Kanditutkielma

Tietojenkäsittelytieteen kandiohjelma

Järjestelmäintegraatioiden tyylit ja suunnittelumallit

Joona Halonen

4.5.2024

Matemaattis-luonnontieteellinen tiedekunta

Helsingin yliopisto

Yhteystiedot

PL 68 (Pietari Kalmin katu 5)
00014 Helsingin yliopisto

Sähkopostiosoite: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Batchelor's Programme in Computer Science

Tekijä — Författare — Author

Joona Halonen

Työn nimi — Arbetets titel — Title

Järjestelmäintegraatioiden tyylit ja suunnittelumallit

Ohjaajat — Handledare — Supervisors

Vanhenpi yliopiston lehtori Lea Kutvonen

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Bachelor's thesis	May 4, 2024	25 pages, 23 appendix pages

Tiivistelmä — Referat — Abstract

Write your abstract here.

In addition, make sure that all the entries in this form are completed.

Finally, specify 1–3 ACM Computing Classification System (CCS) topics, as per https://dl.acm.org/ccs. Each topic is specified with one path, as shown in the example below, and elements of the path separated with an arrow. Emphasis of each element individually can be indicated by the use of bold face for high importance or italics for intermediate level.

ACM Computing Classification System (CCS)
General and reference → Document types → Surveys and overviews
Applied computing → Document management and text processing → Document management → Text editing

Avainsanat — Nyckelord — Keywords

ulkoasu, tiivistelmä, lähdeluettelo

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Järjestelmäintegraatiot *(Enterprise Application Integration, EAI)* tarkoittaa ohjelmistoja, jotka yhdistävät eli integroivat organisaation olemassa olevia erillisiä järjestelmiä toimimaan yhdessä.

Tarve EAI:lle kehittyi kun organisaation jo olemassa olevien järjestelmien hajautunut tieto piti yhdistää tiedonkulun optimoimiseksi ja automatisoimiseksi. Esimerkiksi organisaation asiakkuudenhallinnasta vastaavan ohjelmiston pitäisi pystyä keskustelemaan myynninhallinnointiohjelmiston kanssa pitääkseen asiakastilastot ajan tasalla. Järjestelmät jäivät helposti erillisiksi saarekkeiksi eivätkä jakaneet tietoa keskenään. Syntyi niin sanottuja "savupiippujärjestelmiä", joissa järjestelmä oltiin suunniteltu pitämään data esimerkiksi firman yksittäisen osaston sisällä, eikä järjestelmän suunnitteluvaiheessa oltu huomioitu datan jakamista; järjestelmästä puuttui esimerkiksi ulospäin suuntavat ohjelmointirajapinnat.

Järjestelmäintegraatio mahdollistaa näiden eri järjestelmien keskinäisen keskustelun toisilleen. Näin voidaan rakentaa yhteinen järjestelmä jo olemassa olevista palasista [21, sivu 15]. Useissa tapauksissa organisaation sisäisiä järjestelmiä yhdistetään myös ulkoisiin järjestelmiin tai muiden organisaatioiden kanssa [20]. Organisaation olemassa olevat järjestelmät voi olla kehitetty eri käyttöjärjestelmille, hyödyntävät eri ohjelmointikieliä tai tietokantaratkaisuita. Väliin tarvitaan integraatio, joka hoitaa kommunikoinnin ohjelmistojen välillä

Vuonna 2020 EAI-markkinan arvioitiin olevan 9,26 miljardin dollarin (USD) arvoinen [22] ja koostuu ohjelmistojättien (Microsoft, Oracle, Salesforce IBM) ja pienempien yksittäisten yritysten (Workato, SnapLogic, Tibico) tarjoamista järjestelmistä.

2 Integraatioiden lähestymistavat

Esittelen tässä luvussa suosituimpia jäsentely ja lähestymistapoja järjestelmäintegraatioille. Eri lähestymistavat perustuvat viitatuimpaan järjestelmäintegraatioarkkitehtuurin kirjallisuuteen. Luvun tarkoituksena on antaa kattokategorioita erilaisille teknisille lähestymistavoille ja antaa historiallista kontekstia eri arkkitehtuurisuuntauksille.

2.1 Integraatiotasot

Teoksessaan [21] luokittelee integraatioiden lähestymistavat neljään eri tasoon

1. Datataso: Dataa liikutellaan eri tietovarastojen välillä ja mahdollinen datan prosessointi ja muokkaus toteutetaan siirron yhteydessä. Kyseisellä menetelmällä on myös paljon yhtäläisyyksiä tyypillisen datavaraston *(data warehouse)* toteutuksen kanssa.

2. Rajapintataso: Tasolla pyritään hyödyntämään tarkoitusta varten tilattujen tai paketoitujen sovellusten, esimerkiksi SAP, PeopleSoft, tarjoamia ohjelmointirajapintoja.

3. Metoditaso: Tällä tasolla pyritään hyödyntämään jaettua sovelluslogiikka. Esimerkiksi hyöntyntämällä sovelluspalvelimia *(application server)* metodienjakamiseen tai käyttämällä hajautettuja objekteja ja/tai etäkutsuja *(Remote Procedure Call, RPC)* hyödyntäviä teknologioita.

4. Käyttöliittymätaso: Integraatioiden yhdistävä tekijä on olemassa olevan sovelluksen käyttöliittymä kun sovellus ei tarjoa muita keinoja datan jakamiseen. Tämä toteutetaan hyödyntymällä ruudun tiedonharavointi tekniikoita (screen scraping).

2.2 Tekniset lähestymistavat

Järjestelmäintegraatioiden arkkitehtuurin määrää pitkälti liiketoiminnan tarpeet ja niiden asettamat vaatimukset. Integraatioiden tekniset lähestymistavat voi luokitella neljään erilaiseen yläkategoriaan [16, sivu 64].

1. Tiedostojen siirto
2. Jaettu tietokanta
3. Etäproseduuriherätys
4. Sanomat

Tiedostojen siirto: Tiedostojen siirrossa integroitavat sovellukset voivat toimia itsenäisesti ja yhden sovelluksen muutokset eivät vaikuta toisen sovelluksen toimintaan kunhan sovellukset toimivat sovituilla tiedostotyypeillä ja tiedoston nimeämisstrategioilla. Integroijan vastuulla on taata, että tiedostot muutetaan toisen sovelluksen ymmärtämään muotoon. Lähestymistapana tiedostojen siirto on yksinkertainen eikä vaadi lisätyökaluja, koska yleisten tiedostotyyppien (JSON, XML, CSV) tuki löytyy käytännössä jokaisesta ohjelmointikielestä tai integraatiotyökalusta. Data-analytiikassa ja datavarastojen saralla on myös muita yleisiä tiedostomuotoja kuten Apache Parquet, jolle löytyy myös tuki useista ohjelmointikielistä ohjelmistokirjaston muodossa [3]. Sovitusta tiedostomuodosta tulee käytännössä integroitavien sovellusten välinen rajapinta.

Tiedostojen siirron heikkouksina on tiedostojärjestelmäoperaaatioiden hallinnointi ja datan synkronointi. Integraatiokehittäjien vastuulle jää siis tiedostojen lukitseminen kirjoitusoperaatioiden ajaksi tai kirjoitusten ajoittaminen jotta ne eivät mene päällekkäin lukemisen kanssa, tiedostojen nimeämiskäytännöt, tiedostojen arkistointi ja poistaminen. Jos integroitavilla sovelluksilla ei ole pääsyä samoille levyille niin kehittäjien ratkaistavaksi jää myös tiedoston siirtäminen oikealle laitteelle. Datan synkronointi järjestelmien välillä tuo oman haasteensa, koska tiedostojen siirtoa tapahtuu yleensä harvakseltaan. Esimerkiksi jos asiakkuudenhallintajärjestelmä tuottaa tiedostoja datan synkronointia varten vain kerran päivässä ja laskutusjärjestelmä lähettää laskut aikaisemmin samana päivänä, niin osa laskuista on jo saattanut lähteä vanhaan osoitteeseen, jos asiakas on päivittänyt osoitetietojaan aikaisemmin saman päivän aikana.

Jaettu tietokanta: Jaetun tietokannan etuna tiedostojen siirtoon on muutosten nopea propagoituminen eri sovelluksille. Samassa tietokannassa uusin tieto on saatavilla eri sovelluksille lähes välittömästi. Nopea tiedon liikkuminen tekee virheiden havaitsemisesta ja korjaamisesta helpompaa. Etuna on myös, että tietokantojen datamallit takaavat yhtenevän datan esitysmuodon verrattua tiedostojen siirtoon. Datan synkronoinnista ja kirjoitus- ja lukuvuoroista vastaa tietokannan hallintajärjestelmä *(DBMS, Database Management System)* ja transaktioiden hallinnointijärjestelmä antaa hyvät työkalut datan eheyden takaamiselle.

Gregor Hohpen ja Bobby Woolfin teos [16, sivu 69] korostaa myös SQL-pohjaisten relaatiotietokantojen yleisyyttä. Integraatiokehittäjien ei tarvitse opetella uutta teknologiaa tai taistella uuden tiedostoformaatin kanssa vaan kehittäjät voivat työskennellä laajalti tunnettujen relaatiotietokantojen parissa. Valtaosa ohjelmointikielistä ja kehitystyökaluista tukee SQL:n kanssa työskentelyä joten jaetun tietokannan kanssa työskentely on suoraviivaista ja adoptointi helppoa.

Saman tietokannan käyttö estää datan tulkitsemiseen liittyvien ongelmien, kuten semanttisen dissonanssin *(semantic dissonance)* pitkittymistä, missä samaa dataa voidaan tulkita ristiriitaisilla tavoilla. Koska integroitavat sovellukset käyttävät samaa datalähdettä, niin nämä tulkintakysymykset on kohdattava varhaisessa vaiheessa integraatiokehitystä, eikä vasta tuotannossa jossa data voi olla jo yhteensopimatonta tulkintaeroista johtuen.

Jaetun tietokannan suunnitteluhaasteisiin sisältyy yhtenäisen skeemaan suunnittelu, jota useampi eri sovellus pystyy tehokkaasti hyödyntämään. Usein useamman sovelluksen tuomat vaatimukset johtavat monimutkaiseen tietokantaskeemaan jonka käytön kehittäjät kokevat haastavaksi. Yhtenäisen skeeman suunnittelua voi myös vaikeuttaa "poliittiset"haasteet, koska tietokannan suunnittelu voi johtaa aikataulujen venymiseen ja kommunikaatiohaasteisiin tietokantaa hyödyntävien eri yksiköiden välillä. Lisää suunnitteluhaasteita tuo ulkoiset ohjelmistot. Lähes poikkeuksetta kaupalliset ohjelmistot tukevat vaan omaa ohjelmiston mukana tulevaa tietokantaformaattia eivätkä taivu siitä poikkeavaan tietokantaskeemaan. Vastaavia haasteita tuo sovellukset jotka on peritty toiselta organisaatiolta esimerkiksi yrityskaupan yhteydessä. Sovellusten jälkeenpäin tehtävä jatkokehittäminen jaettua tietokantaa hyödyntäväksi on yleensä työlästä ja kallista. Kun jaettuun tietokantaan yhdistettyjen sovellusten määrä lisääntyy niin ratkaisu voi aiheuttaa suorituskykyhaasteita, varsinkin jos luku- ja kirjoitusoperaatiot kohdistuvat vaan muutamaan tietokantatauluun. Jos sovellukset on hajautettu useammalle laitteelle ja tietokanta niiden kanssa, jotta sovelluksilla on lokaali pääsy kantaan, niin hajauttaminen tuo omat haasteensa. Pääosin datan hajauttamistaktiikkojen muodossa ja lisää näin ratkaisun kompleksisuutta.

Etäproseduuriherätys:*(Remote Procedure Invocation)* Edelliset lähestymistavat keskittyivät pääosin datan jakamiseen, mutta näissä lähestymistavoissa pienet datanmuutokset voivat johtaa eri toimintoihin useiden sovellusten taholta. Osoitteen vaihto voi olla yksinkertainen kentän muutos tai laukaista useita rekisteröinti- ja lakiprosesseja useassa eri sovelluksessa. Jaettu tietokanta ei mahdollista minkäänlaista datan kapselointia ja tämä yksi iso datalähde tekee datamuutosten havaitsemisesta ja muutosten vaatimien prosessien

aktivoimisesta haastavaa. Tiedostojen siirto tarjoaa suoraviivaisen tavan reagoida datan muutoksen, mutta tämä tapahtuu yleensä viiveellä johtuen tiedostojen synkronoimisen haasteista. Jaetun tietokannan kapseloimattomuus tarkoittaa myös integraatioiden ylläpidon joustamattomuutta. Muutokset yhdessäkään integroidussa sovelluksessa vaikuttavat jaettuun tietokantaan ja tietokantamuutokset voivat aiheuttaa kauas kantautuvia muutoksia tietokantaa käyttävien sovellusten kesken.

