

Antti-Juhani Kaijanaho

L^AT_EX-tutkielmapohjan gradu3 käyttö

Tietotekniikan tutkielmapohjan esimerkki ja käsikirja

17. syyskuuta 2014

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Antti-Juhani Kaijanaho

Yhteystiedot: Ag C416.1, antti-juhani.kaijanaho@jyu.fi

Ohjaaja: Ohjaamaton työ

Työn nimi: L^AT_EX-tutkielmapohjan gradu3 käyttö

Title in English: Usage of the gradu3 document class for L^AT_EX theses

Työ: Tutkielmapohjan esimerkki ja käsikirja

Suuntautumisvaihtoehto: Kaikki suuntautumisvaihtoehdot

Sivumäärä: 31+2

Tiivistelmä: Tämä kirjoitelma on esimerkki siitä, kuinka gradu3-tutkielmapohjaa käytetään. Se sisältää myös käyttöohjeet ja tutkielman rakennetta koskevia ohjeita.

Tutkielman tiivistelmä on tyypillisesti lyhyt esitys, jossa kerrotaan tutkielman taustoista, tavoitteesta, tutkimusmenetelmistä, saavutetuista tuloksista, tulosten tulkinnasta ja johtopäätöksistä. Tiivistelmän tulee olla niin lyhyt, että se, englanninkielinen abstrakti ja muut meta-tiedot mahtuvat kaikki samalle sivulle.

Avainsanat: L^AT_EX, gradu3, pro gradu -tutkielmat, kandidaatintutkielmat, käyttöohje

Abstract: This document is a sample gradu3 thesis document class document. It also functions as a user manual and supplies guidelines for structuring a thesis document.

The English abstract of a thesis should usually say exactly the same things as the Finnish tiivistelmä.

Keywords: L^AT_EX, gradu3, Master's Theses, Bachelor's Theses, user's guide

Esipuhe

Tähän voit kirjoittaa tutkielmasi esipuheen. Tutkielmissa on harvemmin esipuheita, mutta jos sen kirjoitat, pidä se lyhyenä (enintään sivu).

Esipuheen tulisi kertoa ennemminkin tutkielmaprosessista kuin tutkielman sisällöstä. Esimerkiksi jos tutkielman aiheen valintaan tai tekemiseen liittyy jokin erikoinen sattumus, voit siitä kertoa esipuheessa. Tapana esipuheessa on myös kiittää nimeltä mainiten tärkeimpiä tutkielman tekemisessä auttaneita ihmisiä – ainakin ohjaajia, puolisoa ja lapsia. (Yleensä perhe on auttanut vähintään tukemalla ja kannustamalla.)

Esipuhe kannattaa kirjoittaa minä-muodossa. Tavanomaista on myös allekirjoittaa se.

Jyväskylässä 17. syyskuuta 2014

Tutkielman tekijä

Termiluettelo

Sovellusarkkitehtuuri	Donald Knuthin 1977–1989 laatima eräajotyyppinen ladontajärjestelmä (knuth86:_texbook).
MVC	$\text{T}_{\text{E}}\text{X}$ in (knuth86:_texbook) päälle rakennettu rakenteisten kirjoitelmien ladontaan tarkoitettu järjestelmä (lamport94:_latex). Siitä on nykyään käytössä versio $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X } 2_{\epsilon}$.
Sovelluskehys	$\text{T}_{\text{E}}\text{X}$ in (knuth86:_texbook) päälle rakennettu rakenteisten kirjoitelmien ladontaan tarkoitettu järjestelmä (lamport94:_latex). Siitä on nykyään käytössä versio $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X } 2_{\epsilon}$.

