *et al*., 2012] Sílvia Oliveira, Jorge Jorge and José M González-Méijome. Dynamic accommodative response to different visual stimuli (2D vs 3D) while watching television and while playing Nintendo 3DS Console. *Ophthalmic & Physiological Optics* **32** (2012), 383-389.

[Pank, 2008] Bob Pank, *The Digital Fact Book: Converged Media*. Quantel, 2008.

[Pölönen *et al*., 2012] Monika Pölönen, Toni Järvenpää and Batrice Bilcu. Stereoscopic 3D entertainment and its effect on viewing comfort: Comparison of children and adults. *Applied Ergonomics* **44** (2013), 151-160.

[Satava and Jones, 2002] Richard M. Satava and Shaun B. Jones. Medical Applications of Virtual Environments. In: Kay M. Stanney (ed.), *Handbook of Virtual Environments: Design, Implementation, and Applications*. Lawrence Erlbaum Associates (2002), 937-957.

[Sharma *et al*., 2008] Pooja Sharma, Pamela A. Sample, Linda M. Zangwill and Joel S. Schuman. Diagnostic Tools for Glaucoma Detection and Management. *Survey of Ophthalmology* **53**(6) (2008), S17-S32.

[Votanopoulos *et al*., 2008] Konstantinos Votanopoulos, F. Charles Brunicardi, John Thornby and Charles F. Bellows. Impact of Three-Dimensional Vision in Laparoscopic Training. *World Journal of Surgery* **32**(1) (2008), 110-118.

[Wang *et al*., 2010] Xiao Hui Wang, Janet E. Durick, Amy Lu, David L. Herbert, Carl R. Fuhrman, Joan M. Lacomis, Cynthia A. Britton, Diane C. Strollo, Sherry S. Shang, Saraswathi K. Golla and Walter F. Good. Compare Display Schemes for Lung Nodule CT Screening. *Journal of Digital Imaging* **24**(3) (2011), 478-484.

Videopeliohjelmien ylläpidettävyyden parantaminen suunnittelumalleilla

Miikka Kähkönen

Tiivistelmä.
Tutkielma esittelee peliohjelmien ylläpidettävyyden kannalta mielenkiintoisia suunnittelumalleja keskittyen tekoäly-, grafiikka- ja pelilogiikkaohjelmoinnin kannalta korkeantason ylläpidettävyyteen vaikuttaviin ominaisuuksiin ja niihin liittyviin suunnittelumalleihin.

Avainsanat ja -sanonnat: Peliohjelmointi, suunnittelumallit, ylläpidettävyys, siltamalli, tarkkailijamalli, strategiamalli
CR-luokat: D.2.2, K.8.0

1. Johdanto

Videopelit ovat kehittyneet suureksi osaksi viihdeteollisuutta. Videopeliteollisuuden viimeisin suuri tuotos, Rockstarin GTA 5, tuotti yli miljardi dollaria ensimmäisellä myyntiviikollaan. GTA 5 rikkoi seitsemän Guinness-ennätystä, joista yksi oli nopeimmin miljardi dollaria kerännyt viihdetuote (entertainment property). [McLaughlin, 2013] Videopeliteollisuudesta on siis tullut viime vuosikymmenellä mielenkiintoinen tutkimusalue myös taloudellisista syistä.

Videopelien yhteiskunnallista merkitystä selittävien tieteellisten tutkimusten määrä on kasvanut videopelien suuren suosion ja teollistumisen myötä, mutta videopelien tuotantoa on tieteellisesti tutkittu vähemmän. [Stacey and Nandhakumar, 2005] Videopeleille on perustettu omia tutkintolinjojaan, ja esimerkiksi Kajaanin ammattikorkeakoulussa voi valmistua pelikehittäjäksi.

Videopelien suosio, sekä teollinen että akateeminen, on johtamassa pelikehitystä ad hoc -kehitystavasta järjestelmällisempään suuntaan. Videopeleillä on kuitenkin sellaisia ominaisuuksia, joita tietopohjaisilla järjestelmillä ei ole. Tietopohjaisilla järjestelmillä tarkoitan perinteisiä tietojärjestelmiä, jotka on kehitetty tietyille tilaajille. Videopelit eroavat tietopohjaisista järjestelmistä erityisesti niiden käyttötarkoitukseltaan. Videopelien tarkoituksena on olla viihteellisiä, toisin kuin yleensä tietopohjaisten järjestelmien tarkoituksena on olla ratkaisu johonkin rajattuun ongelmaan. [Stacey and Nandhakumar, 2005] Käsittelen ylläpidettävyyden merkitystä peliohjelmoinnissa ja sitä, miten suunnittelumalleilla voidaan parantaa videopelien ylläpidettävyyttä.

Peli-sanalla on useita merkityksiä; tässä tutkielmassa viittaan sanalla videopeleihin. Määrittelen aluksi peliohjelmoinnin ja sen osa-alueet. Peliohjelmoinnin määrittelyn jälkeen esittelen ylläpidettävyyden ja sen huomioimisen peliohjelmoinnissa. Tämän jälkeen määrittelen suunnittelumallit ja peliohjelmien ylläpidettävyyden kannalta keskeisiä malleja.

Lopuksi tarkastelen kuhunkin peliohjelmoinnin osa-alueeseen soveltuvaa suunnittelumallia.

2. Peliohjelmointi ja ylläpidettävyys

Pelien kasvaminen varsinaiseksi teollisuudeksi on johtanut kooltaan yhä suurempien pelien kehittämiseen. Aikaisemmin mainittu GTA 5, jonka budjetti oli noin 270 miljoonaa dollaria, on kustannuksiltaan verrattavissa laajoihin tietopohjaisiin järjestelmiin. Peleillä on sellaisia ominaisuuksia, joita tietopohjaisilla järjestelmillä ei ole, kuten taiteellinen ympäristö ja epävakaat markkinat. Nämä ominaisuudet vaikuttavat järjestelmäkehitykseen ja erityisesti vaatimusten määrittelyyn.

Toisin kuin suuret tietopohjaiset järjestelmät, joilla on yleensä vain yksi tilaaja, pelit myydään yksityishenkilöille, joten niiden taloudellinen tuotto on riippuvainen markkinoiden mielenkiinnosta peliin. Tämä tarkoittaa, että ei ole olemassa varsinaista asiakasta, joka kertoisi, vastaako järjestelmä hänen vaatimuksiaan, vaan pelien tuottajien on otettava riski arvioidessaan markkinoiden kiinnostusta peliä kohtaan. Varsinaisten vaatimusten puute johtaa pelin muuttumiseen kehitysvaiheessa ja myös julkaisun jälkeen. [Stacey and Nandhakumar, 2005] Vaikka pelien kehitysympäristö on omalaatuinen, ovat ne pohjimmiltaan tietojärjestelmiä, joiden kehittämisessä voidaan siten soveltaa osittain jo tutkittua tietoa järjestelmien ominaisuuksista.

Tietojärjestelmissä ylläpidolla tarkoitetaan ohjelmiston kykyä mukautua asiakkaan muuttuviin tarpeisiin [Sommerville, 2007]. Järjestelmän ylläpidettävyys on laaja termi ja yleensä se koostuu järjestelmän ymmärrettävyydestä, joustavuudesta, muokattavuudesta ja laajennettavuudesta. Pelien tapauksessa määrittelen, että laitteistojen välinen siirrettävyys on osa ylläpidettävyyttä. Tämä määrittely on luonteva siksi, että nykyään yhä useammin pelit julkaistaan useille eri konsoleille, PC:lle tai mobiililaitteille. Useat alustat, käyttäjien kiinnostus ja epävakaat vaatimukset luovat peleille muuttuvan kehitysympäristön ja näiden muutosten huomioimisen mahdollistamiseksi pelin tulisi olla ylläpidettävä. Steve Rabin kehottaakin olla kovakoodaamatta mitään. [DeLoura, 2000]

Pelit kattavat laajan määrän eritasoisia järjestelmiä tekstiseikkailuista laajoihin multimediajärjestelmiin. Laajasta toimintaympäristöstä johtuen peleissä on useita ohjelmointialueita, joiden suuren määrän takia käsittelen niistä vain grafiikka-, tekoäly ja pelilogiikkaohjelmointiin liittyvää ylläpidettävyyttä. [Wikipedia, 2013a]

Grafiikka, tekoäly ja pelilogiikka ovat kaikki peliohjelmoinnin osa-alueita. Rajoitun näiden osa-alueiden tarkasteluun, koska ne löytyvät lähes jokaisesta pelistä. Grafiikkaohjelmoinnissa keskitytään pelin visuaaliseen esitykseen. Tekoälyohjelmointi käsittää vastustajan ja muiden peliolioiden toimintojen valinnan. Pelilogiikalla tarkoitan sitä osaa peliohjelmoinnista, joka vastaa pelin yleisen kulun toiminnasta, peliolioiden luomisesta ja olioiden välisistä suhteista. Seuraavaksi käsittelen ylläpidettävyyteen liittyviä korkean tason ominaisuuksia ja kolmannessa luvussa esittelen, kuinka suunnittelumalleja voi

hyödyntää niiden saavuttamiseksi.

2.1 Grafiikka

Grafiikkaohjelmointi käsittää objektien visuaalisen esityksen ohjelmoimista. Se sisältää usein vektori- ja matriisilaskentaa sekä lineaarialgebraa. Grafiikkalaskuista voi tulla monimutkaisia, minkä takia niille on oma prosessori näytönohjaimessa, joka on yksi tietokoneen pääkomponenteista. Prosessorin lisäksi näytönohjaimessa on tuki erilaisiin grafiikkakirjastoihin kuten OpenGL tai DirectX. Usein grafiikkaohjelmointi käsittää ainoastaan näiden kirjastojen hyödyntämistä ja olioiden piirtorutiinien tekemistä. Ylläpidettävyyden kannalta grafiikkaohjelmoinnissa on abstraktilla tasolla olennaisia ominaisuuksia.

Pelien epävakaat vaatimukset voivat johtaa tarpeeseen muuttaa grafiikkaolioita. Ylläpidettävyys grafiikkaohjelmoinnissa merkitsee mahdollisuutta muuttaa kaikkea ruudulle tulostettavaa. Pelien porttaamisen mahdollistamiseksi grafiikkaohjelmoinnissa tulisi huomioida eri alustojen eroavat grafiikkamoottorit sekä tehokkuudet. Ylläpidettävyyden kannalta grafiikoiden säätäminen tehokkuuden ja laadun välillä tulisi olla mahdollista.

Grafiikan piirtäminen pitäisi pystyä määrittämään siten, että ruudulla näkyvä tarkkuus voi muuttua tilanteen mukaan. Tämä tulee konkreettisesti esille olion liikkuessa ruudulla kauemmas katsojasta, mitä kutsutaan nimellä level-of-detail eli LOD. [Sanchez-Crespo, 2003] Ylläpidettävyyden kannalta tämä tarkoittaa olion muokattavuutta. Tarkkuuden säätäminen on mielekästä, koska tarkemmasta grafiikasta seuraa raskaammat grafiikkalaskut. [Sanchez-Crespo, 2003]

Grafiikan tarkkuuden lisäksi voi olla tarvetta mahdollistaa erilaisten 2D- tai 3D-mallien, tekstuurien tai värien käyttö olioissa. Ylläpidettävyyden kannalta olioiden visuaalinen toteutus tulisi olla joustava ja mahdollistaa uudelleenkäyttöä. Olioiden visuaalisen esityksen kanssa voi joutua kokeilemaan useita yhdistelmiä estetiikan takia, mutta tekstuureiden ja mallien välillä on myös tehokkuuteen liittyviä eroja. Tarkemmat tekstuurit ja mallit johtavat raskaampiin grafiikkalaskuihin. Monesti on myös tarvetta käyttää samoja tekstuureita eri malleille, tai samalle mallille eri tekstuureita. Esimerkkinä tästä voisi olla moninpeli, jossa on erivärisiä joukkueita ja niissä pelaajia, joilla on erinäköisiä hahmoja. Hahmojen erilaisuudesta huolimatta niiden väritystä tulisi pystyä muokkaamaan joukkueen väriin sopivaksi.

2.2. Tekoäly

Tekoäly peleissä on erilaista kuin akateeminen tekoäly. Buckland [2005] huomauttaa, että yleisesti akateeminen tekoäly jaetaan vahvaan ja heikkoon tekoälyyn, joista ensimmäinen pyrkii simuloimaan ihmisen toimintaa ja jälkimmäinen hyödyntämään tekoälyä johonkin reaalimaailman ongelmaan. Tämä johtaa kahteen eroon pelien tekoälyn ja akateemisen tekoälyn välillä.

Ensimmäinen Bucklandin huomio on, että pelien resurssit ovat rajalliset. Akateemisessa tekoälyn tutkimuksessa yleensä haetaan optimaalisinta ratkaisua, eikä resurssien

käytön hallinnointi ei ole yhtä olennaista kuin peliohjelmoinnissa. Tämä johtuu siitä, että laskuaika ja muistiavaruus ovat rajalliset jopa tehokkaissakin pöytäkoneissa. Mikäli peli halutaan siirtää eri alustalle, tekoälyn tulee pystyä huomioimaan myös alustojen resurssien erot. Ylläpidettävyys vaatii siis tekoälyohjelmoinnilta joustavuutta.

Toisena Bucklandin huomiona on tekoälyn tarkoitus. Pelien tekoälyn ei välttämättä ole tarkoitus olla ihmisen toiminnan simulaatio tai optimaalisin ratkaisu kyseessä olevaan ongelmaan. Sen sijaan pelien tekoälyn tarkoitus on tuoda peliin mielenkiintoa ja viihdytystä. Bucklandin mukaan pelien tekoälyn tulisi antaa hyvä vastus, mutta hävitä enemmän kuin voittaa, sekä saada pelaaja tuntemaan itsensä älykkääksi tai ovelaksi.

Tekoälyn on siis tarkoitus olla viihdyttävä, mutta koska viihtyminen on subjektiivista, on tekoälyn asetusten hienosäätäminen välttämätöntä kehitysvaiheessa ja yleensä myös pelin julkaisun jälkeen. Tekoälyä ohjelmoidessa on siis ylläpidettävyyden takia välttämätöntä huomioida tekoälyn älykkyystason muokattavuus.

Älykkyyden muuttaminen voi olla helppoa esimerkiksi hidastamalla tekoälyn reaktioaikaa. Todellisuudessa kyseinen säätely ei välttämättä johda mielenkiintoiseen tekoälyyn, sillä pelaaja voi olettaa tekoälyn kokeilevan erilaisia tapoja voittaa pelaaja. Ylläpidettävyyden kannalta näitä toimintoja olisi hyvä pystyä laajentamaan ja muokkaamaan, peliin halutaan esimeriksi uusi ase, joka käyttö olisi tekoälylle mielekäs ratkaisuvaihtoehto.

Tekoälyä voi olla yhdellä tai usealla vastustajalla ja se voi olla keskenään eritasoista. Esimerkki tästä voisi olla ammuskelupeli, jossa perinteiset rivisotilaat odottavat pelaajan osumista, kun taas heidän johtajansa pyrkii hakemaan suojaa läheisistä rakennuksista. Ylläpidettävyyden kannalta tekoälyn tulisi olla joustava, eli mahdollistaa se useissa eri olioissa ja älykkyystasoissa.

2.3. Pelilogiikka

Peli voidaan ajatella sääntöinä ja niitä noudattavina olioina. Pelilogiikka on peliohjelmoinnin osa-alue, joka sisältää säännöt ja oliot sekä niiden toteuttamisen. On siis selvää, että pelilogiikan laajuus riippuu pelin sääntöjen määrästä ja niiden monimutkaisuudesta. Esimerkiksi Tetriksen kaltaisen pelin pelilogiikka sisältää palikat, pelialueen, pisteiden laskun, pelivalikot, syötteiden tunnistuksen ja palikoiden törmäystunnistimen toimintoineen. Toisaalta pelin säännöt ovat yksinkertaiset: erimuotoiset palikat tippuvat tietyllä nopeudella pelialueelle ja pelaaja koettaa saada palikat täyttämään täysiä rivejä. Mikäli rivi täyttyy, se häviää ja pelaaja saa siitä tietyn määrän pisteitä. Peli loppuu, kun uusi palikka ei mahdu tulemaan pelialueelle. Moniin peleihin kuuluu myös fysiikkamoottori, jonka hallinta on osa pelilogiikkaa. En käsittele fysiikkamoottoria sen laajuuden vuoksi, ja koska sen toiminta vaihtelee peleittäin.

Jos pelin säännöt tai pelioliot muuttuvat – kuten pelikehityksessä usein tapahtuu – vaikuttaa muutos myös pelilogiikkaan. Tämän vuoksi pelin vaatimusten epävakaus johtaa myös pelilogiikan vaatimusten epävakauteen. Tästä esimerkkinä Bungie Studiosin suosittu ensimmäisen persoonan ammuntapeli Halo oli alun perin suunniteltu reaaliaikastrate-

giapeliksi [Boulding, 2003]. Ylläpidettävyyden kannalta tulisi mahdollistaa joustava pelilogiikka, jotta pelin kehitysvaiheessa esiin tulleita ideoita voidaan testata.

Menestyviin peleihin halutaan yleensä luoda lisää sisältöä myös pelin julkaisun jälkeen, esimerkiksi lisäosien muodossa. Tällä hetkellä suosittu tapa uuden sisällön julkaisemiseen on ladattava sisältö eli downloadable content (DLC). Pelien lisäosien uudistukset eivät aina ole pelkästään tarinallisia tai kosmeettisia, vaan usein myös uusia toimintoja. Tästä on esimerkki Relic Entertainmentin Dawn of War -sarja, jonka alkuperäisosassa tarina siirtyi taistelusta toiseen lineaarisesti. Pelin lisäosassa Dark Crusade tuli uusi ominaisuus, planeetan valloitus. Planeetan valloitus muutti tarinan kulun dynaamiseksi kamppailuksi, jossa pelaajan tuli paitsi käydä uusia taisteluita, myös suojata jo vallattuja alueita vihollisten hyökkäyksiltä. [Wikipedia, 2013b] Pelin logiikan kannalta se oli täysin uusi ominaisuus ja tällaisen ominaisuuden lisääminen on haaste pelilogiikan ylläpidettävyydelle, sillä se laajentaa itse pelilogiikkaa ja todennäköisesti tekee myös muutoksia vanhoihin komponentteihin. Mikäli pelilogiikan ylläpidettävyys on huono, näin monipuolisen ominaisuuden lisääminen tulisi kalliiksi. Pelilogiikan toteutuksessa tulisi ylläpidettävyys olla yksi keskeisimmistä laadun vaatimuksista ja eritoten laajennettavuus tulisi siten pitää mielessä.

Pelilogiikka sisältää myös syötteiden tunnistamisen ja siten syötteiden hallinta on ylläpidon kannalta mielenkiintoinen. Pelien siirtäminen eri alustoille tarkoittaa monesti myös hallintalaitteiden muuttumista, esimerkiksi PC:n näppäimistö ja hiiri -kombinaatio voi vaihtua mobiililaitteen kosketusnäyttöön. Ylläpidettävyyden kannalta syötelaitteiden muuttuminen tulisi ottaa huomioon pelilogiikkaa toteuttaessa.

3. Olio-ohjelmointi ja suunnittelumallit

Abstraktisti määriteltynä olio käsittää jonkin kokonaisuuden, joka voidaan erottaa sen ympäristöstä [Koskimies, 2000]. Näihin olioihin perustuvasta ohjelmoinnista tai suunnittelusta käytetään sanaa olioperustainen. Olioperustainen tapa kuvata asioita on muita ohjelmointitapoja, kuten proseduraalista ohjelmointia, laajempi ja se vastaa hyvin ihmisen tapaa hahmottaa maailmaa. Koskimies [2000] huomioi myös, että olioparadigmalla on muita ohjelmointiparadigmoja paremmat lähtökohdat hyvän käytettävyyden ja ylläpidon saavuttavuuteen. En käsittele olioperustaisuuden valitsemista peliohjelmointiin, sillä se on alan standardi, mutta esimerkiksi Gold [2004] perehtyy asiaan ja toteaa olio-ohjelmoinnin sopivan hyvin pelikehitykseen.

3.1 Suunnittelumallit yleisesti

Ohjelmoinnissa yleisesti on kyse ongelmien ratkaisemisesta. Monesti pelkkä ratkaisu ei riitä, sillä ongelma tulisi ratkaista myös jossain määrin tehokkaasti, varsinkin pelien kohdalla. Olio-ohjelmoinnin monipuolisuuden vuoksi ongelmilla on yleensä lukuisia ratkaisuvaihtoehtoja ja tehokkaan tavan löytäminen vaatii yleensä kokemusta. Kokeneemmatkin ohjelmoijat joutuvat kuitenkin kehittämään ratkaisujaan useaan otteeseen.

[Gamma et al., 1994] Gamma ja muut määrittelevät suunnittelumallien olevan dokumentoituja, testattuja ja toimivia ratkaisuja, jotka ovat kehittyneet asiantuntijoiden omien uudelleenkäytettyjen ratkaisujen tuloksena.

Suunnittelumallit ovat siis ratkaisuja tiettyyn ongelmaan. Ratkaisuja ja malleja samaan ongelmaan voi olla useita, minkä vuoksi sopivimman mallin löytämiseksi Gamma ja muut ehdottavat että mallin dokumentista tulisi löytyä mallin käytön seuraukset, nimen, ongelman ja itse ratkaisun lisäksi. Seuraamukset yleensä koskevat tilan ja ajankäyttöä, eli muistia ja laskutehoa. Toisaalta mallit vaikuttavat myös ohjelman joustavuuteen, laajennettavuuteen ja siirrettävyyteen [Gamma et al., 1994], jotka ovat osa ylläpidettävyyttä. Mallit vaikuttavat siis järjestelmän ominaisuuksiin ja niitä voidaan hyödyntää jo arkkitehtuurin suunnitteluvaiheessa.

3.2 Suunnittelumallit peliohjelmoinnissa

Ylläpidettävyydellä tarkoitetaan siis järjestelmän kykyä vastaanottaa muutoksia ja siihen voidaan vaikuttaa suunnittelumalleilla. Oikein käytettynä ne parantavat ainakin peleissä joustavuutta ja uudelleenkäytettävyyttä. [Ampatzoglou and Chatzigeorgiou, 2006] Luvussa 2 esittelin peliohjelmoinnin ylläpidettävyyden kannalta mielenkiintoisia huomioita grafiikka-, tekoäly- ja pelilogiikkaohjelmoinnissa. Grafiikkaohjelmoinnissa näitä olivat muokattavat visuaaliset toteutukset peliolioille sekä mahdollinen siirrettävyys eri alustoille. Tekoälyohjelmoinnin kannalta mielenkiintoista oli tekoälyn vahventaminen, heikentäminen ja uusien ratkaisujen sekä rinnakkaistoteutusten mahdollistaminen. Pelilogiikkaohjelmoinnissa tulisi mahdollistaa alijärjestelmien muuttaminen, esimerkiksi hallintalaitteiden vaihtuminen, uusien suhteiden lisääminen ja toimintalogiikan muuttaminen.

Suunnittelumalleja on kymmenittäin, ja useat mallit voivat ratkaista saman ongelman. Olen valinnut käsiteltäväksi kaksi ylläpidettävyyttä parantavaa mallia kutakin käsiteltävää peliohjelmoinnin osa-aluetta kohden.

3.2.1 Grafiikka

Grafiikkaohjelmoinnissa voidaan käyttää silta- ja tilamallia mahdollistamaan rinnakkaistoteutukset, selkeä rakenne ja siirrettävyys.

Siltamallin tarkoituksena on erottaa abstraktio ja sen toteutus siten, että niitä voidaan muokata toisistaan riippumatta [Gamma et al., 1994]. Ampatzoglou ja Chatzigeorgiou [2004] toteavat, että mallia voidaan hyödyntää kaikissa 3D-olioissa, joissa olion malli ja sen värittäminen on mahdollista erottaa. Mallin hyötynä on se, että usein peleissä tarvitsee värittää olioita samantyylisesti, esimerkiksi pelaajan valitsemalla värillä. Toisaalta voidaan myös haluta, että pelaajalla on useita eri hahmoja tai rakennuksia, jotka kaikki pystytään värittämään pelaajan valitsemalla värillä. Abstraktion ja sen toteutuksen erottelu mahdollistaa lisäksi asetusten sekä toteutuksen muuttamisen ajonaikaisesti. Siltamallia käyttävä järjestelmä on laajennettavissa, sillä abstraktiota sekä toteutusta voidaan laajentaa erillisesti milloin tahansa. [Gamma et al., 1994] Siltamallin avulla on myös mahdollista

erottaa laitekeskeiset ominaisuudet ja mahdollistaa esimerkiksi laitekohtainen versio näytölle tulostamisesta. Tämä helpottaa huomattavasti järjestelmän siirrettävyyttä. Siltamalliin perehdytään tarkemmin kohdassa 4.1.

Tilamalli mahdollistaa toiminnan muuttamisen olion sisäisen tilan muuttuessa [Gamma et al., 1994]. Grafiikkaa ohjelmoidessa Ampatzoglou ja Chatzigeorgiou huomioivat useita eri ominaisuuksia, joita mallilla voidaan toteuttaa. Esimerkiksi tällaisesta ominaisuudesta on tarkkuusaste eli level-of-detail (LOD). Tarkoitan tarkkuusasteella peliolion graafista tarkkuutta. Esimerkiksi peliolion tarkkuutta voidaan haluta muuttaa sen paikasta riippuen. Mikäli peliolio on ruudusta katsottuna lähellä, halutaan tarkkuus korkeaksi. Toisaalta mitä kauempana peliolio on ruudusta, sitä vähemmän tarkkuutta se vaatii. Tilamalli on laajennettavissa helposti ja se on selkeä, sillä tilan vaihtaminen tapahtuu yhden muuttujan avulla. [Gamma et al., 1994] Tilamallia voidaan hyödyntää kaikissa tapauksissa, joissa olion sisäinen tila vaikuttaa sen grafiikkaan [Ampatzoglou and Chatzigeorgiou, 2006]. Tilamallin avulla voidaan tehdä grafiikasta joustavaa ja muokattavaa. Tilamallin ovat tarkasti määritelleet ja dokumentoineet Gamma ja muut [1994].

3.2.2 Tekoäly

Tekoälyn ohjelmoinnissa voidaan käyttää strategia- ja komposiittimallia laajennettavuuden, muokattavuuden sekä usean ratkaisun mahdollistamiseksi.

Strategiamalli määrittää algoritmiperheen, jonka algoritmeja voidaan muuttaa toisistaan riippumatta [Gamma et al., 1994]. Strategiamallia voidaan käyttää esimerkiksi shakkipelin eritasoisten tekoälypelaajien simulaatiossa, jossa jokainen tekoälypelaaja voi käyttää erilaisia heuristiikkoja liikkeiden muodostamiseen. Mallin käytön sijaan heuristiikkavalinnat voidaan toteuttaa myös esimerkiksi ehtolauseilla, mutta mallia käytettäessä ratkaisu on helpompi ylläpitää [Ampatzoglou and Chatzigeorgiou, 2006]. Strategiamalliin perehdytään tarkemmin kohdassa 4.2.

Komposiittimallin tarkoitus on luoda olioista puumalli siten, että käsiteltäessä oliota ei ole väliä, onko kyseessä yksi komposiittiolio vai niistä muodostuva ryhmä. [Gamma et al., 1994] Komposiittimallin käyttöä tekoälyssä on käsitellyt Buckland [2005]. Komposiittimallia voidaan hyödyntää rakennettaessa toimintastrategiaa peliagentille. Kirjassa komposiittimallia hyödynnetään hierarkkisessa maalipohjaisessa sovittelumallissa (Hierarchical goal-based arbitration design). Komposiittimallia voidaan käyttää tilanteessa, joissa agentille asetetaan tehtäviä, jotka voivat sisältää alitehtäviä. Komposiittimalliin lisätään toimintojen arviointialgoritmi, joka valitsee, minkä tehtävän agentti aloittaa. Buckland [2005] toteaa, että mallia on helppo laajentaa. Komposiittimallin monipuolisuuden johdosta se on myös joustava. Komposiittimallin ovat tarkasti määritelleet ja dokumentoineet Gamma ja muut [1994].

3.2.3 Pelilogiikkaohjelmointi

Pelilogiikan ohjelmoinnissa tarkkailijamalli mahdollistaa olioiden väliset relaatiot ja fasadimalli selkeyttää ohjelman rakennetta.

Tarkkailijamalli (Observer pattern) mahdollistaa yhden suhde moneen -tyyppisen, riippuvuuden olioiden välille siten, että useat oliot voivat saada tiedon tarkkailtavan olion muuttamisesta [Gamma et al., 1994]. Pelin reagoiminen pelaajan toimintaan on monissa peleissä keskeistä. Tarkkailijamallin käyttäminen siihen, että pelimaailma voi reagoida pelihahmoon reaaliajassa, on selkeä ratkaisu. Tarkkailijamallia voidaan hyödyntää myös pelin sisäisessä logiikassa. Ampatzoglou ja Chatzigeorgiou [2006] antavat tästä esimerkin jalkapallovalmennuspelistä, jossa pelaajaoliot noudattavat joukkueelle annettuja toimintoja. Pelaajia voidaan asettaa tarkkailemaan joukkuetta, jolloin joukkueen aloittaessa harjoittelun pelaajat päivittyvät kyseisen hajottelun mukaisesti. Tarkkailijamalli luo vain abstraktin riippuvuuden, minkä vuoksi esimerkiksi kun joukkueeseen asetettu pelaaja saa tiedon jostain harjoittelusta, on pelaajaolion päätettävissä, kuinka se tietoon reagoi. Pelaajaolion oma tila voi vaikuttaa sen reagoimiseen, eli esimerkiksi jos pelaaja on merkitty loukkaantuneeksi, niin pelaajaolio voi jättää harjoittelutiedotukseen reagoimatta. Tarkkailijamallilla on siis mahdollista luoda monipuolisia ja muokattavia tapoja reagoida pelaajan tekemiin päätöksiin ja se mahdollistaa siten joustavan ja laajennettavan tavan olioiden välisten suhteiden luomiseen. Tarkkailijamalliin perehdytään tarkemmin kohdassa 4.3.

Fasadimallilla on tarkoitus yhdistää useita alijärjestelmiä yhteiseen korkeammantason rajapintaan, mikä lisää käytettävyyttä ja vähentää alijärjestelmien keskinäisiä riippuvuuksia [Gamma et al., 1994]. Fasadimallia kutsutaan monesti myös nimellä hallinnointiluokka. [DeLoura, 2000] Pelilogiikkaohjelmoinnissa voi olla useita alijärjestelmiä, esimerkiksi tallennus- ja latausjärjestelmä, käyttöliittymäjärjestelmä sekä äänijärjestelmä. Niiden toiminnot eivät välttämättä korkealla tasolla muutu pelin kehityksen aikana, mutta esimerkiksi laitteiston muuttuessa alijärjestelmiä voidaan joutua sovittamaan uuteen laitteistoon. Fasadimallin avulla alijärjestelmien muuttaminen ei vaikuta muihin järjestelmiin, mutta muutokset tulee huomioida fasadimallin rajapinnassa. [DeLoura, 2000] Fasadimalli mahdollistaa siis muutokset pelin alijärjestelmiin rikkomatta kuitenkaan muita pelin osioita. Tämä johtaa laajennettavuuden, muokattavuuden ja siirrettävyyden kannalta parempaan ylläpidettävyyteen. Fasadimallin ovat tarkasti määritelleet ja dokumentoineet Gamma ja muut [1994].