Etäproseduuriherätys mahdollistaa mekanismin jossa sovellus voi kutsua toisen sovelluksen funktiota, jakaa vain tarvittavan datan ja kutsua funktiota joka kertoo datan vastanottajalle miten toimia jaetun datan kanssa. Jos sovellus tarvitsee toisen sovelluksen dataa se voi kysyä sitä siltä suoraan. Vastaavasti jos sovelluksen tarvitsee muokata toisen sovelluksen dataa niin se voi tehdä funktiokutsun. Jokainen sovellus vastaa oman datansa eheydestä ja jokainen sovellus voi tehdä muutoksia omaan dataansa, vaikuttamatta muiden sovellusten tilaan.

Etäproseduuriherätyksen mahdollistavat teknologiat ovat myös yleisiä ja tuttuja kehittäjille. Etäproseduurikutsu *(Remote Procedure Call, RPC)* teknologiat ja kirjastot ovat tunnettuja ja yleisesti käytettyjä. Teoksessa [16, sivu 71] Martin Fowler listaa CORBA, COM, .NET Remoting ja Java RMI esimerkkeinä ja mainitsee, että web-palveluiden yleistyessä http-yhteyksiä hyödyntävät lähestymistavat kuten SOAP ja XML ovat tulleet kehittäjien suosikeiksi. Varsinkin kun http-yhteyksien kanssa on helppo työskennellä, koska useimpien yritysten palomuurit sallivat http-liikenteen. Teoksen julkaisun jälkeen REST ja JSON ovat pitkälti korvanneet SOAP:in ja XML:än web-palveluiden suosituimpana lähestymistapana.

Etäproseduuriherätyksellä on mahdollisuus vähentää semanttista dissonanssia, koska sovellukset voivat tarjota usean erillaisen rajapinnan samalle datalle. Eri asiakasohjelmille voidaan tarjota erilainen datanesitysmalli riippuen siitä, mikä asiakasohjelma on kyseessä. Tämä antaa ennemmän mahdollisuuksia esittää datan useammalla eri tavalla verrattuna pelkkään relationaaliseen malliin. Useammat eri rajapinnat tarkoittavat lisää työtä integraatiokehittäjille datan muokkaamisen parissa ja integroitaviensovellusten täytyykin neuvotella mitä rajapintoja ne tulevat toisiltansa käyttämään.

Etäproseduuriherätyksen helppous kehittjille voi myös olla sen haittapuoli jos integraatiokehittäjät eivät tiedosta etäkutsujen suorituskyky- ja luotettavuuseroja verrattuna paikkalisiin kutsuihin. Useat etäkutsut voivat kasaannuttaa näitä ongelmia ja johtaa hitaaseen ja epäluotettavaan järjestelmään.

Vaikka etäproseduuriherätyksen mahdollistama datan kapselointi vähentää sovelluksen

kytköksiä karsimalla suuren yhteisen datalähteen, niin se voi silti aiheuttaa solmukohtia, erityisesti kun kyse on jaksossa - tietyssä järjestyksessä tehtävistä- operaatioista. Integraatioijärjestelmistä näistä muodostuu helposti ongelma, koska vastaavat kytkökset eivät välttämättä aiheuttaisi ongelmia yksittäisessä sovelluksessa, mutta usemman sovelluksen integraatiossa lisäkytkökset tarkoittavat lisäviivettä ja ylimääräisiä verkkokutsuja.

Sanomat:*(Messaging)* Aikaisemmat integraatioiden lähestymistavat keskittyivät joko datan tai toiminnalisuuden jakamiseen. Gregor Hohpen ja Bobby Woolfin mukaan yleinen integraationkehitykesn haaste on saada eri järjestelmät toimimaan yhdessä mahdollisimman viipettä ilman, että järjestelmien välillä on kytköksiä, jotka tekevät järjestelmästä epäluotettavan joko sovelluksen suorittamisen tai kehittämisen kannalta [16, sivu 72]. Tiedostojen siirrossa datan siirtyminen ei ole tarpeeksi viipeetöntä ja sovellusten välinen toiminta tarpeeksi sujuvaa vaikka lähestymistapa estääkin rajoittavien kyskösten muodostamisen. Jaetussa tietokannassa data on jaettu ja datan muutokset ovat responsiivisia, mutta kaikki sovellukset ovat kytköksissä samaan tietokantaan ja lähestymistapa ei mahdollista sovellusten yhteistä toimintaa. Etäproseduuriherätyksen heikkoudet olivat yleiset hajautettujenjärjestelmien sudenkuopat, kuten verkkoviiveet ja verkon luetattavuus, varsinkin jos eäkutsuja käytetään samallalailla kuin paikallisia kutsuja ja lähestymistavassa sovellusten pitää jakaa tietoa toistensa rajapinnoista, mikä lisää kehitystä vaikeuttavien kytkösten määrää.

Sanomien käyttö erityisesti asynkroniseen viestintään on aikaisemmin esiteltyjen lähestymistapojen parhaiden puolien yhdistelmä [16, sivu 73]. Sanomien käyttö mahdollistaa pienien tiheästi kulkevien datapakettien lähetyksen ja tallenttamisen ja tiedosto-operaatioiden yksityiskohtien abstsraktoinnin. Tämä mahdollistaa nopeat skeeman muutokset vastaten yrityksen tarpeisiin. Sovellukset pystyvät jakaa toiminnallisuuksian lähettämällä sanoman toisilleen joka herättää esimerkiksi datanmuokkausproseduurin. Asynkroninen viestintä ei taas vaadi vastaanottajan olemaan saatavilla lähestyhetkellä ja asynkronine viestintä ohjaa kehittäjiä ymmärtämään, että etäyhteyksien käyttäminen on hitaampaa ja suunnittelemaan korkeamman koheesion komponentteja, jolloin etänä tehtävien operaatioiden käyttö on harkitumpaa. Sanomapohjaiset järjestelmät myös mahdollistavat tiedostojen siirron kaltaisten löyhien kytkösten käytön. Sanomia voidaan muokata kesken lähetyksen ilman, että lähettäjä- tai vastaanottajasovelluksen tarvitsee olla tietoinen muokkausoperaation yksityiskohdista. Tämä mahdollistaa integroijien yleislähettävän *(broadcast)* sanomia useammalle vastaanottajalle, valitsevan yhden vastaanottajan useamman joukosta tai valita useasta muunlaisesta topologiasta jotka sallivat integraation irrottamisen sovel-

luksen kehitysprosessista. Sanomien tiheä lähettäminen mahdollistaa säännöllisen datan jakamisen lisäksi toiminnallisuutta. Käsittelyprosessi voidaan käynistää heti kun yksittäinen sovellus saapuu ja asynkronisten kutsujen avulla lähettävän sovelluksen suoritus ei keskeydy odottamaan vastausta.

Sanomien lähetyksen tiheä datanvaihto ei kuitenkaan estä semanttisen dissonanssin syntymistä, varsinkin kuin datan esitysmuoto voi vaihtua useasti sanomia muokatessa. Sanomien lähetyksen tiheys ei myöskään täysin poista samoja datan synkronointihaasteita, joita tiedostojen siirrossa ilmeni. Sanomien siirrossta on jonkin verran viiveitä ja niiden ajoituksella tulee edelleen olemaan merkitystä. Asynkronisuus tuo myös lisä haasteensa integraatioden kehitys vaiheessa. Testaus ja sovellusten virheenpaikkannus tulee olemaan monimutkaisempaa sanomien lähetyksen rinnakkaisuuden takia ja vaati integraatiokehittäjiltä jonkin verran lisätotuttelua. Sanomien käytön löyhät kytkennät lisäävät integroitavien sovellusten koheesioita, mutta tarkoittavat kuitenkin vaikeammin ylläpidettävän "liimakoodin"tarvetta jotta integraatiot saadaan toimimaan yhdessä.

Edellä mainitut haasteet tarkoittavat sanomapohjaisille järjestelmille suunniteltuja lähestymistapoja ja arkkitehtuureja mitkä toistuvat järjestelmissä niiden yksittäisistä eroista huolimatta.

3 Sanomapohjaiset suunnittelumallit

Edellisessä luvussa esitellyistä neljästä kategoriasta Hohpe ja Woolf suosivat sanomien käyttöä integraatioratkaisuissa ja suurin osa teoksesta keskittyy sanomapohjaisen suunnittelumallien esittelyyn [16, sivu 76]. Nämä järjestelmäintegraatioiden suunnittelumalleina *(enterprise integration patterns, EIP)* tunnetut kehitysohjeet ovat vaikuttaneet useamman eri integraatioalustan arkkitehtuuriin.

Vuonna 2018 viimeisintä tekniikka edustavista 48:sta integraatioalustasta yksitoista tuki EIP-malleja [7]. Kaksistoista vuotta kirjan julkaisun jälkeen pidetyssä haastattelussa kirjailijat sanovat, että useimmat avoimen lähdekoodin liikepalveluväylät *(enterprise service bus, ESB)* ovat omaksuneet teoksessa esitellyn mallikielen *(pattern language)* [26]. Samassa haastattelussa Hohpe ja Woolf toteavatkin, että kirjan pysyminen relevanttina yli 12 vuotta julkaisun jälkeen on harvinaista tietokoneaiheisille kirjoille [26]. Woolf selittää kirjan pitkän vaikutuksen johtuvan kirjan keskittymisestä suunnittelumalleihin eikä spesifisiin teknologioihin [26].

Teos sisältää 67 sanomapohjaista suunnittelumallia ja nämä mallit on jaettu seitsemään eri kategoriaan [16]:

1. Sanomajärjestelmät *(messaging systems)*
2. Sanomakanavat *(messaging channels)*
3. Sanomien rakentaminen *(message construction)*
4. Sanomien reititys *(message routing)*
5. Sanomien muokkaus *(message transformation)*
6. Sanomien päätepisteet *(messaging endpoints)*
7. Järjestelmän hallinta *(systems management)*

Tässä tutkielmassa perehdymme vain sanomajärjestelmiin, koska muissa kategorioissa esitetyt EIP:eet sisältyvät sanomäjärjestlemissä esiteltyihin malleihin.

3.1 Sanomajärjestelmät

Järjestelmät kategoria toimii kattokattegoriana muille kategoroille ja tarjoaa konseptit ja termit muille suunnittelumalleille. Kategoria esittelee lähtökohtaiset pohjasuunnittelumallit minkä päälle muut suunnittelumallit rakentavat.

Sanomajärjestelmät *(messaging systems)* on kattokategorian nimi EIP:lle, mutta viittaa myös järjestelmään joka jakelee sanomia. Vaihtoehtoinen termi sanomajärjestelmille on sanomapohjainen väliohjelmisto *(message-oriented middleware)*. Hohpe ja Woof käyttävät termiä vastaavasti teoksessaan [16]. Jos tekstissä viitataan sanomajärjestelmiin kattokategoriana se tarkennetaan erikseen, muussa tapauksessa sanomajärjestelmä viitta edellä mainittuun ohjelmistojärjestelmään joka kuljettaa sanomia.

- Sanomakanava *(Message Channel)*: Kanavia käytetään viestin organisointiin ja jaotteluun sanomajärjestelmän sisällä. Kanava toimii eräänlaisena virtuaalisena putkena lähettäjän ja vastaanottajan välillä. Kehittäjän vastuulle jää kanavien luonti ja organisointi. Sovelluksen lähettäessä dataa, dataa ei lähetetä satunnaisesti mille tahansa kanavalle vaan lähettävä sovellus tietää juuri oikean kanavan jolle data on tarkoitettu. Vastaavasti sama dynamiikkka kuuluu myös vastaaanottavalle sovellukselle; sovellus tietää juuri oikean kanavan mistä odottaa oikeanlaatuista dataa. Kanavat toimivat loogisina osotteina sanomajäjrestelmälle. Kanavan toteutusyksityiskohdat vaihtelevat eri sanomäjärjestelmien välillä [18] [16]. Kanavien käyttö on laskentakapasiteetin osalta halpaa, mutta ei täysin ilmaista; jokainen kanava tarvitsee muistia sanomien datan esitttämiseen ja pysyvien sanomien tapauksesssa – sanomat jotka säilyvät levyllä esimerkiksi virhetilanteessa – myös levytilaa. Kanavia kannataa siis luoda useita, mutta lukumäärän kasvaessa tuhansiin, alkaa sanomajärjestelmän skalautuvuudesta tulla haaste [16].

- Sanoma*(Message)*: Jos kanavaa voi ajatella putkena niin sanomaa voi ajatella putken vetenä, paitsi virran sijaan putkesssa data kulkee yksittäisinä datayksiköinä. Kuten monen muunkin protokollan kohdalla, niin myös sanomakin sisältää otsakkeen *(header)* ja hyötykuorman *(payload)*. Sanomajärjestelmät eivät ota kantaa hyötykuorman sisältöön, mutta vastaanottava sovellus voi reagoida eritavoila hyötykuorman sisältöön. Saapuva sanoma voi aktivoida proseduurin, välittää dataa, ilmoittaa sovellusta tilamuutoksesta tai vaatia sovelluksesta vastausta [16]. Sanomat voi myös lähettää sarjana jos lähetettävä data ei mahdu yhteen sanoman hyötykuormaan. Jos datalla

on rajallinen voimassaoloaika, voi sanoma tarkentaa lähetetyn datan voimassa olo ajan [16]. Hohpen ja Woofin teos sisältää suunnittelumallit näille erillaisille sanoman käyttötavoile.