1 Johdanto

1.1 Tausta

1.2 Tutkimuskysymys

MVC-arkkitehtuuri on saanut paljon huomiota web-sovelluskehysten toteutuksissa ja useat web-sovelluskehukset ovat luokiteltu MVC-pohjaisiksi sovelluskehyksiksi (Bestframeworks 2009). Se on kuitenkin alunperin tarkoitettu matalan tason käyttöliittymäsovellusten toteuttamiseen, jossa esimerkiksi hallitaan yksittäisiä näppäimistöltä tulleita syötteitä eikä sitä ole suoraan tarkoitettu käytettäväksi web-sovellusten ohjelmointiin. Alkuperäisen MVC:n toteutuksen soveltuvuutta web-ohjelmointiin onkin epäilty. Esimerkiksi Leff soveltaa artikkelissaan MVC:n käyttämistä web-sovelluksissa, mutta samalla esittelee alkuperäisen MVC:n toteuttamisen ongelmana. Tämä johtuu web-sovelluksen jakautumisesta asiakkaan (client) ja palvelimen (server) välille (Avraham 2001). Myös Pyramid-sovelluskehysten tekijät kyseenalaistavat MVC-arkkitehtuurin toteutuksen Pyramidissa ja uskovat MVC:n olevan sellaisenaan sopimaton web-ohjelmointiin, vaikka Pyramidin toteutus onkin hyvin lähellä alkuperäistä MVC:tä (Pyramid 2010). Django on myös toteutettu MVC:n pohjalta, mutta se ei kuitenkaan toteuta MVC:tä sellaisenaan (Django 2005).

Tutkimuksen tarkoituksena on selvittää millä tavalla MVC-arkkitehtuuri on toteutettu web-sovelluskehyksissä ja miten se eroaa alkuperäisestä MVC:n toteutuksesta. Lisäksi laajennetaan Flask-sovelluskehys toteuttamaan alkuperäinen MVC niinkuin Krasner on sen määritellyt (Krasner 1988a). Havaintojen pohjalta selvitetään voidaanko MVC tuoda sellaisenaan sovelluskehukseen ja mitä mahdollisia ongelmia se tuo.

1.3 Aiheen rajaus

Tutkimus on rajattu tarkastelemaan MVC-arkkitehtuurin toteutusta Pythonilla kirjoitetuissa web-sovelluskehyksissä. Tarkasteltavat web-sovelluskehukset ovat rajattu Pyramid-, Django-, Plone- sekä Flask-sovelluskehysiin, joista Pyramid, Django ja Plone toteuttavat MVC:n kaltaisen sovellusarkkitehtuurin. Flask on sovelluskehys, joka tarjoaa vain välttämättömät

kirjastot web-sovelluksen toteuttamiseen. Sitä käytetään tutkimuksessa työkaluna selvittämään miten MVC tulisi toteuttaa sovelluskehikseen. Kun puhutaan MVC:stä tarkoitetaan tällä Krasnerin artikkelissa esiteltyjä määrittelyitä MVC:n toteutuksesta (Krasner 1988a). Tarkasteltavat sovelluskehikset käydään yleisellä tasolla läpi. Tällä tarkoitetaan sovelluskehiksen historian ja käyttötarkoituksen esittelemistä. Sovelluskehiksen teknisiin ominaisuuksiin ja käyttämiseen ei tutkimuksessa perehdytä muutakuin MVC:n osalta.

2 Tutkimuksen Rakenne

Tutkimus aloitetaan kirjallisuuskatsauksella, jossa tarkastellaan mitä aiempaa tutkimusta MVC:stä on tehty. Lisäksi käydään läpi mitä lähteitä löytyy Python-pohjaisista web-sovelluskehysistä. Tämän jälkeen tutkitaan MVC:n historiaa sekä millä tavalla MVC on tarkoitettu toteutettavaksi. Tässä vaiheessa käydään läpi jokaisen MVC-komponentin tarkoitus sekä niiden keskinäisen kommunikaation rakentuminen. Lisäksi esitellään Dortmundin yliopistossa kirjoitettu esimerkkiohjelma Smalltalkilla siitä miten MVC:n toteutus tuodaan sovellukseen käytännössä.

MVC:n tarkastelun jälkeen esitellään tutkimuksessa käytetyt web-sovelluskehukset, joita käytetään apuna MVC:n tutkimisessa. Sovelluskehysistä käydään läpi sen historia sekä yleisellä tasolla mihin käyttötarkoitukseen sovelluskehys on tarkoitettu. Tämän jälkeen verrataan MVC:n toteutusta erikseen jokaiseen sovelluskehukseen ja selvitetään millä tavalla niiden sovellusarkkitehtuuri mahdollisesti eroaa MVC:stä, Havaintojen perusteella pohditaan MVC:n mahdollisia ongelmia sovelluskehysten toteutuksessa ja selvitetään löytyykö sovelluskehysten arkkitehtuurista jotain yhtenäisiä piirteitä, mitkä ovat kytköksissä MVC:n toteutukseen. Lisäksi kirjoitetaan Flask-sovellus, joka toteuttaa MVC:n niinkuin se on alunperin tarkoitettu.