4. Suunnittelumallien soveltaminen

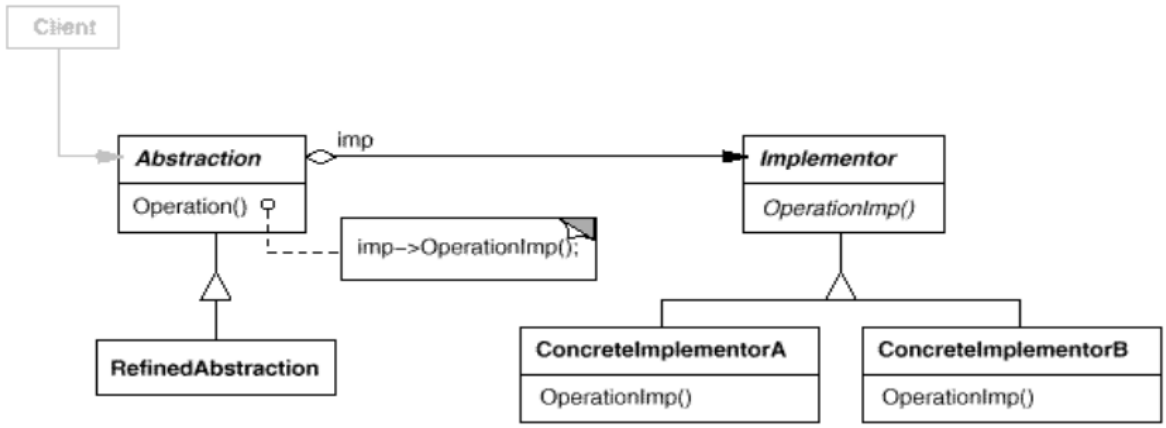
4.1. Siltamalli

Siltamallin tarkoituksena on jakaa toteutus ja abstraktio siten, että niitä voidaan muokata toisistaan riippumatta. Yleisesti perintää hyödynnetään, jos abstraktiolla on useita toteutustapoja, mutta monesti se on liian sitova. Perimisellä toteutettu abstraktio johtaa sen laajentamisen vaikeuteen, sekä sitoo toteutuskoodin alustakohtaiseksi. [Gamma et al., 1994]

Siltamallin oleellisin hyöty ylläpidettävyyteen tulee jo sen määritelmästä. Toteutuksen ja abstraktion vapaa muokattavuus toisistaan riippumatta johtaa ylläpidettävyyden kannalta laajennettavuuteen ja muokattavuuteen. Siltamallin käyttäessä erotetaan rajapinta ja toteutus, mikä mahdollistaa paitsi ajonaikaisen muokkaamisen, myös selkeyttää rakennetta. [Gamma et al., 1994]

4.1.1 Siltamallin toiminta

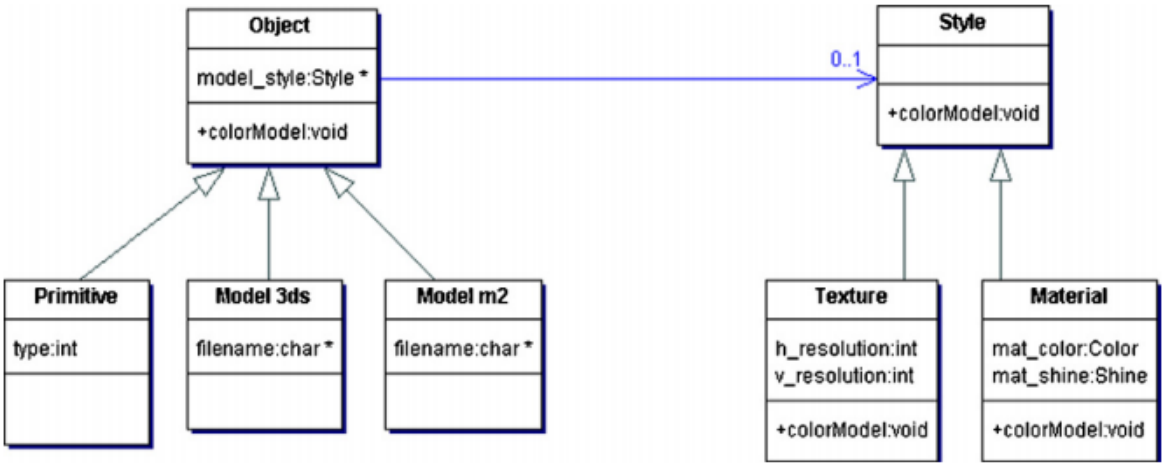
Siltamalli (Kuva 1) jakautuu neljään toimijaan: Abstraktio (Abstraction), JalostettuAbstraktio (RefinedAbstraction), Toteutuuttaja (Implementor) ja VarsinainenToteuttaja (ConcreteImplementor). Abstraktio määrittää abstraktion rajapinnan sekä pitää viitettä Toteuttajaan. JalostettuAbstraktio laajentaa Abstraktion määrittämää rajapintaa. Toteuttaja määrittää toteutuksiin rajapinnan. VarsinainenToteuttaja toteuttaa Toteuttaja-rajapinnan ja määrittää sen varsinaisen toteutuksen. Abstraktio välittää Asiakkaan (Client) toiminta-pyynnöt siihen liitettyyn Toteuttajaan. [Gamma et al., 1994]



Kuva 1. Yleinen siltamalli [Gamma et al., 1994]

4.1.2 Esimerkki siltamallin soveltamisesta

Ampatzoglou ja Chatzigeorgiou [2006] huomioivat, että siltamallia voidaan käyttää kaikissa sellaisissa 3D-olioissa, joissa olio voidaan värittää usealla tavalla. Siltamallia voidaan hyödyntää myös mallin puolella, sillä 3D-malli voi muuttua tai se voidaan ladata erilaisessa muodossa. Esimerkissä (Kuva 2) Abstraktio on Object-luokka, jolla on viite Toteuttajaan eli Style-luokkaan. JalostetutAbstraktiot eli Primitive, Model 3ds ja Model m2, laajentavat Abstraktiota ja TodellisetToteuttajat, eli Texture ja Material, toteuttavat varsinaisen Toteuttajan eli Style-luokan. [Ampatzoglou and Chatzigeorgiou, 2006]



Kuva 2. Esimerkin siltamalli [Ampatzoglou and Chatzigeorgiou, 2006]

Esimerkkissä siltamallin hyödyt konkretisoituvat siten, että Object-luokkaa ja Style-luokkaa voidaan muokata erikseen. Mikäli myöhemmin tulee käyttöön uusi 3D-paketti ja sen lataaja, se voidaan lisätä suoraan malliin muokkaamatta muita abstraktioita. Sama pätee toteutuksen puolella, eli mikäli tulisi esimerkiksi uusi väristandradivaatimus materiaaleille, voidaan se lisätä toteutukseen vain pienin muutoksin. [Ampatzoglou and Chatzigeorgiou, 2006]

Ampatzogloun ja Chatzigeorgioun mukaan siltamallia voidaan siis hyödyntää 3D-mallien toteutuksen ja abstraktion erottamisessa. Siltamallia voidaan yleensä käyttää myös silloin, kun jonkin asian toteutus voi vaihdella. Esimerkiksi grafiikka-ajurien vaihtaminen voi toimia siltamallin avulla, milloin koko pelin grafiikkamoottori voidaan vaihtaa riippuen pelin asetuksista. [Singh, 2012]

4.2. Strategiamalli

Strategiamallin tarkoituksena on luoda eräänlainen algoritmiperhe, jonka algoritmit tekevät pääosin saman asian, mutta erillä tavalla. Mallin algoritmit erotetaan toisistaan siten, että niitä voidaan muuttaa toisistaan riippumatta. Eri algoritmeja kutsutaan mallissa strategioiksi. Strategiamallin tarkoituksena on mahdollistaa eri algoritmin käyttö riippuen sen kutsujasta.[Gamma et al., 1994]

Gamma ja muiden mukaan strategiamallin käytöstä saadaan merkittävää hyötyä luettavuuteen sekä ylläpidettävyyteen, sillä siitä seuraa useita hyödyllisiä ominaisuuksia. Strategiamallin avulla algoritmeja saadaan jaettua ryhmiin, ja siten niiden yhtäläisyyksiä on helpompi nähdä sekä hyödyntää esimerkiksi periyttämisessä. Strategiamalli mahdollistaa myös aliluokittamisen korkeammalla tasolla kuin periyttämisen avulla tapahtuvaan aliluokittamiseen, sillä se ei ota lainkaan kantaa toteutukseen. Algoritmien erittely omiin luokkiinsa vähentää siten ehtolauseiden määrää yhdessä luokassa. Strategiamalli mahdollistaa useita toteutuksia samasta toiminnasta mahdollistaen esimerkiksi vaihtelu ajan ja

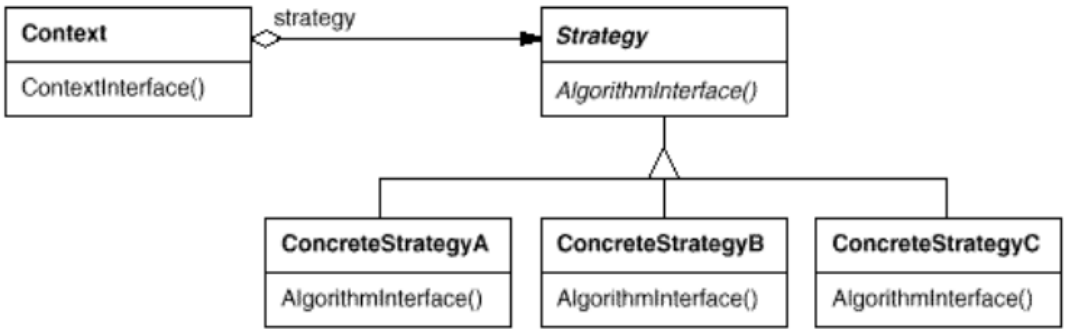
tilan välillä. [Gamma et al., 1994]

Mikään ratkaisu ei ole täydellinen ja Gamma ja muut muistuttavatkin, että strategiamallin käytöstä seuraa myös haittoja. Mikäli strategiat tuottavat eri tuloksia tai saavat eri syötteitä, on kutsujan tiedettävä, mitä strategiaa pitää käyttää, mikä johtaa kutsujan tietoisuuteen strategioista ja toteutuksen kannalta niiden sekoittumiseen (coupling). Ongelmallista voi olla se, etteivät strategiat välttämättä tarvitse samoja tietoja toimiakseen, mikä voi johtaa strategiamallin kutsumiseen ylimääräisillä tiedoilla. Huonossa tapauksessa turhien tietojen luomiseen saatetaan käyttää merkittävästi resursseja. Strategiamalli luo myös useita olioita, joiden hallintaa tulee suunnitella. [Gamma et al., 1994]

4.2.1 Strategiamallin toiminta

Strategiamallissa (Kuva 3) on kolme toimijaoliota: Strategia (Strategy), KonkreettinenStrategia (ConcreteStrategy) ja Konteksti (Context). Strategia määrittelee algoritmeille yhteisen rajapinnan, jota Konteksti käyttää KonkreettisenStrategian määrittelemällä tavalla. KonkreettinenStrategia toteuttaa jonkin algoritmin käyttäen Strategian määrittämää rajapintaa. Kontekstilla on KonkreettisenStrategian asetustiedot, viite Strategiaan ja se voi määrittää rajapinnan, jolla Strategia pääsee Kontekstin tietoihin. [Gamma et al., 1994]

Strategia ja Konteksti toteuttavat yhteistyössä jonkin tietyn algoritmin. Konteksti voi välittää tarvittavat tiedot, tai itsensä, Strategian operaatioille algoritmia tarvittaessa. Algoritmin kutsuja käyttää Kontekstia välittääkseen tiedot Strategialle ja monesti Kontekstiin sijoitetaan myös KonkreettinenStrategiaolio [Gamma et al., 1994].

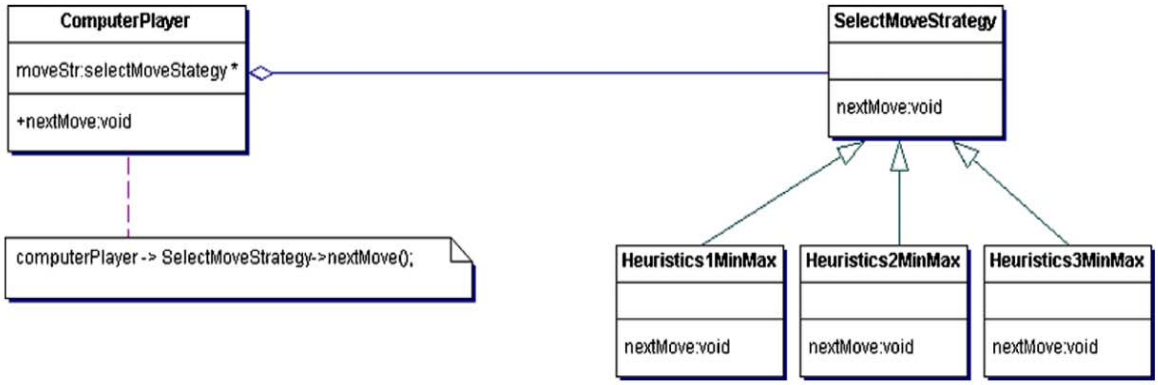


Kuva 3. Yleinen strategiamalli [Gamma et al., 1994]

4.2.2 Esimerkki strategiamallin soveltamisesta

Strategiamallia voidaan käyttää ainakin shakkipelissä, josta Ampatzoglou ja Chatzigeorgiou antavat esimerkin (Kuva 4). Oletetaan, että shakkiturnauksessa on useita tekoälypelaajia, jotka käyttävät erilaisia heuristiikkoja seuraavan liikkeensä laskemiseen. Tässä esimerkissä tekoälypelaaja (ComputerPlayer) vastaa Kontekstia ja Strategia on ValitseLiikeStrategia (SelectMoveStrategy), joka määrittää kaikille strategioille yhteisen rajapinnan. Tekoälypelaaja käyttää ValitseLiikeStrategiaa valitsemaan oikean algoritmin käytten KonkreettistaStrategiaa. KonkreettisetStrategiat, esimerkissä Heuristics1MinMax,

Heuristics2MinMax ja Heuristics3MinMax, toteuttavat algoritmin ValitseLiikeStrategian rajapinnan mukaan.



Kuva 4. Esimerkin strategiamalli [Ampatzoglou and Chatzigeorgiou, 2006]

Strategiamallin tarkoituksena on siis mahdollistaa usean eri toteutuksen saman ongelman ratkaisemiseksi. Peleissä on usein tilanteita, joissa strategiamallia voidaan hyödyntää. Strategiamallia voitaisiin käyttää esimerkiksi liikkumisalgoritmin valinnassa, jolloin eri peliolio käyttää liikkumiseen omia vaatimuksia, mutta yleisesti liikkuminen on kaikille olioille sama. Esimerkiksi ihmisolion tulee väistää puita ja seiniä, mutta kummitusolion ei. [GPWiki, 2012] Strategiamallia voidaan hyödyntää myös ampumistoiminnan valitsemisessa, jolloin ampumistapa erotetaan ampujaoliosta strategiamallin avulla [Cumps Consulting, 2008].

4.3. Tarkkailijamalli

Tarkkailijamallin tarkoituksena on luoda yhden suhde moneen -relaatio olioiden välille. Kun yksi muuttuu, niin siihen asetetut tarkkailijat saavat tiedon muutoksesta ja päivittyvät sen mukaan. [Gamma et al., 1994] Tarkkailijamalli on yksi tunnetuimmista suunnittelumalleista, sillä se on osa MVC-kehystä, joka on tuttu lähes kaikista graafisista ja tapahtumapohjaisista ohjelmista.

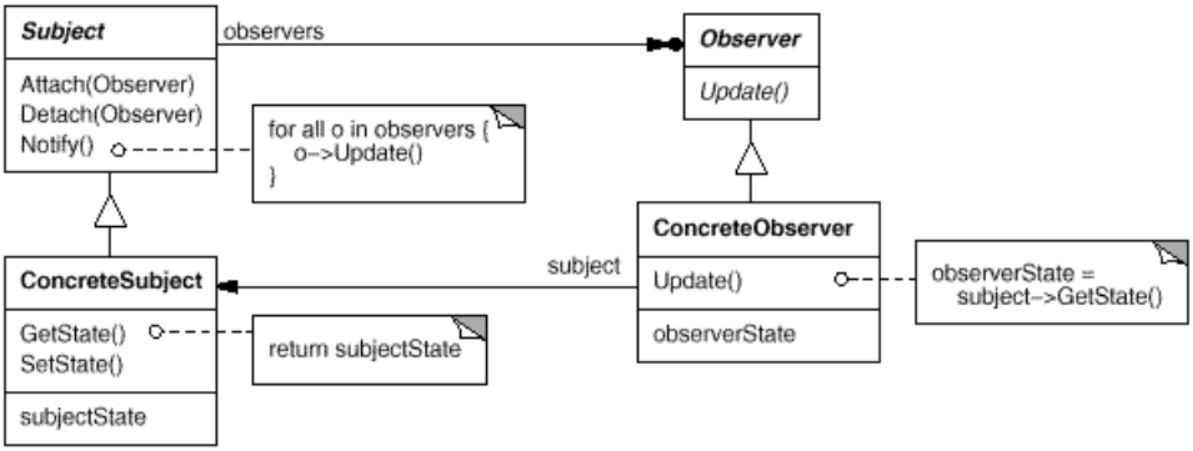
Tarkkailjamallin käyttö mahdollistaa monipuolisen yhteyden eri komponenttien välillä. Tarkkailtavan olion tulee tietää vain viite sitä tarkkailevaan olioon, joten ne ovat hyvin väljästi riippuvaisia toisistaan. Tämä mahdollistaa myös täysin erilaisten olioiden kommunikaation. [Gamma et al., 1994] Tarkkailijamallin avulla voidaan lähettää viesti tuntematta sen vastaanottajaa, mikä tarkoittaa myös sitä, että vastaanottajia voidaan lisätä tai poistaa vaikuttamatta viestittäjään. Koska viesteille ei ole määritelty tiettyä vastaanottajaa, voi vastaanottaja päättää mihin viesteihin se reagoi. [Gamma et al., 1994]

Tarkkailijamallin vapaassa viestinnässä piilee kaksi oleellista vaaraa. Ensinnäkin jos tarkkailtavassa oliossa tapahtuu muutos, ei ole tietoa kuinka kauan järjestelmällä kestää reagoida siihen. Tehokkuuden näkökulmasta pienikin muutos voi laukaista raskaan tapahtumasarjan, kun jokainen tarkkailija reagoi muutokseen omalla tavallaan. Toinen

ongelma piilee muutoksen tiedottamisesta, sillä vähäisen riippuvuuden takia tarkkailijoilla voi olla vaikea selvittää, mikä tarkkailtavassa oliossa on muuttunut. [Gamma et al., 1994]

4.3.1 Tarkkailijamallin toiminta

Tarkkailijamalli (Kuva 5) koostuu neljästä toimijasta: Kohde (Subject) ja Tarkkailija (Observer) sekä niiden varsinaiset toteutukset VarsinainenKohde (ConcreteSubject) ja VarsinainenTarkkailija (ConcreteObserver). Kohteella on tiedossa siihen lisätyt Tarkkailijat sekä palvelut tarkkailijoiden lisäämiseen, poistamiseen ja muutoksesta ilmoittamiseen. Tarkkailija määrittää rajapinnan VarsinaistenTarkkailijoiden päivitystä varten. VarsinainenKohde pitää sisällään tarkkailun kohteena olevan tilan ja lähettää Kohteelle tiedon, mikäli se muuttuu. VarsinaisellaTarkkailijalla on viite VarsinaiseenKohteeseen. Lisäksi se ylläpitää tilaa, joka vastaa VarsinaisenKohteen tilaa, ja toteuttaa Tarkkailija-rajapinnan määrittämän päivityksen, jolla VarsinaisenTarkkailijan tila päivitetään vastaamaan VarsinaisenKohteen tilaa. [Gamma et al., 1994] Tarkkailija siis ilmoittaa VarsinaiselleTarkkailijalle, milloin VarsinaisenKohteen tila on muuttunut ja VarsinainenTarkkailija reagoi siihen päivittämällä oman tilansa vastaamaan muuttunutta tilaa. Päivitystiedon saamisen jälkeen VarsinainenTarkkailija voi joutua selvittämään muutostietoja VarsinaisestaKohteesta. [Gamma et al., 1994]

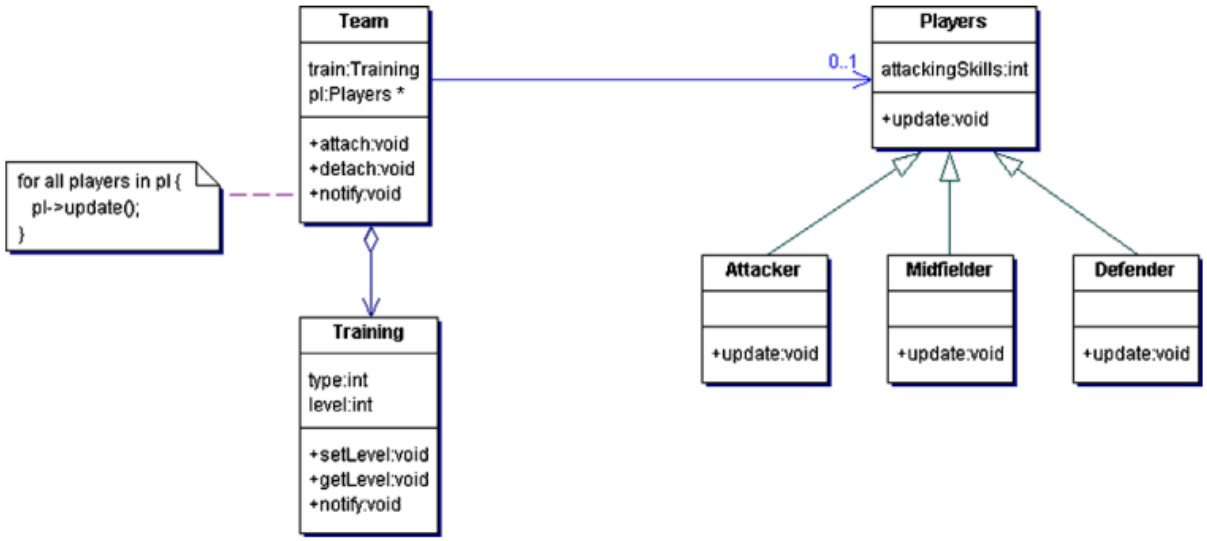


Kuva 5. Yleinen tarkkailijamalli [Gamma et al., 1994]

4.3.2 Esimerkki tarkkailijamallin soveltamisesta

Tarkkailijamallin esimerkkinä (Kuva 6) Ampatzoglou ja Chatzigeorgiou [2006] käyttävät jalkapallojoukkueen manageerointipeliä, jossa Kohteena toimii Team, Tarkkailijana Players, VarsinaisenaKohteena Training ja KonkreettisinaTarkkailijoina toimivat Attacker, Midfielder ja Defender. Esimerkissä Team-luokkaan voidaan lisätä Players-oliota tarkkailijoiksi. Kun Team-luokkaan asetettua Training-luokkaa päivitetään, lähtee siitä ilmoitus Players-oliolle. Players-olion toteuttajat, eli joko Attacker, Midfielder tai Defender, päivittävät oman tilansa vastaamaan päivittynyttä tietoa. Vastuu päivityksestä on kuitenkin Playersin

toteuttajilla, joten esimerkiksi jos jokin pelaajista on merkattu tilaan "kipeä", voi pelaaja ohittaa päivityksen.



Kuva 6. Esimerkin tarkkailijamalli [Ampatzoglou and Chatzigeorgiou, 2006]

Tarkkailijamallin käyttäminen automatisoi tilojen päivitystä, mikäli olion tila on riippuvainen jonkin toisen olion tilasta. Peleissä tällainen suhde löytyy monesta paikasta ja pelien interaktiivisuuden takia tilojen muuttuminen voi tapahtua milloin vain. Tarkkailijamallia voidaan käyttää esimerkiksi vastustajien tekoälyssä siten, että vastustajalle tulee ilmoitus, mikäli pelihahmon tila muuttuu. Esimerkiksi vastustaja voi hyökätä erikoisliikkeellä, mikäli se huomaa, että pelaajan hahmo ei pysty liikkumaan ja väistämään sitä. Tarkkailijamallia voidaan hyödyntää kaikissa käsitellyissä peliohjelmoinnin osa-alueissa ja erikoishuomiona myös käyttöliittymissä.

5. Yhteenveto

Pelien interaktiivisuudesta johtuen niiden sisäinen logiikka on usein hyvin monimutkainen, ja pelien taiteellisen perustan sekä peleille tyypillisen kehitysympäristön johdosta pelien kehittämiseen voi kohdistua hyvin paljon muutoksia. Ylläpidettävyys on järjestelmään kohdistuvien muutosten hallinnointia. Käsittelin grafiikka-, tekoäly- ja pelilogiikkaohjelmoinnin kannalta ylläpidettävyyteen liittyviä huomioita ja esitteli suunnittelumalleja, joilla niiden ylläpidettävyyttä voidaan parantaa. Lopuksi esittelin tarkemmin kolmen suunnittelumallin toimintaa ja avasin niiden käyttömahdollisuuksia peliohjelmoinnissa. Peleihin yleisesti on kohdistunut kauan tutkimuksia, mutta videopelien kehittämiseen liittyviä kysymyksiä on tutkittu vähemmän. Peliteollisuus on nauttinut kuitenkin suurta suosiota viimeisen kymmenen vuoden aikana, ja uskon, että seuraavien vuosien aikana tullaan julkaisemaan huomattavasti enemmän tutkittua materiaalia pelikehityksestä ja sen prosessista tulee nykyistä standardoidumpi.

Viiteluettelo

[Ampatzoglou and Chatzigeorgiou, 2006] Apostolos Ampatzoglou and Alexander Chatzigeorgiou, Evaluation of object-oriented design patterns in game development. *Information and Software Technology* **49**, 5(May 2007), 445-454.

[Boulding, 2003] Aaron Boulding, Halo: The RTS. Available as http://www.ign.com/articles/2003/06/19/halo-the-rts. Checked 21.11.2013.

[Buckland, 2005] Mat Buckland, Programming Game AI by Example, Wordware Publishing, Inc., 2005.

[Cumps Consulting, 2008] Cumps Consulting Blog, Design Patterns - Strategy Pattern. Available as http://www.cumps.be/nl/blog/read/design-patterns-strategy-pattern. Checked 21.11.2013.

[DeLoura, 2000] Mark DeLoura, *Game Programming Gems*, Charles River Media 2000.

[Gamma et al., 1994] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[Gold, 2004] Julian Gold, *Object-oriented Game Development*. Addison-Wesley, 2004.

[GPWiki, 2012] The Game Programming Wiki, Strategy pattern. Available as http://content.gpwiki.org/index.php/Strategy_pattern. Checked 21.11.2013.

[Koskimies, 2000] Kai Koskimies, *Oliokirja*, Talentum, 2000.

[McLaughlin, 2013] Martyn McLaughlin, New GTA V release tipped to rake in £1bn in sales. Available as http://www.scotsman.com/lifestyle/technology/new-gta-v-release-tipped-to-rake-in-1bn-in-sales-1-3081943. Checked 21.11.2013.

[Sanchez-Crespo, 2003] Daniel Sanchez-Crespo, *Core Techniques and Algorithms in Game Programming*. New Riders Publishing, 2003.

[Singh, 2012] Rahul Rajat Singh, Understanding and Implementing Bridge Pattern in C#. Available as http://www.codeproject.com/Articles/434352/Understanding-and-Implementing-Bridge-Pattern-in-C. Checked 21.11.2013.

[Sommerville, 2007] Ian Sommerville, *Software Engineering*, 8th ed.Addison Wesley, 2007.

[Stacey and Nandhakumar, 2005] Patric Stacey and Joe Nandhakumar, Managing Projects in a Games Factory: Temporality and Practices. In: *Proc. of the 38th Annual Hawaii International Conference on System Sciences,* IEEE, 2005, 234a.

[Wikipedia, 2013a] Wikipedia, Game Programmer. Available as http://en.wikipedia.org/wiki/Game_programmer. Checked 21.11.2013.

[Wikipedia, 2013b] Wikipedia, Warhammer 40,000: Dawn of War – Dark Crusade. Available as http://en.wikipedia.org/wiki/Dark_Crusade. Checked 21.11.2013.

Lääketieteellisten opetussimulaatioiden käytettävyydestä

Timo Perälä

Tiivistelmä.
Lääketieteelliset opetussimulaatiot on otettu hyvin vastaan lääketieteen opetuksessa ja niiden käyttämisessä näyttäisi olevan monenlaisia osoitettuja hyötyjä. Tässä tutkielmassa selvitetään kirjallisuuskatsauksen keinoin käsityksiä lääketieteellisten opetussimulaatioiden käytettävyydestä. Käytettävyyttä ei ole tutkittu kovinkaan systemaattisesti lääketieteellisten opetussimulaatioiden viitekehyksessä. Tämä tutkielma tarkastelee pääasiassa olemassa olevan kirjallisuuden lähestymistapoja aihepiiriin. Lääketieteelliset opetussimulaatiot vaikuttaisivat tarvitsevan merkittävästi lisää käytettävyystutkimusta.

Avainsanat ja -sanonnat: Opetussimulaatio, käytettävyys, lääketiede.
CR-luokat: K.3.1, J.3

1. Johdanto

Tämän tutkielman tavoitteena on keskustella kirjallisuuskatsauksen keinoin lääketieteellisten opetussimulaatioiden käytettävyydestä. Tutkielmassa määritellään lyhyesti lääketieteelliset opetussimulaatiot ja käytettävyys ja tuodaan esiin lääketieteellisten opetussimulaatioiden käyttämisen hyötyjä ja haasteita esitettävien tutkimustulosten avulla sekä pyritään julkaistujen tutkimusten kautta etsimään sellaisia käytettävyyden kriteereitä, joita voitaisiin soveltaa erityisesti lääketieteellisiin opetussimulaatioihin. Lääketieteelliset opetussimulaatiot kattavat laajasti erilaisia välineitä ja ohjelmia. Tämän työn puitteissa ei ole mahdollista eritellä erilaisten lääketieteellisten opetussimulaatioiden käytettävyyttä, vaan lääketieteellisiä opetussimulaatioita käsitellään tutkielmassa pääosin yhtenä kokonaisuutena. Tavoitteena on esitellä aiheeseen liittyviä tutkimuksia, keskustella yleisten käytettävyyden kriteereiden soveltuvuudesta lääketieteellisiin opetussimulaatioihin sekä lopuksi pohtia avainkysymyksiä lääketieteellisten opetussimulaatioiden käytettävyydestä ja jatkotutkimustarpeista.

Pedagogiikka ja vuorovaikutteinen teknologia ovat muodostaneet yhdessä kokonaisuuden, josta olen ollut kiinnostunut jo joitain vuosia. Kandidaatin tutkielman aiheen valitseminen oli helppoa, kun molemmat tieteenalat ja kiinnostuksen kohteet yhdistettiin mahdollisuudella perehtyä opetussimulaatioihin. Lääketiede on aina ollut jossain määrin kiinnostuksen kohde lähipiirissä toimivien henkilöiden lääketieteellisen orientaation vuoksi. Valitsin

aiheen sekä kiinnostuksesta sitä kohtaan että voidakseni oppia lisää edelleen voimakkaasti kehittyviltä tieteenaloilta: vuorovaikutteisesta teknologiasta, pedagogiikasta ja lääketieteestä. Suhteellisen nuori vuorovaikutteisen teknologian tieteenhaara tukee monia muita, jo iäkkäitäkin tieteenaloja löytämään uusia teorioita ja uutta tutkittavaa, mallinnettavaa ja toteutettavaa alueilta, joita on saatettu pitää jo lähes läpitutkittuina.

Koko tutkielmaprosessin ajan oli huomioitava, että aineistoa oli läpikäytävänä kahdelta, jopa kolmelta tieteenalalta. Vuorovaikutteisen teknologian käytettävyysaineistot ja lääketieteen opetussimulaatioihin liittyvät aineistot muodostavat tämän tutkielman rungon pedagogisten aineistojen jäädessä vähemmälle käsittelylle. Aineiston keräämis- ja analyysimenetelmät tähän opinnäytetyöhön voidaan jakaa karkeasti kahteen osaan. Ensimmäiseksi suoritettiin lähdeaineistohakuja erilaisista tietokannoista, mm. ACM Digital Librarysta, Sprigerin elektronisista lehdistä, Science Direct -palvelusta ja IEEE Xploresta. Hakuja radikaalisti rajaamalla ja laajoja tuloksia läpi käymällä löytyi aihepiiriin soveltuvia artikkeleita. Toiseksi saatiin toisilta tutkijoilta ja ohjaajalta joitain artikkeleita, joiden viiteluetteloita läpikäymällä päästiin paremmin kohdistamaan hakuja tutkielman aihepiiriin liittyviin artikkeleihin.

Aineistoa on ollut koko tutkielmaprosessin ajan todella laajasti ja verraten suuri määrä julkaisuja on opinnäytetyön mitoitettu laajuus huomioiden jouduttu jättämään viitteiden ulkopuolelle. Kaikissa vaiheissa aineistoja analysoitaessa on pyritty arvioimaan kriittisesti niiden sopivuutta, luotettavuutta ja eettisyyttä.

Tämä opinnäytetyö valmistui lopulta noin puolentoista vuoden työn tuloksena. Tutkielma oli ajoittain koskemattomana jopa kuukausia kerrallaan, jonka vuoksi kenties normaalia laajempi osa kokonaisajasta käytettiin kunkin työvaiheen uudelleentarkasteluun opinnäytetyön tekemisen jatkuessa. Pohjatyö, kirjallisuushaut ja kirjallisuuden analyysi tehtiin yhtenäisesti työn alkuvaiheessa syksyllä 2012 ja toisaalta työn viimeistely tehtiin hyvin tiiviisti marras-joulukuussa 2013.

2. Lääketieteelliset opetussimulaatiot

Luvussa kuvataan lyhyesti lääketieteen tieteellisen tiedon kertymistä ja opetussimulaatioiden muuttuvaa asemaa lääketieteen opetuksessa. Luvussa määritellään lyhyesti lääketieteellinen opetussimulaatio ja sen eri muodot, kuvataan lääketieteellisten opetussimulaatioiden käyttöä, esitetään kirjallisuudesta nousevia määritelmiä lääketieteellisten opetussimulaatioiden yleisiksi kriteereiksi ja lopuksi mainitaan joitain lääketieteellisten opetussimulaatioiden hyötyjä ja ongelmia.

Lääketieteen nopea uudistuminen asettaa erityisvaatimuksia alan opetukseen ja myös opetusvälineiden ja -menetelmien käyttämiseen. Kaufmannin [2001] mukaan lääketieteellinen

tieto kaksinkertaistuu 6-8 vuoden välein. Lääketieteellisen tiedon määrän kasvamisen lisäksi uudet tutkimukset tuottavat koko ajan myös muutoksia ja tarkennuksia olemassa olevaan tietoon. Tämän tietomäärän opettaminen lääketieteen opiskelijoille on kasvava haaste. Ihmismäisten potilassimulaatioiden käyttöönotto 1900-luvun loppupuolella oli valtava askel lääketieteen opetuksen kehityksessä [Rosen, 2008].