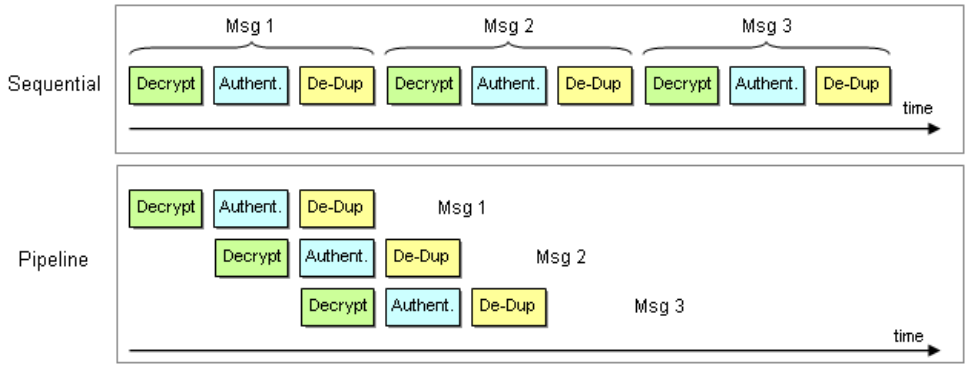
- Putket ja suodattimet *(Pipes and filters)*: Putket ja suodattimet suunnittelumallissa putkena toimivat edellä mainitut kanavat ja suodattimet ovat pieniä itsenäisiä prosessointiaskeleita joilla ei ole kytköksiä muihin prosessointiaskeleihin. Kanavat yhdistävät nämä suodattimet toisiinssa ja suodattimet voivat toimia yksin tai olla yhdistettynä toisiinsa. Vaihtoehtoisesti putkena voi toimia muistissa oleva jono. Suodattimien rajapinta pitää olla tarpeeksi yksinkertainen, että putkien implementaatio on vaihdettavissa ja kiinteitä kytköksiä muihin suodattimiin tai putkiin ei synny. Asynkroninen viestintä komponenttien välillä mahdollistaa suodattimien samanaikaisen käytön. Suodattimien löyhät kytkökset helpottavat testaamista kun suodattimia voi testata itsenäisinä komponentteina jonka lopputuloksen oikeudellisuutta voi helposti verrata siihen sisälle syötettyyn dataan.

  Putket ja suodattimet lähestymistapana on myös saanut suosiota integraatioalustoiden arkkitehtuureissa; neljästäkymmenestäkahdeksasta modernista integraatioalusta kymmenen hyödyntää putket ja suodattimet arkkitehtuurityyliä [7]. Putket ja suodattimet arkkitehtuurille löytyy myös implementaatioohjeita monelle alustalle ja teknologialle. Internet haulla putket ja suodattimet suunnittelumallille löytyy ohjeet seitsemän eri teknologian virallisilta sivuilta, Amazon Web Services, Red Hat Fuse, Spring Integration, Microsoft Azure, Mulesoft Runtime ja Apache Camel [4] [2] [1] [24] [6] [23].

  Suodattamien käyttö ei ole täysin ilmaista. Suodattamien skaalautuvuuden haasteena on se, että jokainen suodatin tarvii oman putkensa datan kuljettamiseen. Jos putken toteutetaa sanomakanavana niin näiden käyttö eivä myöskään ole täysin ilmaisia kuten aikasemmassa suunnittelumallissa on mainittu; vastaavasti muistissa olevat jonot kuluttavat luonnollisesti muistiresursseja. Myös sanomien kulku putkissa ja suodattimista tuo omat haasteensa, koska viestit pitää käsitellä kanavan formaatista sovellukseen käyttämään formaattiin ja sama prosessi pitää tehdä takaperin sanoman lähtiessä, syöden operaatiosyklejä. Pitkät suodatinketjut tarjoavat arkkitehtuurista joustavuutta löyhien kytköstensä ansioista, mutta maksavat potentiaalisen suoritustehon laskemisen muodossa, johtuen toistuvista datan esitystavan muutoksista.

  Asynkronisten sanomakanavien hyödyntäminen mahdollistaa jokaisen suodatinyksikön käsitellä sanoman itsenäisesti omassa säikeessään tai prosessissaan. Tämä johtaa huomattavasti suurenpaan suoritustehoon kun yksikään suodatinyksikkö ei ole odottamassa sanoman käsittelyn päättymistä vaan seuraavaa sanomaa voidaan al-

kaa käsittelemään enne kuin edellinen on valmistunut. Hohpe ja Woolf kutsuvat tätä prosessointiputkeksi *(processing pipeline)* jota selventää kuva 3.1 [16] [18] Kuvassa 3.1 sanoman salaus puretaan *(decrypt)*, autentikoidaan *(authent.)* ja lopuksi tarkistetaan duplikaateilta *(de-dup)*.

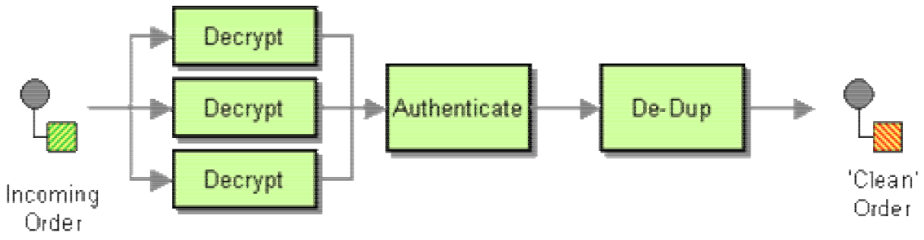


Kuva 3.1: Kuva sanomien peräkkäisen käsittelyn ja prosessointiputken eroista [16]

Peräkkäisessä käsittelyssä, eli ilman asynkronisuutta, sanomat joutuvat odottamaan toisten sanomien käsittelyä. Alemmassa prosessointiputkessa asynkronisuus mahdollistaa suodattimien operaatiot heti kun seuraava sanoma saapuu.

Suodatinketjun suorituskykyä voidaan parantaa tästä entisestään. Edellisessä prosessointiputken esimerkissä heikkoutena on se, että kaikesta hitain suodatinprosessi määrää koko järjestelmän sanomien käsittelytahdin. Tässä tapauksessa hitain suodatin voidaan rinnakkaistaa. Haasteena on varmistaa, että jokaisen sanoman voi käsitellä vain yksi rinnakkaisista suodatinprosesseista. Toinen huomioon otettava aspekti on, että rinnakkaistaminen ei takaa sanomien suoritusjärjestyksen säilymistä. Jos sanomien käsittleyjärjestyksellä on väliä, suodatinprosesseja voi olla vain yksi instanssi. Kolmas haaste on suodattimien tilanhallinta. Tilanhallinta rinnakkaisissa operaatioissa lisää tunnettusti kompleksisuutta, niin suodattimen tilanpalautuminen lähtöpisteeseen operaation valmistuttua yksinkertaistaa huomattavasti suodattimien rinnakkaistamista.

Kuvan 3.2 esimerkissä Hohpe ja Woof näyttävät miten salauksen purkamisen *(decrypt)* käytetty suodatin voidaan rinnakkaistaa kolmeksi rinnakkaiseksi prosessiksi, koska purkamisoperaatio on työläin suodattimista [16]. Esimerkissä duplikaattien tarkastus *(de-dup)* on pidetty peräkkäisenä prosessina, koska duplikaattien havaitseminen tarvitsee sanomahistorian hyödyntämistä eli operaatio ei ole tilaton ja sen rinnakkaistaminen on haastavaa.



Kuva 3.2: Kuva sanomien rinnakkaisesta käsittelystä kun järjestyksellä on väliä [16]

- Sanomareititin *(Message Router)*: Putkien ja suodattimien löyhät kytkennät mahdolistavat uuden prosessointiaskeleen lisäämisen oleamssaolevien väliin. Sanomareitittimen tapauksessa reititin vastaa putkien ja suodattamien suodatinta siinä suhteessa, että se tekee suorittaa prosessointia, mutta suodattimien sijaan reititin on kytketty useampaan ulostulo kanavaan. Reititin ei myöskän muokkaa sanomaa vain on ainoastaan keskittynyt sanomien määränpään valitsemiseen. Reitittimen etuna on löyhä kytkentä muihin komponentteihin; ympärillä olevien sudoattimien ei tarvitse olla tietoisia reittittimen olemassaolosta ja kaikki reitityspäätöksien logiikka elää yhdessä sijainnissa. Jos uusia sanomatyppejä tai reititysmuutoksia toteutetaan niin ainoastaan sanomareitittimen logiikka tarvitsee muokata. Koska saapuvan kanavan kaikki viestit siirtyvät reitittimen läpi yksitellen, on saapuvien viestien käsittelyjäjrestys myös taattu. Löyhän kytkennän ylläpitohaasteet korostuvat sanomareitintä hyödyntäessä. Jos suurin osa järjestelmästä on löyhästi kytketty toiseen niin järjestlmän kokonaiskuvan muodostaminen käy hankalaksi. Hohpe ja Woofin mukaan näitä haasteita voi lieventää hyödyntämällä sanomien kulkuhistoriaa [16, sivu 93].

  Reittitimen haasteina ovat erilaiset pullonkaulat. Reittittimestä voi syntyä ylläpidollinen pullonkaula, koska reittittimen pitää olla tietoinen kaikista vastaanottavista kanavista ja tästä voi syntyä ylläpiolline ongelma varsinkin jos vastaanottokanavien lista vaihtuu tiheään. Toinen mahdollinen pullonkaula on suoritysky. Reititin luonteensa vuoksi lisää ylimääräisen prosessointiaskeleen ja monissa sanomajärjestelmissä sanoman siirtäminen toiselle kanavalle vaati sanomien uudelleen koodausta. Useamman reitittimen rinnakkainen käyttö lieventää ongelmaa, mutta jokatapauksessa reititin tulee vaikuttamaan negatiivisesti sanomien viiveeseen vaikka sanomien käsittelytahti pysyisikin samana [16, sivu 93].

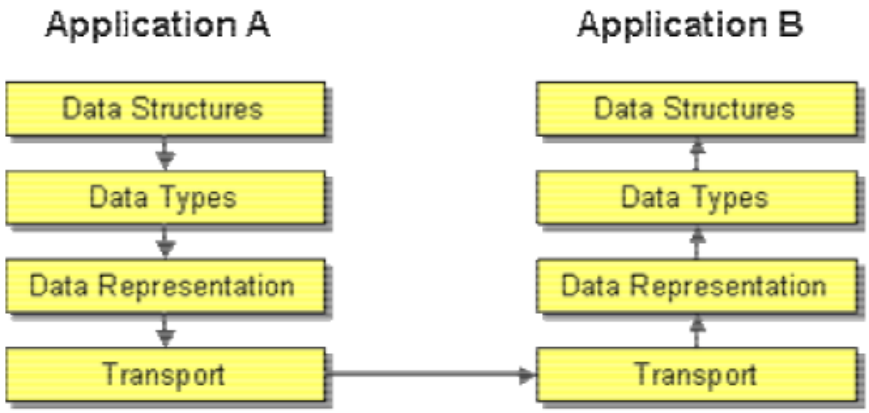
  Reitittimen käyttö voidaan kategorioida useamaan eri tyyppiin. Yksinkertaisin rei-

titin on kiinteä jolloin sanoma kulkee yhdestä kanavasta yhteen vastaanottaja kanavaan. Kiinteiden reitittimen tarkoitus on irrottaa kiinteitä kytkentöjä erilaisista alijärjestelmistä tai välittää sanomia eri integraationratkaisujen välillä. Reititin voi olla sisältöpohjainen *(content-based)* tai ympäristöpohjainen *(context-based)*. Sisältöpohjainen tekee reitityspäätöksiä sanoman sisällön perusteella. Ympäristöpohjainen ottaa huomioon ympäristön tilan ja tekee esimerkiksi kuormituksen tasaamista tai uudelleenreititystä ympäröivän systeemin virhetilanteessa. Reitittimet voi jakaa tilallisiin ja tilattomiin; tilalliset reitittimet ottavat huomioon aikasemmat sanomat ne voivat esimerkiksi estää duplikaattisanomien saapumisen tallentamalla jo saapuneet sanomat. Reitin voi olla myös dynaaminen *dynamic router* ja vastaanottaa konfiguraatiomuutoksia ohjausväylän kautta. Tällä tavalla reititin voi muutta reitityslogiikkaansa ilman koodimuutoksia [16, sivu 94].