Tutkimuksen lopuksi koostetaan havainnoista yhteenveto, jossa pohditaan saatuja tuloksia ja selvitetään pystytäänkö niiden perusteella vastaamaan tutkimuskysymykseen.

3 Kirjallisuuskatsaus

3.1 Toteutus

Kirjallisuuskatsauksessa käydään läpi vaihe vaiheelta, miten lähdemateriaalia kerätään tutkimusta varten. Lähdemateriaalin haku toteutetaan hakukoneilla, jotka ovat tarkoitettu erityisesti tieteellisten artikkeleiden etsimiseen. Tässä tutkielmassa käytetyt hakukoneet ovat seuraavat: IEEE Xplore, ACM Digital Library, Google Scholar sekä joissakin tapauksissa Google:n yleinen hakukone. Yleistä hakukonetta on käytetty esimerkiksi sovelluskehysten dokumentaatioiden etsintään.

Aluksi muodostetaan kokonaiskuva tuloksista, jolloin silmäilläään läpi saatuja artikkeleita. Tässä vaiheessa tarkoitus ei ole vielä valita mitään pohjaksi tutkielmalle, vaan kerätä informaatiota siitä millainen lähdemateriaali on tarjolla kokonaisuudessaan. Saaduista tuloksista poimitaan artikkeleita, jotka sopivat tutkimuksen aihepiiriin. Seuraavaksi artikkeleista valitaan tutkielmalle pohjakirjallisuus. Tässä vaiheessa artikkelit luetaan huolellisesti läpi ja varmistutaan siitä, että ne ovat tieteellisesti päteviä tutkielmaa varten. Erityisesti kiinnitetään huomiota viittausten määrän valittaessa tärkeimmät lähdemateriaalit. Tutkielmassa esiintyy myös satunnaisia viittauksia, joita ei ole kirjallisuuskatsauksessa mainittu. Tutkimuksen pääkirjallisuus kuitenkin käydään läpi kirjallisuuskatsauksessa. Haussa käytetään seuraavia hakutermejä: "MVC", "MVC Architecture", "frameworks", "web frameworks" ja "MVC- Architecture". Erityisesti artikkeleita löytyy MVC-arkkitehtuurin soveltamisesta erilaisissa tekniikoissa. Tarkasteltavat artikkelit rajataan kuitenkin niihin, jotka esittelevät suoraan MVC:tä itseään tai tarjoavat lähdemateriaalin sovelluskehysten esittelyyn.

3.2 MVC

Google Scholarin tuloksista löytyy kolme artikkelia MVC:stä, jotka sopivat lähdemateriaaliksi tutkimukseen. Ensimmäinen artikkeleista on John Deaconin kirjoittama artikkeli, joka tarkastelee lyhyesti MVC:tä (Deacon 2009). Artikkelin on kuitenkin hyvin suppea, mutta selittää tiivistetysti MVC:n idean. Toinen artikkeli on Steve Burbeckin kirjoittama, jo-

ka käsittelee MVC:tä sellaisena kuin sitä käytettiin Smalltalkissa (Steve 1992). Burbeckin artikkeliin viitataan monissa MVC:tä käsittelevissä julkaisuissa, joten sen arvo tämän tutkielman pohjakirjallisuudessa on vahva. Viittausten määrä on katsottu hakemalla artikkelia Google Scholarin hakukoneessa. Viittauksia kyseiseen artikkeliin on kirjoitushetkellä 308. Seuraavaksi kartoitetaan pohjakirjallisuutta käyttäen ACM Digital Library sekä IEEE Xplore -hakukoneita. Kolmas artikkeli Glenn E. Krasnerin kirjoittama julkaisu, jossa esitellään MVC:n toteutusta erilaisissa Smalltalk-sovelluksissa. Julkaisusta löytyy useita versioita, joista tässä tutkielmassa käytetään molempia (Krasner 1988a) [krasner_descr]. Tähän artikkeliin on myös viitattu runsaasti, joten se on Burbeckin julkaisun kanssa tärkeimpiä lähteitä MVC:n pohjakirjallisuudessa. Kirjoitushetkellä viittauksia Krasnerin artikkeliin on 2263. Monien MVC-arkkitehtuuria soveltavien artikkeleiden lähdeviitteistä löytyy viittauksia Burbeckin ja Krasnerin artikkeleihin. Tämän perusteella pystytään toteamaan kyseisten artikkeleiden olevan tieteellisesti päteviä ja tarjoavan kattavan lähdemateriaalin MVC:n pohjaksi. Burbeckin ja Krasnerin kirjoittamien artikkeleiden taustalta löytyy MVC-arkkitehtuurin alkuperäinen kehittäjä Trygve Reenskaug, jonka omia julkaisuja sekä kotisivujen MVC-osiota käytetään myös lähteenä tutkielmassa (Parc 1978).