Opetussimulaatiot on otettu hyvin vastaan lääketieteen opetuksessa ja lääketieteelliset opetussimulaatiot on voimakkaasti kasvava alue sekä tutkimuksessa että opetuskäytössä. Lääketieteen opettajat ovat ymmärtäneet, että simulaatiot ovat lääketieteen opetuksen tulevaisuus [Rosen, 2008]. Lääketieteellisiä opetussimulaatioita on alettu suunnitella ja tuottaa monille lääketieteellisen opetuksen osa-alueille operaatioiden lisäksi. Tutkijat ovat myös kiinnostuneet sosiaalisten tilanteiden simuloimisen mahdollisuuksia [Johnsen *et al.*, 2006]. Opetussimulaatioiden määrä ja niiden kattamat lääketieteelliset toiminnot (kuten vastaanotto tai operointi) laajenevat voimakkaasti. Esimerkiksi virtuaaliympäristöjen tekniikoiden avulla voidaan harjoitella anatomiaa, neulan käyttöä, diagnoosin tekemistä tai leikkauksen suorittamista [Tikka, 2001]. Virtuaaliset maailmat, sekä immersiiviset että verkkopohjaiset, ovat lääketieteen opetuksen innovoinnin etulinjassa [Rosen, 2008]. Lääketieteellisillä opetussimulaatioilla voidaan pyrkiä osaksi opetusta ja harjaantumista hyvin monilla lääketieteen eri osa-alueilla. Lääketieteen laaja kenttä antaa mahdollisuuden suunnitella ja toteuttaa hyvin monipuolisesti erilaisia opetussimulaatioita harjoittelun tueksi.

Opetussimulaatioiden tuottaminen ja hankinta ei kuitenkaan ole aina yksinkertaista. Nikoukaranin ja muiden [1998] mukaan simulaatio-ohjelmistojen kasvava määrä ja laatu, hankintahinta, käyttökustannukset sekä monimutkaisten ohjelmistojen hallintaan liittyvät henkilöstökustannukset vaativat asiantuntijuutta ohjelmistoa valittaessa. Opetussimulaatioiden hyödyt vaikuttavat selkeiltä suhteutettuna muuhun opetukseen ja erityisesti muun opetuksen tukena, mutta panos-tuotossuhteena hyötyä on vaikea arvioida melko korkeiden käyttöönotto- ja käyttökustannusten vuoksi.

2.1. Lääketieteellisen opetussimulaation määritelmä

Lääketieteelliset opetussimulaatiot on erittäin laaja käsite, jota käytetään yleisesti kattokäsitteenä erilaisille simulaatiotyypeille, kuten virtuaalimaailmat (virtual realities, VR), standardipotilaat tai näyttelijät, ihmiskehoa tai sen osia esittävät simulaationuket ja tietokoneohjelmat [Johnsen *et al.*, 2005; Mantovani *et al.*, 2003; Nagle *et al.*, 2009]. Tässä tutkielmassa lääketieteellisiä opetussimulaatioita käsitellään pääosin yhtenä kokonaisuutena, vaikka sekä pedagogisesta ja toiminnallisesta että käytettävyyden näkökulmasta kutakin opetussimulaatiotyyppiä voitaisiin käsitellä erillisenä instanssina. Tutkielmassa keskitytään sellaisiin opetussimulaatiotyyppeihin, joissa käytetään tietokonetta tai tietokoneohjelmistoa tai muuta

teknologiaa osana simulaatiota. Standardipotilaat, joilla tarkoitetaan potilaita esittävien ihmisten vakioituja toimintatapoja simulaatiossa ja näyttelijöiden avulla toteutetut opetussimulaatiot jätetään tässä tarkastelun ulkopuolelle.

Opetussimulaatiot voidaan määritellä laajasti tai suppeasti. Lääketieteellisillä opetussimulaatioilla voidaan tarkoittaa kaikkea sellaista toiminnan harjoittelua, jossa pyritään kuvaamaan aitoa tilannetta vaikka oikeaa potilasta ei ole. Toisaalta erilaisten teknisten apuvälineiden (virtuaalimaailmat, potilasnuket, tietokoneet, muu ääni ja/tai kuva) käyttäminen voidaan nähdä usein ehtona. Suppeampana määritelmänä voidaan pitää sellaisia, joissa opetussimulaatiot kytketään johonkin tiettyyn tekniikkaan, esimerkiksi potilasnukkeihin. Naglen ja muiden [2009] määritelmän mukaan lääketieteellisissä simulaatioissa voidaan käyttää ihmiskehoa tai sen osia esittäviä nukkeja, tietokoneohjelmia tai näyttelijöitä. Nuket ovat yleensä tietokoneohjattuja ja luonnollisen kokoisia, ja niille voidaan toteuttaa erilaisia fysiologisia ominaisuuksia [Nagle *et al.*, 2009]. Fysiologiset ominaisuudet voivat sisältää esimerkiksi hengityksen, verenkierron tai pulssin kuvaamisen.

Simulaatioiden tavoitteena ei Aldrichin [2003] mukaan ole vain suoriutua paremmin tai nopeammin, vaan myös oppia sattumalta virheiden kautta. Harjaantuminen ja määrälliset toistot voivat parantaa merkittävästi jonkin yksittäisen suorituksen onnistumisprosenttia, jolloin on mielekkäämpää toteuttaa harjoitteluvaihe soveltuvilta osin opetussimulaattorilla. Uusi saatu tieto tai jopa totaalinen epäonnistuminen kehittävät henkilön ymmärrystä alkuperäisestä tavoitteesta, riippumatta onnistumisen asteesta [Aldrich, 2003]. Opetussimulaatiot harjaannuttavat opiskelijoita usein toistuvien tilanteiden suorittamisessa, mutta auttavat myös kohtaamaan ja toimimaan harvinaisissa tilanteissa ja toisaalta opettavat kohtaamaan epämukavia tilanteita ja selviämään niistä. Rosenin [2008] mukaan simulaatioiden historiallista taustaa voidaan kuvata esimerkiksi lääketieteellisten simulaatioiden laajimmalla määritelmällä: "todellisen asian, tilanteen tai prosessin imitaatio".

2.2. Lääketieteellisten opetussimulaatioiden käytöstä

Opetussimulaatiot ovat vaihtelevasti käytössä eri opetusinstituutioissa ja työpaikkojen täydennyskoulutuksissa. Simulaatioita käytetään myös eri tavoin, muun muassa vaikeiden leikkausten taidon ylläpitämisessä, potilaan kohtaamisessa esitietojen (anamneesin) saamisen harjoittelemiseksi tai peruselvytysharjoittelussa. Esimerkiksi Yhdysvalloissa lääketieteen opetuksen arviointiin liitetään voimakkaasti simulaatioharjoituksien tulosten arviointia. Standardipotilaat ovat Johnsenin ja muiden [2006] mukaan tällä hetkellä paras tapa opettaa, harjoittaa ja testata lääkäriopiskelijoiden taitoa anamneesin tekemisessä. Standardipotilaiden käyttäminen voi kuitenkin rajoittaa tilanteiden vaihtelevuutta, toistettavuutta, laadunvarmistusta ja vaikuttavuutta [Johnsen *et al.*, 2006]. Kun yhtenä opetussimulaation tarkoituksena

voidaan pitää odottamattomiin ja yllättäviin tilanteisiin valmistaumista, eivät vakioidut potilaita esittävien ihmisten toimintatavat välttämättä takaa parasta mahdollista simulaatioharjoituksen hyödyntämistä. Standardipotilaiden kaltaiset simulaatiot vaikuttaisivat tukevan paremmin mittamiseen pyrkivää simulointia siinä, missä toisten opetussimulaatioiden tavoite voi olla nimenomaisesti harjaantuminen erilaisten vaikeiden tilanteiden ratkaisemiseksi.

Aldrich [2003] huomauttaa, että oppijoiden seuraava sukupolvi on kasvanut tietokonepelien seurassa. He olettavat harjoittelevansa monella tasolla samanaikaisesti ja saavansa välitöntä palautetta visuaalisesta, stimuloivasta, erittäin immersiivisestä ja käyttäjäkeskeisestä ympäristöstä [Aldrich, 2003]. Uudet sukupolvet ovat kokeneempia teknologian kanssa ja vaikuttavat aiempia valmiimmilta toimimaan tietokoneohjelmoitujen simulaatioiden parissa.

Aldrich [2003] mainitsee yhdeksi simulaatioiden käyttötavaksi päätöksenteon harjoittelun ennen kuin vaikea tai vaarallinen tilanne kohdataan oikeassa elämässä, esimerkiksi pelastustyöntekijöillä. Lääketieteelliset opetussimulaatiot voivat harjaannuttaa myös useiden toimijoiden tilanteiden hallinnassa, esimerkiksi leikkaussalitoiminnassa, jossa paikalla on useita hoitajia ja lääkäreitä. Tavoitteena voi toisaalta olla vakioida toimintatapoja ja toisaalta opettaa reagointia yllättäviin tilanteisiin ja kokonaisuuden hallintaa. Simulaattoriharjoittelu voi sisältää tilanteita, joiden kokeminen todellisessa tilanteessa olisi harvinaista tai mahdotonta [Mantovani *et al.*, 2003]. Harvinaisten sairauksien hoito, korkealaatuisen hoitovälineen rikkoutuminen tai monivammapotilaat ovat esimerkkejä tilanteista, joita ei välttämättä kohdata usein. Toiminnan pitäisi olla kuitenkin hyvää myös tällaisissa tilanteissa, joten niiden harjoittelua opetussimulaatioilla voidaan pitää perusteltuna. Tällaiset haastavat ja polveilevat tilanteet eivät kuitenkaan vaikuta olevan sopivia suorituksen mittaamisen näkökulmasta.

2.3. Lääketieteellisten opetussimulaatioiden kriteereistä

Lääketieteellisille opetussimulaatioille voidaan määritellä monenlaisia kriteerejä aina hankinta- ja käyttökustannuksista erilaisten vaikuttavuusarviointien kautta toiminnallisiin kriteereihin, myös käytettävyyskriteereihin. Lääketieteellisestä näkökulmasta asiaa lähestyvässä kirjallisuudessa on pääasiallisesti keskitytty muihin kuin toiminnallisiin kriteereihin. Vaikuttaisi siltä, että toiminnallisuuteen liittyvät kriteerit on johdettu melko suoraan yleisistä käytettävyyden kriteereistä, eikä niitä ole merkittävässä määrin erikseen määritelty lääketieteellisille opetussimulaatioille.

Nikoukaran ja muut [1998] mainitsevat tärkeiksi yleisiksi kriteereiksi mm. toimittajan, dokumentaation ja käyttötuen. Käytettävyysnäkökulmasta simulaation tärkeiksi ominaisuuksiksi mainitaan syötteiden validointi ja virheenkorjaus, peräkkäinen ajo vaihtamalla vain joitain muuttujia, lämmittelyjakso sekä simulaation nopeuden säätömahdollisuus [Nikoukaran *et al.*, 1998]. Nämä käytettävyyskriteerit voidaan katsoa osittain riippuviksi siitä, millai-

sesta opetussimulaatiosta arvioinnissa on kysymys, eikä niitä tule soveltaa sellaisenaan kaikkiin lääketieteellisten opetussimulaatioiden tyyppeihin. Nikoukaranin ja muiden [1998] mukaan simulaation raportointi on tärkeä ominaisuus. Opetussimulaatiosta ei välttämättä saa aina välitöntä palautetta suorituksen onnistumisesta tai onnistumisen asteesta, jolloin jälkikäteen saatu raportti tukee oppijan kehittymistä ja mahdollistaa toiminnan muokkaamisen seuraavaan harjoitukseen. Edelleen simulaation yhteensopivuus esimerkiksi tilasto-ohjelmien toimintalogiikkaan on tärkeää [Nikoukaran *et al.*, 1998]. Käsittelen opetussimulaatioita käytettävyysnäkökulmasta kattavammin luvussa 4.

2.4. Lääketieteellisten opetussimulaatioiden hyötyjä ja haasteita

Kirjallisuuden perusteella opetussimulaatioiden käyttämisessä on monenlaisia osoitettuja hyötyjä. Yksi suurimmista eduista simulaatioille mainitaan olevan yrityksen ja erehdyksen mahdollisuus ilman vahinkoa potilaille tai eläimille [Rosen, 2008]. Johnsenin ja muiden [2006] mukaan sosiaalisen tilanteen kokeminen virtuaalisesti voi vähentää jännitystä todellisessa tilanteessa. Sandersin [2006] mukaan fyysinen läsnäolo mahdollistaa luonnollisemman oppimisen. Lisäksi näyttäisi, että vuorovaikutus todellisten kohteiden kanssa kehittää ongelmanratkaisukykyä [Lok *et al.*, 2003].

Sandersin [2006] mukaan oppimisen rinnastaminen tiedonhankintaan ja oppimisprosessin pelkistäminen mahdollisimman laajan teksti-, kuva-, video- tai audiomateriaalin hankinnaksi väheksyy oppimisessa tarvittavaa ongelmanratkaisukyvyn kehitystä. Sen sijaan Sanders [2008] kehottaa luomaan ympäristöjä, joissa opiskelijoiden täytyy työskennellä, reflektoida ja olla vuorovaikutuksessa toisten kanssa. Aldrichin [2003] mukaan monen käyttäjän simulaatiot parantavat luovuutta, aiheuttavat oppimista monella tasolla ja satunnaistavat käyttäjäkokemusta.

Rosen [2008] mainitsee simulaatioiden hyödyksi mm. automaattisen tilanteen purkamisen (debriefing). Mantovani ja muut [2003] nostavat simulaattoriopetuksen hyödyiksi motivaation parantumisen, yhteistyön ja sopeutumiskyvyn edistämisen sekä arvioinnin helpottumisen. Arvioinnin helpottumisesta opetussimulaatioiden käytön avulla voidaan kuitenkin sanoa olevan myös vastakkaisia näkemyksiä. Suoritusta opetussimulaatiossa voidaan kyllä arvioida, mutta sen perusteella ei välttämättä ole mahdollista suoraan arvioida oppijan toimintaa todellisessa tilanteessa, jota parhaatkaan ohjelmat eivät pysty täysin simuloimaan.

Simulaatioiden käyttämiseen liittyy haasteita myös opetuksellisesta näkökulmasta. Aldrich [2003] mainitsee useita haasteita simulaatioiden kehittämisessä. Simulaatioissa opitun arvioiminen kokeella on vaikeampaa kuin esimerkiksi kirjatiedon [Aldrich, 2003]. Simulaatioiden kautta arviointi on haasteellista ja oikeiden asioiden mittaaminen voi olla vaikeaa. Vapaan ja ohjatun käyttämisen suhteen määrittely on vaikeaa ja tarve vaihtelee käyttäjien ja ta-

voitteen mukaan. Simulaatioissa vaikuttaisi olevan lähes poikkeuksetta jokinlaisia suunnittelemattomia oikomismahdollisuuksia, joiden käyttäminen johtaa parempaan mitattuun tulokseen, mutta joita ei olisi olemassa tai ei voisi käyttää vastaavassa todellisessa tilanteessa. Simulaatioissa vuorovaikutuksen voidaan katsoa pelkistyvän, kun potilasta simuloiva kohde ei reagoi täysin kuten ihminen.

3. Käytettävyys

Mantovanin ja muiden [2003] mukaan simulaation käytettävyys on yksi sen tärkeimmistä vaatimuksista. Nikoukaranin ja muiden [1998] mukaan kriteereiden määrittäminen simulaatio-ohjelmistolle on tärkeää oikeiden valintojen tekemiseksi ohjelmistoa hankittaessa. Tässä luvussa käsitellään käytettävyyttä, annetaan käsitteelle lyhyt, tiivis ja tutkielman viitekehykseen soveltuva määritelmä sekä kuvataan erityisesti käytettävyyskirjallisuudesta nousevia näkökulmia käytettävyyden kriteereistä.

On mahdotonta suunnitella optimaalista käyttöliittymää vain yrittämällä parhaansa, sillä käyttäjillä on rajaton kyky sekä tehdä odottamattomia virhetulkintoja käyttöliittymästä että suorittaa tehtävänsä eri tavalla kuin suunnittelija oletti. Käytettävyyden parantamisella voi olla myös haittavaikutuksia. Käytettävyys voi sisältää myös ristiriitoja, jos esimerkiksi opittavuuden parantaminen heikentäisi samanaikaisesti kokeneen käyttäjän käyttötehokkuutta tai -mukavuutta. Aina ei ole mahdollista saavuttaa optimaalisia tuloksia kaikille käytettävyyden osa-alueille samanaikaisesti. [Nielsen, 1993]

Käytettävyyttä voi kuitenkin arvioida erilaisin menetelmin. Ei kuitenkaan ole päästy yksimielisyyteen erilaisten menetelmien tehokkuudesta [Hartson *et al.*, 2001]. Nielsenin [1993] mukaan käytettävyyden arviointi paljastaa lähes aina joitain käytettävyysongelmia. Käytettävyyden arvioinnin kautta paljastuneiden käytettävyysongelmien määrän ja laadun arvioinnin kautta voidaan toteuttaa kokonaisarviointi käytettävyydestä. Normaalisti kaikkien ongelmien ratkaiseminen on mahdotonta, joten on pystyttävä priorisoimaan merkittävimmät ongelmat ratkaistaviksi [Nielsen, 1993].

3.1. Käytettävyyden määritelmä

Holzingerin [2005] esittelee viisi yleisesti hyväksyttyä attribuuttia, joiden kautta käytettävyyttä voidaan arvioida: opittavuus, tehokkuus, muistettavuus, virheenkäsittely ja tyytyväisyys. Käytettävyyttä on eri tahoilla edellisten lisäksi määritelty monilla eri tavoilla ja erilaisina osa-alueina, kuten yksinkertaisuutena tai monikäyttöisyytenä. Näiden yksittäisten elementtien arvioimisen yhteenvetoa voidaan kutsua käytettävyydeksi. Käytettävyyden arviointiin ei ole yksimielisesti hyväksyttyä tapaa tai mitta-asteikkoa. Käytettävyyttä voisi ajatella myös ordinaaliasteikolla mitattavaksi suureeksi, jolloin asiat voitaisiin laittaa käytettävyyden

perusteella järjestykseen, mutta asioiden etäisyyttä suhteessa toisiinsa ei voida määritellä käytettävyyden näkökulmasta. Jonkin ohjelman käytettävyys voidaan kokea toista parempana, mutta sitä ei voida yksioikoisesti määritellä esimerkiksi kaksi kertaa toista paremmaksi käytettävyyden osalta.

Opittavuus tarkoittaa käytettävyyden näkökulmasta sekä matalaa kynnystä käytön aloittamiseksi, myös käyttötehokkuuden kasvamista käyttömäärän kasvaessa. Lisäksi opittavuus voi sisältää arvion siitä, onko ohjelma suunnattu aloitteleville vai kokeneille käyttäjille ja mukautuuko käyttö käyttäjän taitavuuden kasvaessa. Tehokkuudella tarkoitetaan soveltuvuutta halutun tehtävän ratkaisemiseen. Arvioitava asia voi olla helppokäyttöinen, mutta soveltua huonosti annetun tehtävän ratkaisemiseen, jolloin se ei ole tehokas eikä sitä voi pitää käytettävyydeltään hyvänä annetun asian ratkaisemiseksi. Muistettavuus liittyy opittavuuteen, erityisesti aiemmin mainittuun käyttämistehokkuuden kasvuun käyttömäärän kasvaessa. Käytettävyyden näkökulmasta käyttömäärän pitäisi parantaa tehokkuutta, suhteen riippuessa suuntautumisesta aloitteviin tai kokeneisiin käyttäjiin. Virheettömyys on ideaalitavoite, johon ei juuri koskaan päästä pois lukien korkeintaan kaikkein yksinkertaisimmat käyttöliittymät, jos niissäkään. Virheenkäsittelyllä tarkoitetaan käyttäjän mahdollisuutta havaita tapahtunut virhe ja mukautua virheeseen tai korjata se sekä virheen mahdollisimman pientä negatiivista vaikutusta tehokkuuteen. Tyytyväisyys on yksinkertaisimmillaan käyttäjän mielikuvaa käytettävyydestä ja toiminnon tai asian tekemisen miellyttävyydestä. Käyttömukavuus on oleellinen ja tärkeä osa käytettävyyttä. Hyvä opittavuus, tehokkuus, muistettavuus ja virheenkäsittely yleisesti lisäävät tyytyväisyyttä, mutta tyytyväisyyteen vaikuttavat myös muut tekijät, kuten visuaalisuustekijät, esimerkiksi värien sopiva käyttö.

Nielsen [1993] huomauttaa, että hyödyllisyydellä tarkoitetaan kyseisen asian sopimista annetun tehtävän suorittamiseen. Tuolloin tavoitteen ja tehtävän on oltava selkeitä ja määriteltävissä. Käytettävyyttä arvioidaan yleisesti juuri suhteessa annetun tehtävän suorittamiseen. Esimerkiksi ruuviväänтimen käytettävyyttä voidaan toki arvioida yleisestikin, mutta hyödyllisintä sen käytettävyyden arviointi on suhteessa ruuvien kiinnittämiseen tai irrottamiseen. Tuolloin esimerkiksi kahvan pito väännettäessä on selkeämmin määriteltävissä käytettävyyden osaksi ja parempi pito tarkoittaisi osaltaan parempaa käytettävyyttä.

3.2. Käytettävyyden kriteereistä

Nielsen [1993] painottaa yksinkertaisuuden ja yksityiskohtien merkitystä käytettävyydessä. Teknologian käytettävyys tulisi suunnitella käyttäjien mukaan. Käyttäjät eivät kuitenkaan ole aina oikeassa: esimerkiksi kohdattaessa uutta teknologiaa, jollaisesta käyttäjillä ei ole aikaisempaa kokemusta, voivat käyttäjien oletukset teknologian käyttötavoista olla erilaiset kuin todellinen käyttö. [Nielsen, 1993]

Käytettävyyttä pohdittaessa näytetään käytännössä palattavan usein kysymykseen käyttötarkoituksesta ja käyttäjäkunnan aiemmasta kokemuksesta. Vastakkain voidaan asettaa käyttöliittymän laaja muuntelumahdollisuus tai erilaisten oikoteiden käyttömahdollisuudet ja toisaalta pitkälle viety standardointi ja käyttöliittymän yksinkertaisuus. Edellisen voidaan katsoa sopivan usein kokeneille käyttäjille, jälkimmäisen yleensä aloitteleville käyttäjille.

Nielsen [1993] toteaa, että on helpompaa, jos käyttäjät voivat toimia ilman tukeutumista ohjeisiin, manuaaleihin tai käyttäjätukeen. Yksinkertaisuus korostuu tilanteessa, jossa käyttäjän tulisi pystyä käyttämään tuotetta tai asiaa heti, ilman erillistä oppimisaikaa. Yleensä kattavat dokumentaatiot ja niihin perehtymisen vaatimukset tarkoittavat monimutkaisuutta, jota voidaan joskus pitää käytettävyyttä heikentävänä tekijänä. Toisaalta jotkut kokeneille käyttäjille suunnatut korkean tason kapeaa sektoria käsittelevät toiminnallisuudet voivat vaatia käyttäjiltä perehtymistä dokumentaatioon ennen käytön aloittamista.

Nielsen [1994] on määritellyt julkaisussaan heuristisen lähestymistavan käyttävyyden kriteereiden arviointiin. Nielsenin heuristiikaksi nimetyllä menetelmällä on mahdollista arvioida käytettävyyttä ja etsiä käytettävyyttä parantavia korjauksia tai päivityksiä ja arvioida niiden hyödyllisyyttä sekä asettaa korjauksia prioriteettijärjestykseen.

Nielsenin [1994] mukaan kaikessa käytettävyyssuunnittelussa tulisi huomioida seuraavat asiat:

- Yksinkertainen ja luonnollinen dialogi
  - Graafinen suunnittelu ja värit
  - Vähemmän on enemmän
- Puhu käyttäjän kieltä
  - Kokonaisuus ja metaforat
- Minimoi käyttäjän muistamistarve
- Yhdenmukaisuus / johdonmukaisuus
- Palaute
  - Palauteaika
  - Järjestelmän kaatumisen ilmoittaminen
- Selkeästi osoitetut uloskäynnit
- Oikotiet
- Hyvät virheilmoitukset
- Virheiden välttäminen
  -Vältetään moodeja aina kun mahdollista
- Tuki ja ohjeet.

Yksinkertaisella ja luonnollisella dialogilla tarkoitetaan käytettävän asian ja käyttäjän välistä vuorovaikutusta. Siihen voidaan vaikuttaa esimerkiksi graafisella suunnittelulla ja vä-

reillä sekä pelkistämällä (yksinkertaistamalla) käyttöliittymää mahdollisuuksien mukaan. Varsinkin aloitteleville käyttäjille suunnatut tai erityisen matalan oppimiskynnyksen asiat tulisi esittää mahdollisimman yksinkertaisella terminologialla sekä karttamaisina kokonaisuuksina, jolloin nykyisen toiminnon hahmottaminen suhteessa koko käyttöliittymään on mahdollista. Lisäksi kannattaa käyttää tunnettuja metaforia, kuten esimerkiksi erilaisia valintalistoja vaikkapa asetuksia määriteltäessä. Käyttöliittymäsuunnittelulla pyritään hakemaan yhteyksiä käyttäjien aiempiin kokemuksiin metaforien avulla [Horila *et al.*, 2002]. Käyttöliittymä tulisi suunnitella siten, että käyttäjän muistamisen tarve minimoidaan. Käyttöliittymän tulisi olla yhdenmukainen kaikkialla niin, että samat toiminnallisuudet löytyisivät samoista paikoista läpi koko käyttöliittymän. Käyttöliittymän tulisi antaa palaute halutusta ja virheellisestä toiminnosta mahdollisimman nopeasti, jotta käyttäjän ei tarvitse arvailla suorituksen onnistumista. Virheilmoituksien tulisi olla selkeitä ja neuvoa käyttäjää jatkamis- tai korjaamismahdollisuuksista. Erilaisia moodeja tulisi välttää. Moodilla tarkoitetaan tietynlaisia pakotettuja toimintatiloja, kuten esimerkiksi caps lock -näppäin näppäimistössä, joka asettaa moodiksi isot kirjaimet pienten sijaan.

Ylle on poimittu joitain yleisiä käytettävyyden kriteereitä, joista suurin osa voidaan huomioida ja arvioida riippumatta siitä, minkä käytettävyyttä arvioidaan, fyysistä konetta tai tietokoneohjelmaa yhtä lailla. Näitä käytettävyyden kriteereitä voidaan ja kannattaakin usein tarkentaa riippuen siitä, minkä käytettävyyttä arvioidaan. Käytettävyyden kriteereiden eri osa-alueille voidaan myös antaa erilaisia painoarvoja osana kokonaisarviointia riippuen arvioinnin kohteesta.

4. Lääketieteellisten opetussimulaatioiden käytettävyydestä

Lääketieteen kirjallisuudessa on käsitelty hyvin rajatusti lääketieteellisten opetussimulaatioiden käytettävyyttä. Käytettävyyteen liittyviä kriteereitä huomioidaan lähinnä osana yleisten kriteereiden määrittelyä ja tuolloinkin mainitaan vain jokin tietty ominaisuus. Luon tässä luvussa tiiviin katsauksen lääketieteen kirjallisuudessa esitettyyn lääketieteellisten opetussimulaatioiden käytettävyyteen ja täydennän sitä yleisen käytettävyyden näkökulmasta soveltuvin osin.

Johnsenin ja muiden [2006] tutkimuksessa selvisi, että kohteen puhekyky ja luonnollinen koko olivat erittäin tärkeitä kokonaiskokemuksen kannalta simulaatioissa, joissa käytetään virtuaalitodellisuutta. Todentuntuisuus voidaan usein arvioida tärkeäksi ominaisuudeksi lääketieteellisille opetussimulaatioille. Bearmanin [2001] mukaan simulaatioiden käyttämisen koetaan rajoittavan itseilmaisua. Ilmeisesti simulaation vuorovaikutuksen rajoitukset aiheuttavat yllä mainitun kokemuksen. Käyttäjät saattavat ajatella, ettei simulaatio kuitenkaan

ymmärrä esimeriksi kasvonilmeitä. Kun käyttäjät kokevat, ettei heidän kasvonilmeillään ole merkitystä, simulaatiotilanne ei enää vastaa todellista tilannetta. Ilmeisesti tulisi siis pyrkiä mahdollisimman kattaviin simulaatioihin, joita käytettäessä käyttäjät eivät kokisi simulaatiota rajoittavana tekijänä, vaan suoriutuisivat kuten aidosta tilanteesta. Tämä asettaa merkittäviä vaatimuksia simulaatiosuunnittelulle.

Nikoukaranin ja muiden [1998] mukaan simulaation tärkeitä ominaisuuksia ovat syötteiden validointi ja virheenkorjaus, peräkkäinen ajo vaihtamalla vain joitain muuttujia, lämmittelyjakso sekä simulaation nopeuden säätömahdollisuus. Holzingerin [2005] käytettävyyden määritelmästä poimitut opittavuus, tehokkuus, muistettavuus, virheenkäsittely ja tyytyväisyys tulisi huomioida myös lääketieteellisissä opetussimulaatioissa. Nämä kaikkeen käytettävyyteen liitetyt ominaisuudet kuvaavat hyvin myös lääketieteellisten opetussimulaatioiden käytettävyyteen liitettäviä ominaisuuksia. Edelleen Nielsenin [1994] heuristiikassa mainitut kohdat voitaneen huomioida lähes sellaisenaan opetussimulaation tyypistä riippuen. Yksinkertainen ja luonnollinen dialogi, käyttäjän kielen puhuminen, käyttäjän muistamistarpeen minimointi, yhdenmukaisuus ja johdonmukaisuus, palaute, selkeät uloskäynnit, oikotiet, hyvät virheilmoitukset, virheiden välttäminen sekä tuki ja ohjeet voidaan kaikki nähdä olennaisina elementteinä myös lääketieteellisten opetussimulaatioiden käytettävyyden kriteereitä arvioitaessa.

Lisäksi, simulaatiosta riippuen, voidaan huomioida erilaiset simulaatioharjoituksen ympäristötekijät, kuten käytettävä tila, ylimääräisten henkilöiden läsnäolo sekä käyttöä edeltävä ja käytönaikainen opastus. Nämä elementit saattavat huonosti toteutettuna heikentää todentuntuisuutta ja siten vaikuttaa tyytyväisyyteen. Lisäksi on huomioitava erilaiset aikatekijät, kuten simulaatioharjoituksen pituus suhteessa todellisen tilanteen aikajänteeseen. Esimerkiksi Rosenin [2008] mukaan yli 90 minuutin simulaatioharjoitus on liian raskas.

5. Pohdinta ja yhteenveto

Edellä on käsitelty lääketieteellisiä opetussimulaatioita ja käytettävyyttä lääketieteelliseen ja käytettävyyskirjallisuuteen perustuen lyhyesti käsitteiden ja kriteereiden määrittelyn kautta. Seuraavaksi pohditaan havaintoja ja löydöksiä ja niiden mahdollisia implikaatioita. Lopuksi tehdään kokonaisuudesta yhteenveto, esitetään kritiikkiä ja nostetaan esiin joitain jatkotutkimusmahdollisuuksia.

Katsauksen perusteella vaikuttaisi, että lääketieteellisten opetussimulaatioiden erilaisia kriteereitä on kyllä pohdittu nyt läpi käydyssä lääketieteen kirjallisuudessa, mutta näistä kriteereistä vain pieni osa sisältää käytettävyyden näkökulman ja silloinkin usein vain jokin yksittäinen ominaisuus on huomioitu. Käytettävyyttä ei kyseisen kirjallisuuden perusteella ole

tutkittu kovinkaan systemaattisesti lääketieteellisten opetussimulaatioiden viitekehyksessä. Tehdyt käytettävyystutkimukset ja -selvitykset lienee tehty lähinnä valmistajien omille tuotteille ja tuotekehityksen näkökulmasta. Tutkielmaan sisällytetyn lääketieteen kirjallisuuden eduksi on mainittava, että sieltä nousevat yksittäisetkin lääketieteellisten opetussimulaatioiden käytettävyyttä käsittelevät seikat ovat käytännössä suoraan sovellettavissa lääketieteellisten opetussimulaatioiden käytettävyysarviointeihin.

Käytettävyyskirjallisuudesta on havaittavissa määrällisesti enemmän käytettävyyden kriteereitä, joita voitaisiin soveltaa lääketieteellisiin opetussimulaatioihin. Tutkielmassa käsitelty käytettävyysnäkökulman kirjallisuudesta ponnistava käytettävyyskriteeristö näyttäisi yleisluontoisuudestaan huolimatta olevan melko hyvin sovellettavissa myös lääketieteellisiin opetussimulaatioihin. Nielsenin käytettävyyden kriteerit ja heuristiikka antavat hyvät työkalut käytettävyyden arviointiin myös lääketieteellisissä opetussimulaatioissa. Lääketieteelliset opetussimulaatiot vaikuttaisivat tarvitsevan merkittävästi lisää käytettävyystutkimusta. Tutkielman perusteella vaikuttaa, että toistaiseksi ei ole kyetty määrittelemään tyydyttävästi lääketieteellisten opetussimulaatioiden käytettävyyttä. Määrittelyn tekeminen olisi merkittävä vaihe lääketieteellisten opetussimulaatioiden käytettävyystutkimukselle. Tutkimusta voitaisiin tehdä esimerkiksi perustuen havainnointiin, kyselyihin ja haastatteluihin ja todellisen käytön kirjaamiseen.

Lääketieteelliset opetussimulaatiot on kuitenkin hyvä asettaa viitekehykseensä. Mantovanin ja muiden [2003] mukaan simulaatiot ovat kuitenkin vain oppimisväline muiden joukossa eivätkä ne korvaa opettajaa. Kymmenen vuotta sitten Mantovani ja muut [2003] mainitsevat lisätutkimusten tarpeen simulaatioteknologian vaikutuksista oppimiseen ja tuo tarve on varmasti edelleen olemassa. Rosenin [2008] mukaan simulaatioiden käyttöönoton esteenä on edelleen mm. laitteiston, henkilöstön ja ohjelmistojen korkeat kustannukset.