- Sanomankääntäjä *(Message Translator)*: Sanomankääntäjä on suodatintyyppi, joka mahdollistaa eri dataformaatteja käyttävät sovellukset keskustelemaan keskenään sanomien avulla. Hohpe ja Woof vertaavat sanomankääntäjä EIP:tä sanomajärjestelmille tarkoitetuksi versioksi tunnetusta sovitinsuunnittelumallista [16, sivu 97], joka sai alkunsa Gang of Four teoksesta [8]. Keratauksena aikaisempiin esiteltyihin EIP: hin verrattuna sanomakääntäjän tarkoituksena on vähentää tiukkoja riippuvuuksia arkkitehtuurissa; sanomakanava mahdollisti, että sovellusten ei tarvitse tietää toistensa sijaintia, sanomareititin mahdollisti löyhät kytkennät sovellusten viestien reitittämisessä, joten sanomakääntäjä estää järjestelmien tiukat riippuvuudet toistensa dataformaateista. Sanomakääntäjä ratkaiseee jaetun tietokannan haasteita jota käsiteltiin luvuvssa 2.2. Erityisesti yhtenäisen skeemaan löytäminen ei muodostu ongelmaksi ja järjestelmät on kytketty löyhemmin toisiinsa kääntäjiä hyödyntämällä. Vaihtoehtoinen lähestymistapa sanomakääntäjän sijaan olisi käyttää yhteistä datatyyppiä koko sanomajärjestelmässä, joka jakaa myös yhtäläisyyksiä jaetun tietokannan kanssa 2.2, toteuttamalla samoman datatyypin muunnokset sanomien päätepisteisiin 3.1, mutta valmiissa kaupallisissa sanomajärjestlemiä tukevissa sovelluksissa päätepisteen koodi ei ole muokattavissa. Lisäksi sanomien kääntämislogiikan eläessä päätepisteessä, koodin uudelleenkäyttäminen on haastavaa.

  Hohpe ja Woof [16] jakavat sanomkääntäjän operaatiot eri abstraktiotasoihin, ottaen löyhästi vaikutteita OSI-mallista [19], jakaen kääntämisoperaatiot kuljetus-, datan esitys-, datatyyppi- ja tietorakennekerroksiin (sovelluskerros). Kuljetuskerrokseen kuuluvat protokollat kuten http, mqtt, JMS. Datan esityskerros on kiin-

nostunut datan muodosta kute JSON, XML ja enkoodauksesta kuten UTF-8. Datatyyppi viittaa soveluksen muistinsisäisiin esityistapohin esim. ovatko postinumerot merkkijonoja vai kokonaisnumeroita ja ovatko esim. aikaleimat aina UTC-ajassa vai sisätävätkö aikavyöhykeinformaation. Tietorakennekerros (tai sovelluskerros) ottaa kantaa logiisiin kokonaisuuksiin, kuten eri dataentiteettien assosisaatioihin ja niiden kardianaliteetin esim. onko dataentiteettien välillä moni-moneen-yhteyksiä. Jakamalla kääntämisoperaatiot usempaan kerrokseen voidaan operaatioita käsitellä itsenäisesti omassa abstraktiotasossaan, kuten kuvassa 3.3, ilman riippuvuutta muihin kääntäjiin ja kääntämisoperaation koodin uudelleenkäyttäminen on suoraviivaista. Myös eri abstraktiotasojen operaatiot ovat vaihdettavissa toisiin. Esimerkiksi datan esityskerrosessa *(Data Representation)* voidaankin päättää, että data esitetäänkin JSON:ina CSV:een sijaan ilman, että vaihto aiheuttaa muutoksia muiden tasojen operaatioissa.



Kuva 3.3: Sanomakääntäjän operaatiot voivat kohdistua kaikkiin abstraktiotasoihin [16]

- Sanomien päätepiste *(Message Endpoint)*: Verrattuna aikaisempiin esiteltyihin kattokategorian EIP:in, sanoman päätepiste on puuttuva palanen, joka mahdollistaa integroitavan sovelluksen liittämisen osaksi sanomaviestintää hyödyntävää kokonaisuutta; sanomien päätepiste on siis asiakasohjelmisto, joka yhdistää sovelluksen sanomakanavaan ja mahollistaa sanomien lähettämisen tai vastaanottamisen. Sanomajärjestselmät ovat palvelinohjelmia ja sanomien päätepiste toimii niiden asiakasohjelmana. Sanomien päätepiste vastaanottaa sovelluksen komennon tai datan ja muuttaa sen sanomaksi ja lähettää sanoman sanomakanavalle ja päinvastoin sanomia

vastaanottavalle päätepisteelle. Hohpe ja Woof rajaavat yhden päätepisteinstanssin vain vastaanottamaan tai lähettämään, eikä toteuttamaan molempia eli sovellus hyödyntäisikin useampaa päätepistettä jos sen pitäisi kommunikoida useammalle sanomakanvalle [16, sivu 106]. Sovellus voi kuitenkin hyödyntää usemapaa päätepisteinstanssia vaikka sanomat lähtisivätkin vai yhdelle kanavalle, mahdollistaakseen instanssien skaalatutumisen useammalle samanaikaiselle säikeelle.

Päätepisteen koodi on kustomoitu kyseiselle sovellukselle ja sanomajärjestelmän asiakasrajapinnalle sopivaksi. Päätepiste kapseloi sanomajärjestelmän sovellukselta joten sovellus ei ole tietoinen sanomajärjestelmästä. Esimerkiksi sanomajärjestelmän asiakasrajapinnan muutokset vaikuttavat vain päätepisteen koodiin, eikä lainkaan itse integroitavaan sovellukseen.

Päätepiste EIP:estä on muokattuja versioita riippuen siitä haluuako sanomia vastaanottava päätepiste käyttää erilaista vastaanottostrategiaa kuten kiertokyselyä tai tapahtumapohjaista reagointia. Useammat vastaanottavat päätepisteet voivat kilpailla sanomien käsittelystä *(computing consumer)* tai jakaa viestin lähettämisen eri päätepisteelle sisäine logiikan mukaan *(message dispatcher)*. Vastaanottava päätepiste voi vastaavasti olla idempotenssi niin toistuivat sanomat eivät aiheuta järjestelmässä virhetiloja [16, sivu 106].

4 Sanomajärjestelmien kehitystä

4.1 Kirjailijoiden omaa modernisointia

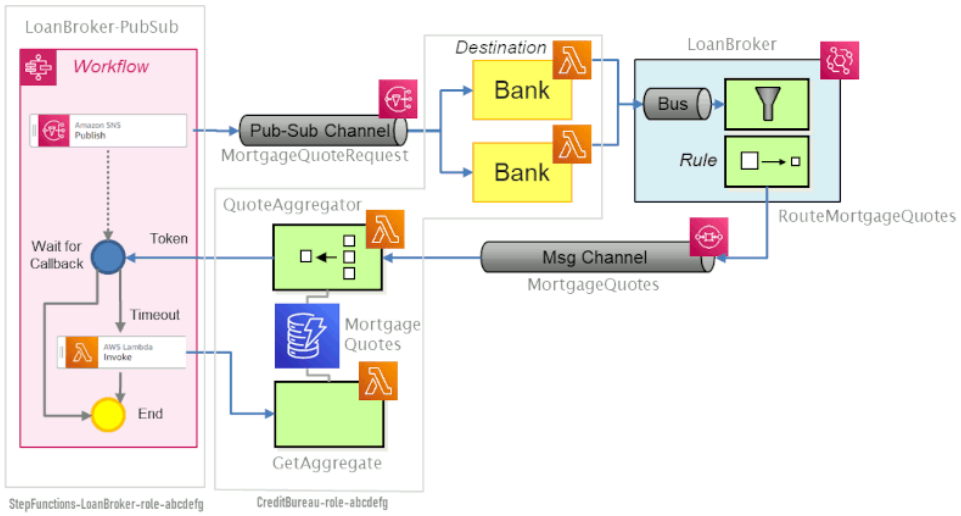
EIP-teoksen [16] julkaisun jälkeen kirjailijat (pääasiassa Gregor Hohpe) ovat pyrkineet pitämään sisällön relevanttina. Vaikka kirjan arkkitehtuurisisältö itsessään on pysynyt relevanttina ja vaikuttaa selkeästi nykypäivän järjestelmäintegraatioratkaisuissa ja asynkronisten sanomien käytössä, on kirjan koodiesimerkit ja teknologiamaininnat vanhentuneet; jota on myös tässä tutkielmassa pyritty korjaamaan tuoreemmilla esimerkeillä jotka ovat tämänpäivän opiskelijalle tuttuja.

Hohpen henkilökohtainen blogi sisältää useamman päivitetyn esimerkin tunnetuista EIP-suunnittelumallista, mutta nyt implementoituina käyttäen pilviteknologioita:

- Sisältöpohjainen reititin, EIP:een kategoriassa sanomien reititys, modernisoitiin käyttäen Googlen pilvialustan *(GCP)* Google Cloud Functions palvelitonta arkkitehtuuria vuonna 2017 [12]. Alkuperäisen teoksen C# Microsoft Message Queuing pohjainen toteutus vaihtui Node.js pohjaiseen, Google Cloud Pub/sub kirjastoa hyödyntävään toteutukseen.

- EIP-teoksessa esitelty monimutkaisempi esimerkki lainanvälitttäjästä *(Loan Broker)* [16, sivu 317] toteutettuun kolmella eri tavalla: Java / Apache Axis web-palvelimena, C# / Microsoft Message Queuing sanomajonona ja Tibco ActiveEnterprise julkaise ja tilaa kanavana. Vuonna 2021 Hohpe aloitti sarjan blogikirjoituksia joissa lainanvälittäjä esimerki toteutetaan uudelleen käyttäen Amazon Web Services *(AWS)* pilviteknologioita [9]. Lainanvälittäjä esimerkki toteuttettiin pilvituotteilla AWS Step Functions, AWS Lambda, AWS Simple Notification Service, AWS Simple Queuing Service ja AWS DynamoDB. Katso kuva 4.1.

  Myöhemmissä sarjan osissa Hohpe keskittyy ohjelmiston julkaisemiseen AWS:ää hyödyntäen [13] [14]. Hohpe myös argumentoi, että EIP-suunittelumallit ovat itse pilviarkkitehtuuriratkaisun lisäksi hyödyllisiä abstraktioita koodin julkaisua automatisoidessa [14]. Julkaisussa käytettävät teknologiat ovat AWS CloudFormation, AWS Simple Storage Service, AWS Serverless Application Model ja AWS Cloud Development Kit.



Kuva 4.1: Kuva EIP:een lainanvälittäjä esimerkistä AWS:än palveluita hyödyntämällä. [13]

- Hohpen esimerkkien modernisointi jatkuu samana vuonna, lainanvälittäjä esimerkin siirtämisenä AWS:ltä Google Cloudille [15]. Ohjelman siirto ja kuvan 4.1 AWS palvelut kuvaantuivat yksi yhteen GCP:een palveluille. Esimerkissä hyödynnettiin GCP Workflows, GCP Cloud Functions, GCP PubSub, GCP DataStore teknologioioita. Julkaisun automaatiossa huomattavaa oli, että AWS CloudFormationille ei löytynyt suoraa vastaavuutta GCP:een tarjonnasta ja vastaava toiminnalisuus pitää löytää kolmannen osapuolen ratkaisuista [15].

EIP-teoksessa esiteltyjen suunnitelumallien esimerkkejä on myös vähitellyn uudistettu EIP:een kotisivulle [11]. Tällä hetkellä 67 suuniittelumallista 14 on modernisoitu. Modernisoidut on koottu yhteen paikkaan Hohpen blogiin [10]. Aiemmin mainittujen pilvialustaesimerkkien lisäksi EIP-suunnittelumalleja on kyseisissä esimerkeissä toteutettua Golangilla, Apache Kafkalla, Apache Camelilla, Mule ESB:llä, RabbitMQ:lla ja Microsost Azurella.

Edellä mainitut esimerkit, mukaan lukien listatut pilvialustaesimerkit, löytyvät julkisesta githubissa isännöidystä tietovarastosta [17].

4.2 Tilalliset protokollat

EIP:een toinen sudenkuoppa on tilalliset *(stateful)* protokollat. Hohpe myöntää, että tilallisesta suunnittelumalleista voisi tulla EIP toinen voluumi ja tilallisten suunnittelumallien

sisällyttäminen oikeuttaisi "Enterprise Integrations Patterns"otsikon käyttöä [26]. Hohpe on alkanut kerämään tilallisia suunnittelumalleja sivulle [11], mutta toistaiseksi suunnittelumallit eivät ole poikeneet uutta kirjaa. Verrattuna EIP:een Hohpen tilalliset mallit käyttävät sanomien sijaan keskusteluja *(conversations)* jonka ympärille suunnittelumallien abstarktiot on rakennettu.

Integraatiotrendien ja kaupallisten sekä avoimen lähdekoodin alustojen analyysissä todettiin, että lähes kaikki alustat tarjoavat omat tilalliset "keskustelunsa" ja mahdollisuudet tallennustilan käyttöön [25]. Tosin tämä alustojen tarjoama tuki todetaan alkeelliseksi. Palvelukeskeisen arkkitehtuurin *(Service Orieted Architecture, SOA)* kirjallisuudesta löytyy kuitenkin keskustelusuunnittelumalleja mitä ei löydy integraatioalustojen toteutuksista [25]; julkaisussa [5] suunnittelumallit yhdestä-moneksi *(one-from-many)* ja yhdestä-moneen *(one-to-many)* ovat monenvälisiä keskustelusuunnittelumalleja jotka todetaan puuuttuvan olemassaolevsita toteutuksista [25].