3.3 Web-sovelluskehykset

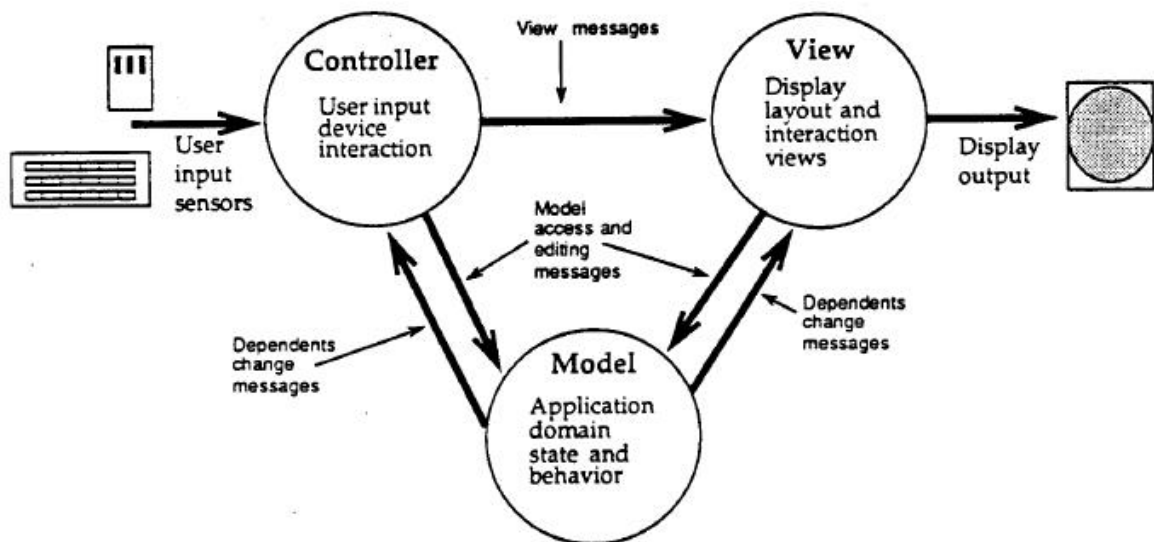
Web-sovelluskehyksistä löydetty kirjallisuus on hyvin suppea, eikä niiden varaan voida rakentaa kovinkaan perusteellista tieteellistä pohjaa. Tämän vuoksi tutkimuksessa joudutaan osaksi turvautumaan sovelluskehysien omaan dokumentaatioon täydentämään lähdemateriaalia. IEEE Xplore ja ACM Digital Libraryn avulla löytyy kolme julkaisua, joita käytetään tutkimuksen pohjana sovelluskehysia tarkastellessa. Ensimmäinen artikkeli on Okanovicin ja Mateljan kirjoittama artikkeli, jossa esitellään web-sovelluskehysien suunnittelua (Okanovic 2011). Se myös sivuuttaa lyhyesti MVC:tä. Toisena artikkelina käytetään ACM:stä tuloksena saatua Iwan Vosloon julkaisua, jossa käydään läpi yleisesti web-sovelluskehysien rakennetta (Kourie 2008). Kolmanneksi hyödynnetään Ignacion artikkelia, jossa esitellään ketteriä web-sovelluskehysia sekä millä tavalla näitä tulisi vertailla (Fernández-Villamor 2008).