Lääketieteellisiä opetussimulaatioita käytetään kansainvälisesti osin myös suoritusarvioinneissa. Arvioinnista on lähteiden mukaan kahdenlaista kantaa: osa sanoo simulaatiotilanteiden arvioimisen olevan vaikeaa, toisaalla taas mainitaan arvioinnin helpottuvan. Lääketieteellisten opetussimulaatioiden soveltuvuudesta suoritusarviointeihin olisi tarpeellista tehdä tutkimuksia.

Tämä tutkielma ei kykene käsittelemään kattavasti lääketieteellisten opetussimulaatioiden käytettävyyttä, vaan pääasiassa tarkastelemaan olemassa olevan kirjallisuuden lähestymistapoja aihepiiriin. Sekä lääketieteessä että käytettävyystutkimuksessa on erittäin laajasti kirjallisuutta, joka käsittelee ainakin osittain tämän tutkielman aihepiiriä. Tuosta laajasta kirjallisuudesta on kuitenkin tutkielman laajuuteen suhteutettuna valikoitunut vain osa käytettäväksi. Kirjallisuuteen perehdyttiin tutkielmassa viitattua kirjallisuutta laajemmin, mutta siitä huolimatta rajausta ja valikointia jouduttiin suorittamaan jo tutkielman tekemisen alkuvai-

heessa. Tutkielman reliabiliteetti arvioidaan melko korkeaksi, mutta validiteetin osalta jää joitain epäselvyyksiä. Valitusta kirjallisuudesta poimitut näkökulmat ovat melko korkealla todennäköisyydellä edustavia, mutta tutkielman rajauksesta ja suppeudesta johtuen ei voida täysin pitävästi väittää, että valikoitunut kirjallisuus on kaikilta osin ollut soveltuva aiheen tutkimiseen ja jotain oleellisesti aihetta käsittelevää kirjallisuutta on voinut jäädä tutkielman ulkopuolelle.

Viiteluettelo

[Aldrich, 2003] Clark Aldrich, *A Field Guide to Educational Simulations*. Learning Circuits. American Society for Training and Development. January, 2003.

[Bearman and Cesnik, 2001] Margaret Bearman and Branko Cesnik, Comparing student attitudes to different models of the same virtual patient, *Stud. Health Technol. Inform.* **84**(2),1004-8.

[Hartson et al., 2001] H. Rex Hartson, Terence S. Andre and Robert C. Williges, Criteria for evaluating usability evaluation methods, *International Journal of Human-Computer Interaction* **13**(4), 373–410.

[Holzinger, 2005] Andreas Holzinger, Usability engineering for software developers, *Communications of the ACM* **48**(1), (January 2005), 71-74.

[Horila et al., 2002] Mikko Horila, Petri Nokelainen, Antti Syvänen and Jan Överlund, *Pedagogisen Käytettävyyden Kriteerit ja Kokemuksia OPIT-oppimisympäristön Käytöstä Hämeenlinnan Normaalikoulussa Syksyllä 2001. DL-projektin osaraportti*. Hämeen ammattikorkeakoulu, ISBN 951-784-146-9 Hämeenlinna 2002.

[Johnsen et al., 2006] K. Johnsen, R. Dickerson, A. Raij, C. Harrison, B. Lok, A. Stevens, et al., Evolving an immersive medical communication skills trainer. *Presence: Teleoperators & Virtual Environments* **15**(1), 33-46.

[Kaufmann, 2001] C. R. Kaufmann, Computers in surgical education and the operating room, *Annales Chirurgiae et Gynaecologiae* **90**, 141-146.

[Lok et al., 2003] B. Lok, S. Naik, M. Whitton and F. Brooks, Effects of interaction modality and avatar fidelity on task performance and sense of presence in virtual environments, *Precence: Teleoperators and Virtual Environments*, **12**(6), 615-628.

[Mantovani et al., 2003] F. Mantovani, G. Castelnuovo, A. Gaggioli and G. Riva, Virtual reality training for health-care professionals. *CyberPsychology & Behavior* **6**(4), 389.

[Nagle et al., 2009] Beth M. Nagle, Jeanne M. McHale, Gail A. Alexander, Brian M. French, Incorporating scenario-based simulation into a hospital nursing education program, *The Journal of Continuing Education in Nursing* **1**(40), 18-25.

[Nielsen, 1993] Jakob Nielsen, *Usability Engineering*, Morgan Kaufmann Publishers, London 1993.

[Nielsen 1994] Jakob Nielsen, *Usability Inspection Methods*, Morgan John Wiley & Sons.

[Nikoukaran et al., 1998] J. Nikoukaran, V. Hlupic and R. J. Paul, Criteria for simulation software evaluation, In: *Proceedings of the 30th Simulation Conference* (Dec 1998), 399-406.

[Rosen, 2008] Kathleen R. Rosen, The history of medical simulation, *Journal of Critical Care*, **23**(2), (June 2008), 157–166.

[Sanders, 2006] R. Sanders, The impenderable bloom: reconsidering the role of technology in education. *Innovate* **2**(6).

[Tikka, 2001] Maarit Tikka, Virtuaaliympäristöjen tekniikat ja niiden käyttö lääketieteen sovelluksissa, Pro gradu -tutkielma, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto, 2001.

Lajittelualgoritmit ja rinnakkaislaskenta Javassa

Antti Reunamo

Tiivistelmä.
Tässä tutkielmassa käsitellään lajittelualgoritmeja ja esitellään Java-ohjelmointikielellä toteutettu rinnakkaistettu ShellSort-algoritmi. Sen lajittelunopeutta verrataan sekä tavalliseen ShellSortiin että Javan omiin sisäänrakennettuihin lajittelijoihin.

Avainsanat ja -sanonnat: Java, lajittelu, algoritmit, rinnakkaisuus.
CR-luokat: D.1.3

1. Johdanto

Lajittelu on yksi tietojenkäsittelyn perusongelmista. Lajittelua tarvitaan miltei kaikkialla, missä ollaan tekemisissä tiedon tallentamisen ja esittämisen kanssa. Miten voidaan lajitella jonkin tietorakenteen alkiot esimerkiksi suuruusjärjestykseen tehokkaasti? Tehokkuuden lisäksi lajittelulle saattaa olla tilanteesta riippuen myös muita vaatimuksia esimerkiksi muistinkäytön osalta.

Tarkkoja ohjeita siitä, miten joku tietty laskenta tai tietojenkäsittelyllinen toimenpide suoritetaan alusta loppuun, kutsutaan *algoritmiksi*. Algoritmi siis yhdistää jonkun tietyn syötteen johonkin tiettyyn tulokseen. Tietojenkäsittelyssä algoritmit koostuvat sarjasta peräkkäin suoritettuja käskyjä, joilla annettua tietoa käsitellään [Heineman *et al.*, 2008]. Lajitteluun on olemassa useita perusalgoritmeja, joiden toimintaperiaatteet voivat erota toisistaan huomattavastikin. Näistä on kehitetty edelleen lukuisia erilaisia variaatioita sekä algoritmien yhdistelmiä erilaisiin tilanteisiin sopiviksi. Tässä tutkielmassa keskitytään erityisesti Donald Shellin [1959] laatimaan ShellSort-lajittelualgoritmiin, josta lisää myöhemmin.

Tietokoneen suorituksessa olevaa ohjelmaa sekä sen hallitsemia resursseja, kuten muistia, kutsutaan *prosessiksi*. Prosessi voi sisältää yhden tai useampia *säikeitä* (thread), jotka suorittavat varsinaisen työn. Tietokoneen s*uoritin* (processor) voi suorittaa vain yhtä säiettä kerrallaan, jolloin useamman säikeen samanaikainen suoritus edellyttää useampaa suoritinta tai suoritinydintä. Laskentaa usealla suorittimella samanaikaisesti kutsutaan *rinnakkaiseksi laskennaksi* (parallel processing). Kyseessä voi siis olla kokonaan eri sovellusten samanaikainen suoritus tai yhden sovelluksen sisällä

tapahtuva useiden säikeiden samanaikainen suoritus. Algoritmien rinnakkaislaskenta kuuluu jälkimmäiseen kategoriaan. [Stallings, 2011]

Tietokonelaitteistojen kehitys näyttää yhä enemmän suuntautuvan perinteisen kellotaajuuden kasvattamisen sijaan suorittimien määrän lisäämiseen. Ohjelmistojen sekä niiden sisältämien algoritmien on myös muututtava tämän kehityksen myötä, ja niiden on hyödynnettävä enemmän rinnakkaislaskentaa [Mattson and Wrinn, 2008]. Tutkielmassa selvitetään, miten ShellSort-lajittelualgoritmi saadaan rinnakkaistettua ja paljonko se käytännössä hyötyy siitä.

Rinnakkaislaskennan haasteena on se, miten suoritus käytännössä jaetaan eri suorittimien kesken. Samanaikaisuus synnyttää helposti ongelmia ja ristiriitatilanteita, joissa sekä käsiteltävän tiedon eheys että suorituksen normaali jatkuminen ovat uhattuina. Optimaalisessa tapauksessa suoritettava algoritmi ja sen käyttämä tieto pitäisi pystyä jakamaan osiin, jotka eivät olisi tekemisissä toistensa kanssa suorituksen aikana. Tähän kuitenkin harvoin päästään, ja käytännössä eri suoritussäikeet joutuvat tahdistamaan tekemisiään keskenään. Tämä vähentää rinnakkaislaskennasta saatavaa hyötyä.

Tässä tutkielmassa käsitellään lajittelualgoritmeja ja rinnakkaislaskentaa sekä yleisellä tasolla että Java-ohjelmointikielessä. Luvussa 2 perehdytään lajittelun ongelmaan ja esitellään lajittelualgoritmityyppejä sekä niiden toiminta- ja valintaperiaatteita. Luvun lopussa vilkaistaan Javan sisäänrakennettuja lajittelualgoritmeja.

Luvussa 3 esitellään rinnakkaisuuden käsitteitä ja periaatteita, ja luvussa 4 selvitetään, miten ShellSort-algoritmi on saatu rinnakkaistettua. Luvussa 5 verrataan sen lajittelunopeutta niin Javan sisäänrakennettuihin lajittelijoihin, tavalliseen ShellSort-algoritmiin kuin yksinkertaiseen lisäyslajitteluunkin. Pääasiallinen painopiste on lähinnä lajittelunopeuksien vertailussa sekä kyseisten algoritmien yleisessä esittelyssä.

Tutkimuksessa selvisi, että ShellSort-algoritmi rinnakkaistui kohtuullisen hyvin ja sen suoritusnopeus parantui verrattuna tavalliseen ShellSortiin. Rinnakkainenkaan ShellSort ei kuitenkaan aivan voittanut Javan omaa peruslajittelijaa, ja sen oma rinnakkainen lajittelija oli ylivoimaisesti nopein.

2. Lajittelualgoritmit

Lajittelualgoritmeiksi kutsutaan algoritmeja, joiden avulla voidaan järjestää jonkin tietorakenteen alkiot haluttuun, esimerkiksi kasvavaan tai laskevaan, järjestykseen [Heineman *et al.*, 2008]. Lajittelualgoritmit perustuvat usein alkioiden vertailuun keskenään, mutta lajittelun voi suorittaa myös

muullakin tavoin. Esimerkiksi laskentalajittelu pohjautuu alkioiden jakauman laskemiseen [Heineman *et al.*, 2008]. Lisäksi on olemassa erikoistuneempia algoritmeja, kuten esimerkiksi topologinen lajittelu suunnattujen graafien solmujen järjestämiseen [Cormen *et al.*, 2001]. Tässä työssä keskitytään kuitenkin ainoastaan yksinkertaisten taulukkojen lajitteluun tarkoitettuihin lajittelualgoritmeihin.

2.1. Aikakompleksisuus

Lajittelualgoritmin suoritusaika on verrannollinen lajiteltavan syötteen kokoon. Tätä a*symptoottista suoritusaikaa* eli suoritusajan kasvunopeutta suhteessa syötteen kokoon voidaan mitata eri tavoilla, esimerkiksi *Ordo-notaatiolla*. Tämä lajittelualgoritmin *aikakompleksisuus* voidaan ilmoittaa kolmella eri tavalla, parhaimmassa, keskimääräisessä tai pahimmassa mahdollisessa tapauksessa. [Heineman *et al.*, 2008]

Lajittelualgoritmien aikakompleksisuus riippuu monesti syötteen järjestysasteesta. Monet lajittelualgoritmit toimivat parhaassa mahdollisessa tapauksessa $O(n)$-ajassa eli lineaarisessa ajassa. Tällöin algoritmin suoritusaika on suoraan verrannollinen syötteen kokoon. Pahimman tapauksen suoritusaika on joillakin lajittelualgoritmeilla $O(n^2)$. Tämä syntyy tyypillisesti käyttämällä kahta sisäkkäistä silmukkaa, joissa jokaista syötteen alkiota kohti koko syöte käydään kokonaan tai lähes kokonaan uudelleen läpi. Näin toimivat esimerkiksi lisäyslajittelu sekä kuplalajittelu [Heineman *et al.*, 2008].

Ratkaisevinta on tietää algoritmin suorituksen kesto keskimääräisessä sekä pahimmassa mahdollisessa tapauksessa. Keskimääräisessä siksi, että se on aika, joka suoritukseen yleensä tarvitaan, ja pahimmassa, koska siihen pitää osata varautua. Nopeimmat lajittelualgoritmit toimivat sekä keskimääräisessä että pahimmassa tapauksessa ajassa $O(n \log n)$ [Heineman *et al.*, 2008]. Neliöllisessä ajassa toimivat lajittelijat alkavat etenkin suuremmilla syötteillä olemaan niin hitaita, että niitä ei ole enää mielekästä käyttää.

2.2. Lajittelualgoritmityyppejä

Lajittelualgoritmit voidaan jakaa *vakaisiin* (stable) ja *epävakaisiin* (unstable) lajittelualgoritmeihin [Heineman *et al.*, 2008]. Vakaat algoritmit säilyttävät syötteenä saadun tietorakenteen samansuuruisten alkioiden suhteellisen järjestyksen, mutta epävakaat eivät välttämättä tee niin. Joissakin tapauksissa, kuten esimerkiksi yksinkertaisessa numeroiden tai sanojen järjestämisessä, tällä ei ole merkitystä. Tämä pätee myös silloin, kun kaikki järjestettävät alkiot ovat erisuuruisia. Toisaalta esimerkiksi avain-arvo -pareja

avaimen perusteella järjestettäessä tällä saattaa olla merkitystä. Epävakaa lajittelualgoritmi voidaan toteuttaa myös vakaana versiona, mutta silloin se voi olla hitaampi ja viedä enemmän muistia.

Lajittelualgoritmit voidaan luokitella edelleen myös sen perusteella, miten paljon ne tarvitsevat ylimääräistä muistitilaa lajittelun suorittamiseen. Ylimääräisen muistin tarvetta kutsutaan algoritmin *tilakompleksisuudeksi*. Jos algoritmi lajittelee syötteenä saamansa tietorakenteen suoraan, vaatien korkeintaan kiinteän, syötteen koosta riippumattoman määrän apumuistia, niin sitä kutsutaan *paikallaan toimivaksi*. Tällaisen lajittelualgoritmin tilakompleksisuus on $O(1)$. Lajittelualgoritmit voidaan myös jakaa *online*- ja *offline*-algoritmeihin. Online-algoritmit lajittelevat alkioita sitä mukaa kun niitä syötetään, kun taas offline-algoritmien pitää ensin saada koko syöte ennen kuin lajittelu voidaan aloittaa.

Monessa tapauksessa lajittelualgoritmien aika- ja tilakompleksisuus riippuvat toisistaan niin, että nopeus on suoraan suhteessa tarvittavaan muistitilaan. Esimerkiksi tasapainotettu hakupuu on sekä nopea että vakaa, mutta vaatii koko syötteen verran apumuistia [Cormen *et al.*, 2001].

Niin sanotut *hajota ja hallitse* -algoritmit toimivat niin, että ne jakavat käsiteltävän ongelman useammaksi itsenäiseksi aliongelmaksi [Heineman *et al.*, 2008]. Nämä aliongelmat jaetaan edelleen rekursiivisesti yhä pienemmiksi osasiksi. Lopuksi, kun kaikki osat on käsitelty, ongelma on ratkaistu. Tämän tyyppisen algoritmin rinnakkaistaminen on yleensä hyvin yksinkertaista, kunhan aliongelmat ovat riippumattomia toisistaan. *QuickSort*-lajittelualgoritmi [Hoare, 1962] on hyvä esimerkki tällaisesta lähestymistavasta.

2.3. Lajittelualgoritmin valinta

Käytettävän lajittelualgoritmin valintaperusteisiin vaikuttaa moni asia, kuten esimerkiksi se, onko algoritmin oltava vakaa vai ei. Hyvin yksinkertaiseen ja pienimuotoiseen lajitteluun riittää yksinkertainen lisäyslajittelu, mutta lajiteltavan syötteen koon kasvu rajaa nopeasti tehottomimmat neliölliset algoritmit pois käytettävien listalta. Myös lajitteluun käytettävissä olevan muistin määrä vaikuttaa valintaan, ja muistitilan ollessa hyvin rajoitettu on käytettävä paikallaan toimivia algoritmeja. Toisaalta muistitilan niin salliessa on järkevää hyödyntää nopeita, muistia runsaasti käyttäviä algoritmeja.

Lajittelualgoritmin valinta on siis hyvin tapauskohtainen ja riippuu lajiteltavasta syötteestä, käytettävästä laitteistosta sekä muista vaatimuksista. Käytännössä kuitenkin tyydytään yleensä käyttämään ohjelmointikielten

kirjastojen valmiita lajittelufunktioita, jotka riittävät hyvin tavalliseen käyttöön.

2.4. ShellSort

ShellSort on Donald Shellin [1959] kehittämä paikallaan toimiva vertailuun perustuva lajittelualgoritmi, joka pohjautuu yksinkertaiseen lisäyslajitteluun. ShellSort ei ole vakaa, eli se saattaa vaihtaa samanarvoisten alkioiden suhteellista järjestystä. Se lajittelee sitä nopeammin, mitä suurempi syötteen järjestyksen aste on, ja optimaalisella valmiiksi lajitellulla syötteellä sen aikakompleksisuus on $O(n \log n)$ [Heineman *et al.*, 2008].

Shellin algoritmi toimii niin, että se jakaa syötteen alitaulukoiksi, jotka lajitellaan tavallisella lisäyslajittelulla toisistaan riippumatta. Jakaminen tapahtuu niin, että kuhunkin alitaulukkoon otetaan mukaan syötteen joka n:s alkio. Tällä tavalla syötteestä saadaan muodostettua n kappaletta erillisiä alitaulukoita, joissa ei siis ole yhteisiä alkioita.

Kukin alitaulukko lajitellaan erikseen käyttämällä lisäyslajittelua, ja lajittelun jälkeen syötteen sanotaan olevan n-lajiteltu [Sedgewick, 1996]. Tämä tarkoittaa sitä, että mikä tahansa alkio syötteestä on lajiteltu oikeaan kohtaan omassa n-alitaulukossaan. Tämän jälkeen n:n arvoa vähennetään ja jako sekä lajittelu suoritetaan uudelleen.

Algoritmi siis vertailee aluksi kaukana toisistaan olevia alkioita, ja vertailtavien alkioiden välimatkaa vähennetään algoritmin edetessä. Tällä tavalla algoritmi saa nopeasti poistettua epäjärjestystä syötteestä ja alkiot saadaan siirrettyä suurin piirtein oikeille paikoilleen. Lopuksi n:n arvolla 1 koko syöte muodostaa yhden lajiteltavan alitaulukon. Tämä vastaa tavallista lisäyslajittelua koko syötteelle [Shell, 1959]. Koska alkiot ovat ennen tätä viimeistä läpikäyntiä jo lähes oikeilla paikoillaan, on lopussa suoritettava koko syötteen lisäyslajittelu hyvin nopea toimenpide alkutilaan verrattuna, sillä syötteen lajitteluaste vaikuttaa voimakkaasti lisäyslajittelun nopeuteen. Kuva 1 havainnollistaa kokonaislukutaulukon lajittelua ShellSort-algoritmilla.

Indeksi	0	1	2	3	4	5	6	7	8	9
Alkuperäinen	9	5	3	7	6	5	4	2	1	8
5-lajiteltu	5	4	2	1	6	9	5	3	7	8
3-lajiteltu	1	3	2	5	4	7	5	6	9	8
1-lajiteltu	1	2	3	4	5	5	6	7	8	9

Kuva 1. Taulukon ShellSort-lajittelu n-arvoilla 5, 3 ja 1.

Shellsortin keskimääräinen aikakompleksisuus on epäselvä ja riippuu siitä, miten alitaulukoihin jako suoritetaan eli miten n:t valitaan [Sedgewick, 1996]. Sedgewickin mukaan tämä tekeekin algoritmista mielenkiintoisen, sillä emme voi olla varmoja, etteikö löytyisi sellaista n-sarjaa, joka tekisi Shellsortista yhtä tehokkaan tai jopa tehokkaamman kuin parhaat tunnetut lajittelualgoritmit.

On esitetty useita ehdotuksia siitä, miten alitaulukoihin jako kannattaisi suorittaa, jotta algoritmi tekisi mahdollisimman vähän vertailuita. Shell [1959] itse käytti kaavaa $\left\lfloor \frac{N}{2^k} \right\rfloor$, kun N on lajiteltavan taulukon koko ja k:n arvoa kasvatetaan aina yhdellä algoritmin edetessä. Ensimmäinen n-arvo saadaan siis jakamalla taulukon koko kahdella ja pyöristämällä saatu arvo alaspäin lähimpään kokonaislukuun. Seuraava arvo saadaan jakamalla taulukko neljällä ja niin edelleen, kunnes lopulta päädytään lukuun yksi.

Knuth [1998] puolestaan ehdotti sarjaa $n_k = 3(n_{k-1}) + 1$, jossa $n_1 = 1$. Tästä saadaan n-arvot 1, 4, 13, 40, 121 ja niin edelleen. Lajittelu aloitetaan sillä arvolla, joka on kolmannes lajiteltavan taulukon koosta. Ciuran [2001] sarja 1, 4, 10, 23, 57, 132, 301, 701 on toistaiseksi osoittautunut käytännössä tehokkaimmaksi. Se ei pohjaudu mihinkään tiettyyn kaavaan, vaan on saatu kokeellisesti [Ciura, 2001]. Optimaalisten alitaulukoiden muodostamiseen vaikuttavat seikat liittyvät monimutkaisiin matemaattisiin ilmiöihin [Sedgewick, 1996], eikä niihin syvennytä tarkemmin tässä tutkielmassa.

2.5. Lajittelualgoritmit Javassa

Java-ohjelmointikielen *Java Collections Framework* -luokka sisältää valmiina joitakin lajittelualgoritmeja eri tietotyypeille. Objekteille Java 7:ssa on käytössä vakaa MergeSort-algoritmiin pohjautuva TimSort, ja yksinkertaisille tietotyypeille muunneltu versio QuickSort-algoritmista. Java 8 sisältää uutena myös rinnakkaisen lajittelumetodin *Arrays.parallelSort*, joka käyttää MergeSort-pohjaista rinnakkaistettua algoritmia. [Oracle, 2013]

3. Rinnakkaisuus ja rinnakkaislaskenta

Tietokonelaitteistojen kehityksessä on parhaillaan tapahtumassa suuri rakenteellinen muutos. Tavallisessa tietokoneessa on tyypillisesti ollut vain yksi varsinaisen laskentatyön tekevä suoritin, ja useampia suorittimia on ollut lähinnä vain erikoisohjelmistoja käyttävissä raskaaseen laskentaan tai muuhun haastavaan ammattikäyttöön tarkoitetuissa palvelimissa. Nykyään vastaavia ratkaisuja on käytössä yhä enemmän myös kuluttajatason

laitteissa, ja esimerkiksi uusimmissa matkapuhelimissa on jo useampia suorittimia.

3.1. Kohti useampaa suoritinydintä

Tietokoneiden suoritusnopeuden jatkuvaa kasvua on jo opittu pitämään itsestäänselvyytenä. Nopeutta onkin vuosien varrella onnistuttu kasvattamaan koko ajan, ja laskentateho on kasvanut alkuajoista huimasti. Puhutaan *Mooren laista*, jonka mukaan teho kaksinkertaistuu noin kahden vuoden välein. Laki on ennustanut kehitystä poikkeuksellisen tarkasti pian jo viidenkymmenen vuoden ajan [Hill and Marty, 2008]. Moore [1965] tutki ja ennusti mikropiirille pakattujen elektronisten komponenttien määrän kasvua, ja määrä onkin kasvanut useita kertaluokkia: vuonna 1965 yhdelle piirille saatiin viisikymmentä komponenttia ja vuonna 2007 jo liki viisi miljardia [Forsell *et al.*, 2008].

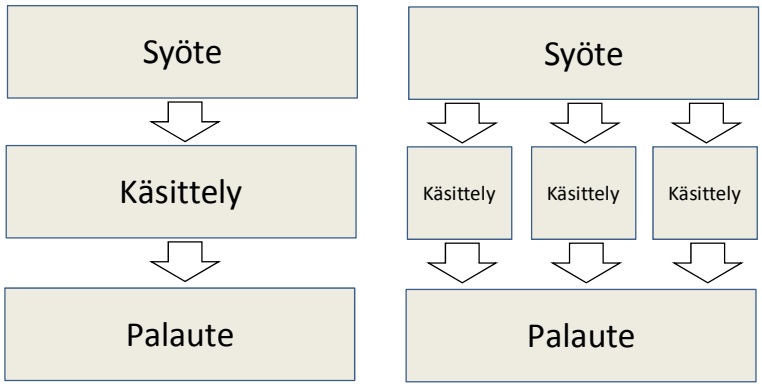
Perinteisesti suorittimiin on haettu lisää nopeutta kasvattamalla kellotaajuutta, lisäämällä välimuisteja sekä kehittelemällä muita aputoimintoja. Forsellin ja muiden [2008] mukaan nämä vanhat keinot alkavat kuitenkin olla loppuun käytettyjä. Fysiikan lait asettavat komponenttien koolle ja pakkaustiheydelle tietyt rajat, joita ei ainakaan nykytekniikalla pystytä alittamaan. Korkeampi kellotaajuus lisää suorittimen energiankulutusta ja sitä kautta lämmön tuottoa, ja Stallingsin [2010] mukaan kellotaajuuden kasvattamisessa onkin jo saavutettu käytännön asettama raja.

Vaikka yksittäisen suoritinytimen nopeuden yläraja alkaakin häämöttää, komponenttien pakkaustiheyttä on vielä mahdollista pienentää jonkin verran [Forsell *et al.*, 2008]. Tämän seurauksena samalle mikropiirille on ryhdytty mahduttamaan yhden sijaan useampia suoritinytimiä, jolloin puhutaan *moniydinsuorittimesta* (multicore, chip multiprocessor) [Stallings, 2010]. Näin laskentatehoa saadaan ainakin teoriassa kasvatettua vielä jonkin aikaa. Tässä tutkielmassa keskitytään nimenomaan moniydinsuorittimiin rinnakkaisuuden lähteenä, sillä suurin osa kuluttajatason suorittimista kuuluu tähän kategoriaan kuten tutkimuksessa käytetty tietokonekin.

3.2. Rinnakkaislaskennan haasteet

Useamman suorittimen tai suoritinytimen samanaikaista hyödyntämistä ohjelman suorituksessa kutsutaan *rinnakkaislaskennaksi* (kuva 2). Rinnakkaista tietojenkäsittelyä voidaan toteuttaa monella eri tavalla ja tasolla, ja se voidaan jakaa hieno-, keski- ja karkearakeiseen synkronointivälin perusteella [Stallings, 2010]. Synkronointiväli on käskyjen määrä, jonka välein järjestelmä tahdistaa suoritettavia prosesseja. Karkearakeinen rinnakkais-

uus tarkoittaa useiden prosessien samanaikaista suoritusta järjestelmässä, ja keskirakeinen on yhden sovelluksen sisällä tapahtuvaa useiden säikeiden rinnakkaista suorittamista.



Kuva 2. Sarjallinen ja rinnakkainen algoritmi.

Rinnakkaislaskentaan kytkeytyy uudenlaisia ongelmia, jotka liittyvät esimerkiksi siihen, miten ohjelman suoritusta käytännössä ositetaan eri ytimille. Lisäksi moniydinympäristössä ytimet jakavat keskenään saman keskusmuistin sekä oheislaitteet, joten resursseja tarvitaan myös niiden yhteiskäytön hallinnoimiseen [Forsell *et al.*, 2008]. Käytännössä vastuu tästä laitteiston kontrolloinnista kuuluu käyttöjärjestelmälle [Stallings, 2010].

Rinnakkaislaskennan varsinainen ongelma on kuitenkin ohjelmistopuolella. Suurin osa vanhoista tietokoneohjelmista on kehitetty yhdellä suorittimella suoritettavaksi, eivätkä ne osaa hyödyntää useampaa suoritinta [Hill and Marty, 2008]. Tällaisten ohjelmien rinnakkaistaminen voi olla haasteellista, sillä ohjelmakoodia saatetaan joutua muuttamaan hyvinkin paljon.

Rinnakkaislaskennan kautta saatava hyöty riippuu siitä, miten hyvin käsiteltävä ongelma saadaan jaettua pienemmiksi, itsenäisiksi rinnakkain suoritettaviksi osiksi [Forsell *et al.*, 2008]. Jotkin tehtävät sopivat tähän hyvin, toiset huonommin. Sellainen osa ohjelmakoodia, jota ei voi rinnakkaistaa ja joka on pakko suorittaa peräkkäin, kutsutaan *sarjalliseksi*.

Gene Amdahlin jo 1960-luvulla esittelemän lain mukaan rinnakkaislaskennasta saatavaa hyötyä rajoittaa juuri suoritettavan ohjelman sarjallinen osuus, sillä ytimien määrän lisääminen ei nopeuta tämän osan suoritusta [Amdahl, 1967; Hill and Marty, 2008]. Vaikka ohjelmasta vain viisi prosenttia olisi pakko suorittaa sarjallisesti, teoreettinen nopeuden kasvu rajattomallakin määrällä ytimiä olisi vain 20-kertainen yhteen ytimeen verrattuna.

Hillin ja Martyn [2008] mukaan nykyaikaisen tietokoneen suorittimessa on yleensä 2–8 suoritinydintä, ja vuonna 2020 suoritinytimiä saattaa olla yhdellä mikropiirillä jo 200. Näin suuri määrä ytimiä tekee sarjallisista osis-

ta todellisen pullonkaulan ja edellyttää vanhojen ohjelmointikäytäntöjen perinpohjaista uudelleentarkastelua [Mattson and Wrinn, 2008]. Käytettäviä ohjelmistoja sekä niihin sisältyviä algoritmeja tulisi siis rinnakkaistaa, jotta useammista suorittimista saataisiin täysi hyöty irti.

Java-ohjelmointikielessä on nykyään hyvä tuki rinnakkaisuudelle. Säikeiden ja niiden synkronoinnin hallintaan on tarjolla lukuisia apuluokkia, joita voi käyttää hyödyksi luotaessa rinnakkain toimivia sovelluksia ja algoritmeja. Rinnakkaisia ohjelmia voi laatia joko hyvin yksityiskohtaisella tasolla tai sitten käyttämällä apuna korkean tason komponentteja, kuten esimerkiksi *Executor*-luokkaa, jotka helpottavat rinnakkaisuuden hallintaa huomattavasti. [Oracle, 2013; Göetz *et al.*, 2005]

4. Testiohjelma ja koeasetelma

Tutkimus suoritettiin itse laaditun Java-ohjelman avulla neliytimisellä tietokoneella. Sovelluksessa oli erikseen pääohjelma, joka suoritti lajiteltavan taulukon laatimisen sekä lajittelunopeuksien mittaamisen. Lisäyslajittelusta sekä ShellSortista tehtiin omat luokat ja koe suoritettiin näitä apuna käyttäen. Kukin lajittelutesti suoritettiin 5 kertaa, ja käytetty aika mitattiin millisekunnin tarkkuudella. Tulokseksi otettiin näiden mittausten keskiarvo.

4.1. Rinnakkainen ShellSort-algoritmi

Normaalissa ShellSortissa kukin n-alitaulukko käydään erikseen läpi tavallisen lisäyslajittelun avulla. ShellSortin rinnakkaistamiseksi tavalliseen ShellSortiin lisättiin ylimääräinen toiminto, joka jakaa nämä alitaulukot eri säikeille suoritettavaksi. Kukin säie saa käynnistyessään parametrina sekä oman numeronsa että säikeiden kokonaislukumäärän. Näitä tietoja on hyödyntämässä säikeen sisällä yksi ylimääräinen silmukka, jonka ohjaamana säie lajittelee koko syötteestä vain sille määrätyt alitaulukot.

Säikeiden luomiseen ja hallintaan käytetään *ExecutorService*-luokkaa sekä *ThreadPoolia*, ja säikeiden synkronointiin *CountDownLatch*-objektia [Göetz *et al.*, 2005]. Erillisille säikeille ei luoda omaa kopiota käsiteltävästä taulukosta, vaan kaikki säikeet käsittelevät samaa, syötteenä saatua alkuperäistä taulukkoa. Tämä on mahdollista, koska kaikki kulloinkin käsittelyssä olevat n-alitaulukot ovat toisistaan riippumattomia eivätkä sisällä samoja alkioita. Näin säikeet voivat vapaasti muokata omaa osuuttaan taulukosta, eikä ristiriitoja synny.