Koska nykypäivän integraatiot keskittyvät enenmmän pilveen, integraatiovaatimukset ovat alkaneet kallistua tilallisiin talennustilaa vaativiin skenaarioihin mihin EIP ei vastaa ja kerätyt keskustelusuunnittelumallit ovat myös keskenräisiä vastatakseen [25]. Paremmin muodostetuille keskustelusuunnittelumalleilla löytyisi tarvetta.

Huomioitavaa on, että keskustelusuunnittelumalleja ei voi kuitenkaan verrata palvelukeskeisen arkkitehtuurin koreografia- *(choreography)* tai vuorovaikutussuunnittelumalleille *(interaction patterns)*, koska keskustelumallit kuvaavat monimutkaisempa tehtäviä kuin datan lähettämisen ja vastaanottamisen [25].

4.3 Virtausprotokollat

Synkrononisia- ja virtausprotokolleja *(streaming protocol)* ei ole huomioitu järjestelmäintegraatioisssa eikä niiden suunnittelumalleissa, minkä hohpe ja woof myöntävät haastattelussa [26]. Erityisesti virtausprotokollien puute aiheuttaa haasteita esimerkiksi big data järjestelmien integraatioissa [25]. Samassa haastattelussa hohpe ja woof toteavatkin, että virtausprotokollien suunnittelumallien dokumentointi parantaisi sanomien käytön ja virtauskäyttötapausten yhtäläisyyksien ymmärtämistä ja johtaisi eip:een kaltaisten suunnittelumallien löytämiseen [25].

Virtausdatan käsittelemistä eip-malleilla on kokeiltu laitteistokiihdyttämisen kanssa hyödyntämällä ohjelmoitavia porttimatriiseja [**Dannritter2017**]. Virtausdatan käsittelyä var-

ten eip-malleja laajennettiin kahdella uudella suunnittelumallilla: kuormituksen tasaaja *(load balancer)* ja liittäjä reititin *(join router)*. Kuormituksen tasaaja lähettää sanomaan yhteen useasta kanavasta hyödyntämällä yleisiä kuromituksen tasaus mekanismeja. Liittäjä reititin liittää useasta eri kanavasta saapuvat sanomat yhdelle kanavalle ohjelmoidun logiikan mukaisesti.

4.4 Virheenhallinta

Virheenhallinnasta: To handle erroneous situations during message processing, escalate them and make systems more fault-tolerant, error handling is seen as a major aspect [5], [45]. Hohpe et al. [3], [70] do only cover Dead Letter Channel as solution and sketch some ideas about the topic. Overall, in the literature, the topic is neither addressed from a pattern, formalization, nor modeling perspective. While [5] mentions missing patterns and formalization, Merkel et al. [45] lists Balancing and Distribution, as well as [69] mentions Fault-tolerance and Message Scheduling as missing aspects. Similarly, the insight into the current state of affairs, called monitoring, for services and cross-cloud are seen as important topics in [45], [61], [62][25]

4.5 Formalisointi

5 Yhteenveto

Lähteet

[1] *3. Spring Integration Overview*. 2024. URL: https://docs.spring.io/spring-integration/docs/4.2.5.RELEASE/reference/html/overview.html (viitattu 02. 02. 2024).

[2] *4.4. Pipes and Filters Red Hat JBoss Fuse 6.0 | Red Hat Customer Portal*. 2024. URL: https://access.redhat.com/documentation/en-us/red_hat_jboss_fuse/6.0/html/implementing_enterprise_integration_patterns/msgsys-pipes (viitattu 02. 02. 2024).

[3] *Apache Parquet*. 2023. URL: https://parquet.apache.org/ (viitattu 11. 08. 2023).

[4] *Application integration patterns for microservices: Orchestration and coordination | AWS Compute Blog*. 2024. URL: https://aws.amazon.com/blogs/compute/application-integration-patterns-for-microservices-orchestration-and-coordination/ (viitattu 02. 02. 2024).

[5] A. Barros, M. Dumas ja A. H. T. Hofstede. "Service Interaction Patterns". *Lecture Notes in Computer Science* 3649 (2005), s. 302–318. ISSN: 1611-3349. DOI: 10.1007/11538394_20. URL: https://link.springer.com/chapter/10.1007/11538394_20.

[6] *Enterprise Integration Patterns Using Mule | MuleSoft Documentation*. 2024. URL: https://docs.mulesoft.com/mule-runtime/latest/understanding-enterprise-integration-patterns-using-mule (viitattu 02. 02. 2024).

[7] D. L. Freire, R. Z. Frantz, F. Roos-Frantz ja S. Sawicki. "Survey on the run-time systems of enterprise application integration platforms focusing on performance". *Software: Practice and Experience* 49 (3 maaliskuu 2019), s. 341–360. ISSN: 1097-024X. DOI: 10.1002/SPE.2670. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/spe.2670%20https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2670%20https://onlinelibrary.wiley.com/doi/10.1002/spe.2670.

[8] E. Gamma, R. Helm, R. Johnson ja J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994, 139ff. ISBN: 0-201-63361-2. URL: https://archive.org/details/designpatternsel00gamm/page/139.

[9] G. Hohpe. *Loan Broker Implementation with AWS Step Functions - Enterprise Integration Patterns*. 2021. URL: https://www.enterpriseintegrationpatterns.com/ramblings/loanbroker_stepfunctions.html (viitattu 03. 05. 2024).

[10] G. Hohpe. *Modern Examples for Enterprise Integration Patterns - Enterprise Integration Patterns*. 2017. URL: https://www.enterpriseintegrationpatterns.com/ramblings/eip1_examples_updated.html (viitattu 03. 05. 2024).

[11] G. Hohpe. *Overview - Enterprise Integration Patterns 2*. 2017. URL: https://www.enterpriseintegrationpatterns.com/patterns/conversation/index.html (viitattu 02. 03. 2024).

[12] G. Hohpe. *Serverless Integration Patterns on Google Cloud Functions - Enterprise Integration Patterns*. 2017. URL: https://www.enterpriseintegrationpatterns.com/ramblings/google_cloud_functions.html (viitattu 03. 05. 2024).

[13] G. Hohpe. *Serverless Loan Broker @ AWS, Part 4: Automation - Enterprise Integration Patterns*. 2021. URL: https://www.enterpriseintegrationpatterns.com/ramblings/loanbroker_automation.html (viitattu 03. 05. 2024).

[14] G. Hohpe. *Serverless Loan Broker @ AWS, Part 5: Integration Patterns with CDK - Enterprise Integration Patterns*. 2022. URL: https://www.enterpriseintegrationpatterns.com/ramblings/loanbroker_cdk.html (viitattu 03. 05. 2024).

[15] G. Hohpe. *Serverless Loan Broker @ GCP - Enterprise Integration Patterns*. 2022. URL: https://www.enterpriseintegrationpatterns.com/ramblings/loanbroker_gcp_workflows.html (viitattu 03. 05. 2024).

[16] G. Hohpe ja B. Woolf. "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions: Hohpe, Gregor, Woolf, Bobby: 9780321200686: Amazon.com: Books". *Addison-Wesley Professional; 1 edition* (2004), s. 736. URL: https://books.google.com.cu/books?hl=es&lr=&id=bUlsAQAAQBAJ&oi=fnd&pg=PR7&dq=HOHPE,+G.+AND+WOOLF,+B.+Enterprise+Integration+Patterns:+Designing,+Building,+and+Deploying+Messaging+Solutions.+edited+by+I.+PEARSON+EDUCATION.+Edtion+ed.+Boston,+MA,+USA:+Addis.

[17] G. Hohpe ja B. Woolf. *GitHub - spac3lord/eip: Code for modern EIP examples*. 2023. URL: https://github.com/spac3lord/eip (viitattu 04. 05. 2024).

[18] *Home - Enterprise Integration Patterns*. 2024. URL: https://www.enterpriseintegrationpatterns.com/ (viitattu 26. 01. 2024).

[19] *ISO/IEC 7498-1:1994 - Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*. URL: https://www.iso.org/standard/20269.html.

[20] P. Johannesson ja E. Perjons. "Design principles for process modelling in enterprise application integration". *Information Systems* 26 (3 toukokuu 2001). ISSN: 03064379. DOI: 10.1016/S0306-4379(01)00015-1.

[21] D. S. Linthicum. *Enterprise application integration*. 2000, s. 21–23.

[22] Mordorintelligence. *Enterprise Application Integration Market | Growth, Trends, Forecasts (2020 - 2025)*. 2020. URL: https://www.mordorintelligence.com/industry-reports/enterprise-application-integration-market (viitattu 11. 05. 2020).

[23] *Pipeline :: Apache Camel*. 2024. URL: https://camel.apache.org/components/4.0.x/eips/pipeline-eip.html (viitattu 02. 02. 2024).

[24] *Pipes and Filters pattern - Azure Architecture Center | Microsoft Learn*. 2024. URL: https://learn.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters (viitattu 02. 02. 2024).

[25] D. Ritter, N. May ja S. Rinderle-Ma. "Patterns for emerging application integration scenarios: A survey". *Information Systems* 67 (heinäkuu 2017), s. 36–57. ISSN: 0306-4379. DOI: 10.1016/J.IS.2017.03.003.

[26] O. Zimmermann, C. Pautasso, G. Hohpe ja B. Woolf. "A decade of enterprise integration patterns: A conversation with the authors". *IEEE Software* 33 (1 2016), s. 13–19. ISSN: 07407459. DOI: 10.1109/MS.2016.11.

Liite A Sample Appendix

You can add one or more appendices to your thesis.

Liite B Instructions for LaTex

B.1 General Setup

In the HY-CS-main.tex file you will find the following STEPS 0–5. Below you can find related instructions.

STEP 0 – Access the thesis template

- Import the thesis template into a new Overleaf project. The easiest way to do it is to:
  - Obtain a zip file of the LaTeX template from the webpage of your programme.
  - Go to https://www.overleaf.com/edu/helsinki and login to Overleaf with your university credentials.
  - Go to the list of your projects at https://www.overleaf.com/project, click "New Project" and "Upload Project"., the projects under your account
  - Then upload the zip with the template.
  - You are now ready to write your thesis in Overleaf by editing the template, you can start by renaming the project.

STEP 1 – BSc or MSc thesis?

1. Select whether your are writing BSc (tkt) or MSc (csm for CS) thesis.
2. Select your language: `finnish`, `english`, or `swedish`.
3. If you are writing MSc select your line / track.

STEP 2 – Set up your personal information

1. Specify the title of your thesis with `\title{}`.
2. Specify your name to the author field with `\author{}`.
3. Specify the names of your supervisors of the thesis with `\supervisors{}`.

4. Specify the keywords of the thesis with `\keywords{}`.
5. Specify the ACM classification terms of the thesis with `\classification{}`. See https://dl.acm.org/ccs for more information.

STEP 3 – Write your abstract

- You can have the abstract in multiple languages with the `otherlanguages` environment. The example below shows how to provide an English abstract:

```
\begin{otherlanguage}{english}
\begin{abstract}
Your abstract text goes here.
\end{abstract}
\end{otherlanguage}
```

STEP 4 – Writing your thesis

1. There are some minimal contents and instructions below
2. Remove, or comment out, this appendix from your thesis.

STEP 5 – Set your bibliography style

- The default is Author-Year style (Einstein, 1905), but it can be easily changed to numbered [1] or alphabetical [Ein05] , as the examples of these are in comments.
- Discuss the style to use with your supervisor.

B.2 Bibliography in Latex

The bibliography is defined in a separate `.bib` file. For this template, it is named `bibliography.bib` and includes the content show in Figure B.1.

Chapter Bibliography lists all the works that you refer to in your text. You refer to the works in the bibliography using an appropriate *citation key*.

References are done using `\citep{einstein}`, which generates in text a citation formatted according to the selected style [**einstein**], or `\citep{latexcompanion,knuth99}`, which

generates [**latexcompanion**, **knuth99**]. As examples of a different kinds of citations (see how these look in the Latex source), we can write [**einstein**] to refer to the work written by **einstein** in **einstein**, because the work by **einstein** appears in the bilbliography included in this template.

Note that there are different possible styles for the bibliography and citation keys. Consult your supervisors on the chosen style – and once you arrive at a preferred style, use it consistently throughout the thesis.

```
@article{einstein,
    author =       "Albert Einstein",
    title =        "{Zur Elektrodynamik bewegter K{\"o}rper}. ({German})
        [{On} the electrodynamics of moving bodies]",
    journal =      "Annalen der Physik",
    volume =       "322",
    number =       "10",
    pages =        "891--921",
    year =         "1905",
    DOI =          "http://dx.doi.org/10.1002/andp.19053221004"
}

@book{latexcompanion,
    author    = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
    title     = "The \LaTeX\ Companion",
    year      = "1993",
    publisher = "Addison-Wesley",
    address   = "Reading, Massachusetts"
}

@book{knuth99,
    author    = "Donald E. Knuth",
    title     = "Digital Typography",
    year      = "1999",
    publisher = "The Center for the Study of Language and Information",
    series    = "CLSI Lecture Notes (78)"
}
```

Kuva B.1: Examples of bibliographic reference in .bib file.