Toteutus vaatii kuitenkin säikeiden synkronointia siinä vaiheessa, kun kaikki n-alitaulukot on lajiteltu. Tällöin on odotettava sitä, että kaikki säikeet ovat käsitelleet omat osuutensa. Jo valmistuneet säikeet eivät voi läh-

teä lajittelemaan uusia pienempiä alitaulukoita ennen kuin viimeinenkin säie on valmistunut. Jos näin kävisi, niin taulukon sisältö menisi lopulta sekaisin kahden eri säikeen operoidessa mahdollisesti samoja alkioita.

Alitaulukkoihin jakamisen perusteena käytettiin Ciuran [2001] esittämää sarjaa (1, 4, 10, 23, 57, 132, 301, 701), jota jatkettiin ylöspäin kertomalla aina edellinen arvo luvulla 2.25. Yllättäen parhaaseen tulokseen päästiin, kun n-arvot 23, 132 ja 301 jätettiin kokonaan pois. Sama ilmiö tapahtui myös rinnakkaistamattomassa ShellSortissa. Myös joitakin muita sarjoja kokeiltiin ja niillä oli vaikutusta lajittelunopeuteen, mutta yksikään toinen sarja ei kuitenkaan päässyt nopeudessa lähelle tätä muokattua Ciuran sarjaa.

4.2. Testiohjelma

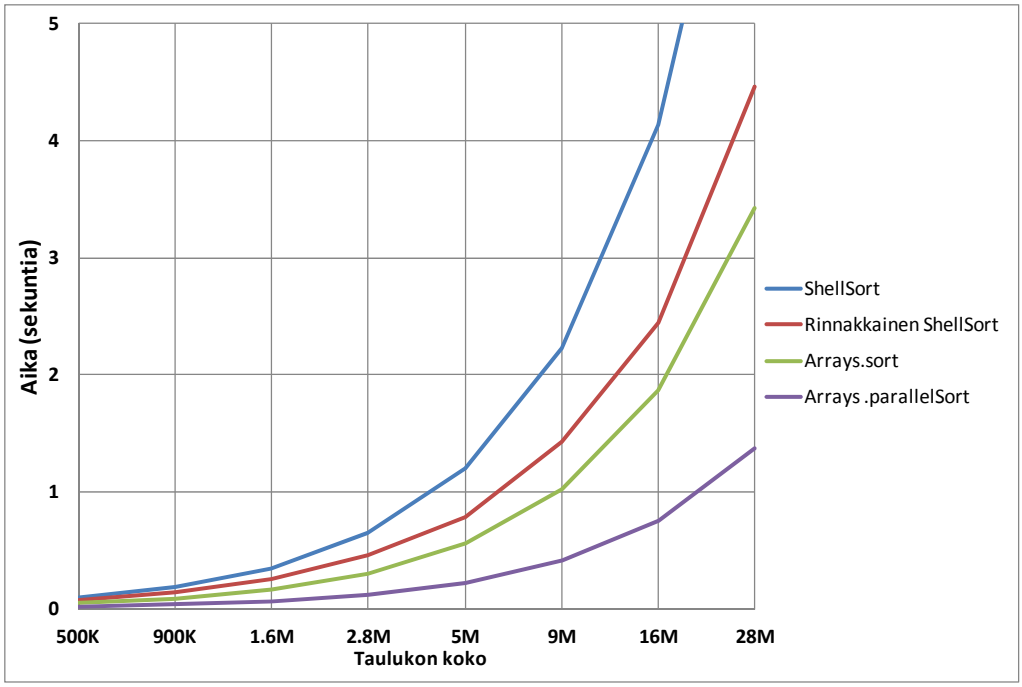
Pääohjelma laati silmukassa erikokoisia kokonaislukutaulukoita, joita se täytti satunnaisilla luvuilla nollan ja suurimman mahdollisen kokonaisluvun väliltä. Tästä laaditusta taulukosta otettiin kopio aina jokaista lajittelualgoritmia varten, jotta lajiteltava sisältö olisi sama jokaisella algoritmilla ja saadut ajat olisivat vertailukelpoisia keskenään. Normaali ShellSort sekä lisäyslajittelu toteutettiin hyvin samankaltaisina kuin ne on esitelty kirjallisuudessa [Cormen *et al.*, 2001; Sedgewick, 1996]. Lisäyslajittelu osoittautui niin hitaaksi, että sitä ei käytännön syistä enää käytetty suurempien syötteiden kanssa.

5. Tulokset

ShellSortin sekä Javan omien lajittelijoiden testitulokset ovat kuvassa 3. Rinnakkainen ShellSort osoittautui nopeammaksi kuin rinnakkaistamaton, mutta ei yltänyt Javan omien lajittelijoiden vauhtiin. Javan rinnakkaistettu lajittelija oli ylivoimaisesti nopein, ja lisäyslajittelu oli odotetusti hitain. Pienillä alkiomäärillä lajitteluun kulunut aika vaihteli eri suorituskerroilla suhteellisesti huomattavan paljon. Vaikka lopputulos on useamman mittauksen keskiarvo, niin keskihajonta oli niin suuri, että alle 50000 alkion taulukoiden lajitteluajat ovat lähinnä suuntaa-antavia.

Taulukon koko	ShellSort	Rinnakkainen ShellSort	Arrays.sort	Arrays .parallelSort	Lisäys-lajittelu
987	0,0006	0,001	0,0002	0,0001	0,0008
1742	0,0002	0,0014	0,0002	0,0002	0,001
3076	0,0002	0,0012	0,0004	0,0002	0,0026
5430	0,0006	0,0012	0,0002	0,0004	0,0086
9587	0,001	0,0022	0,0008	0,0008	0,0264
16926	0,0022	0,0086	0,0014	0,0006	0,0854
29882	0,0044	0,0068	0,0026	0,0012	0,2612
52755	0,0078	0,009	0,0046	0,0018	0,8142
93136	0,015	0,013	0,008	0,0036	2,538
164426	0,0282	0,024	0,0136	0,0064	7,9194
290283	0,053	0,0416	0,0264	0,0104	24,725
512474	0,0992	0,0782	0,0486	0,0228	77,412
904738	0,185	0,141	0,0896	0,0394	243,93
1597252	0,3482	0,2574	0,1648	0,068	-
2819838	0,6496	0,4604	0,3026	0,1264	-
4978226	1,2048	0,7898	0,5566	0,228	-
8788711	2,2338	1,4324	1,0254	0,4134	-
15515854	4,1344	2,439	1,8644	0,7476	-
27392154	7,6638	4,462	3,4282	1,3701	-

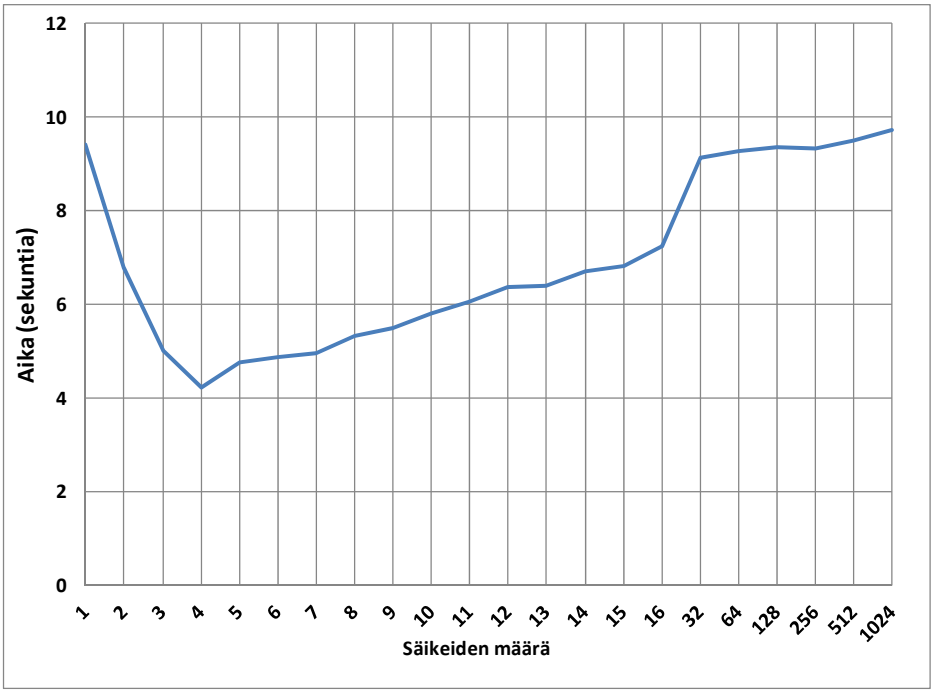
Kuva 3. Eri lajittelualgoritmien nopeuksia sekunneissa mitattuna.



Kuva 4. Lajitteluajan kasvu suhteessa lajiteltavan taulukon kokoon.

Lajitteluun kulunut aika kasvoi lajiteltavan taulukon koon myötä eri algoritmeilla eri tavalla (kuva 4). ShellSortin tarvitsema aika kasvoi selvästi

jyrkimmin ja Javan rinnakkaisen lajittelijan hitaimmin. Kuvasta on jätetty kokonaan pois lisäyslajittelu, sillä sen kuvaaja erosi hyvin paljon muista, eikä sen sisällyttäminen olisi ollut mielekästä.



Kuva 5. Säikeiden määrän vaikutus lajitteluaikaan 28 miljoonalla alkiolla.

Myös käytettävien säikeiden määrä vaikutti voimakkaasti rinnakkaisen ShellSortin lajittelunopeuteen etenkin suuremmilla tietomäärillä (kuva 5). Paras tulos saatiin käyttämällä neljää säiettä, ja tämä liittyy siihen, että testikoneessa oli neliytiminen prosessori [Stallings, 2011]. Suuremmalla säikeiden määrällä niiden luominen ja hallinta vaativat niin paljon ylimääräistä suoritinaikaa, että lisätehoa ei enää saada, koska säikeet joutuvat jakamaan samoja suoritinytimiä keskenään.

Taulukon koon ollessa pieni lajitteluajat vaihtelivat huomattavan paljon erityisesti kummallakin ShellSort-algoritmilla. Tämä saattoi johtua siitä, että taulukon sisältö vaikutti niiden lajittelunopeuteen voimakkaammin kuin Javan omiin lajittelijoihin. Toisaalta syynä saattoi olla se, että itse lajittelun ollessa nopea toimenpide tietokoneessa samaan aikaan tapahtuneet käyttöjärjestelmän satunnaiset toiminnot saattoivat hidastaa suoritusta ja muodostaa suhteellisen ison osan käytetystä ajasta.

6. Yhteenveto

Näyttää siltä, että tulevaisuudessa laskentatehon kasvu tulee nimenomaan rinnakkaislaskennan suunnalta. Tämän vuoksi rinnakkaisen suorittamisen periaatteet pitäisi ottaa huomioon jo ohjelmien ja algoritmien suunnittelu-

vaiheessa. Erityisesti olisi keskityttävä sarjallisten osien osuuden minimointiin, sillä niiden merkitys tulee kasvamaan huomattavasti [Hill and Marty, 2008]. Kääntäjien sekä muiden ohjelmointityökalujen on osattava hyödyntää rinnakkaisuuden tarjoamat mahdollisuudet sekä tunnistettava siihen liittyvät sudenkuopat. Tuotettu koodi olisi optimoitava rinnakkaiseen suoritukseen.

Rinnakkaislaskenta tarjoaa haasteistaan huolimatta kuitenkin mahdollisuuden nopeuden kasvun jatkumiseen [Mattson and Wrinn, 2008]. Siinä missä tehon lisäys on aiemmin saatu ilman suurempia muutoksia aiempaan ohjelmakoodiin, siirtyminen rinnakkaislaskentaan vaatii myös ohjelmoijilta uudenlaisten tapojen ja tekniikoiden omaksumista.

Lajittelualgoritmien merkitys ei tule tulevaisuudessakaan vähenemään, sillä käsiteltävien tietomäärien kasvaessa niiden käyttö tulee varmasti lisääntymään yhä enemmän. Nykytekniikka mahdollistaa valtavien tietomäärien keräämisen ja analysoinnin, eikä tästä selvitä ilman hyviä lajittelualgoritmeja. Päinvastoin tarvitaan uusia, suurelle tietomäärälle kehitettyjä hyvin rinnakkaistettuja algoritmeja, joiden avulla uuden ajan tietojenkäsittely saadaan toimimaan sujuvasti.

Vaikka ShellSort onkin selvästi hitaampi algoritmi kuin nopeimmat nykyiset lajittelualgoritmit, niin se on niihin verrattuna kuitenkin yksinkertaisempi ja helpompi toteuttaa. Lisäksi muistimäärän ollessa rajoitettu on ShellSortin $O(1)$-tilavaatimus parempi kuin esimerkiksi QuickSortin ja MergeSortin keskimääräinen $O(\log n)$ tai pahimman tapauksen $O(n)$ [Cormen *et al.*, 2001]. Koska ShellSort rinnakkaistuu kohtuullisen hyvin, saattaa sille olla käyttöä vielä tulevaisuudessakin.

Viiteluettelo

[Amdahl, 1967] Gene Amdahl, Validity of the single processor approach to achieving large scale computing capabilities. In: *Proc. of the 1967 Spring Joint Computer Conference* (1967), 483–485.

[Ciura, 2001] Marcin Ciura, Best increments for the average case of Shellsort. In: *Proc. of the 13th International Symposium on Fundamentals of Computation Theory* (2001), 106–117.

[Cormen *et al.*, 2001] Thomas Cormen, Charles Leiserson, Ronald Rivest and Clifford Stein, *Introduction to Algorithms*. MIT Press, 2001.

[Forsell *et al.*, 2008] Martti Forsell, Ville Leppänen ja Martti Penttonen, Rinnakkaistietokoneen uusi tuleminen. *Tietojenkäsittelytiede* **28** (2008), 55–65.

[Göetz *et al.*, 2005] Brian Göetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes and Doug Lea, *Java Concurrency in Practice.* Addison-Wesley, 2005.

[Heineman *et al.*, 2008] George Heineman, Gary Pollice and Stanley Selkow, *Algorithms in a Nutshell.* O'Reilly Media, 2008.

[Hill and Marty, 2008] Mark Hill and Michael Marty, Amdahl's law in the multicore era. *Computer* **41**, 7 (2008), 33–38.

[Hoare, 1962] Tony Hoare, Quicksort. *The Computer Journal* **5**, 1 (1962), 10–16.

[Knuth, 1998] Donald Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison Wesley, 1998.

[Mattson and Wrinn, 2008] Tim Mattson and Michael Wrinn, Parallel programming: Can we PLEASE get it right this time? In: *Proc. of the 45th Annual Design Automation Conference* (2008), 7–11.

[Moore, 1965] Gordon Moore, Cramming more components onto integrated circuits. *Electronics* **38**, 8 (1965), 114–117.

[Oracle, 2013] Oracle Corporation, Java Platform Standard Edition 7 documentation. http://docs.oracle.com/javase/7/docs/index.html (Checked 20.12.2013).

[Sedgewick, 1996] Robert Sedgewick, Analysis of Shellsort and related algorithms. In: *Proc. of the 4th European Symposium on Algorithms* (1996).

[Shell, 1959] Donald Shell, A high-speed sorting procedure. *Commun ACM* **2**, 7 (1959), 30–32.

[Stallings, 2010] William Stallings, *Computer Organization and Architecture: Designing for Performance.* Prentice Hall, 2010.

[Stallings, 2011] William Stallings, *Operating Systems: Internals and Design Principles.* Pearson Education, 2011.

Virtuaalitodellisuuden terapeuttisista mahdollisuuksista

Elias Roihuvuo

Tiivistelmä.
Virtuaalitodellisuutta on tutkittu erilaisissa kliinisissä tarkoituksissa. Tässä tutkielmassa käsittelen virtuaalitodellisuutta ja sen erinäisiä hoidollisia tutkimuskohteita. Tutkielman painopiste on erilaisten terveydellisten tilojen hoitamisessa virtuaalitodellisuuden avulla.

Avainsanat- ja sanonnat: Virtuaalitodellisuus, kivun hoito, skitsofrenia, fobiat, ruumiinkuvan häiriö
CR-luokat: H.5.1

1. Johdanto

Virtuaalitodellisuus herättää usein lähinnä pelillisiä ja viihteellisiä mielleyhtymiä, mutta sitä voidaan soveltaa myös lääketieteessä. Virtuaalitodellisuuden hyödyntäminen erilaisissa hoidollisissa tarkoituksissa on viimeisten viidentoista vuoden aikana saanut osakseen kasvavaa kiinnostusta. Kyseessä on niin psyykkisten kuin fyysistenkin tilojen hoidon uudenlainen lähestymistapa, jonka toteuttamisen käytännössä on mahdollistanut tietokoneiden ja oheislaitteiden nopea kehittyminen.

Virtuaalitodellisuuden asema hoitomuotona on lupaavista tuloksista huolimatta edelleen vakiintumaton, eikä sen roolia itsenäisenä hoitona tai muiden hoitokäytäntöjen tukena ole vielä tarkkarajaisesti määritelty.

Tämän tutkielman tarkoituksena on luoda katsaus virtuaalitodellisuuteen sekä etenkin siihen, millaisten sairauksien tai hoitoa vaativien tilojen tapauksessa virtuaalitodellisuutta on tutkittu. Aluksi käsittelen virtuaalitodellisuutta sekä vaikuttimia sen lääketieteelliseen tutkimukseen. Tämän jälkeen esittelen virtuaalitodellisuuden hyödyntämistä hoidettaessa kipua, fobioita, skitsofreniaa ja ruumiinkuvan häiriöitä. Lopuksi kokoan yhteen tutkielman keskeisen sisällön ja teen loppupäätelmät.

2. Virtuaalitodellisuus

Termi *virtuaalitodellisuus* (virtual reality) esiintyi ensimmäisen kerran vuonna 1986, jolloin Jaron Lanier esitteli sen [Yoh, 2001]. Käsitteenä virtuaalitodellisuus voidaan ymmärtää monella tavalla, ja pyrinkin tässä luvussa selventämään sen määritelmää siten kuin tutkielman aiheen kannalta on olennaista.

Virtuaalitodellisuus toteutetaan tyypillisesti käyttämällä laitteistoja, jotka tuottavat ärsykkeitä eri aistijärjestelmille. Näköaistimukset voidaan toteuttaa käyttämällä päähän asetettavia visiirinäyttöjä (head mounted display), kuuloaistimukset tilaäänijärjestelmien avulla ja tuntoaistimukset erilaisin asustein, kuten käsinein.

Tiukan teknisestä näkökulmasta arvioituna virtuaalitodellisuuden voidaankin käsittää tarkoittavan sellaista teknologista ympäristöä, jossa käyttäjä voi kokea asioiden tapahtuvan todellisen kaltaiseen tapaan. Tällaisia ympäristöjä voivat olla esimerkiksi erilaiset simulaatiot, mutta tulkinnan mukaan näiden keinotekoisten ympäristöjen elämykselliset ulottuvuudet ovat toissijaisia. [Yoh, 2001]

Virtuaalitodellisuus voidaan kuitenkin mieltää myös laajemmaksi ilmiöksi tarkastelemalla sitä kokemuksellisesta näkökulmasta. Tämän näkemyksen mukaan virtuaalitodellisuus ei ole vain teknisesti luotu ja ulkopuolelta tarkasteltava mukaelma jostakin fyysisen maailman osasta, vaan itsenäisempi ilmiö sisältöineen. Tässä yhteydessä käytetään käsitettä *läsnäolo* (presence). Läsnäololla tarkoitetaan tunnetta ja tietoisuutta ympäröivästä maailmasta. Läsnäolon tunne ei kuitenkaan ole erottamattomasti sidoksissa yksinomaan fyysiseen ympäristöön, vaan sen voidaan mieltää muodostuvan myös niistä tulkinnoista, joita ympäristön ärsykkeistä tehdään. Niinpä läsnäolon kokemusta ei synny, ellei näitä ärsykkeitä aktiivisesti käsitellä. Esimerkkinä voidaan pitää vaikkapa ajatuksiin uppoutumista, jolloin ympäröivä todellisuus tuntuu häviävän, kun läsnäolon kokemus kohdistuukin mielikuvien maailmaan.

Oleellisimpia tekijöitä virtuaalisen läsnäolon kannalta ovat muun muassa virtuaalitodellisuuden hallitsevuus konkreettiseen ympäristöön nähden sekä tunne luodussa ympäristössä olemisesta sen sijaan, että se koettaisiin vain katseltavana kohteena [Fornells-Ambrojo *et al.*, 2008]. Voidaankin ajatella, että juuri läsnäolon ja vuorovaikutteisuuden ansiosta virtuaaliympäristössä tapahtuvat kokemukset ovat havainnoitsijan näkökulmasta yhtä todellisia kuin fyysisenkin maailman kokemukset. Juuri tämän ansiosta virtuaalitodellisuuden mahdollisuudet esimerkiksi käyttäytymisterapiassa ovat herättäneet kiinnostusta. [Yoh, 2001]

Läsnäolon ohella toinen keskeinen käsite on *immersio,* joka voidaan virtuaalitodellisuuden tapauksessa mieltää yleisesti todentuntuisuudeksi ja luonteeltaan tekniseksi läsnäolon edellytykseksi. Immersion syvyys on riippuvainen virtuaalitodellisuuden toteuttavan järjestelmän ominaisuuksista, kuten näkökentän laajuudesta ja luodun näkymän resoluutiosta sekä vuorovaikutuksen moniaistisuudesta. [Yoh, 2001]

Lääketieteen ammattilaisten kannalta eräs suurimmista hoidollisen virtuaalitodellisuuden houkutuksista pohjautunee sen kustannustehokkuuteen. Mikäli virtuaalitodellisuus voisi osaltaan korvata kliinikon aktiivisen osallistumisen, voitaisiin saavuttaa taloudellisia säästöjä sekä myös lisätä hoidon tavoitettavuutta.

Levac ja Galvin [2013] ovat kuitenkin esittäneet, ettei virtuaalitodellisuutta tule käsittää itsenäiseksi hoitomuodoksi, ellei se kykene huolehtimaan kaikista perinteisesti kliinikon vastuulla olevista hoitojen osa-alueista. Heidän mukaansa ei toistaiseksi ole kehitetty järjestelmää, joka itsenäisesti ja ilman kliinikon osallistumista pystyisi esimerkiksi räätälöimään kullekin potilaalle sopivan yksilöllisen hoidon tai antamaan yksilökohtaista ohjeistusta ja palautetta.

Virtuaalitodellisuus tulisi mieltää pikemminkin nykyisiä hoitomuotoja tukevaksi apuvälineeksi, jonka tehokas hyödyntäminen edellyttää kliinikoilta aktiivista osallistumista. Tämän tulkinnan mukaan virtuaalitodellisuuden käyttöä koskevat vastaavat edellytykset kuin muitakin hoidollisia apuvälineitä, kuten esimerkiksi fysioterapeuttisia laitteita. Virtuaalitodellisuuden tehokas hyödyntäminen edellyttää, että hoitohenkilökunta on riittävän perehtynyt tekniikkaan sekä osaa käyttää sitä asiantuntevasti ja turvallisesti. Parhaimmillaan virtuaalitodellisuus voisi tukea hoitoa sekä kliinikon että potilaan näkökulmasta. Kliinikon voisi olla mahdollista huomioida potilaiden yksilölliset tarpeet entistä tarkemmin sekä myös kasvattaa potilaiden motivaatiota osallistua usein tylsäksi tai turhauttavaksi koettuun terapiaan. [Levac and Galvin, 2013]

3. Virtuaalitodellisuuden tutkimuseettisiä näkökulmia

Potilaiden turvallisuus on aina huomioitava huolellisesti käytettävästä hoitomenetelmästä riippumatta. Potilaiden turvallisuudesta ja hyvinvoinnista on huolehdittava erityisen tarkoin tutkittaessa uutta ja kokeellista hoitomuotoa, jollainen virtuaalitodellisuuskin on.

Kenties eräs suurimmista lääketieteellisen tutkimuksen eettisistä ongelmista kumpuaa siitä, kuka todella hyötyy eniten uusien menetelmien tutkimuksesta. Virallisesti innovaatioiden tulisi aina hyödyttää ensisijaisesti potilaita, mutta myös tutkijoiden henkilökohtaisten ja itsekkäiden motiivien mahdollisuus on otettava realistisesti huomioon. Mikäli tutkijat pyrkivät ensisijaisesti esimerkiksi niittämään mainetta uusilla innovaatioilla, saatetaan potilaiden oikeuksia herkästi tai jopa tietoisesti laiminlyödä ja tutkimukset suorittaa välittämättä mahdollisesta inhimillisestä kärsimyksestä. Esimerkkejä tällaisesta valitettavasta menettelystä voisivat olla sähköhoito ja lobotomia, joita aikanaan pidettiin mullistavina keksintöinä ja käytettiin ahkerasti ilmeisistä haitoista huolimatta.

Whalley [1995] esittää, että lääketieteen tutkijoiden tarve läpimurtoihin on erityisen pakottava sellaisilla osa-alueilla, joihin liittyy uraauurtavaa tekniikkaa. Myös virtuaalitodellisuustutkimuksissa käytetään potilasryhmiä, joiden valmiudet puntaroida kokeisiin osallistumista voivat olla merkittävästi heikentyneet. Tällaisiin ryhmiin kuuluvat etenkin psyykkisistä sairauksista tai vajavaisuuksista kärsivät potilaat, jotka saattavat olla kykenemättämiä itsenäiseen päätöksentekoon ja mahdollisesti helposti tutkijoiden ohjailtavissa. Lisäksi on huomioitava, että juuri tällaisten potilaiden kohdalla pyritään muokkaamaan käyttäytymistä haluttuun suuntaan virtuaalitodellisuuden avulla, eikä tutkijoiden moraalista vastuusta voi liiaksi korostaa.

Hyödynnettäessä virtuaalitodellisuutta on varmistuttava myös siitä, että keinotekoinen maailma ei järkytä käyttäjien henkistä tasapainoa. Yksinkertaisimmillaan tämä voi tarkoittaa mielekkäiden ja turvallisten tuntuisten virtuaalisten ympäristöjen luomista. Käyttäjien reaktioita ja järjestelmien mahdollisia toimintahäiriöitä on kuitenkin erittäin vaikeaa, ellei peräti mahdotonta ennustaa tarkasti. Äkillinen häiriö esimerkiksi kivunlievitykseen käytettävässä järjestelmässä voisi olla traumaattinen tilanne, ja niinpä myös tämän hoitomuodon höydyt ja haitat on aina punnittava perusteellisesti ja tapauskohtaisesti.

4. Virtuaalitodellisuuden käyttö kivun hoidossa

Tässä luvussa käsittelen kipua ja perusteita virtuaalitodellisuuden käytölle sen hoidossa. Kansainvälisen kivuntutkimusseura IASP:n määritelmän mukaan kipu on "epämiellyttävä aistimus ja tunnepohjainen elämys, johon liittyy kudosvaurio tai sen uhka tai jota kuvataan kudosvaurion käsittein" [Koulu ja Tuomisto, 2007].

Kipu voi olla tavanomaisista hoidoista huolimatta suuresti elämänlaatua heikentävää ja jopa invalidisoivaa, joten uusien, entistä tehokkaampien tai entisiä menetelmiä tukevien hoitomuotojen kehittäminen on tutkimuksen kohteena.

Länsimaisen lääketieteen piirissä kivun hoidossa käytetään tyypillisesti niin sanottuja analgeetteja. Analgeeteilla tarkoitetaan lääkeaineita, jotka muovaavat kipujärjestelmän toimintaa ja siten vaimentavat kipua aiheuttamatta unta tai anestesiaa, jossa ulkoisiin ärsykkeisiin reagointi estyy täysin [Koulu ja Tuomisto, 2007].

Varsinaisten analgeettien ohella kipua voidaan hoitaa myös lääkkeettömien menetelmien avulla. Tällaisia menetelmiä ovat muun muassa hypnoosi, mielikuvaharjoitukset sekä tietoinen ponnistelu kognitiivisesti kuormittavien tehtävien, kuten vaikkapa päässälaskujen parissa. Näiden menetelmien toiminta pohjautuu näkemykseen, jonka mukaan ihmisen resurssit tietoiseen havain-

nointiin ovat rajalliset. Myös kipu kilpailee näistä resursseista, ja niinpä kivun osakseen saamaa huomiota ja siten sen koettua voimakkuutta voidaan vähentää ohjaamalla huomio johonkin muuhun asiaan. Kun kivun havainnointiin ei riitä resursseja, se ei enää saavuta tietoisuutta. [Hoffmann *et al.*, 2000]

Esimerkiksi Schmitt ja muut [2011] esittävät, että virtuaalitodellisuus voisi juuri immersiivisyytensä ja vuorovaikutteisuutensa ansiosta olla erityisen tehokas ja soveltuva huomionohjausmenetelmä. Läsnäolokokemuksen virtuaalisessa ympäristössä ajatellaan olevan parhaimmillaan niin voimakas, että sen avulla saavutetaan perinteisiä mielikuvamenetelmiä parempi irtautuminen epämiellyttävästä tilanteesta.

5. Käytännön kokemuksia kivun hoidosta

Virtuaalitodellisuuden soveltuvuutta kivun hoidossa on tutkittu etenkin nuorten palovammapotilaiden haavanhoitotoimenpiteiden ja fysioterapian yhteydessä.

Palovammapotilaat kärsivät usein erittäin voimakkaista kivuista, joita ei välttämättä kyetä lievittämään riittävästi perinteisen kipulääkityksen keinoin. Voimakkaiden, morfiinijohdannaisten analgeettien käyttöä rajoittavat varsinkin nuorilla niihin liittyvät useat sivuvaikutukset ja ongelmat, joita ovat esimerkiksi riippuvuuden kehittymisen ja hengityslaman riski [Schmitt *et al.*, 2011]. Kivun hoito on ensiarvoisen tärkeää sujuvan toipumisen kannalta, sillä voimakkaiden kipukokemusten on etenkin nuorilla havaittu johtavan pitkäaikaisvaikutuksiin, kuten kivulle herkistymiseen ja heikentyneeseen psyykkiseen hyvinvointiin [Kipping *et al.*, 2012].

Kivun hoitoon tarkoitetun virtuaalitodellisuuden toteutuksessa on yleensä pyritty saavuttamaan mahdollisimman voimakas immersio, eli voimakas uppoutuminen ja eläytyminen luotuun keinotekoiseen ympäristöön. Kolmiulotteisen vaikutelman aikaansaamiseksi on käytetty korkearesoluutioisia ja laajan näkökentän mahdollistavia visiirinäyttöjä [Hoffmann *et al.*, 2000; Schmitt *et al.*, 2011], jotka myös rekisteröivät käyttäjän päänliikkeet ja mukauttavat esitettävän näkymän niitä vastaaviksi. Lisäksi voidaan seurata käyttäjän kädenliikkeitä ja antaa tuntopalautetta, jonka ansiosta kohteiden käsittely virtuaalitodellisuudessa on todentuntuisempaa.

Yksi esimerkki käytetyistä virtuaalitodellisuusympäristöistä on SnowWorld, jota ovat hyödyntäneet ainakin Schmitt ja muut [2011]. SnowWorld muodostuu mukautuvasta lumisesta ympäristöstä, jossa käyttäjä voi heittää lumipalloilla erilaisia kohteita, kuten lumiukkoja ja pingviinejä. Hoffmann ja muut [2000] ovat käyttäneet samankaltaista, kohteiden manipuloinnin mahdollistavaa virtuaaliympäristöä.

Potilaat arvioivat hoitojen aikana kokemiaan tuntemuksia visuaalisella asteikolla. Kartoitettavia kokemuksia olivat muun muassa kivun voimakkuus ja epämiellyttävyys, viihtyvyys toimenpiteiden aikana, virtuaalitodellisuuden aiheuttama pahoinvointi, virtuaalitodellisuuden kohteiden realistisuus sekä virtuaalitodellisuuden immersiivisyys. Hoffmann ja muut [2000] ja Schmitt ja muut [2011] ovat saavuttaneet samankaltaisia, kannustavia tuloksia. Potilaiden kivut vähentyivät merkitsevästi, virtuaalitodellisuudesta aiheutuvaa pahoinvointia ei esiintynyt juuri lainkaan ja kokemus virtuaalimaailmaan uppoutumisesta säilyi hoitokerrasta toiseen. Tulosten tulkinnassa on kuitenkin huomioitava, että Hoffmann ja muut [2000] suorittivat vain alustavan tapaustutkimuksen. Lisäksi Schmitt ja muut [2011] toteavat käyttäneensä useita eri mallisia visiirinäyttöjä ja myöntävät puutteet esimerkiksi osallistujien sokkoutuksessa ja koetun pahoinvoinnin kartoituksen kontrolloinnissa.

Käytettäessä halvempaa ja valmiina kokonaisuutena markkinoitavaa yksinkertaisempaa tekniikkaa sekä tavanomaisia videopelejä hyödyt ovat jääneet vaatimattomimmiksi [Kipping *et al.*, 2012]. Tämä voidaan tulkita osoitukseksi siitä, että potilaan tietoinen keskittyminen virtuaalitodellisuuteen ja immersion kokemus ovat välttämättömiä hoitomuodon onnistumiseksi. Ellei potilas koe olevansa läsnä keinotekoisessa ympäristössä ja vuorovaikutuksessa sen kanssa, hänen huomionsa ei irtaannu tarpeeksi samanaikaisesti suoritettavien toimenpiteiden tuottamasta kivusta.

Virtuaalitodellisuutta on tutkittu palovammoista aiheutuvien kipujen hoidossa vasta kokeiluluontoisesti, joten sen laajemmasta käytöstä ei ole vielä kokemuksia. Alkuhankaluudet ovat tyypillisiä uusille teknologioille, eikä niitä ole mahdollista ennustaa täysin varmasti pelkkien kokeellisten asetelmien pohjalta.

Markus ja muut [2009] ovat selvittäneet, millaisia käytännön vaatimuksia virtuaalitodellisuusjärjestelmien hyödyntäminen sairaalaolosuhteissa asettaa henkilökunnalle. Hoitoa toteuttaessaan henkilökunnan on huolehdittava useista vaiheista. Näitä ovat muun muassa käytettävän laitteiston eristäminen hygieniasyistä, järjestelmien käynnistäminen ja ohjelmistojen lataaminen sekä laitteiston säätäminen potilaalle henkilökohtaisesti. Lisäksi laitteisto on vielä purettava ja desinfioitava hoitokerran päätyttyä.

Tutkimuksessa kutakin hoitokertaa toteutti kaksi koulutettua työntekijää, joista toinen huolehti tekniikasta ja toinen annettavasta hoidosta. Yksittäisen hoitosession kesto kaikkine järjestelyineen oli keskimäärin 59 minuuttia, joista kuitenkin vain noin 13 minuuttia kului potilaan ohjeistamiseen sekä varsinaiseen hoitamiseen ja loput noin 46 minuuttia laitteistoon liittyviin käytännön toimiin, joissa ilmeni myös teknisiä hankaluuksia. [Markus *et al.*, 2009]

Hoitomuodon hyödyntäminen esimerkiksi tavanomaiseen mielikuvaharjoitteluun verrattuna vaatii enemmän niin henkilöstöä kuin aikaakin. Lisäksi järjestelmien kustannukset saattavat rajoittaa niiden hyödyntämistä, sillä Sharar ja muut [2007] toteavat, että voimakkaan immersion mahdollistavan laitteiston kustannukset voivat kohota kymmeniin tuhansiin dollareihin. Sairaalaympäristöissä on usein pulaa näistä kaikista resursseista, ja niinpä nykyisten virtuaalitodellisuusjärjestelmien käyttökelpoisuutta hoitomuotona on arvioitava huolellisesti vaatimusten ja saavutettujen hyötyjen suhteen. Tulevaisuudessa kehittyneemmät ja halvemmat laitteet sekä erikoistuneempi henkilökunta saattavat ratkaista havaitut ongelmat.

6. Virtuaalitodellisuuden käyttö fobioiden hoidossa

Fobioilla tarkoitetaan sellaisia pelkoja, jotka kohdistuvat tiettyyn tilanteeseen tai kohteeseen, ja jotka ovat aiheuttajaansa nähden epätavallisen voimakkaita [Huttunen, 2013a]. Tyypillisiä fobioita ovat esimerkiksi akrofobia eli korkeanpaikankammo, araknofobia eli hämähäkkien pelko sekä lentopelko. Pelot ja niihin liittyvä ahdistuneisuus saattavat olla niin voimakkaita, että ne johtavat pelon aiheuttajan aktiiviseen välttelemiseen ja rajoittavat normaalia elämää tuntuvasti. Esimerkiksi voimakas lentopelko saattaa käytännössä täysin estää urakehityksen ja akrofobia voi tehdä vaikkapa parvekkeella oleskelusta mahdotonta.

Bush [2008] väittää, että jopa noin 19 miljoonaa amerikkalaista kärsii jostakin fobiasta, ja että vaivastaan huolimatta heistä vain 15-40 prosenttia hakeutuu hoitoon. Hoidon karttamisen syynä pidetään sietämätöntä ahdistuneisuutta, jota jo ajatus pelkojen ja niitä aiheuttavien tilanteiden kohtaamisesta aiheuttaa. Näin ollen on erittäin perusteltua pyrkiä kehittämään uusia hoitomuotoja, jotka olisivat sekä tehokkaita että nykyisiä helpommin lähestyttäviä. Virtuaalitodellisuuden hyödyntämiseen fobioiden hoidossa onkin osoitettu kasvavaa kiinnostusta aina 1990-luvun puolivälistä lähtien.

Fobioita hoidetaan tyypillisesti hyödyntämällä käyttäytymisterapiaa, jossa potilas altistuu pelkojensa aiheuttajalle vähitellen. Käytännössä tämä voi toteutua esimerkiksi akrofobian tapauksessa siirtymällä kerros kerrallaan yhä korkeammalle. Teorian mukaan potilas tottuu vähittäisen altistumisen myötä kohtaamaan pelon aiheuttajan ja oppii tulemaan toimeen tilanteessa kumpuavien negatiivisten tuntemusten kanssa. Tavoitteena on luoda pelkojen aiheuttajaa kohtaan uudenlainen käyttäytymismalli, joka korvaa aiemman, ahdistusta ja pelkoa aiheuttaneen suhteen. [Krijn *et al.*, 2004]

Vuodesta 1992 alkaen esillä on ollut virtuaalitodellisuudessa tapahtuva altistumishoito, josta käytetään lyhennettä *VRET* (virtual reality exposure thera-

py). VRET saattaa tarjota käyttökelpoisen vaihtoehdon todellisissa tilanteissa ja ympäristöissä tapahtuvalle altistukselle. VRET:llä arvellaan olevan todelliseen altistukseen nähden useita suotuisia puolia, jotka voisivat parantaa potilaiden hoitomyöntyvyyttä ja saavutettuja tuloksia. Esimerkiksi lentopelon hoito todellisen lentomatkustuksen avulla on paitsi kallista myös heikosti kontrolloitavissa, sillä olosuhteet ovat usein arvaamattomia. Lisäksi tosielämässä esimerkiksi agorafobian eli avarien paikkojen pelon hoito altistuksen avulla voi olla erittäin haasteellista, sillä potilaita vaivaavat usein myös samanaikainen paniikkihäiriö sekä pelko itsekontrollin julkisesta menettämisestä.

VRET mahdollistaa altistuksen tarkoin säädellyssä ja turvallisessa ympäristössä. Tilanteet ovat helposti toistettavissa, ja lisäksi niihin voidaan nopeasti tehdä haluttuja muutoksia. [Bush, 2008]

Toimiakseen tarkoituksenmukaisesti VRET:n on ensinnäkin kyettävä luomaan kokemus todellisesta läsnäolosta keinotekoisessa ympäristössä sen sijaan, että potilas vain kokisi katselevansa sitä esimerkiksi videon tapaan. Toiseksi VRET:n on kyettävä herättämään potilaassa samankaltaisia negatiivisia tunnereaktioita kuin todellisenkin altistuksen, sillä muuten altistumishoito ei toimi tarkoituksenmukaisesti. Kolmanneksi VRET:n avulla saavutettujen käyttäytymismallien on luonnollisesti säilyttävä myös tosielämän ärsykkeiden alaisina. [Krijn *et al.*, 2004]

6.1. Käytännön kokemuksia VRET-menetelmästä

Tässä kohdassa esittelen, miten virtuaalitodellisuusympäristöjä on hyödynnetty erinäisten fobioiden hoidossa sekä millaisia tuloksia niiden avulla on saavutettu. Tarkasteltavia fobioita ovat agorafobia, akrofobia, araknofobia sekä lentopelko. Käsittelen kunkin fobian erikseen omassa kappaleessaan.

Agorafobian hoidossa virtuaalisesti koettavia ympäristöjä ovat olleet muun muassa ostoskeskus ja kuumailmapallolento, jotka pyrkivät jäljittelemään oireita aiheuttavia ympäristöjä [Krijn et al., 2004]. Lisäksi on käytetty ympäristöjä, joissa potilaan tehtävänä on suorittaa tavanomaisia arkisia askareita, kuten käydä ostoksilla [Bush, 2008]. Tulokset ovat olleet jokseenkin kannustavia. Niiden perusteella on kuitenkin vielä liian varhaista tehdä kovin kauaskantoisia johtopäätöksiä, sillä ongelmina ovat olleet muun muassa riittämätön läsnäolon kokemus, vähäiset osallistujamäärät sekä epävarmuus suotuisten tulosten pysyvyydestä.

Akrofobian hoidon yhteydessä virtuaalisina ympäristöinä on käytetty korkeaa tornia sekä lasiseinäistä hissiä. Keinotekoisten näkö- ja kuuloärsykkeiden lisäksi tutkimuksissa on käytetty myös konkreettista elementtiä, kuten pitelyn mahdollistavaa kaidetta. Saavutetut tulokset ovat olleet hyviä, ja ne ovat myös

olleet verrattavissa todellisen altistuksen avulla saatuihin tuloksiin. Tulosten on havaittu säilyneen ainakin kuuden kuukauden seurantajakson ajan. Valtaosa VRET-tutkimuksista onkin keskittynyt nimenomaan akrofobiaan. [Bush, 2008; Krijn et al., 2004]

Hoidettaessa araknofobiaa potilaiden on ollut mahdollista tarkastella virtuaalista hämähäkkiä vaihtelevilta etäisyyksiltä, minkä lisäksi myös hämähäkin koskettamista on simuloitu tuntopalautteen avulla. Kokeissa saavutettujen tulosten perusteella VRET voisi olla käyttökelpoinen hoitomuoto araknofobiaan. Asian perusteellisempi tutkimus on kuitenkin tarpeen, jotta hoidon tehosta voitaisiin vakuuttautua. [Krijn et al., 2004]

Lentopelko on erityisen oleellinen VRET-hoidon käyttöaihe, sillä se on todelliseen altistukseen nähden huomattavasti kustannustehokkaampaa. Lisäksi sen avulla on vaivatonta simuloida vaihtelevia sääolosuhteita sekä lennon eri vaiheita, kuten nousua ja laskeutumista. Toteutettuja virtuaaliympäristöjä ovat olleet esimerkiksi helikopterilento sekä asteittain toteutettu kokonainen lentomatka, joka on kattanut matkustamisen vaiheet aina lentokentälle saapumisesta laskeutumiseen asti. Joissakin tapauksissa läsnäolon kokemusta on lisäksi tehostettu luomalla lentämiselle ominaista tärinää. VRET vaikuttaa lupaavalta lentopelon hoitomuodolta. Lisätutkimuksia kuitenkin tarvitaan, ja erityisen tärkeää olisi vertailla VRET-hoitoa ja käytännön altistukseen tukeutuvia terapiamenetelmiä, joiden toiminnallinen sisältö olisi sinänsä sama. [Krijn et al., 2004]

6.2. Päätelmiä VRET-menetelmästä

Tutkimustulosten valossa VRET on osoittautunut hoitamatta jättämistä tehokkaammaksi etenkin akrofobian, lentopelon sekä araknofobian lievittämisessä. Tutkimuksissa ei kuitenkaan ole erityisesti syvennetty siihen, kuinka vaikuttavia tuloksia yksinään VRET:n avulla on saavutettavissa verrattuna muihin hoitomuotoihin, eikä myöskään huolellisesti järjestettyjä, perinpohjaisia tutkimuksia ole vielä tehty tarpeeksi.

Yhtenä ongelmana on ollut, etteivät käytetyt virtuaaliympäristöt ole kaikissa tapauksissa kyenneet luomaan riittävän voimakasta läsnäolon tunnetta, joka on ensiarvoisen tärkeää VRET:n vaikuttavuuden kannalta. Toistaiseksi VRET vaikuttaakin tässä suhteessa sijoittuvan tavallaan tavanomaisen mielikuvaharjoittelun ja konkreettisen altistuksen välimaastoon.

VRET:n avulla ei välttämättä pystytä saavuttamaan todellisen altistuksen veroisia hoitotuloksia, mutta helpottamalla tilanteisiin eläytymistä pelkkiin mielikuvaharjoituksiin verrattuna se voi kuitenkin tarjota terapeuttien hyödynnettäväksi käyttökelpoisen ja huomionarvoisen hoitokeinon. [Krijn *et al.*, 2004]

7. Virtuaalitodellisuuden käyttö skitsofrenian hoidossa

Skitsofrenia on vakava mielenterveyden häiriö, joka alkaa useimmiten nuorella aikuisiällä. Sairaudelle ominaisia oireita ovat erilaiset harhat, hajanainen puhe ja käytös sekä niin sanotut negatiiviset oireet. Negatiiviset oireet voidaan yleisesti käsittää puutosoireiksi, kuten tunneilmaisujen ja motivaation voimakkaaksi latistumiseksi. Skitsofrenian esiintyvyys väestössä on noin yksi prosentti, ja nykyisistä hoitomenetelmistä huolimatta 20-40 prosentilla sairastuneista etenkin negatiiviset oireet jatkuvat vuosia tai jopa vuosikymmeniä. [Huttunen, 2013b].

Skitsofreniapotilaiden toimintakykyä arki- ja työelämässä haittaavat usein motivaation puute sekä heikentyneet sosiaaliset taidot. Toimintaterapian avulla voidaan saavuttaa myönteisiä tuloksia näiden kykyjen kohentamisessa, mutta muun muassa motivaation puute, sosiaalinen ahdistuneisuus ja köyhtynyt ilmaisu voivat merkittävästi vaikeuttaa hoidon onnistumista. Tutkimus onkin toistaiseksi keskittynyt pääasiassa vuorovaikutustaitojen valmentamiseen [Park et al., 2011; Tsang and Man, 2013] sekä niissä ilmenevien vajavaisuuksien tunnistamiseen [Dyck *et al*., 2010].

7.1. Käytännön kokemuksia

Tässä kohdassa keskityn tarkastelemaan virtuaalitodellisuuden avulla toteutettavia kuntouttavia hoitoja, joiden tarkoituksena on voimistaa skitsofreniaa sairastavien potilaiden motivaatiota osallistua toimintaterapiaan ja parantaa yleisiä kognitiivisia sekä sosiaalisia taitoja, kuten äänenkäyttöä, nonverbaalista viestintää sekä keskustelutaitoja. Näiden taitojen harjaannuttamisen tavoitteena on, että potilaat pystyisivät toimimaan yhteiskunnassa mahdollisimman normaalisti osallistuen työntekoon sekä tavanomaiseen sosiaaliseen kanssakäymiseen.

Parkin ja muiden [2011] tutkimuksessa potilaat suorittivat erilaisia keskusteluharjoituksia virtuaalitodellisuudessa. Terapia oli sisällöltään samankaltaista kuin tavanomaisestikin, ja koostui erilaisista tilanneharjoituksista. Näitä olivat esimerkiksi keskustelun aloittaminen ja ylläpitäminen, vaatimusten esittäminen keskustelukumppanille ja vastaavasti tämän vaatimusten torjuminen sekä myönteisten ja kielteisten tunteiden esittäminen. Toteutettaessa terapiaa virtuaalisina keskustelukumppaneina olivat todellisten henkilöiden sijasta virtuaalihahmot, ja myös potilas itse ohjasi virtuaalihahmoa luodussa ympäristössä.

Tulosten perusteella potilaat osoittivat enemmän kiinnostusta virtuaalitodellisuudessa suoritettuihin vuorovaikutusharjoituksiin kuin tavanomaisiin rooliharjoituksiin. Tämä selittynee ainakin osittain sillä, että virtuaalimaailmassa potilaat tunsivat olonsa turvallisemmaksi ja saattoivat siten osallistua harjoi-

tuksiin vapautuneemmin ja ilman pelkoa virheiden tekemisestä. Virtuaalitodellisuusharjoitukset paransivat etenkin potilaiden keskustelutaitoja, kun taas tavanomaiset harjoitukset olivat tehokkaampia nonverbaalisen viestinnän harjaannuttamisessa. On kuitenkin syytä huomioida, että käytetty laitteisto saattaa hankaloittaa potilaiden suoriutumisen arviointia. Esimerkiksi visiirinäyttö ymmärrettävästi vaikeuttaa silmänliikkeiden ja kasvonilmeiden seurantaa. [Park *et al.*, 2008]

Erilaisten kognitiivisten häiriöiden esiintyminen on skitsofreniapotilailla tavallista. Tällaisia tiedon prosessointiin liittyviä ongelmia voivat olla muun muassa tarkkaavaisuuden, muistin sekä päättely- ja päätöksentekokyvyn heikentyminen. Kognitiivisten ongelmien vuoksi onnistunut paluu työelämään saattaa olla vaikeaa, ja niinpä virtuaalitodellisuutta onkin kokeiltu näistä vaikeuksista kärsivien potilaiden kuntoutuksessa.

Tsanging ja Manin [2013] kliinisessä tutkimuksessa pyrittiin selvittämään, millaisia tuloksia voidaan saavuutta virtuaalitodellisuuteen pohjautuvalla työvalmennuksella, josta käytetään lyhennettä *VRVTS* (virtual reality-based vocational training system). Potilaille luotiin skenaario vaatekaupasta, jossa heidän tuli toimia myyjän roolissa suorittaen samankaltaisia tehtäviä kuin todellisessakin tilanteessa. Näitä olivat esimerkiksi vaatteiden järjestely ja asiakaspalvelu vaihtelevine ongelmatilanteineen. Tällaisessa toimenkuvassa menestyminen edellyttää monipuolisia vuorovaikutustaitoja sekä myös järjestelmällisyyttä ja ongelmanratkaisukykyjä.

Potilaiden palautteen mukaan VRVTS oli mielenkiintoisempaa ja myös hyödyllisempää kuin tavanomainen, terapeutin johdolla tapahtuva ryhmäharjoittelu. Potilaat kokivat, ettei heidän tarvinnut olla huolissaan mahdollisista virheistään virtuaalitodellisuudessa, ja etenkin tämän katsotaan myötävaikuttaneen VRVTS:n koettuun miellyttävyyteen.

Tutkimuksen tulokset olivat yleisesti ottaen lupaavia, ja eräissä kognitiivisissa testeissä havaittiin tiedollisten toimintojen kohentumista, joskaan muutokset eivät olleet tilastollisesti merkitseviä tavanomaiseen ryhmäohjaukseen verrattuna. VRVTS:ään osallistuneet potilaat menestyivät tositilannetta mukailevassa harjoituksessa silti paremmin kuin perinteiseen terapiaan osallistuneet verrokit, mikä saattaa viitata harjoittelun avulla hankittujen taitojen parempaan yleistyvyyteen. Tutkijat kuitenkin huomauttavat, että potilaat olivat laitoshoidossa, eikä todellista seurantaa heidän työllistymisestään ja sopeutumisestaan yhteiskuntaan siten voitu tehdä. [Tsang and Man, 2013]

Skitsofreniaa sairastavilla henkilöillä on usein vaikeuksia muodostaa mielekkäitä tulkintoja toisten ihmisten tunnetiloista heidän kasvonilmeittensä perusteella, mikä voi osaltaan olla esteenä normaalille sosiaaliselle toiminnalle.

Ilmeiden ja niiden kautta välittyvien nonverbaalisten viestien tunnistamisen opettelu rooliharjoitusten avulla on kuitenkin vaivalloista. Lisäksi se vie paljon aikaa ja vaatii tiivistä sitoutumista niin potilaalta kuin henkilökunnaltakin. Näistä seikoista johtuen on koettu mielekkääksi selvittää, voisivatko virtuaaliset hahmot korvata terapeutin kanssa kasvokkain tapahtuvat harjoitukset.

Tutkimusten perusteella on voitu päätellä, että potilailla ilmenee virtuaalisten kasvojen kohdalla vastaavia ilmeiden tunnistamis- ja tulkintaongelmia kuin elävien ihmisten tapauksessa. Havainto tukee ajatusta siitä, että virtuaaliset ilmeet saattaisivat olla hyödyllisiä tunnetilojen tunnistamisen ja tulkitsemisen harjaannuttamisessa. Potilaat eivät välttämättä tulkitse virtuaalisia kasvoja yhtä pelottaviksi ja uhkaaviksi kuin todellisia, joten ne voivat tarjota oivallisen mahdollisuuden harjoittaa sosiaalisia taitoja ilman häiritsevän voimakkaita ärsykkeitä. Keinotekoisia kasvonilmeitä on myös mahdollista muunnella hyvin hienovaraisesti, mikä mahdollistaa harjoitusten haasteellisuuden asteittaisen säätelyn erittäin tarkasti. [Dyck *et al.*, 2010]

Tietokoneiden avulla generoidut kasvot on myönnettävä jossakin määrin epärealistisiksi ja pelimäisiksi, eikä niillä välttämättä voida helposti ja yksiselitteisesti esittää kuin selkeimmät perusilmeet. Kuitenkin juuri tähän pohjautuen virtuaalisilla ilmeillä voisi olla myös diagnostista merkitystä, sillä kyvyttömyys tunnistaa niiden avulla esitettyjä perustunteita voidaan tulkita osoitukseksi merkittävästä vuorovaikutustaitojen alentumisesta. [Dyck *et al.*, 2010]

7.2. Turvallisuusnäkökulmia

Poikkeavat käsitykset todellisuudesta sekä toisten ihmisten motiiveista ja aikeista ovat erittäin tunnusomaisia piirteitä skitsofreniapotilaiden oireistossa. Koska potilaiden todellisuudentaju saattaa siis olla voimakkaastikin vääristynyt ja esimerkiksi erilaisten sairaalloisten epäluulojen sävyttämä, heidän kohdallaan on kiinnitettävä erityistä huomioita virtuaalitodellisuuskokemusten turvallisuuteen. Seuraukset perussairauden hoitotasapainon kannalta voisivat olla katastrofaaliset, jos potilas ei esimerkiksi pystyisikään erottamaan keinotekoisessa ja todellisessa ympäristössä tapahtuneita asioita toisistaan, vaan ajautuisi psykoottiseen kriisiin hoitomenetelmän vaikutuksesta.

Virtuaalitodellisuuden turvallisuutta tässä potilasryhmässä on tutkittu kokein, joissa syyllistävistä ja vainoharhaisista ajatushäiriöistä kärsivät henkilöt matkustivat muutaman minuutin simuloidussa metrossa.

Mahdollisimman perinpohjaisen eläytymisen luomiseksi simulaatio luotiin käyttäen niin sanottua CAVE-järjestelmää, joka rakentuu lattiatasosta sekä kolmesta pystysuuntaisesta tasosta. Virtuaaliset näkymät heijastetaan näille pinnoille. Tilan puitteissa on mahdollista liikkua luonnolliseen tapaan, jolloin

kokemus läsnäolosta kyseisessä tilassa korostuu avaruudellisten ulottuvuuksien ansiosta. [Fornells-Ambrojo *et al.*, 2008]

Muina matkustajina metrovaunussa oli tietokonein luotuja hahmoja eli avataria. Hahmot suunniteltiin käyttäytymään neutraalisti, eivätkä ne mukauttaneet toimintaansa testihenkilön käyttäytymisen perusteella.

Tutkimuksen perusteella virtuaalitodellisuuden käyttö on turvallista myös syyllistävistä harhaluuloista kärsivien potilaiden kohdalla, vaikkakin heillä havaittiin lievää taipumusta kokea ympäristö ja kanssamatkustajina olleet avatarit vihamielisempinä kuin terveillä verrokeilla. Mahdollisuus herättää potilasryhmälle tunnusomaisia ajatushäiriöitä hallitusti virtuaalitodellisuudessa saattaa myös tarjota uudenlaisen hoitomuodon. Virtuaalitodellisuutta voitaisiin kenties soveltaa osana skitsofreniapotilaiden käyttäytymisterapiaa, joka tukisi uudenlaisten ajatusmallien omaksumista perustavanlaatuisten oireiden osalta. [Fornells-Ambrojo *et al.*, 2008]

8. Virtuaalitodellisuuden käyttö ruumiinkuvan häiriöiden hoidossa

Ruumiinkuvalla tarkoitetaan yksilön mielikuvaa siitä, millainen hänen kehonsa on. Ruumiinkuva on monimutkainen ja syvästi henkilökohtainen käsite, johon kytkeytyy erilaisia tiedollisia ja tunnepitoisia kokemuksia. Niinpä ruumiinkuvalla katsotaan olevan yhteyksiä muun muassa syömiskäyttäytymiseen, itsetuntoon, fyysiseen aktiivisuuteen ja seksuaaliseen käyttäytymiseen. Tässä luvussa keskityn käsittelemään ruumiinkuvan häiriöiden hoitamista syömishäiriöiden kannalta.

Tunnetuimpien syömishäiriöiden, anoreksian eli laihuushäiriön ja bulimian eli ahmimishäiriön, ajatellaan aiheutuvan vääristyneestä ruumiinkuvasta. Molempien sairauksien taustalla on perusteeton käsitys omasta lihavuudesta, minkä seurauksena potilaat ajautuvat äärimmäisyyksiin tavoitellessaan ihannevartaloaan. Laihuushäiriölle on ominaista pakonomainen laihduttaminen niin kuntoilun kuin ruuan välttelemisenkin avulla, jolloin seurauksena voi olla henkeä uhkaava nälkiintyminen [Huttunen, 2008]. Bulimiaan sitä vastoin liittyy toistuvaa pakonomaista ahmimista, jota seuraa usein pyrkimys ehkäistä painonnousua esimerkiksi tahallisesti oksentamalla.

Ruumiinkuvahäiriöistä kärsivillä on kehostaan tyypillisesti erittäin vankka ja itsepintainen mielikuva, ja niinpä sen muovaaminen psykoterapeuttisin keinoin voi olla hyvin vaikeaa. Virtuaalitodellisuuden on arveltu soveltuvan tähän tarkoitukseen erityisesti siksi, että sen avulla käyttäjä voi itse kokea ristiriidan, joka vallitsee hänen todellisen ruumiinrakenteensa ja oman ruumiinkuvansa välillä.

Hoidettaessa ruumiinkuvan häiriöistä kärsiviä syömishäiriöisiä potilaita virtuaalitodellisuusympäristö on usein rakennettu niin, että sen avulla voidaan puuttua sekä ruumiinkuvaan että syömiskäyttäytymiseen.

Käytännössä tämä on toteutettu esimerkiksi siten, että virtuaalisessa keittiössä tarkkaillaan potilaan tunnereaktioita ja rohkaistaan häntä omaksumaan normaali suhde ravitsemukseen. Ruumiinkuvaan pyritään puolestaan vaikuttamaan muun muassa siten, että käyttäjän on virtuaaliympäristössä kohdattava mittojensa mukainen hahmo. Ajatuksena on, että käyttäjä tulisi voimakkaammin tietoiseksi omasta kehonkuvastaan ja sen suhteesta todellisuuteen. Tutkijat esittävät tämän perustuvan siihen, että käyttäjän asentoaistin antama palaute poikkeaa virtuaalitodellisuuden tuottamista ärsykkeistä. Tällöin potilas on tavallaan pakotettu käsittelemään suhdettaan ruumiiseensa tietoisesti. [Ferrer-García and Gutíerrez-Maldonado, 2012]

Tutkimustulosten perusteella potilaat pitävät virtuaalitodellisuudessa kehostaan saamaansa palautetta puolueettomana. Näkemys puolueettomasta arvioijasta vähentää palautteen herättämää epäluuloisuutta, ja tämän myötä heidän on helpompi hyväksyä ajatus ruuminkuvansa vääristyneisyydestä. Lisäksi on raportoitu merkittävää ja tavanomaisten hoitomuotojen veroista lieventymistä tyypillisessä oireistossa, kuten hoikkuuden tavoittelussa, sosiaalisessa epävarmuudessa ja täydellisyyden tavoittelussa. Tulosten on havaittu paitsi säilyvän vuoden ajan myös parantuneen entisestään ajan myötä, mikä viittaa perustavanlaatuiseen ruumiinkuvan muutokseen. Hyvistä tuloksista huolimatta tutkijat korostavat, että hoitomuoto vaatii jatkotutkimuksia. Heidän mukaansa jatkossa tulisi laajempien ja kontrolloitujen tutkimusten avulla vertailla hoidon tehoa muihin menetelmiin nähden sekä asettaa aiemmat tutkimukset meta-analyysin alaisiksi. [Ferrer-García and Gutíerrez-Maldonado, 2012]

9. Yhteenveto ja päätelmät

Tässä tutkielmassa tarkoituksenani oli selvittää, mitä virtuaalitodellisuudella tarkoitetaan lääketieteellisessä tutkimuksessa, mitä tutkimuseettisiä ongelmia sen käyttöön liittyy sekä millaisissa käyttökohteissa sen soveltuvuutta on tutkittu.

Pyrkimyksenäni oli esittää eri käyttökohteissa saavutetut tulokset puolueettomasti siten, että myönteisten tulosten ohella huomioiduiksi tulivat tasapuolisesti myös tutkimuksissa todetut puutteet. Tulosten luotettavuuden kohentaminen ja tutkimuseettisten näkökulmien asianmukainen huomioiminen ovat eräitä keskeisimpiä seikkoja, jotka aihepiirin jatkotutkimuksessa on huomioitava. Eettisten näkökulmien merkitys korostuu entisestään sovellettaessa kokeellista hoitomuotoa vajaakykyisiin potilaisiin, kuten skitsofreenikoihin.

Pidän virtuaalitodellisuuden terapeuttisten sovellusten tutkimusta hyvin tärkeänä, sillä ne saattavat tulevaisuudessa osoittautua tehokkaiksi monissa vaikeahoitoisissa sairauksissa. Tutkimus edellyttää monialaista yhteistyötä, ja siksi myös kiinnostusta on onnistuttava herättämään laajalti eri alojen asiantuntijoiden keskuudessa. Uusien menetelmien tapauksessa on aina olemassa riski siitä, että innostus hiipuu eikä niiden täyttä potentiaalia saavuteta.

Vaihtoehtoinen lähestymistapa aihepiiriin olisi ollut esimerkiksi virtuaalitodellisuuden käsittely teknisemmästä näkökulmasta, jolloin varsinaisen vuorovaikutteisen teknologian rooli olisi voinut olla suurempi.

Lisäksi olisin voinut laajentaa tutkielmaa esittelemällä tutkimuksia myös muista käyttökohteista, kuten aivohalvauspotilaiden kuntoutuksesta, traumaperäisestä stressireaktiosta tai addiktiokäyttäytymiseen puuttumisesta. Näissä tutkimuksissa ei kuitenkaan nähdäkseni ole menetelmien tai tulosten osalta ilmennyt sellaisia erityisiä seikkoja, joita ei olisi jo huomioitu.

Viiteluettelo

[Bush, 2008] Jimmy Bush, Viability of virtual reality exposure therapy as a treatment alternative. *Comput. Hum. Behavior.* **24** (2008), 1032-1040.

[Dyck *et al*., 2010] Miriam Dyck, Maren Winbeck, Susanne Leiberg, Yuhan Chen and Klaus Mathiak, Virtual faces as a tool to study emotion recognition deficits in schizophrenia. *Psychiatry Res.* **179** (2010) 247-252.

[Ferrer-García and Gutíerrez-Maldonado, 2012] Marta Ferrer-García and José Gutíerrez-Maldonado, The use of virtual reality in the study, assesment, and treatment of body image in eating disorders and nonclinical samples: A review of the literature. *Body Image* **9** (2012) 1-11.

[Fornells-Ambrojo et al., 2008] Miriam Fornells-Ambrojo, Chris Barker, David Swapp, Mel Slater, Angus Antley and Daniel Freeman, Virtual reality and persecutory delusions: Safety and feasibility. *Schizoph. Res*. **104**, 1-3 (Sept. 2008), 228-236.

[Hoffmann *et al.*, 2000] Hunter G. Hoffmann, Jason N. Doctor, David R. Patterson, Gretchen J. Carrougher and Thomas A. Furness III, Virtual reality as an adjunctive pain control during burn wound care in adolescent patients. *Pain* **85**, 1-2 (Mar. 2000), 305-309.

[Huttunen, 2013a] Matti Huttunen, Määrekohtainen pelko (fobia), http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00394. Viitattu 19.12.2013.

[Huttunen, 2013b] Matti Huttunen, Skitsofrenia (F20), http://www.terveyskirjasto.fi/terveyskirjasto/tk.koti?p_artikkeli=dlk00148. Viitattu 19.12.2013.

[Huttunen, 2008] Matti O. Huttunen, *Lääkkeet mielen hoidossa*. Kustannus Oy Duodecim, 2008.

[Kipping *et al.*, 2012] Belinda Kipping, Sylvia Rodger, Kate Miller and Roy M. Kimble, Virtual reality for acute pain reduction in adolescents undergoing burn wound care: A prospective randomized controlled trial. *Burns* **38,** 5 (Aug. 2012), 650-657.

[Koulu ja Tuomisto, 2007] *Farmakologia ja Toksikologia*. Kustannusosakeyhtiö Medicina, 2007.

[Krijn *et al.,* 2004] M Krijn, P.M.G. Emmelkamp, R.P Olafsson and R Biemond, Virtual reality exposure therapy of anxiety disorders: A review. *Clin. Psychol. Rev.* **24**, 3 (July 2004), 259-281.

[Levac and Galvin, 2013] Danielle E. Levac and Jane Galvin, When is virtual reality "therapy"?. *Arch. Phys. Med. Rehab.* **94** (2013), 795-798.

[Markus *et al.*, 2009] L.A. Markus, K.E. Willems, C.C. Maruna, C.L. Schmitz, T.A. Pellino and J.R. Wish, Virtual reality: feasibility of implementation in a regional burn center. *Burns* **35** (2009), 967-969.

[Park *et al.*, 2011] Kyung-Min Park, Jeonghun Ku, Soo-Hee Choi, Hee-Jeiong Jang, Ji-Yeon Park, Sun I. Kim and Jae-Jin Kim, A virtual reality application in role-plays of social skills training for schizophrenia: A randomized, controlled trial. *Psychiatry Res.* **189**, 2 (Sept. 2011), 166-172.

[Schmitt *et al.*, 2011] Yuko S. Schmitt, Hunter G. Hoffmann, David K. Blough, David R. Patterson, Mark P. Jansen, Maryam Soltani, Gretchen J. Carrougher, Dana Nakamura and Sam R. Sharar, A randomized, controlled trial of immersive virtual reality analgesia, during physical therapy for pediatric burns. *Burns* **37** (2011), 61-68.