B.3 Some instructions about writing in Latex

The following gives some superficial instructions for using this template for a Master's thesis. For guidelines on thesis writing you can consult various sources, such as university courses on scientific writing or your supervisors.

For more detailed instructions, just google, e.g., "Overleaf table positioning", and your

chances of finding good info are pretty good.

B.4 Figures

Besides text, here are simple examples how you can add figures and tables in your thesis. Remember always to refer to each figure in the main text and provide them with a descriptive caption.

Figure B.2 is an example of a figure in the document (see the source about how to add them).

Kuva B.2: University of Helsinki flame-logo for Faculty of Science.

B.5 Tables

Table B.1 gives an example of a table. Remember always to cite the table in the main text, table captions go on top of the table.

Taulukko B.1: Experimental results.

Experiment	1	2	3
A	2.5	4.7	-11
B	8.0	-3.7	12.6
$A + B$	10.5	1.0	1.6

Liite C Tutkielmapohjan käyttöohjeet

C.1 Ensiaslkeleet

`HY-CS-main.tex` tiedosto sisältää viisi askelta STEPS 0–5. Alla on kuvattu, mitä nämä askeleet tarkoittavat ja miten niitä seuraamalla luot pohjan tutkielmallesi.

STEP 0 – Kopioi tutkielmapohja

- Hae tutkielmapohja uuteen Overleaf-projektiin. Tämä käy helpoiten seuraavasti:
  - Lataa Latex-pohjan zip-tiedosto koulutusohjelman sivuilta.
  - Mene osoitteeseen `www.overleaf.com/edu/helsinki` ja kirjaudu Overleafiin yliopiston tunnuksillasi.
  - Overleafissa (`https://www.overleaf.com/project`), klikkaa "New Project" and "Upload Project".
  - Valitse lataamasi tutkielmapohjan zip-tiedosto.
  - Nyt voit lähteä kirjoittamaan tutkielmaasi suoraan pohjaan, voit aloittaa esim. vaihtamalla projektin nimen.

STEP 1 – BSc vai MSc tutkielma?

1. Valitse (tiedostossa `HY-CS-main.tex`) oletko tekemässä BSc (tkt) vai MSc (csm tietojenkäsittely) tutkielmaa.
2. Valitse kieli jolla kirjoitat tutkielman: `finnish`, `english` tai `swedish`.
3. Jos olet kirjoittamassa maisterintutkielmaa, valitse linja/opintosuunta.

STEP 2 – Aseta henkilökohtaiset tietosi

1. Kirjoita alustava otsikko tutkielmallesi: `\title{}`.
2. Kirjoita oma nimesi kohtaan `\author{}`.
3. Lisää ohjaajien nimet `\supervisors{}`.

4. Määrittele avainsanat `\keywords{}`.
5. Määritä tutkielmasi ACM luokittelutermit `\classification{}`. Ks. lisätietoa: https://dl.acm.org/ccs.

STEP 3 – Kirjoita tiivistelmä

Voit kirjoittaa tiivistelmän (koko tiivistelmäsivu) eri kielillä `otherlanguages`-ympäristön avulla. Alla esimerkki jolla kirjoitat englanninkielisen tiivistelmän muulla kuin englannin kielellä kirjoitettuun tutkielmaan:

```
\begin{otherlanguage}{english}
\begin{abstract}
Your abstract text goes here.
\end{abstract}
\end{otherlanguage}
```

STEP 4 – Kirjoita tutkielma

1. Kirjoittamisesta Latexilla löydät hieman ohjeita alempaa.
2. Poista tämä liite ja muu ohjeistus tutkielmastasi, esim. kommentoimalla.

STEP 5 – Aseta kirjallisuuslähdeluettelon tyyli

- Oletustyylin tekijä-vuosi, eli (Einstein, 1905), voit vaihtaa viittaustyylin (tiedostossa `HY-CS-main.tex`) helposti (eri mallit kommentoituna) esim. numeroituun [1], tai aakkostyyliin [Ein05]. Lisää ohjeita liittyen viittaustyylin säätämiseen BibTEXissä löytyy verkosta: https://ctan.org/pkg/biblatex
- Sovi käytettävä tyyli ohjaajasi kanssa.

C.2 Kirjallisuusviitteet Latexissa

Kirjallisuuslähteet ylläpidetään erillisessä .bib-tiedostossa. Tässä tutkielmapohjassa käytetyt kirjallisuuslähteet, joista esimerkkejä kuvassa C.1, löytyvät tiedostosta `bibliography.bib`.

```
@article{einstein,
    author =       "Albert Einstein",
    title =        "{Zur Elektrodynamik bewegter K{\"o}rper}. ({German})
        [{On} the electrodynamics of moving bodies]",
    journal =      "Annalen der Physik",
    volume =       "322",
    number =       "10",
    pages =        "891--921",
    year =         "1905",
    DOI =          "http://dx.doi.org/10.1002/andp.19053221004"
}

@book{latexcompanion,
    author    = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
    title     = "The \LaTeX\ Companion",
    year      = "1993",
    publisher = "Addison-Wesley",
    address   = "Reading, Massachusetts"
}

@book{knuth99,
    author    = "Donald E. Knuth",
    title     = "Digital Typography",
    year      = "1999",
    publisher = "The Center for the Study of Language and Information",
    series    = "CLSI Lecture Notes (78)"
}
```

Kuva C.1: Esimerkkejä kirjallisuuslähteiden kuvaamisesta .bib-tiedostossa.

Viitteet kirjallisuuslähteisiin muodostetaan komennolla `\citep{einstein}`, josta generoituu tekstiin valitun viittaustyylin mukaisesti muotoiltu viite [**einstein**], tai `\citep{latexcompanion,knu` josta tekstiin puolestaan generoituu [**latexcompanion**, **knuth99**]. Voit esimerkiksi kirjoittaa [**einstein**] viitataksesi julkaisuun, jonka on kirjoittanut **einstein** vuonna **einstein**, kun vain lähde **einstein** on oikein lisättynä kirjallisuuslähdetiedostossa (katso miltä nämä näyttävät Latex lähdekoodissa).

Tekstissä viitatut kirjallisuuslähteet tulevat automaattisesti viiteluetteloon. Kirjallisuuslähteiden tietojen oikeellisuus ja yhdenmukaisuus .bib-tiedostossa vaikuttavat luonnollisesti siihen, miten tiedot tutkielmassa näyttäytyvät. Tämä on syytä huomioida, sillä esim. verkosta valmiiksi BibTEX muodossa löytyvien tietojen täydellisyyten tai samanmuotoisuuteen ei pidä sokeasti luottaa.

Keskustele viittaustyylin valinnasta ohjaajan kanssa.

C.3 Joitain ohjeita Latexilla kirjoittamiseen

Seuraavassa on joitain ohjeita tämän tutkielmapohjan käyttöön maisterintutkielmassa. Kirjoittamisohjeita löytyy useasta eri lähteestä. Voit esimerkiksi tutustua kandidaatintutkielman ohjeisiin. Ohjaajan kanssa on hyvä keskustella aikaisessa vaiheessa työn rakenteesta.

Yksityiskohtaisia ohjeita Latexin käyttämäsestä saa parhaiten hakemalla verkosta, esim. haku englanniksi "Overleaf table positioning"tuottaa oletettavasti aika toimivan vastauksen.

C.4 Kuvat

Kuva C.2 toimii esimerkkinä kuvan lisäämisestä työhön (katso tarkemmin mallia Latex lähdekoodista). Muista myös viitata jokaiseen kuvaan tekstissä.

Kuva C.2: Helsingin yliopiston logo matemaattis-luonnontieteellisen tiedekunnan värein.

C.5 Taulukot

Taulukossa C.1 on esimerkki kokeellisten tulosten raportoinnista taulukkona. Muista myös viitata jokaiseen taulukkoon tekstissä.

Taulukko C.1: Kokeelliset tulokset.

Koe	1	2	3
A	2.5	4.7	-11
B	8.0	-3.7	12.6
$A + B$	10.5	1.0	1.6

Liite D Johdanto

Kaikessa julkaistavaksi tarkoitetussa tekstissä kirjoittajan luomisen ja esitystavan vapautta rajoittavat monet ohjeet ja tarkatkin määräykset.

Parhaimmillaan lukijalle ja kirjoittajalle yhteinen, tuttu säännöstö luo eräänlaisen tukiverkoston, joka tukee sanoman siirtymistä vääristymättä. Kirjoituksen lukija löytää kirjoituksesta helpommin olennaisen sisällön, jos kirjoituksen ulkoasu ja sisällön rakenne vastaavat hänen tottumuksiaan. Sama koskee myös kirjoittajaa. Noudattaessaan valmista esitystapamallia kirjoittajan ei tarvitse käyttää aikaansa itse työn kannalta toissijaisten seikkojen miettimiseen, vaan hän voi keskittyä hiomaan tekstin sisältöä. Siksi kannattaa harjoitella myös työn ulkoasua koskevien ohjeiden noudattamista, vaikka omasta mielestään osaisikin valita esitykselleen ohjetta paremman muodon.

Tämä kirjoitus on tarkoitettu Helsingin yliopiston Tietojenkäsittelytieteen osastoon alempien opinnäytteiden ja harjoitusten ulkoasun ja rakenteen ohjeeksi. Ohje soveltuu siten kandidaatintutkielman kirjoittamisen kurssille, ohjelmistotuotantoprojekteihin, seminaareihin ja pro gradu -tutkielmiin. (Kirjoitus on päivitetty uusintapainos aiemmista ohjeista, jotka kurssin luennoijat ovat laatineet [**erkio01**, **erkiomakela96**, **erkio94**, **verkamo92**].)

Tyylimäärittely on saatavissa pdflatex- ja word-versiona. Tyylimäärittelyitä valitessa on huomattava ohjeet tekstien syöttöön liittyvästä koodauksesta (UTF8,ISO 8859-15). Tämän kirjoituksen tukena sopivat käytettäväksi tavanomaiset latex- tai word-oppaat.

Liite E Kirjoituksen rakenne

Tarkastellaan aluksi tieteelliseltä tekstiltä odotettuja kirjoituksen osia. Samoihin asioihin on luonnollisesti syytä kiinnittää huomiota myös muussa teknisessä kirjoittamisessa. Huomattakoon, että tämä teksti ei ole tieteellinen teksti, eikä siten itse sisällä kaikkia niitä elementtejä, jotka tieteellisen tekstin sisällölliseen antiin kuuluvat. Tällaisia puutteita ovat esimerkiksi johdannon tutkimuskysymyksen asettelun puuttuminen sekä arvoivan materiaalin puute tekstin lopussa, sekä yhteenvedon latteus. Teksti rajoittuu siten otsikkonsa mukaisesti vain tekniseen sisällön asetteluun.

E.1 Tiivistelmä

Tiivistelmäsivu sisältää seuraavat osat: työn bibliografiset tiedot, tiivistelmäteksti, aiheluokat ja avainsanat. Bibliografiset tiedot koostuvat työn otsikosta, tekijän nimestä, julkaisupaikan tiedoista, julkaisuajankohdasta ja sivumäärästä.

Tiivistelmäteksti on lyhyt, yleensä yhden kappaleen mittainen (maksimissaan noin 100 sanaa) selvitys kirjoituksen tärkeimmästä sisällöstä: mitä on tutkittu, miten on tutkittu ja mitä tuloksia on saatu.

Aiheluokat kuvataan ACM Computing Classification System -luokituksen (CCS) luokituksen mukaisesti. Luokittelussa käytetään täysia polkuja juurisolmun CCS osoittamista lähtöposteistä lehtisolmuihin. Polkuja voi antaa 1-3 aihepiirien soveltuvuuden mukaan, mitä alempi opinnäyte, sen vähemmän polkuja se tarvitsee. Poluissa tasot erotetaan toisistaan nuolella eteenpäin. Kun polun nimisanoja arvioidaan suhteessa työn sisältöön, merkitään boldface-fontilla tärkein termi, italics-fontilla toiseksi tärkein. Näin menetellään, mikäli jotkin termeistä ovat olennaisesti paremmin kuvaavia kuin muut polun termit. Nimettyjen polkujen lisäksi lukija voi siten tarkastella lisäulottuvuutena myös tärkeiksi merkittyjen termien joukkoa sinänsä. Avainsanoiksi valitaan kirjoituksen sisältöä keskeisesti kuvaavia käsitteitä.

E.2 Johdanto

Johdannon tarkoituksena on kertoa yleiskielisesti työn tavoite. Kerrotaan (kuten tiivistelmässäkin, mutta laveammin), mitä on tutkittu, miten on tutkittu ja mitä tuloksia on saatu. Jotta kysymyksenasettelu ja tulokset on lukijan helppo oikein tulkita on syytä aloittaa johdanto asettelemalla tutkimus asiayhteyteensä, esimerkiksi kertomalla aluksi, minkälaisessa yhteydessä tarkasteluun otettavat haasteet esiintyvät ja keiden on ratkaisuista tarkoitus hyötyä.