[Sharar et al., 2007] Sam R. Sharar, Gretchen J. Carrougher, Dana Nakamura, Hunter G. Hoffmann, David K. Blough and David R. Patterson, Factors influencing the efficacy of virtual reality distraction analgesia during post-burn physical therapy: preliminary results from 3 ongoing studies. *Arch. Phys. Med. Rehabil.* **88**, 12 (Dec. 2007), 43-49.

[Tsang and Man, 2013] Mayie M.Y. Tsang and David W.K. Man, A virtual reality-based vocational training system (VRVTS) for people with schizophrenia in vocational rehabilitation. *Schizoph. Res.* **144** (2013), 51-62.

[Whalley, 1995] L.J. Whalley, Ethical issues in the application of virtual reality to medicine. *Comput. Biol. Med.* **25**, 2 (Mar. 1995), 107-114.

[Yoh, 2001] Meyung-Sook Yoh, The reality of virtual reality. In: *Proc. of the Seventh International Conference on Virtual Systems and Multimedia (VSMM'01)*, 666-674.

NoSQL-tietokannat

Ville Toni

Tiivistelmä.
Yhä useammat organisaatiot tallentavat dataa NoSQL-tietokantoihin. Tämä tutkielma tarkastelee NoSQL-tietokantojen ja perinteisten relaatiotietokantojen ominaispiirteitä sekä sitä, kuinka hyvin ne ovat laajennettavissa. Eri tietokantojen käyttötarkoituksista annetaan myös esimerkkejä.

Avainsanat ja -sanonnat: tietokannat, relaatiotietokannat, NoSQL, SQL
CR-luokat: H.2.4

1. Johdanto

Erikokoiset ja erityyppiset tietokannat ovat läsnä lähes jokaisen ihmisen elämässä. Ilman niitä ja niitä hyödyntäviä tietojärjestelmiä modernin yhteiskunnan ylläpitäminen olisi lähestulkoon mahdotonta. Tietokanta on yksinkertaisesti kokoelma, johon on tallennettu samankaltaisia, toisiinsa liittyviä *tietoja* (dataa). Esimerkiksi ihmisten nimet, puhelinnumerot ja osoitteet yhteen kerättynä muodostavat tietokannan – oli tallennusvälineenä sitten osoitekirja, muistivihko tai tietokoneen kiintolevy. Tietokannan tulee lisäksi olla kosketuksissa reaalimaailman tapahtumiin ja sillä tulee olla määriteltynä jokin aktiivinen käyttäjäryhmä, joka on kiinnostunut sen sisällöstä. Tietokantaan tallennetaan myös *metatietoa* (meta data), eli tietoa tiedosta. Metatiedon avulla voidaan pitää esimerkiksi kirjaa siitä, kuka tiedon on lisännyt ja koska. [Elmasri and Navathe, 2010]

Erityyppisiä tietokantoja on kehitetty jo 1960-luvulta lähtien. Relaatiotietomalliin perustuvat tietokannat [Codd, 1970] ovat olleet kuitenkin näistä kaikkein suosituimpia, sillä ne ovat tarjonneet korkean suorituskyvyn lisäksi yksinkertaisen matemaattisen mallin tiedon käsittelyyn. Lisäksi niiden toimintalogiikkaan kuuluu olennaisena osana tiedon *eheys* (consistency) ja yhteensopivuus eri laitteistojen kesken. Relaatiotietokantoja hallitaan *tietokannan hallintajärjestelmillä* (relational database management systems), joista yleisimpiä ovat Oracle, MySQL, SQL Server, IBM DB2 ja PostgreSQL. Viimeisimmän Gartnerin vuosiraportin mukaan Oracle on relaatiotietokantojen markkinajohtaja 48 % markkinaosuudellaan [Graham et al., 2013].

Relaatiotietokannan käsittelyä varten on kehitetty standardoitu kyselykieli SQL (Structured Query Language), jonka avulla pystyy hallitsemaan tietokantojen kaavioita ja käyttöoikeuksia sekä käsittelemään tietokantojen sisältämiä tietoja. Kielen standardoimisesta huolimatta eri relaatiotietokannoilla on pieniä eroavaisuuksia sen rakenteessa ja termistössä, mutta kaikki järjestelmät tukevat SQL:n yleisimpiä ko-

mentoja. Nykyinen SQL:n standardi on seitsemäs uudisversio vuodelta 2011. Vuosien varrella standardiin on lisätty uusia tietotyyppejä, ehtolauseet ja muun muassa *ajanjaksojen vertailukeinoja* (temporal predicates). Muitakin kyselykieliä, esimerkiksi QBE ja Quel, on kehitetty, mutta SQL on ylivoimaisesti käytetyin.

Relaatiotietokantoja kohtaan on myös esitetty kritiikkiä. Niiden rajat tulevat vastaan erityisesti *laajennettavuudessa* (scalability), *saatavuudessa* (availability) ja *virheensietokyvyssä* (fault-tolerance) [Sutinen, 2010]. Lisäksi relaatiotietokantojen kaavioiden suunnitteluun vaaditaan asiantuntemusta ja virheellisellä suunnittelulla tiedon eheys saattaa kärsiä. Relaatiotietokannan rakenne saattaa olla monimutkainen jo pienenkin sovelluksen kohdalla, eikä sen hallinta välttämättä ole helppoa. Tässä tutkielmassa laajennettavuudella tarkoitetaan suorituskyvyn muutosta järjestelmän kasvaessa, saatavuudella viitataan järjestelmän pystyssä pysymiseen tiettynä ajanjaksona ja virheensietokyky mittaa, kuinka hyvin järjestelmä pystyy jatkamaan toimintaansa virhetilanteissa.

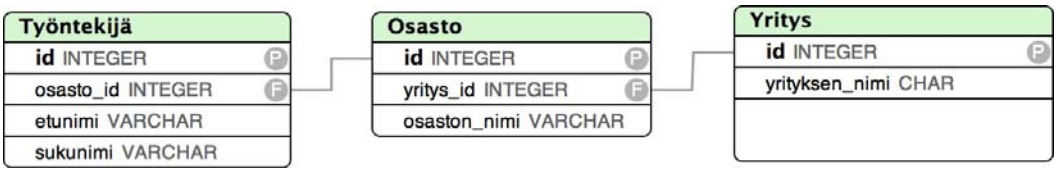
Tallennetun tiedon räjähdysmäinen kasvu 2000-luvulla ja teknologian kehittyminen ovat tarjonneet uusia haasteita ja toisaalta ratkaisuja tietokantojen maailmassa. Yhä useammin palveluiden ja sovellusten kasvaessa tietokannat osoittautuvat pullonkauloiksi. Relaatiotietokannat kehitettiin alunperin yhden tietokoneen järjestelmiä varten ja aivan eri luokan palvelinkapasiteetille kuin mitä tänä päivänä on mahdollista hankkia. Esimerkiksi prosessoritehon, muistin ja varsinkin tallennustilan kustannukset ovat vähentyneet merkittävästi relaatiotietokantojen alkuaikoihin nähden.

Haasteisiin ovat vastanneet useat ns. *NoSQL*-tietokannat (Not only SQL), joiden idea esiteltiin ensimmäisen kerran vuonna 1998 ja varsinainen läpimurto nähtiin vuonna 2009. Näiden uusien, jo tuotantokäytössä kykynsä osoittaneiden tietokantojen pääperiaatteita ovat helppo laajennettavuus, korkea suorituskyky suurtenkin tietomäärien kohdalla ja tietorakenteiden vapaampi muoto. Tässä tutkielmassa vertailen relaatiotietokantoja ja NoSQL-tietokantoja. Teen myös katsauksen suosituimpiin NoSQL-tietokantaohjelmistoihin ja keskityn erityisesti MongoDB-ohjelmistoon. Lisäksi arvioin sekä relaatio- että NoSQL-tietokantojen soveltuvuutta erilaisissa tilanteissa.

2. Relaatiotietokannat

Relaatiotietokanta koostuu relaatioista eli tauluista, jotka sisältävät ennalta kategorisoitua tietoa. Jokainen taulu sisältää yhden tai useamman tiedon kategorian eli sarakkeen. Taulujen rakennetta ei ole tarkoitus muuttaa jälkikäteen, joskin se on mahdollista. Itse sisältö voi kuitenkin muuttua useinkin. Tauluihin tallennetaan tietoa riveinä, joilla kullakin on yhdestä tai useammasta sarakkeen arvosta muodostuva pääavain, joka erottaa eri rivit toisistaan. Usein tietokannan rakenteessa eli skeemassa on tarve yhdistää erilaisia tauluja kuten työntekijä, osasto ja yritys. Kuvassa 1 taulujen

liittäminen toisiinsa tapahtuu *taululiitoksilla* (joins), jotka ovat riippuvaisia viitattavan taulun *pääavaimesta* (primary key) ja viittaavan taulun *vierasavaimesta* (foreign key). Vierasavainten ja liitosoperaatioiden avulla tietokantaohjelmisto muodostaa kolmesta erillisestä taulusta yhden uuden taulun, joka sisältää kaikkien kolmen taulun tiedot. Kyselyä voi ja kannattaakin rajata siten, ettei tarpeettomia sarakkeita käytetä. [Elmasri and Navathe, 2010]



Kuva 1. Taululiitokset

Relaatiotietokannat toimivat parhaiten hyvin organisoidun tiedon kuten myyntilukujen tallennukseen. Toisaalta taas huonosti strukturoidun datan kuten kuvien tai tekstidokumenttien tallennus relaatiotietokantaan ei ole optimaalista [Leavitt, 2010], joskin se on mahdollista erilaisten binääritietotyyppien avulla. Jopa yhteystietojen tallentaminen voi olla haastavaa, mikäli tallennettavalla kohteella on esimerkiksi monta puhelinumeroa. Tällöin voi olla perusteltua luoda oma taulu puhelinnumeroille ja yhdistää se muut yhteystiedot sisältävään tauluun vierasavaimen avulla. Yhteen sarakkeeseen voi tallentaa useita arvoja esimerkiksi pilkulla eroteltuna, mutta se sotii tiedon eheyttä [Codd, 1970] vastaan ja hankaloittaa tiedon käsittelyä.

Relaatiotietokantoihin kuuluu olennaisena osana transaktiot ja niiden ACID-ominaisuudet (Atomicity, Consistency, Isolation, Durability). Transaktio on joukko tapahtumia, joilla suoritetaan jokin tehtävä, esimerkiksi tilisiirto henkilöltä A henkilölle B. ACID-ominaisuuksien toteuttaminen edellyttää, että kaikki transaktiot joko tapahtuvat kokonaan tai eivät ollenkaan. Lisäksi transaktioiden tulisi muuttaa tietokannan tilaa siten, ettei ristiriitoja synny ja etteivät tietokannan muut toiminnot ole niistä mitenkään riippuvaisia. Tietokannan tulee myös pystyä varmistamaan lukutransaktioiden oikeellisuus kirjoitusoperaatioiden jälkeen [Elmasri and Navathe, 2010]. ACID-periaatteen toteutuminen on ehdottoman tärkeää, mikäli sovellus, esimerkiksi verkkopankki, ei saa ikinä päätyä virheelliseen tilaan [Pokorny, 2010]. Toisaalta on paljon *kuluttajille suunnattuja pilvitietokantoja* (consumer cloud databases), joiden kohdalla tiedon eheys ei ole ehdotonta. Esimerkiksi sosiaalisen median päivityksen kommentit voivat ilmestyä pienellä viiveellä vaikuttamatta palvelun tarjoamaan käyttäjäkokemukseen.

2.2. Relaatiotietokantojen laajennettavuus

Relaatiomalli ja sen ACID-periaatteet luovat haasteen relaatiotietokantojen laajennettavuuteen, mikä ilmenee tilanteessa, jossa tietokantapalvelimen resurssit eivät enää

kykene vastaamaan kysyntään ja tietokanta täytyisi jakaa kahden tai useamman eri palvelimen kesken. Kuten johdannossa mainittiin, relaatiotietokanta on alkujaan tarkoitettu käytettäväksi yhdellä palvelimella. Siksi niiden laajennettavuus on suunniteltu hoidettavan vertikaalisesti eli tehoa lisäämällä. Jossain vaiheessa yksittäisen palvelimen fyysiset rajat kuitenkin tulevat vastaan, ja kuormaa on pakko jakaa toisten palvelimien kanssa. Näin on myös mahdollista saavuttaa parempi virheensietokyky – vaikka yksi tietokantapalvelin kaatuisikin, järjestelmä voi jatkaa toimintaansa.

Mikäli pullonkaulaksi muodostuu tietokannan lukunopeus, sitä voi parantaa lisäämällä *klusteriin* (cluster) *pääpalvelimen* (primary) tueksi *kopiopalvelimia* (replica), jolloin lukuoperaatiot jakaantuvat palvelimien kesken tasaisesti. Esimerkiksi luottokorttiostoja käsittelevän palvelun kohdalla suurin osa *siirrännästä* (input/output) koostuu kirjoitusoperaatioista. Vain pääpalvelin voi ottaa vastaan kirjoitusoperaatioita, eivätkä silloin edellä mainitut keinot auta tilanteessa. Lukuoperaatioiden kohdalla tietokannan kuormitusta on mahdollista vähentää lisäksi välimuistin avulla. Kun tietokantahakujen tulokset tallennetaan väliaikaisesti palvelimen keskusmuistiin, tiedonhaku nopeutuu huomattavasti eikä tietokantayhteyttä tarvitse avata niin usein.

Horisontaalisessa laajentamisessa järjestelmään lisätään useita yhtä tehokkaita palvelimia sen sijaan, että kasvatettaisiin yksittäisen palvelimen tehoja. Relaatiotietokantojen laajentamisen kohdalla suurimmaksi ongelmaksi tulee *tiedon jakaminen palvelimien välillä* (sharding). Laajentamisen voi tehdä monilla eri tavoilla. Esimerkiksi suuri käyttäjätietoja sisältävä taulu voitaisiin jakaa sukunimen ensimmäisen kirjaimen tai maanosan mukaan. Tiedon jakaminen tekee tietokantojen hallinnasta monimutkaisempaa, sillä ohjelmakoodissa täytyy ottaa huomioon tiedon jaottelu. Lisäksi tietokantojen kaavioiden muutokset, indeksien hallinta ja varmuuskopiointi hankaloituvat.

Relaatiotietokantojen tiedon jakamiseen on kehitetty kaupallisia sovelluksia kuten MySQL Cluster ja Oracle RAC, mutta nekään eivät loppujen lopuksi tarjoa yhtä joustavia mahdollisuuksia horisontaaliseen laajentamiseen kuin NoSQL-tietokannat. Eikä varsinkaan pienemmillä yrityksillä välttämättä ole varaa niiden ylläpitoon, sillä esimerkiksi MySQL Clusterin yrityslisenssit alkavat 5000 dollarista palvelinta kohden.

3. NoSQL-tietokannat

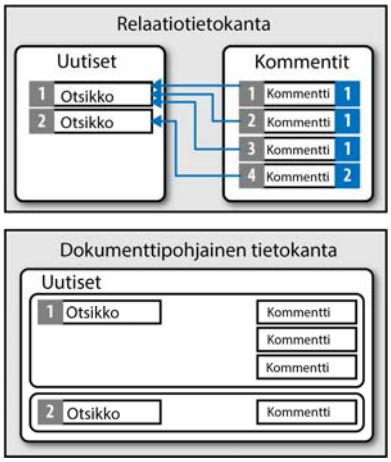
NoSQL-tietokannoilla tarkoitetaan perinteisestä relaatiomallista poikkeavia tietokantoja, joiden eduiksi katsotaan erityisesti suorituskyky ja laajennettavuus. Tässä tutkielmassa NoSQL-tietokannoilla viitataan dokumenttipohjaisiin, avain-arvo -pohjaisiin ja kenttäpohjaisiin tietokantoihin.

NoSQL-tietokantojen termistö eroaa jonkin verran relaatiotietokannoista. Esimerkiksi relaatiotietokantojen kohdalla puhutaan riveistä. NoSQL-tietokantojen kohdalla riviä vastaa *dokumentti* (document), johon voi tallentaa käytännössä mitä tahansa da-

taa, esimerkiksi listoja, olioita ja binääridataa. Dokumentti eroaa rivistä siten, että sen sarakkeita ei ole määritelty missään kaaviossa erikseen vaan ne luodaan ajonaikaisesti. [Cattel, 2010]

Johdannossa käytetyn osoitekirjaesimerkin tapauksessa NoSQL-tietokantaan voitaisiin tallentaa sellaisiakin käyttäjän yhteystietoja, joita varten ei ole luotu saraketta. Näin ensimmäinen dokumentti saattaisi sisältää vain yhteyshenkilön nimen ja numeron, mutta toisella yhteystiedolla voisi lisäksi olla osoite, sähköpostitili, kotisivuosoite ja niin edelleen. Relaatiotietokannan kohdalla on muutama mahdollisuus: joko tietokantataulua muutetaan joka kerta, kun yhteystietojärjestelmään halutaan lisätä uusi tietokenttä, tai sitten arvoja tallennetaan ei-skalaarisesti, yhteen sarakkeeseen. Näistä molemmat vaihtoehdot ovat huonoja järjestelmän ylläpidettävyyden kannalta. Kolmannessa ja järkevimmässä vaihtoehdossa voitaisiin luoda lisäkenttiä varten uusi taulu, joka koostuisi esimerkiksi sarakkeista “käyttäjä”, “ominaisuus” ja “arvo”. Näin yhteystieto-taulu säilyisi järkevän kokoisena sisältäen suurimman osan yleisistä tiedoista. Näin ei myöskään jouduttaisi puuttumaan tietokannan rakenteeseen eikä tietokannan eheys kärsisi.

Kaaviovapaus ei kuitenkaan ole yksiselitteistä NoSQL-tietokantojen kohdalla. Se, ettei tietokannalla tarvitse olla määriteltynä rakennetta, ei tarkoita, ettei sellaista kannattaisi käyttää. Kuten johdannossa todettiin, tietokantaan kerätään samankaltaisia, toisiinsa liittyviä tietoja. Se pitää paikkansa myös NoSQL-tietokantojen kohdalla. Havainnollistava esimerkki rakenteen vapaudesta on liitoksien toteuttaminen saman tietopaketin sisällä. Relaatiotietokannan kohdalla huomattiin, että taululiitoksilla oli mahdollista yhdistää tauluja. NoSQL-tietokantojen tapauksessa työntekijää kuvaavassa tietokokoelmassa voitaisiin mainita myös osasto ja yritys samassa yhteydessä. Tosin työntekijä-osasto-yritys -suhde on esimerkki rakenteellisesta tiedosta, joka luultavasti on parempi tallentaa relaatiotietokantaan. Jotkin NoSQL-tietokannat mahdollistavat liitokset kokoelmien välillä, mutta se ei ole suositeltavaa. Liitokset vaikeuttavat tiedon jakamista eri palvelimien välillä ja kyselyt hidastuvat. Kuvassa 2 esitetään yksinkertainen tietokantarakenne uutisista ja niiden sisältämistä kommenteista sekä relaatiotietokannassa että dokumenttipohjaisessa tietokannassa.



Kuva 2. Toisiinsa liittyvien tietojen yhdistäminen
relaatiotietokannassa ja dokumenttipohjaisessa tietokannassa

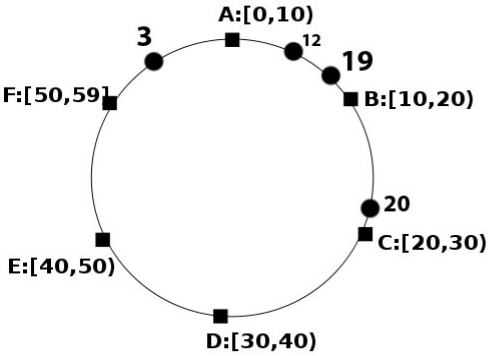
NoSQL-tietokantojen vastine ACID-ominaisuuksille on *CAP*-ominaisuudet (consistency, availability, partition tolerance), johon kuuluu seuraavat kolme sääntöä: 1. Tietokannan käyttäjillä on aina pääsy tietojen samaan versioon. 2. Tietokannan pitää onnistuneesti vastata käyttäjien tekemiin kyselyihin tilanteessa kuin tilanteessa. 3. Järjestelmä pystyy jatkamaan toimintaansa siitäkin huolimatta, että jokin tietokannan sisäinen viesti ei mene perille tai jokin sen osista kaatuu. *CAP-teoreeman* (CAP theorem) mukaan mikään horisontaalisesti laajennettu järjestelmä ei voi toteuttaa kaikkia kolmea sääntöä samanaikaisesti. Lisäksi horisontaalisesti laajennetut järjestelmät eivät voi olla samaan aikaan sekä eheitä että saavutettavissa, joten vaihtoehtoina ovat käytännössä 1. ja 3. tai 2. ja 3. sääntö. Relaatiotietokantojen hallintaohjelmistot suosivat yleisesti vain tiedon yhdenmukaisuutta (1. sääntö). [Pokorny, 2010]

NoSQL-tietokantojen kehitystä ovat johtaneet suuret verkkotoimijat kuten Amazon, joka ensimmäisenä julkaisi vuonna 2007 tietojaan kehittämästään Dynamo-tietokannasta. Google on kehittänyt oman tietokannan nimeltä BigTable, jota Facebookin tekemä Cassandra-tietokanta jäljittelee, ja joka on nykyään avoimen lähdekoodin alainen Apache-projekti. Kaikilla edellä mainituilla toimijoilla yhteistä on todella mittavan kokoiset tietokannat. Esimerkiksi Facebook on kertonut käsittelevänsä tietoa n. 500 teratavua päivittäin.

NoSQL-tietokantoihin liittyy relaatiotietokantojen tapaan myös ongelmia, joita ovat SQL-kielen puuttuminen, tiedon luotettavuuteen ja eheyteen liittyvät ongelmat sekä hallintaohjelmistojen puutteet [Leavitt, 2010]. Lisäksi NoSQL-tietokantoihin siirtyminen on usein työlästä, sillä monilla yrityksillä on käytössä suuria ja mahdollisesti todella vanhoja tietokantasovelluksia, joiden toimintalogiikka ei välttämättä tue siirtymistä.

3.2. NoSQL-tietokantojen laajennettavuus

NoSQL-tietokantojen laajennettavuus perustuu lähinnä kahteen asiaan. Kuten edellä mainittiin, NoSQL-tietokannat eivät toteuta ACID-periaatetta, ja toisekseen niiden sisältämät tiedot tallennetaan levyille siten, että niitä on helppo jakaa eri palvelimien välillä. Useimmat NoSQL-tietokannat käyttävät tiedon jakamisessa *johdonmukaista hajautusta* (consistent hashing), joka kehitettiin alun perin jakamaan tasaisesti verkkosivuihin kohdistuvia kutsuja muuttuvan palvelinjoukon välillä [Karger et al., 1997].



Tunniste-arvo	Palvelin	Kopio-palvelin
3	A	B,C
12	B	C,D
19	B	C,D
20	C	D,E
37	D	E,F
40	E	F,A
54	F	A,B

Kuva 3. Johdonmukainen hajautus. Neliöt kuvaavat palvelimia ja ympyrät dokumentteja.

Johdonmukainen hajautus on helppo ymmärtää kuvan 3 avulla. Siinä jokaiselle tallennetun dokumentin avaimelle lasketaan *yksilöivä tunnistearvo* (hash) ja sama tunnistearvo annetaan sattumanvaraisesti myös jollekin järjestelmässä olevalle *palvelimille* (node). Kun uusi dokumentti tallennetaan tietokantaan, se "tiputetaan" kehälle satunnaiseen kohtaan ja sen tunnistearvo lähetetään seuraavalle kehällä sijaitsevalle palvelimelle. Näin tieto jakaantuu palvelimien välillä tasaisesti. Jokainen palvelin on tietoinen pelkästään sen sisältämien dokumenttien tunnistearvoista ja mahdollisista kopiopalvelimista.

Tietoja haetaan kulkemalla kuvassa 3 olevaa kehää myötäpäivään. Esimerkiksi tunnistearvolla 20 olevan dokumentin tunnistearvosta tehdään kyselyt ensin palvelimilla A ja B ilman tuloksia kunnes dokumentti löytyy palvelimelta C. Kuvasta voi myös nähdä, että dokumentista on olemassa kopio palvelimilla D ja E. Kopiopalvelimien avulla vältetään tiedon menetys vaikka palvelin C kaatuisi. Lisäksi tietojen monistaminen mahdollistaa useampien palvelimien vastata hakuihin, jolloin myös samanaikaisia lukuoperaatioita voidaan suorittaa enemmän. Myös hakuajat pienenevät, sillä kehää ei tarvitse kiertää niin pitkään. Näin toimittaessa jokainen palvelin voi ottaa vastaan myös kirjoitusoperaatioita, mikä todettiin relaatiotietokantojen laajentamisen kohdalla olevan monimutkaista.

Kun järjestelmään lisätään uusi palvelin, se ilmestyy satunnaiseen kohtaan kehällä. Samalla myös kehällä uuden palvelimen vieressä olevat dokumentit siirtyvät uuden palvelimen alaisuuteen. Esimerkiksi jos kuvan 3 kehälle lisättäisiin uusi palvelin tunnistearvon 3 ja palvelimen A väliin, tunnistearvolla 3 oleva dokumentti siirtyisi sen alaisuuteen palvelimelta A. Kehällä olevat palvelimet jakavat keskenään yhtä suuren osan tunnistearvoavaruudesta, jotta hajautus olisi mahdollisimman tasaista. Palvelimien tunnistearvoalueet muuttuvat aina, kun uusi palvelin lisätään tai poistetaan. Esimerkiksi palvelimen poistuessa kehältä sen sisältämistä tiedostoista vastaa kopiopalvelin ja sen hallitsema tunnistearvoalue liitetään kehällä lähimmän palvelimen tunnistearvoalueeseen. [Strauch, 2011]

Kuten termin nimestäkin voi päätellä, dokumenttien hajautuksen täytyy olla johdonmukaista. Se tarkoittaa yksinkertaisesti sitä, että tunnistearvojen täytyy olla järjestelmästä riippumattomia ja täysin satunnaisia. Jos esimerkiksi *hajautustoiminto* (hashing function) perustuu tietyn sovelluksen aikaleimoihin, ruuhka-aikoina suoritettavat toiminnot saattavat kohdistua aina samoihin palvelimiin. Yleisimpiä *hajautusalgoritmeja* (hashing algorithm) ovat SHA-0, SHA-1, SHA-2 ja MD5. Käytännössä vanhimpia algoritmeja ei enää kuitenkaan käytetä, sillä niiden tarjoama tunnistearvoavaruus on liian pieni nykyaikaisten sovellusten tarpeisiin.

Johdonmukainen hajautus tarjoaa kuorman tasaisen jakamisen, mutta se myös mahdollistaa järjestelmän laajentamisen *massatuotantopalvelimilla* (commodity server). Massatuotantopalvelimien ei tarvitse olla tehokkaita, sillä ne vastaavat vain rajatusta määrästä dokumentteja. Mikäli palvelimet eivät ole keskenään samankaltaisia, kuorma ei välttämättä jakaudu tasaisesti tai ylimääräiset palvelinresurssit voi jäädä käyttämättä. Massatuotantopalvelimien komponenttien hinnat ovat edullisia verrattuna tehopalvelimiin ja ne tehdään helposti vaihdettaviksi. Näin vialliset palvelimet saadaan korvattua uusilla mahdollisimman helposti ja edullisesti.

3.3. Erityyppiset NoSQL-tietokannat

NoSQL-tietokannat voidaan jakaa niiden tietomallien perusteella *avain-arvo -pohjaisiin tietokantoihin* (key-value database), *dokumenttipohjaisiin tietokantoihin* (document database), *kenttäpohjaisiin tietokantoihin* (column database) ja *graafitietokantoihin* (graph database).

Avain-arvo -pohjaiset tietokannat ovat NoSQL-tietokannoista rakenteeltaan yksinkertaisimpia. Niiden tietomalli mahdollistaa arvojen hakemisen ja muokkaamisen ennalta määriteltyjen avaimien perusteella. Avain-arvo -pohjaisten tietokantojen tärkeimmät ominaisuudet ovat korkean tason laajennettavuus ja tiedon eheys. Niiden tarkoituksena ei ole tarjota monipuolisia kysely- tai analyysiominaisuuksia. Esimerkiksi taululiitokset ja erilaiset *yhdistelmäfunktiot* (aggregation function), kuten SUM ja MAX on usein jätetty toteuttamatta. Avain-arvo -pohjaiset tietokannat ovat olleet

olemassa jo pitkään (esim. Berkeley DB), mutta vasta viime vuosina uusia sovelluksia on syntynyt runsaasti Amazonin Dynamo-tietokannan innoittamana. [Strauch, 2011] Tunnetuimpia avain-arvo -pohjaisia tietokantoja ovat edellä mainitut BerkeleyDB ja DynamoDB sekä Redis ja Riak [DB-Engines, 2013].

Dokumenttipohjaiset NoSQL-tietokannat ovat askeleen pidemmälle kehitettyjä versioita avain-arvo -pohjaisista tietokannoista. Niiden tietomalli on monipuolisempi mahdollistaen esimerkiksi avain-arvo -parien ryhmittelyn dokumenttien sisällä. Dokumenteillekaan ei kuitenkaan määritellä mitään tiettyä rakennetta, jota niiden tulisi seurata. Dokumenttien esitysmuoto vaihtelee eri tietokantojen välillä, mutta useimmat käyttävät kuvaukseen formaatteja kuten XML, YAML, JSON ja BSON. Dokumenttien sisältö vastaa löyhästi relaatiotietokantojen sarakkeita ja rivejä, mutta niiden rakenne on vapaampi. [Strauch, 2011] Dokumenttipohjaisista tietokannoista tunnetuimpia ovat Apachen kehittämä CouchDB, Facebookin kehittämä Cassandra sekä MongoDB, jota käsittelen tarkemmin seuraavassa luvussa 4.

Kenttäpohjaisten tietokantojen ominaispiirteenä on tallentaa tietoa sarakkeittain rivien sijaan. Taustalla on ajatus tehostaa varsinkin *liiketoimintatiedon hallintaa ja analysointia* (business intelligence), sillä sarakepohjaiseen tietokantaan on nopea soveltaa yhdistelmäfunktioita [Strauch, 2011]. Tämä tosin edellyttää, että tietokannan rakenne on ennalta määritelty. Tehokkuus ilmenee siten, ettei ylimääräisiä tietoja haeta kyselyiden kohdistuessa aina tiettyihin sarakkeisiin. Rivipohjaisessa toteutuksessa taululiitosten kohdalla käsitellään lopputuloksen kannalta myös epäolennaista dataa. Toisaalta kenttäpohjaisiin tietokantoihin tehtävät kirjoitusoperaatiot ovat hitaampia, sillä ne vaativat enemmän transaktioita eri sarakkeisiin. Kenttäpohjaiset tietokannat soveltuvatkin lähinnä suuriin tietovarastoihin, joihin kohdistetaan raskaita ja monipuolisia lukuoperaatioita. Tunnetuimpia kenttäpohjaisia tietokantaohjelmistoja ovat Googlen BigTable, SybaseIQ sekä Apachen HBase. Myös Facebookin kehittämä Cassandra-tietokanta voidaan määritellä sarakepohjaiseksi [Leavitt, 2010].

4. MongoDB

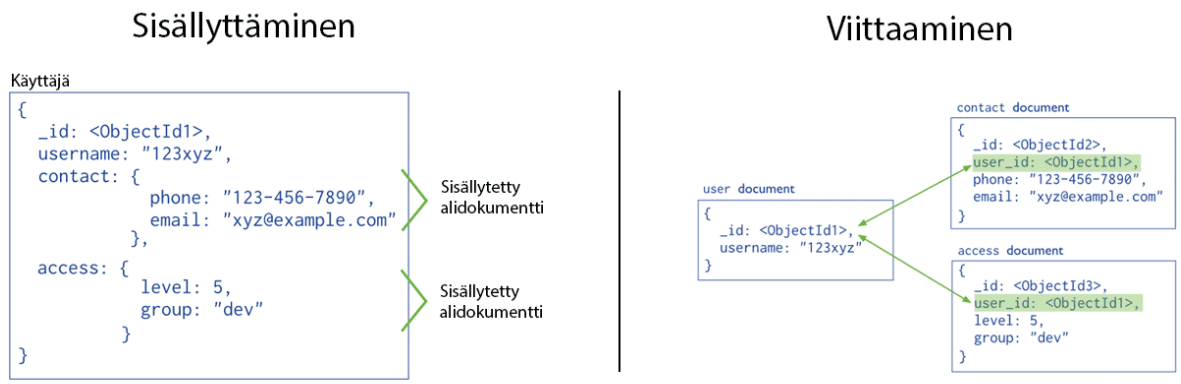
MongoDB on vuonna 2009 10gen-organisaation (nyk. MongoDB) kehittämä, C++-kielellä kirjoitettu dokumenttitietokanta. MongoDB on Google Trends -palvelun mukaan maailman suosituin NoSQL-tietokanta. Se on todella suorituskykyinen ja se käyttää relaatiotietokantoihin verrattuna paljon vähemmän keskusmuistia. Lisäksi se tarjoaa helppokäyttöiset työkalut tietokantojen laajentamiseen ja tiedon replikointiin. MongoDB on julkaistu avoimen lähdekoodin lisenssillä ja sillä on laaja, kasvava ja ketterä kehittäjäyhteisö. [MongoDB] Osaksi MongoDB:n suosiota selittää sen tarjoama pehmeä lasku relaatiotietokantojen maailmasta. Se tarjoaa monia tuttuja ominaisuuksia, kuten useiden sarakkeiden indeksoinnin ja monipuolisen kyselykielen. Lisäksi MongoDB tarjoaa mahdollisuuden tehdä kyselyitä paikkatietojen perusteella.

4.1. Tietomalli

MongoDB-palvelin voi sisältää resurssien rajoissa rajattoman määrän tietokantoja. Jokainen tietokanta koostuu joukosta *kokoelmia* (collection), jotka pitävät sisällään joukon olioita eli dokumentteja. Dokumentit koostuvat edelleen joukosta avain-arvo -pareja, joista avain on jokin arvon yksilöivä tunnistearvo. Dokumenteilla on mukautuva kaaviorakenne, eli kokoelman sisältämillä dokumenteilla ei tarvitse olla keskenään samanlaista rakennetta. Lisäksi samannimiset *kentät* (field) voivat sisältää eri tietotyyppejä. Tällöin tosin jää ohjelmoijan vastuulle tarkistaa tiedon tyyppi ennen sen käyttöä sovelluksessa. Mikäli saman kokoelman dokumenttien rakenne eroaa huomattavasti toisistaan, niistä on vaikea hakea tietoa tehokkaasti.

Kuten relaatiotietokantojen kohdalla huomattiin, usein erityyppiset tiedot liittyvät toisiinsa ja ne halutaan esittää yhdessä. MongoDB:n kohdalla on kaksi tapaa yhdistää tietoa: *sisällyttäminen* (embedded data) ja *viittaus* (references). Kuvasta 4 nähdään, miten sisällyttämisessä käyttäjäolioon liittyvät dokumentit lisätään sen sisälle omina avain-arvo -pareinaan. Tällöin tiedot saadaan haettua yhdellä tietokantakyselyllä.

Viittauksessa käyttäjä-olio liitetään viittaaviin dokumentteihin sen tunnistearvon (vrt. pääavain) avulla. Toisin kuin sisällyttäminen, viittaaminen muistuttaa relaatiotietokannan normalisointikäytäntöä, joka pyrkii tietokannan eheyteen muun muassa vähentämällä tiedon redundanssia. Viittaamisessa tarvitaan kyselyitä eri kokoelmiin, jotka saattavat sijaita eri palvelimilla aiheuttaen ylimääräistä viivettä.



Kuva 4. Vasemmalla tietomalli, joka käyttää sisällyttämistapaa yhdistäessään dokumentteja. Oikealla viittausta käyttävä tietomalli. [MongoDB]

Kummallakin tietomallilla on vahvuutensa ja heikkoutensa. MongoDB:n dokumentaatio neuvoo käyttämään sisällyttämistä tilanteissa, joissa kokoelmien välillä on

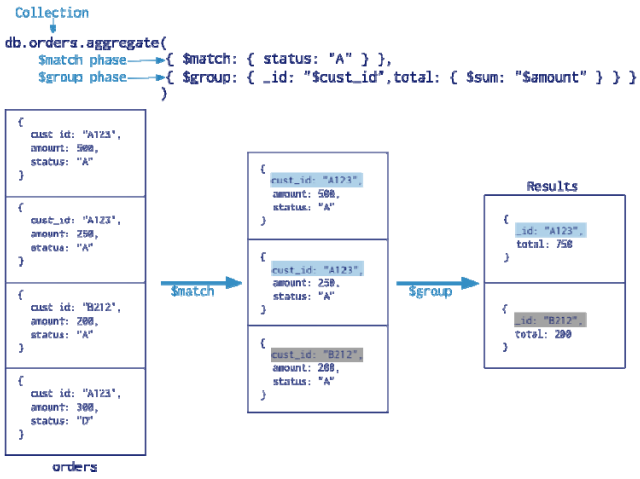
paljon sisällyssuhteita tai yksi-moneen -suhteita. Varsinkin jälkimmäisessä tilanteessa sisällyttämismalli tarjoaa helpon tavan käsitellä *isäntä-* (parent) ja *jälkeläisobjekteja* (child) saman dokumentin sisällä. Viittausta suositellaan käytettävän, jos sisällyttäminen luo liikaa päällekkäistä dataa ilman merkittävää suorituskyvyn kasvua tai kun dokumenttien väliset suhteet ovat monimutkaisia. Sisällyttämisen heikkoudeksi voidaan lisäksi laskea, että yhden dokumentin koko saattaa kasvaa liian suureksi. MongoDB:n BSON-formaattia käyttävien dokumenttien maksimikoko on 16 megatavua. Mikäli 16 megatavun rajoitus realisoituu, MongoDB tarjoaa suuria, esimerkiksi videotiedostoja varten GridFS-määrittelyn. Siinä dokumentit tallennetaan tietokantaan pienissä osissa ja kootaan takaisin yhteen dokumenttia haettaessa. [MongoDB]

MongoDB tarjoaa transaktioiden *atomisuuden* (atomicity) vain *päivitys-* (update) ja *poistokyselyille* (delete). Jotta toiminnallisuuden saa käyttöönsä, täytyy $atomic-*lippumuuttuja* (flag) asettaa true-tilaan ja lisätä se kyselyn hakuehtoihin. MongoDB ei tue monimutkaisia transaktioita eikä *lukkoja* (transactional lock), sillä tällöin varsinkin horisontaalisesti laajennetuissa ympäristöissä suorituskyky heikkenee. Näin MongoDB myös välttää tietokannan *lukkiutumistilanteet* (dead lock) ja tietokannan toiminnot saadaan pidettyä yksinkertaisina ja ennustettavina. [Strauch, 2011]

4.2. Yhdistelmäfunktiot

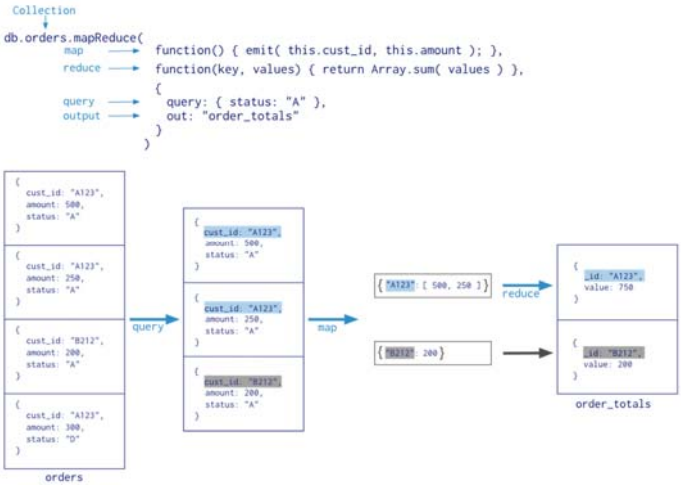
Normaalit hakufunktiot palauttavat tietokannasta dokumentteja sellaisenaan. Yhdistelmäfunktiot käyvät niin ikään läpi dokumentteja mutta ryhmitellen niitä yhteen ja palauttaen tietokantaohjelmiston laskeman tuloksen, kuten keskiarvon, summan tai lukumäärän. MongoDB tarjoaa kolme erilaista suoritusmallia yhdistelmäfunktioille, jotka ovat Aggregation Pipelines, Map-Reduce ja Single Purpose Aggregation Operations. Kaikki yhdistelmäfunktiot ottavat vastaan mielivaltaisen määrän dokumentteja ja palauttavat tulokset yhtenä tai useampana dokumenttina.

Kuvassa 5 esitetty Aggregation Pipelines -menetelmä pohjautuu data pipeline -konseptiin, jossa joukko käsiteltäviä olioita (tässä tapauksessa dokumentteja) muodostavat sarjan, jonka tietyn olion *paluuarvo* (output value) on seuraavan olion *syötearvo* (input value). Menetelmässä funktiolle annetaan ensin match-määre, joka vastaa SQL:n where-ehtoja. Group-määre määrittää, minkä arvon mukaan dokumentit ryhmitellään ja mitä laskutoimitusta käytetään. Käyttäjän ei tarvitse keskittyä suorituskykyyn tätä menetelmää käytettäessä, sillä MongoDB optimoi kyselyt automaattisesti. [MongoDB]



Kuva 5. Aggregation Pipelines -menetelmä [MongoDB]

Map-Reduce -toiminto (kuva 6) pohjautuu MapReduce-ohjelmointiparadigmaan, joka on kehitetty erityisesti suurien tietomassojen monipuoliseen aggregointiin. Se koostuu kahdesta funktiosta, Map ja Reduce. Näistä Map-funktio hoitaa datan *suodatuksen* (filtering) ja *lajittelun* (sorting), minkä jälkeen Reduce-funktio suorittaa itse aggregoinnin. MongoDB:n Map-Reduce -funktiot toteutetaan JavaScript-skriptikielellä, joka on myös itse dokumenttien pohjarakenteena. Map-Reducen vahvuutena pidetään virheensietokykyä ja sen kykyä suorittaa useita tehtäviä samaan aikaan. Esimerkiksi MongoDB:n kohdalla tämä tarkoittaa, että tietoja voidaan noutaa ja muokata monelta palvelimelta samaan aikaan käyttäen useita prosessoriytimiä. [Bonnet *et al.*, 2011]



Kuva 6. Map-Reduce -menetelmä [MongoDB]

Yhden toiminnon yhdistelmäfunktiot (Single Purpose Aggregation Operations) ovat joukon pelkistetyimpiä ja niiden käyttö on kaikkein lähimpänä SQL:n yhdistelmäfunktioita. Näistä funktioista Count palauttaa kyselyä vastaavien dokumenttien määrän, Distinct palauttaa vain kenttien uniikit arvot. Group ryhmittelee tuloksia kentän ja sen arvon sekä ehtolauseen mukaan. Groupin käytössä on rajoituksia, sillä sitä ei voi käyttää usealle palvelimelle jaettujen dokumenttien kanssa. Lisäksi sen antaman tulosjoukon tulee olla alle 16 megatavua.

Näistä kolmesta menetelmästä MongoDB:n dokumentaatio suosittelee käytettävän Aggregation Pipelines -menetelmää sen paremman suorituskyvyn ja yhtenäisen käyttöliittymän vuoksi. Map-Reduce tarjoaa kuitenkin joitain toimintoja, joita ei ole saatavilla Aggregation Pipelines -menetelmässä. Mainittakoon, että esimerkiksi yhden toiminnon yhdistelmäfunktioista varsinkin Count on erittäin helppokäyttöinen ja tehokas. [MongoDB]

4.3. Indeksointi

Relaatiotietokantojen tapaan MongoDB mahdollistaa indeksien liittämisen dokumenttien kenttiin. Indeksien sisältämät tiedot tallennetaan *B-puuna* (B-tree) ja niiden avulla kasvatetaan kyselyiden suoritusnopeutta. Indeksit nopeuttavat varsinkin koko kokoelmaa koskevia hakuja, mutta toisaalta ne myös aiheuttavat ylimääräistä kuormitusta lisäys- ja poisto-operaatioiden yhteydessä, jolloin indeksit päivittyvät. MongoDB ohjeistaakin, että indeksien käyttöä tulisi suosia lähinnä kokoelmissa, joiden käyttö on lukupainotteista. Indeksejä voi lisätä myös sisällytettyjen dokumenttien kenttiin. Esimerkiksi blogikirjoitus-dokumentti saattaa sisältää kirjoittaja-dokumentin, jolloin voisi pohtia indeksin lisäämistä kirjoittajan nimi-kenttään. [MongoDB]

Oletuksena MongoDB luo indeksin dokumenttien _id-kenttään, joka tunnistaa eri dokumentit toisistaan. Käyttäjällä on kuitenkin mahdollisuus estää automaattinen indeksien luonti, mikäli on olemassa tarve luoda indeksit manuaalisesti. MongoDB käsittelee oletuksena indeksien muokkausoperaatioita kuin mitä tahansa muitakin tietokantaoperaatioita. Haittapuolena kyseessä oleva kokoelma lukkiutuu indeksien rakentamisen ajaksi estäen kaikki CRUD-operaatiot (Create, Read, Update, Delete). Mikäli käyttäjä haluaa poiketa oletuksesta, indeksien muokkauksen voi siirtää taustaprosessiksi. Tällöin indeksointi hidastuu ja jotkin hallintatoiminnot ovat pois käytöstä sen aikana. Toisaalta näin toimittuna indeksointi ei vaikuta mitenkään CRUD-operaatioiden toimintaan. [MongoDB]

MongoDB tarjoaa mahdollisuuden lisätä geospatiaalisia indeksejä, joiden avulla voidaan helposti ja tehokkaasti suorittaa kyselyitä etäisyyksistä, alueista ja paikkatiedosta. Tyypillisesti luodaan yksi geospatiaalinen indeksi kokoelman kenttään, joka sisältää GeoJSON-objekteja. Itse kyselyt toimivat samaan tapaan kuin kaikki muutkin

hakulausekkeet. Menetelmää käyttäen voidaan etsiä esimerkiksi kahden eri alueen yhteiset kohteet tai vaikkapa vapaavalintaisen monikulmion muotoisen alueen sisältämät kohteet. Myös monista sosiaalisen median sovelluksista tuttu etäisyyden mittaaminen kohteiden välillä onnistuu. Esimerkiksi FourSquare käyttää omassa palvelussaan MongoDB:tä ja sen geospatiaalisia indeksejä. [MongoDB]

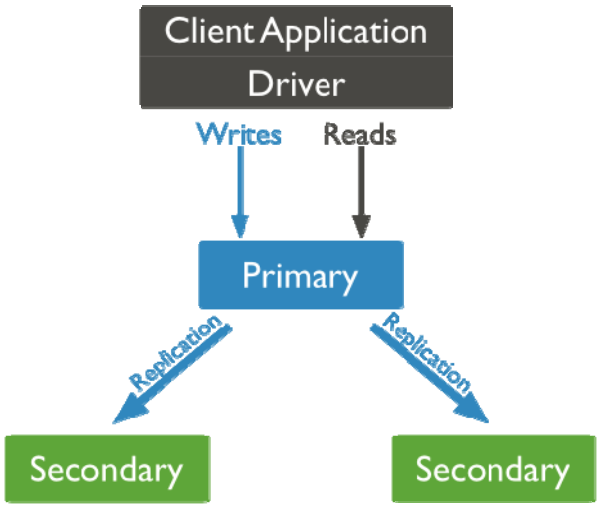
4.4. Laajennettavuus

Kuten tässäkin tutkielmassa on todettu, tietokannan horisontaalinen laajentaminen koostuu tietokantojen kopioimisesta pääpalvelimen ja kopiopalvelimien välillä sekä niiden muodostamien klustereiden monistamisesta siten, että dataa voidaan jakaa mahdollisimman helposti eri klustereiden välillä. MongoDB:n vahvuutena voidaan pitää hajautetun ja varmistetun tietokantaympäristön luomisen helppoutta. Tämän tutkielman puitteissa käyn MongoDB-tietokannan laajentamisen läpi tiivistetysti.

4.4.1. Replikointi

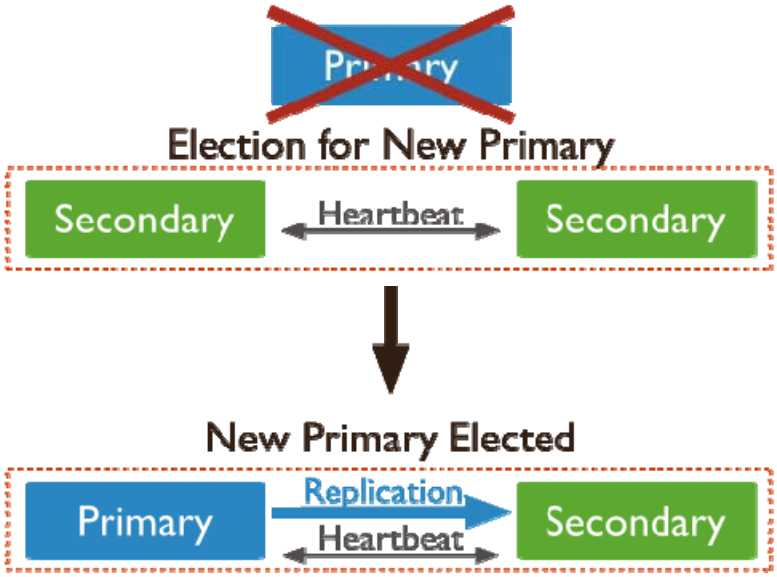
MongoDB tarjoaa kaksi tapaa tietokantapalvelimien kopioimiseen: Master-Slave- ja Replica Sets -kokoonpanot. Master-Slave -kokoonpanossa valitaan tietokantapalvelimien joukosta yksi isäntäpalvelin, jonka kopiopalvelimia kaikki loput palvelimet ovat. Isäntäpalvelin ottaa vastaan kaikki kirjoitusoperaatiot ja välittää ne kopiopalvelimille. Lukuoperaatiot jakaantuvat näin kaikkien palvelimien kesken. Menetelmää ei kuitenkaan voi pitää kovin ylläpidettävänä tai vikasietoisena, sillä oletuksena kopiointi pää- ja kopiopalvelimen välillä pysähtyy, mikäli kopiopalvelimen tiedot ovat liikaa jäljessä pääpalvelimen tiedoista. Master-Slave -menetelmä ei myöskään tarjoa *automaattista vaihtamista* (failover) pääpalvelimen ja kopiopalvelimen välillä, mikäli pääpalvelimeen tulee esimerkiksi laitevika. [MongoDB]

Kuvassa 7 näkyvä Replica Sets -kokoonpano on nykyään pitkälti korvannut Master-Slave -kokoonpanon sen lueteltujen heikkouksien vuoksi. Replica Sets -menetelmä koostuu myös pää- ja kopiopalvelimista, mutta tiedon kopiointi palvelimien välillä tapahtuu asynkronisesti. Asynkroninen kopiointi tarkoittaa sitä, ettei pääpalvelin jää odottamaan vahvistusta kopiopalvelimelta lähettäessään sille dataa. Tämä mahdollistaa nopeamman tiedonvälityksen palvelimien välillä. Haittapuolena on, etteivät kopiopalvelimet välttämättä aina tarjoa kaikkein uusinta tietoa, joka löytyy varmasti ainoastaan pääpalvelimelta.



Kuva 7. Kolmen palvelimen Replica Sets -kokoonpano [MongoDB]

Palvelimet tarkkailevat keskenään toisensa tiloja lähettämällä *sykäyksiä* (“heart beat”) toisillensa. Mikäli ajaudutaan tilanteeseen, ettei pääpalvelin vastaa sykäyksiin, kokoonpanon palvelimet äänestävät keskenään uuden pääpalvelimen (kuva 8), joka aloittaa kirjoitusoperaatioiden vastaanottamisen. Entisen pääpalvelimen palattua toimintaan se asettuu kopiopalvelimeksi. Äänestäviä palvelimia tulee olla aina pariton määrä, jotta äänestystulos saadaan aikaiseksi. Kokoonpanoon voidaan lisätä *äänestäjäpalvelimia* (arbiter), jotka eivät toimi tietokantoina vaan pelkästään äänestävät tilanteessa, jossa uusi pääpalvelin valitaan. Myös tietyn palvelimen äänen merkitystä voidaan kasvattaa. Kokoonpanossa voi olla maksimissaan yksi pääpalvelin ja 11 kopio- tai äänestäjäpalvelinta. [MongoDB]



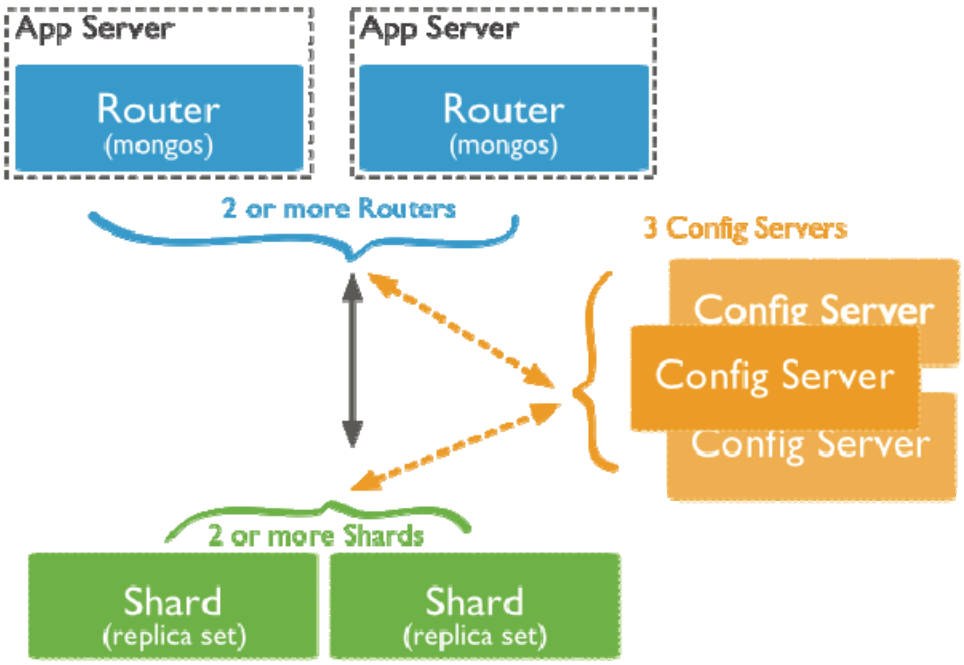
Kuva 8. Uuden pääpalvelimen äänestys ja valinta [MongoDB]

Replica Sets -kokoonpanolle voidaan luoda palvelinkeskuskohtaisia asetuksia erilaisten tilanteiden varalle. Asetuksilla voidaan säätää erityisesti tiedon eheyden ja saatavuuden tasoa. Esimerkiksi kirjoitusoperaatiot voidaan (vastoin MongoDB:n oletusta) asettaa hyväksytyiksi vasta, kun ne on kopioitu suurimmalle osalle kopiopalvelimista. Voidaan myös määrittää, että pääpalvelin vaihtaa tilansa kopiopalvelimeksi ja lopettaa kirjoitusoperaatioiden vastaanottamisen, mikäli se ei saa vastausta vähintään puolelta kokoonpanon kopiopalvelimista. Lisäksi asetukset voidaan säätää siten, että pääpalvelimen vikaantuessa keskeneräiset tiedonsiirto-operaatiot keskeytetään ja niitä voidaan erikseen jatkaa manuaalisesti myöhemmin.

Asetuksia muutettaessa tulee pohtia tarkkaan, onko siihen hyvät perusteet, sillä muutokset saattavat sotia NoSQL-tietokantojen pääperiaatteita vastaan. Monipuolisuus ja säädettävyys täytyy kuitenkin ehdottomasti lukea MongoDB:n eduksi, sillä osaavissa käsissä se on monipuolisesti ja helposti räätälöitävissä.

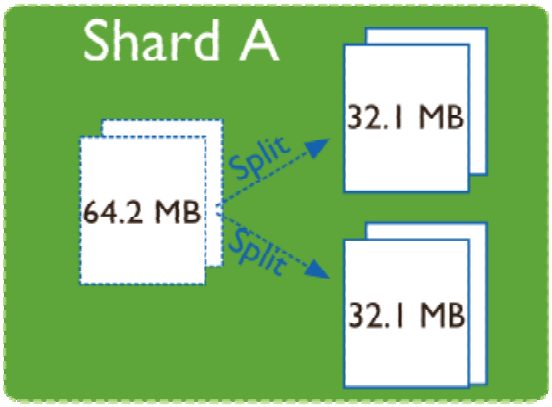
4.4.2. Sharding

MongoDB toteuttaa tiedon jakamisen usean palvelimen kesken kolmea komponenttia käyttäen. Jaettu MongoDB-klusteri (kuva 9) koostuu *kokoelman osista* (shard), *kyselyreitittimistä* (Query Routers) ja *asetuspalvelimista* (Config servers). Kokoelman osat sisältävät itse datan ja usein ne toteuttavat Replica Sets -menetelmän, jotta tietovarastot olisivat mahdollisimman eheät ja hyvin saatavilla. Asetuspalvelimet säilövät klusterin metatiedot. Kokoelman osat kommunikoivat jatkuvasti asetuspalvelimien kanssa lähettämällä tietoa niiden sisältämästä datasta. Kyselyreitittimet puolestaan ohjaavat näiden tietojen perusteella operaatiot oikeille kokoelman osille ja huolehtivat vikatilanteissa liikenteen uudelleenohjauksesta. [MongoDB]



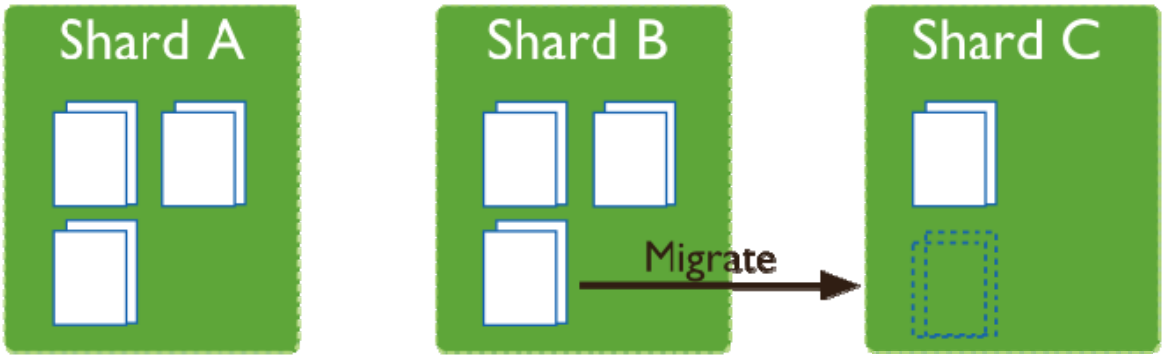
Kuva 9. MongoDB-klusteri [MongoDB]

MongoDB tarjoaa kaksi tapaa hajauttaa tietoa kokoelman osien välillä. Ensimmäinen tapa on arvojoukkoperusteinen hajautus, jossa tietokokoelma jaetaan *kokoelman osan tunnistearvojen* (shard keys) perusteella ei-päällekkäisiin pieniin lohkoihin (chunks), jotka saavat tietyn minimi- ja maksimiarvon. Tästä johtuen dokumentit, joilla on lähellä toisiaan oleva tunnistearvo, saattavat päätyä lähelle toisiaan hajautusalueella. Toinen tapa on käyttää yksilöivien tunnistearvojen perusteella tapahtuvaa hajautusta, joka perustuu kohdassa 3.2 esiteltyyn johdonmukaiseen hajautukseen. Tätä menetelmää käyttäen voidaan varmistua siitä, että tiedot hajautuvat tasaisesti kaikkien kokoelman osien välillä. Ainoa huono puoli on, että arvojoukkoperusteisessa hajautuksessa kyselyreitittimien on helpompi päätellä, mistä kokoelman osasta haluttu data löytyy. Johdonmukaisessa hajautuksessa tietoja joudutaan hakemaan useammalta palvelimelta. [MongoDB]



Kuva 10. MongoDB:n Splitting-taustaprosessi [MongoDB]

Käytettiin kumpaa hajautustapaa tahansa, jossain vaiheessa ajaudutaan tilanteeseen, jossa jonkin kokoelman osan fyysiset rajat tulevat vastaan tai eri kokoelman osat ovat erikokoisia. MongoDB ajaa näiden tilanteiden varalta jatkuvasti kahta taustaprosessia. Splitting-prosessi (kuva 10) huolehtii, että kun jokin kokoelman osan lohko kasvaa yli sen sallitun rajan, se jakaantuu kahdeksi yhtä suureksi lohkoksi. Jokaisella kyselyreitittimen sisältävällä palvelimella pyörivä Balancing-prosessi huolehtii siitä, että kuorma jakaantuu kokoelman osien välillä tasaisesti. Sen toiminta perustuu siihen, että se siirtää lohkoja suurimman ja pienimmän määrän lohkoja sisältävien palvelimien välillä (kuva 10). [MongoDB]



Kuva 11. MongoDB:n Balancing-taustaprosessi [MongoDB]

5. Yhteenveto

Olen tässä tutkielmassa käynyt läpi relaatiotietokantojen ja NoSQL-tietokantojen keskeisimpiä eroja ja pohtinut erityisesti, miten niiden laajennettavuus eroaa toisistaan. Lopuksi käsittelin MongoDB-tietokantaohjelmistoa, joka on lyhyestä historiastaan huolimatta vakiinnuttanut paikkansa monenlaisten sovellusten kohdalla.

Relaatiotietokannoilla on edelleen vahva asema varsinkin ACID-periaatetta noudattavien tai monimutkaisia laskenta- ja raportointitoimintoja käyttävien sovellusten keskuudessa [Stonebraker, 2010]. Relaatiotietokantojen osaajia on myös huomattavasti enemmän johtuen relaatiomallin pitkästä historiasta. Lisäksi standardoitu SQL-kieli mahdollistaa yhteisen kielen kehittäjien keskuudessa. On kuitenkin osoitettu, ettei relaatiotietokantojen laajentaminen ole yhtä helppoa ja tehokasta kuin NoSQL-tietokantojen kohdalla. Myöskään niiden ennalta määritelty rakenne ei välttämättä tue parhaalla mahdollisella tavalla ketterän ohjelmistokehityksen vaatimaa joustavuutta.

NoSQL-tietokantoja varten ei ole vielä kehitetty juurikaan edistyneitä hallintaohjelmistoja, joiden avulla tietokantoja voisi hallinnoida graafisen käyttöliittymän avulla. Tähän tosin voi helposti ennustaa tulevan muutoksen jo lähivuosien aikana, kun NoSQL-tietokannat kehittyvät ja niiden käyttöaste laajenee. Myös laitteistojen ja ohjelmistojen yhteensopivuuteen keskitytään entistä enemmän [Leavitt, 2010]. NoSQL-tietokantojen ehdoton vahvuus on niiden laajentamisen helppous. Esimerkiksi MongoDB mahdollistaa kymmenien palvelimien klusterin muodostamisen minuuteissa – ja siihen kykenee ohjeita seuraamalla kuka tahansa asiaan hieman perehtynyt. Sama ei onnistu relaatiotietokannoilla – ei ainakaan yhtä helposti.

On kuitenkin selvää, etteivät relaatiotietokannat tule häviämään mihinkään vielä pitkään aikaan. Ja jottei todellisuus hämärtyisi, on syytä todeta, että myös relaatiotietokannat on mahdollista virittää todella vaativaan käyttöön. Esimerkiksi Facebook tallentaa edelleen suuren osan tiedoistaan MySQL-relaatiotietokantoihin. Tosin täl-

laisen järjestelmän rakentaminen ja ylläpitäminen on äärimmäisen haastavaa ja aikaa vievää. Viimeaikaisista trendeistä päätellen voisi olettaa, että relaatiotietokantojen ja NoSQL-tietokantojen sekakäyttö tulee lisääntymään [Leavitt, 2010]. Esimerkiksi juuri Facebook on toteuttanut osan järjestelmästään Cassandra-tietokannan päälle.

NoSQL-tietokannan valintaa sovelluksen tietovarastona kannattaa harkita, mikäli tiedon eheys ei ole elintärkeää ja käsiteltävät tietomäärät ovat suuria. Valinnassa täytyy lisäksi ottaa huomioon eri NoSQL-tietokantojen tietomallit ja ominaisuudet. Osaavissa käsissä mitä tahansa järjestelmää on mahdollista laajentaa lähes rajattomasti, mutta varsinkin MongoDB:n kaltaiset tietokantaohjelmistot tarjoavat tavallisille käyttäjille mahdollisuuden rakentaa järjestelmiä, mihin relaatiotietokantojen maailmassa vain gurut pystyvät.

Viiteluettelo

[Bonnet, et al., 2011] Laurent Bonnet, Anne Laurent, Michael Sala, Benedicte Laurent, Nicolas Sicard. REDUCE, YOU SAY: What NoSQL can do for data aggregation and BI in large repositories. *Database and Expert Systems Applications* (2011), 483-488.

[Cattel, 2010] Rick Cattel, Scalable SQL and NoSQL data stores, *ACM SIGMOD* **39**, 4, (2010), 12-27.

[Codd, 1970] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM* **13**, 6, (1970), 377–87.

[DB-Engines, 2013] DB-Engines Ranking http://db-engines.com/en/ranking.

[Elmasri and Navathe, 2010] Elmasri R. and Navathe S.B. *Fundamentals of Database Systems.* Sixth edition. Addison Wesley, 2010.

[Graham et al., 2013] Colleen Graham, Joanne Correia, David Coyle, Fabrizio Biscotti, Matthew Cheung, Ruggero Contu, Yanna Dharmasthira, Tom Eid, Chad Eschinger, Bianca Granetto, Hai Hong Swinehart, Sharon Mertz, Chris Pang, Asheesh Raina, Dan Sommer, Bhavish Sood, Marianne D'Aquila, Laurie Wurster, and Jie Zhang. Market share: all software markets, Worldwide 2012, 29.3.2013.

[Karger et al., 1997] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (1997), 654-663.

[Leavitt, 2010] Neal Leavitt, Will NoSQL databases live up to their promise? *Computer* **43**, 2 (2010), 12-14.

[MongoDB] MongoDB 2.4 documentation. http://docs.mongodb.org/manual.

[Pokorny, 2011] Jaroslav Pokorny, NoSQL databases: a step to database scalability in web environment. ln: *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, WAS '11, (2011), 278-283.

[Stonebraker, 2010] Michael Stonebraker. SQL databases v. NoSQL databases. *Communications of the ACM* **53**, 4, (2010), 10-11.

[Strauch, 2011] Christof Strauch. NoSQL databases. Lecture Selected Topics on Software-Technology Ultra-Large Scale Sites, Manuscript. Stuttgart Media University http://www.christof-strauch.de/nosqldbs.pdf.

[Sutinen, 2010] Olli Sutinen, NoSQL – factors supporting the adoption of non-relational databases. M. Sc. thesis, Dept. of Computer Sciences, University of Tampere, 2010.