Johdannon pituus määräytyy suhteessa koko kirjoitelman pituuteen. Parisivuinen kirjoitus ei erikseen otsikoitua johdantoa kaipaa, sillä se itsessään on laajennettu tiivistelmä. Kymmensivuisen kirjoituksen johdanto voi olla vaikkapa sivun tai puolentoista mittainen. Pro gradu -tutkielman 50-70-sivuiseen kokonaisuuteen tuntuu 2-4-sivuinen johdanto kohtuulliselta.

Johdanto kertoo siis lyhyessä, yleistajuisessa muodossa koko kirjoitelman kysymyksenasettelun, juonen sekä tulokset ja johtopäätelmät. Tämän luettuaan lukija voi päätellä, haluaako syventyä asiaan tarkemmin lukemalla koko kirjoituksen.

E.3 Käsittelyluvut

Käsittelylukujen työnjako määräytyy käsiteltävän asian luonteen mukaisesti. Lukijan ohjailemiseksi kukin pääluku kannattaa aloittaa lyhyellä kappaleella, joka paljastaa mikä kyseisen luvun keskeisin sisältö on ja kuinka aliluvuissa asiaa kehitellään eteenpäin. Erityisesti kannattaa kiinnittää huomiota siihen, että lukijalle ilmaistaan selkeästi miksi kutakin asiaa käsitellään ja miten käsiteltävät asiat suhtautuvat toisiinsa.

Jäsentelyongelmista kielivät tilanteet, joissa alilukuja on vain yksi, tai joissa käytetään useampaa kuin kahta tasoa (pääluku ja sen aliluvut). Kolmitasoisia otsikointeja saatetaan tarvita joissakin teknisissä dokumenteissa perustellusti, mutta nämä muodostavat poikkeuksen.

Perusohjeena on käyttää tekstin rakenteellisesti painokkaita paikkoja, kuten lukujen avauksia ja teksikappaleiden aloitusvirkkeitä juonenkuljetukseen ja informaatioaskeleiden sitomiseen toisiinsa. Tekstikappaleiden keskiosat, samoin kuin lukujen keskiosat selostavat asiaa vähemmän tuntevalle yksityiskohtia, kun taas aihepiirissä jo sisällä olevat lukijat voivat alkuvirkkeitä silmäilemällä edetä tekstissä tehokkaasti eksymättä tarinan juonesta.

Kullakin kirjoittajalla on oma temponsa, joka välittyy lukijalle tekstikappaleiden pituudessa ja niihin sisällytettyjen ajatuskulkujen mutkikkuudessa. Kussakin tekstikappaleessa pitäisi pitäytyä vain yhdessä informaatioaskelessa tai olennaisessa päättelyaskelessa, muuten juonen seuraaminen käy raskaaksi olennaisten lauseiden etsiskelyksi. Yksivirkkeisiä tekstikappaleita on syytä varoa.

E.4 Lähdeviittausten käyttö

Olennaisia opittavia asioita viittaustekniikoissa ovat viitteen paikka tekstissä, oikea lähdeluettelojärjestys valitun viitetyylin parina sekä taito ja tahto noudattaa annettua tyylimääräystä. Väitöskirjoissa ja lehti- tai konferenssiartikkeleissa tekstin hyväksyminen riippuu myös näiden yksityiskohtien asianmukaisesta käsittelystä. Tästä syystä laitoksella nähdään tarpeelliseksi opiskelijoiden tutustua edes pinnallisesti myös muihin tyylilajeihin ja oppia käyttämään automatisoituja muotoilutyökaluja tehokkaasti, jolloin tyylimuutokset ovat tehokkaita.

Lähdeviitteet sijoitetaan aina virkkeen sisäpuolelle. Siten esimerkiksi tekstikappaleen lopussa irrallaan oleva viite ei ole asiallinen. Tilanne ei muutu, vaikka viite sujautettaisiin tekstikappaleen viimeisen virkkeen sisään. Lähdeviittauksen yhteyteen merkitään mukaan tarkentavat sivunumerot, mikäli lukijan olisi työlästä löytää asianomainen kohta viitatusta lähteestä.

Tehokkaita viitteensijoittelupaikkoja ovat esimerkiksi uuden käsitteen nimeämiskohta ja virkkeen loppu kun kyseessä on lähteestä lainattu väite. On myös muistettava lainausmerkkien käyttö silloin kun tehdään suoria lainauksia.

Tekstin jäsentelyn on tuotava selkeästi esiin, mihin asiaan viite liittyy. Samalla tulee ymmärrettäväksi se, kuinka pitkään tekstikatkelmaan ko. viite liitetään. Ei ole siten asiallista aloittaa lukua nimeämällä yhtä tai useampaa lähdettä luvun taustaksi, vaan viitteitä on kiinnitettävä täsmällisemmin väitteisiin ja käsitteisiin. Luvun avaus viitetiedolla voi olla oire myös suuremmasta ongelmasta: lähderiippuvuudesta. Aloitteleva kirjoittaja helposti toistaa lähteestä oppimaansa ilman että tarpeellinen analysointi ja prosessointi suhteessa muuhun opittuun olisi vielä tapahtut.

Viitteillä ja sanamuodolilla on myös tuotava selkeästi esiin se, mikä teksissä on lainattua ja mikä oman pohdinnan ja valikoinnin tulosta.

Lähdeviittauksiin käytetään Tietojenkäsittelytieteen osastolla numeroitua tyyliä ja APA-

tyyliä, valinnan näiden välillä tekevät kunkin ryhmän valvoja ja ohjaaja yhdessä. Numeroitu tyyli on esimerkiksi IEEE- ja ACM-julkaisuissa yleisesti käytetty ja puolustaa siten paikkaansa. APA-tyyli on poikkeuksellinen ns. kovissa tieteissä, mutta monet valvojista pitävät siitä sen luettavuuden vuoksi. Numeroita joutuu nimiä useammin tarkistamaan lähdeluettelosta, sillä tarkastus- ja arviointiprosessiin kuuluu arvioida myös lähteiden valitaa ja niiden käyttötapaa.

E.5 Yhteenveto

Yhteenveto vaatimattomimmillaan on vain lyhyt kertaus kirjoituksen keskeisistä asioista. Arvokkaamman yhteenvedon saa aikaan kommentoimalla työn tulosten arvoa, työn liittymistä ympäristöön ja tulevaisuudennäkymiä. Tällaiset arviot huolellisesti perusteltava.

E.6 Lähdeluettelon laatiminen

Tieteellisen kirjoittamisen kurssin töiden lähdeluetteloiden laatimisessa noudatetaan seuraavia ohjeita.

Niiden taustalla on kaksi keskeistä pyrkimystä: tehdä viitatun lähteen hankkiminen luettavaksi mahdollisimman helpoksi ja ilmaista, millaisen arviointiprosessin läpi käyneeseen kirjoitukseen vedotaan. Näistä syistä

- lähdeviitteen tulee aina olla niin tarkka, että lähde on sen perusteella tunnistettavissa ja löydettävissä luetteloista ja kirjastoista,
- erityyppisten lähteiden (kirjat, konferenssit, lehdet) on erotuttava toisistaan ja
- luettelon eri osien tulee olla mahdollisimman yhdenmukaisia, erityisesti lähdetyypin sisällä.

Riippumatta käytettävästä viitetyylistä, lähteet ovat Tietojenkäsittelytieteen osaston opinnäytteiden lähdeluetteloissa tekijän nimen mukaisessa aakkosjärjestyksessä, saman tekijän (tekijäryhmän) työt julkaisuajan mukaisessa järjestyksessä. Jos jollakin lähteellä ei ole henkilötekijää, se aakkostetaan julkaisun nimen mukaisesti.

Kustakin lähteestä annetaan seuraavat tiedot, edelleen viitetyylistä riippumatta:

- (tarvittaessa lähdeviitelyhenne).
- tekijän tai tekijöiden nimet (sukunimi, etunimien alkukirjaimet) alkuperäisessä järjestyksessään; jos tekijöitä on enemmän kuin kolme, voidaan toimia siten, että vain ensimmäinen tekijä nimetään ja muiden tilalle kirjoitetaan *et al.*
- julkaisun tai artikkelin nimi alkuperäisessä muodossaan
- julkaisupaikan tiedot:
  - kirjasta: kustantaja, julkaisupaikka (voidaan jättää pois, jos kyseessä on tunnettu kustantaja), vuosi ja
  - lehtiartikkelista: lehden nimi, volyymi, numero, vuosiluku ja kuukausi (suluissa),
  - artikkelikokoelmassa (esim. konferenssijulkaisussa) ilmestyneestä artikkelista:
    - kokoelman nimi, toimittaja, kustantaja, julkaisupaikka ja vuosi *tai*
    - konferenssin nimi, järjestäjä, paikka ja aika,
  - raportista: julkaisusarja, raportin numero, julkaisupaikka, julkaisija ja vuosi ja
  - www-lähteestä: verkko-osoite, voimassaoloajankohta, mahdollisesti viittausajankohta hakasuluissa
- sivunumerot, mikäli lähteenä käytetty julkaisu on artikkeli tai kokoomateoksen itsenäinen luku.

Normaaliin suomalaiseen tapaan artikkelin nimessä ainoastaan ensimmäinen sana kirjoitetaan isolla alkukirjaimella, sen sijaan konferenssien ja kokoelmajulkaisujen nimissä käytetään isoa alkukirjainta jokaisen sanan alussa (artikkelisanoja ja prepositioita lukuunottamatta). Katso mallia oheisista esimerkeistä. Kokoelman nimen edessä on syytä selvyyden vuoksi käyttää sanaa *Teoksessa*, paitsi kun on kysymys konferenssijulkaisusta, jonka nimi alkaa lyhenteellä *Proc.* (sanasta Proceedings). Tällöin ei tarvita mitään täydennystä. Tämän eron näkee esimerkiksi vertaamalla lähdeviitteiden "[**dantowsley90**]" ja "[**gannonetal89**]" ulkoasuja.

WWW-lähteiden käytössä on syytä muistaa, että verkossa julkaisukynnys on olematon. Kannattaa siten keskittyä tunnettujen tieteellisten kustantajien julkaisuihin ja niihin teknisiin standardeihin, joille WWW on ainoa julkaisukanava. Mikäli sama julkaisu on saatavissa myös perinteisessä muodossa, viitataan ensisijaisesti siihen ja käytetään verkko-osoitetta lisätietona. Lähdeluettelossa on annettu esimerkit useita kanavia julkaistusta

kirjoituksesta [**abiteboul**, **dietinger**] sekä pelkästään WWW-julkaisuna leviävästä standardista [**bray**].

Erityisesti varoitetaan Wikipedian käytöstä tieteellisessä tekstissä. Vaikka sen avulla on helppo alustavasti tutustua joihin aihepiireihin ja asiantuteva lukija voisi teksin kelvolliseksi tiettynä hetkenä hyväksyäkin, ei se foorumina millään lailla täytä tieteellisesti vertaisarvoidun tutkimusfoorumin kriteerejä. Jos Wikipedia-artikkelia ei mitenkään malta ajankuvana olla mainitsematta, käytettäköön jotain muuta kuin lähdeviitetekniikkaa tähän taiteelliseen otteeseen, vaikkapa alaviitteitä. Olennaista silloinkin on, että tieteellinen sisältö ei tule tällä korvatuksi vaan sen puute korostetuksi.

WWW-lähteeseen viittaamisessa pätevät samat periaatteet kuin perinteisiin lähteisiin viitattaessa: lähdeviitteessä ilmaistaan otsakkeet, kirjoittajat, toimittajat js muut seikat. Eroa on ainoastaan verkko-osoitteen ja sen voimassaoloajankohdan ilmaisemisessa. Mikäli lähde on julkaistu ainoastaan verkossa, voidaan web-osoitetta (URL) käyttää vastaavasti kuin perinteisen julkaisun paikannusinformaatiota (lehden ja se numeron julkaisutiedot). Lähdeluettelossa on WWW-viittausten yhteydessä aina syytä ilmaista päivämäärä, jolloin linkin voimassaolo ja lähteen sisältö on tarkastettu. Esimerkkeinä verkkoviitteistä soveltuvat seuraavat:

- Gergen, Kenneth (1999) Narrative, Moral Identity and Historical Consciousness: a Social Constructionist Account. http://www.swarthmore.edu/SocSci/kgergen1/text3.html. Haettu 11.6.1999.
- Ritala-Koskinen, Aino and Valokivi, Heli (2006) The Role of Development Skills in Social Work Practice Education in Finland. Social Work and Society, The International Online-Only Journal 4(2006)1. http://www.socwork.net/2006/1/series/transition/ritalakoskinenv Viitattu 30.8.2006.
- Heinisuo, Rami and Ekholm, Kai (1997) Elektronisen viittaamisen opas. Jyväskylän yliopiston kirjaston julkaisuja n:o 40. Jyväskylä: Jyväskylän yliopiston kirjasto. http://www.pori.tut.fi/ multisil/evo/. Viitattu 29.8.2006.

Kirjoituksen lähdeluettelossa luetellaan täsmälleen ne lähteet, joihin viitataan kirjoituksen tekstiosassa. Tämän kirjoituksen lähdeluettelo on tarkoitettu lähinnä esitystavan esimerkiksi, mistä syystä siinä on ”ylimääräisiä” lähteitä.

Pääsääntöisesti julkaisun tai artikkelin nimen perään tulee piste, samoin kunkin lähteen bibliografisten tietojen perään. Muut tiedot erotetaan toisistaan pilkulla. Useimmissa ta-

pauksissa voidaan noudattaa teknisten välineiden antamaa mallia, sillä edellytyksellä, että ylläolevat vaatimukset muuten täyttyvät.

Liite F Ulkoasulliset seikat

Tässä luvussa käsitellään yleisimpiä tekstin tekniseen esittämiseen liittyviä seikkoja.

F.1 Työn osien järjestys

Kirjoituksen alussa on aina erillinen, mallin mukainen kansilehti. Toisena sivuna on tiivistelmäsivu, sen jälkeen sisällysluettelo (yksi tai useampia sivuja) ja sitten varsinainen teksti. Sivunumerointi aloitetaan vasta ensimmäiseltä tekstisivulta (arabialaisella ykkösellä). (Tarkat jättävät ykkössivun numeromerkittä.) Sisällysluetteloon merkitään kaikki (numeroidut) otsikot ja vastaavat sivunumerot. Monet tekstinkäsittelyjärjestelmät muodostavat itse sisällysluettelon, jolloin kirjoittajan ei tarvitse huolehtia luettelon sivunumeroiden päivittämisestä tekstin kehittyessä. Sisällysluettelosivu ja sitä edeltävät sivut voidaan haluttaessa numeroida erikseen (roomalaisin numeroin) esimerkiksi tämän mallin mukaisesti.

Varsinaisen tekstin jäljessä, mutta itse työhön kuuluvana on ensimmäisenä lähdeluettelo, jonka otsikkoa ei numeroida. Lähdeluettelon jälkeen sijoitetaan mahdolliset liitteet, jotka otsikoidaan ja varustetaan sisäisillä sivunumeroilla.

Mikäli kuvista, algoritmeista ja taulukoista halutaan tehdä yhtenäinen luettelo, sijoitetaan luettelot sisällysluettelon jälkeen. Luetteloiden käyttöarvosta on eriäviä mielipiteitä, joten niiden laatimiseen ei varsinkaan ilman tekstinkäsittelyjärjestelmän tukea kannata ryhtyä ilman tarkastajan erityistä toivetta.

Mikäli kirjoitukseen erityissyistä halutaan liittää aakkosellinen hakemisto, sijoitetaan se lähdeluettelon jälkeen ennen liitteitä. Indeksi merkitään sisällysluetteloon samoin kuin lähdeluettelo (numeroimaton luku). Mikäli indeksin tekemiseen ryhdytään, on syytä käyttää tekstinkäsittelyjärjestelmän tarjoamaa automatiikkaa.

Teksin luonnollisen juonenkuljetuksen mukana esiin tulevien käsitteiden määrittelyjen sijasta ei pidä yrittää sen enempää pakata kaikkia määritelmiä johdantoon kuin laatia johdantoa ennen käsitelistaa tai lyhenteiden selityslistaa. Kumpikaan ei sovi tavanomaiseen argumentoivaan tieteelliseen tekstityyliin, vaikka teknisessä yhteydessä niillä liitteinä voi olla lisäarvoa.

F.2 Tekstin yleinen sijoittelu

Lopullinen tutkielmaversio voi olla yksi- tai kaksipuoleiseksi aseteltua ja riviväliltään 1,5 tai 1. Erityyppisissä teksteissä haasteet ja asetteluvaatimukset voivat olla erilaiset. Erota kappaleet toisistaan yhdellä tyhjällä rivillä tai käytä tekstinkäsittelytyökalujen ominaisuuksia hyödyksesi ja määrittele tekstikappaleiden väliin jäävä tila hieman normaalia riviväliä suuremmaksi.

Kirjoituksen lukujen, kuvien ja taulukoiden erottumisen kannalta tärkein keino on riittävän tilan käyttö niiden ympärillä. Kuvan ja nimekkeen tulee olla selkeästi yksi kokonaisuus, joka eroaa muusta tyhjän tilan rajaamana. Kuvan tai taulukon on aina numerointinsa ja nimekkeensä kanssa mahduttava yhdelle sivulle tai varmasti kaksipuolisena paperidokumenttina tarkasteltavassa tekstissä aukeamalle. Kuvissa fonttikoko ei saa alittaa 8 pistettä.

Jos uusi luku tulisi alkamaan aivan sivun alareunasta (vain yksi tai kaksi riviä varsinaista tekstiä), aloita mieluummin uusi sivu. Jokaista uutta lukua ei kuitenkaan ole tarpeen — etenkään lyhyessä kirjoituksessa — aloittaa uudelta sivulta: jos kirjoituksessa on paljon melkein tyhjiä sivuja, lukija voi epäillä, että kirjoittaja on yrittänyt saada kirjoituksensa näyttämään pitemmältä kuin se onkaan. Tyhjää tilaa kannattaa käyttää hyödyksi myös kuvien ja taulukoiden yhteydessä. Erityisesti jos kirjoituksessa käytetään kauttaaltaan samaa tekstityyppiä, tyhjät rivit ovat välttämättömiä erottamaan esimerkiksi tekstiä ja taulukkoa toisistaan. Tyhjä tila on halpaa, mutta se lisää selkeyttä ja luettavuutta.

F.3 Kuvat ja taulukot

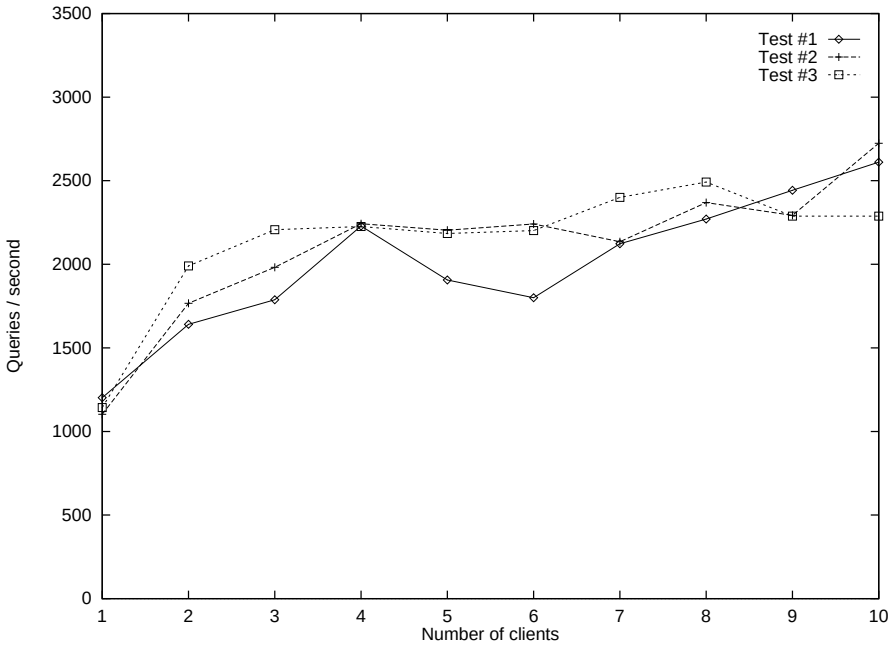
Kuva tai taulukko sijoitetaan mahdollisimman lähelle (ensimmäistä) tekstikohtaa, jossa siihen viitataan, ei kuitenkaan kyseistä viittausta aikaisemmaksi. Tekstissä on syytä myös kertoa, mitä kuvalla halutaan havainnollistaa. Kuvan voi lukea monella eri tavalla, joten lukijaa on ohjattava.

Kuvaa ei pidä sijoittaa välittömästi luvun otsikon alle, vaan on aloitettava tekstillä. Kuvaa ei pidä sijoittaa keskelle tekstikappaletta (saati virkettä), paitsi jos kuva tulee sivun alkuun tai loppuun eikä kappaleen jatkumisesta tule epäselvyyttä.

Kuvan ei aina tarvitse olla välittömästi viittaavan kappaleen perässä. Esimerkiksi viittauskohdan ja vasta seuraavalle sivulle mahtuvan kuvan väliin jäävää sivun loppuosaa ei jätetä

tyhjäksi. Kuvaa ei kuitenkaan pidä viedä seuraavaa sivua kauemmas viittauskohdasta.

Varsinaista kuvan esittämistä havainnollistaa kuva F.1. Huomiota on kiinnitettävä kuvan osien ja tekstimerkintöjen näkyvyyteen, kuvan numerointiin ja otsikointiin.



Kuva F.1: Kuvan elementit.

Kuvien kokoon on kiinnitettävä huomiota. Käytettyjen merkintöjen on oltava helposti luettavissa ja selkeät. Esimerkiksi suorituskykykäyriä esitettäessä akselit on nimettävä, asteikot merkittävä ja käytetyt yksiköt tuotava selkeästi esiin. Samankaltaisia asioita esitettäessä useammalla kuvalla on syytä käyttää samaa mittakaavaa vertailun helpottamiseksi.

Kuvan otsikko kirjoitetaan kuvan alle ja sen tulee olla mieluummin lyhyt ja ytimekäs kuin liian selittelevä. Samoin toimitaan taulukoiden otsikoinnissa.

Kuvat ja taulukot numeroidaan juoksevasti. Pitkissä teksteissä käytetään kaksitasoista numerointia (esimerkiksi Kuva 3.1) pääluvuittain, lyhyissä riittää yksitasoinen numerointi.

Kuva- ja taulukko-otsikoiden yhdenmukaiseen esitystyyliin on syytä kiinnittää huomiota, samoin mm. välimerkkeihin. Luontevaa on käyttää kuvatekstin lopussa pistettä, ovathan useimmat kuvateksteistä virkkeitä.

(Kuvien ja taulukoiden otsikointityyli vaihtelee kustantajittain ja julkaisuittain. Samoin

tuntuu suositeltava käytäntö Tietojenkäsittelytieteen laitoksen sisällä vaihtelevan taulukon otsikon sijainnin suhteen.)

F.4 Otsikot

Otsikoissa voi käyttää muusta tekstistä poikkeavaa kirjasintyyppiä, alleviivausta, suurempaa kirjasinkokoa tms. erotuskeinoa, yleensä kuitenkin vain yhtä näistä, koska kovin monta erilaista kirjasintyyppiä ja -kokoa tekee ulkoasusta helposti sekavan. Otsikoiden esitystavan on oltava johdonmukainen läpi koko kirjoituksen. Numeroimattomia ”ylimääräisiä” otsikoita ei tule yleensä käyttää.

F.5 Mallin käyttö

Voit käyttää tätä kirjoitusta mallina oman opinnäytteesi ulkoasua varten. Eri tekstinkäsittelyjärjestelmissä käytössä olevat yksityiskohdat kuten kirjasintyypit ja -koot ja rivivälit poikkeavat toisistaan, joten pienet poikkeamat ovat toki hyväksyttäviä.

Tieteellisen kirjoittamisen kurssin luennoilla ja liitteenä olevassa ohjeessa annetut töiden ohjeelliset sivumäärät koskevat työtä, joka vastaa ulkoasultaan tätä ohjetta (kirjasinkoko 12 pistettä). Tässä tekstissä keskimääräinen rivin pituus lienee noin 80 merkkiä ja sivun pituus 35-40 riviä. Sivumääriin lasketaan varsinaisen tekstiosuuden pituus ja lähdeluettelo (arabialaisin numeroin numeroitu osuus), ei kansilehteä, tiivistelmää eikä sisällysluetteloa. Sivumääräarviossa otetaan huomioon hyvin vajaat sivut, joita syntyy paljon lyhyiden lukujen ja taittotyyliin määritellyn luvun avauksen pakottaminen oikeanpuolimmaiselle sivulle.

Liite G Yhteenveto

Tämän kirjoituksen tarkoituksena on toimia muistilistana eräistä esitystavallisista säännöistä, joihin harjoitusten ja tutkielmien kohdalla on syytä kiinnittää huomiota.

Annetut ohjeet on laitoksen henkilökunta muotoillut yhdessä keskustellen ja noudattaen oman tieteenalansa perinteitä. Eri erikoistumisaloilla ja erilaisillaa määräävässä asemassa olevissa julkaisufooruilla käytänteet vaihtelevat ja nuorten tutkijoiden onkin tiedostettava ero yleisten sisältöohjeiden ja teknisten muotoilusääntöjen välillä. Aina tekstin valmistuessa on tarkastettava erikseen, täyttääkö se annetut pituusrajoitteet ja vastaako se annettuja muotoiluohjeita, olivatpa ne kuinka pikkutarkkoja tahansa. Tarkasta sääntöjen noudattamisesta syntyy yhteinäisyyttä kokoovan julkaisun tasolla, mikä helpottaa lukijoiden työskentelyä.

Tämä ohje vastaa vain asettelullisiin kysymyksiin ja sen rinnalla on syytä tutustua materiaaliin ja luentoihin, joissa keskitytään tekstin varsinaiseen sisältöön. Olennaisin väline on kuitenkin akateemisesti pidemmälle ehtineen, jo julkaisuja rakentaneen ohjaajan palaute ja mentorointi.