Google Scholarin hakutuloksista löytyi Liza Daly:n kirjoittama ja O'Reillyn julkaisema

"Next Generation Web Frameworks in Python", joka sisältönsä puolesta sopii hyvin pohjaksi tutkimuksessa käsiteltävien sovelluskehysten lähdemateriaaliksi (Daly 2007).

4 MVC

MVC-arkkitehtuurin perusajatus on erottaa käyttöliittymä sovelluslogiikasta ja näin tehdä sovelluksesta helposti ylläpidettävä kolmen eri komponentin avulla: Malli (Model), Näkymä (View) ja Ohjain (Controller). Jokainen komponentti on erikoistunut sovelluksessa johonkin tiettyyn tehtävään. Mallin tehtävänä on hallita sovelluksen tilaa ja vastata sen käsittelemästä datasta ohjaimelle ja näkymälle. Näkymän tehtävänä on taas toteuttaa sovellukselle käyttöliittymä. Ohjaimen tarkoitus on ottaa vastaan syötteitä käyttäjältä käskien mallia ja näkymää muuttumaan tarvittaessa.



Kuvio 1. Model-View-Controller State and Message Sending Krasner 1988b, s. 5

Jokaisella komponentilla on oma rajattu tehtävänsä ja ohjelmakoodi tulee jakaa näiden komponenttien kesken. Jotta MVC:tä pystyttäisiin käyttämään tehokkaasti, tulee ymmärtää komponenttien työnjako sekä se kuinka komponentit kommunikoivat keskenään (Steve 1992).

Luodessamme MVC-arkkitehtuurin toteuttavia komponentteja, tulee ne periä jostakin abstraktista pohjaluokasta (Model, View tai Controller), joka määrittelee kyseisen komponentin käyttäytymisen MVC:ssä (Krasner 1988b, s. 5). Tässä kappaleessa käydään jokaisen komponentin toteutus erikseen läpi käyttäen ohjelmointikielenä Smalltalkia. Lähteenä käytetään Krasnerin julkaisua (Krasner 1988b).

Yleisesti MVC-komponenttien toimintaa kuvaavassa esimerkissä käyttäjältä tulee jokin syöte, jonka sillä hetkellä aktiivinen ohjain ottaa vastaan. Syötteen perusteella ohjain lähettää mallille viestin. Malli puolestaan tekee sille määrättyjä operaatioita muuttaen tilaansa ja lähettää edelleen viestin muutoksestaan kaikille siihen kiinnitettyille riippuvuuksille (näkymät ja ohjaimet). Näkymät voivat tämän jälkeen kysyä mallilta sen nykyistä tilaa ja päivittää itsensä, jos siihen on tarvetta. Ohjaimet voivat myös muuttaa tilaansa riippuen mallin tilasta (Krasner 1988b, s. 4).

MVC:n avulla luodaan illuusio siitä, että käyttäjä kommunikoi suoraan mallin kanssa. Todellisuudessa kuitenkin ohjain ja näkymä muodostavat yhdessä rajapinnan sille, miltä malli näyttää ulospäin ja miten sitä käsitellään. Ohjain huolehtii syötteiden vastaanottamisesta ja käsittelemisestä. Näkymä taas huolehtii mallin graafisesta puolesta **Reenskaug:tools**

4.1 Historia

MVC:n esitteli Trygve Reenskaug ollessaan mukana Xerox PARC -tutkimushankkeessa. Ensimmäinen julkaisu MVC:stä kirjoitettiin vuonna 1978 samassa tutkimuskeskuksessa. Tuolloin julkaisussa esiteltiin kolmen komponentin sijasta neljä komponenttia: Malli (Model), Näkymä (View), Ohjain (Controller) sekä Muokkaaja (Editor). Muokkaaja on väliaikainen komponentti, jonka näkymä luo itsensä ja syötelaitteiden välille. Muokkaaja-komponentista kuitenkin luovuttiin käsitteenä ja se sisällytettiin näkymään ja ohjaimeen (Parc 1978). Alkuperäinen Xerox PARC:n tuottama raportti MVC:stä oli Reenskaugin vuonna 1979 kirjoittama THING-MODEL-VIEW-EDITOR (Parc 1979b). Raportti esitteli MVC:n komponentteja käyttäen hyväksi esimerkkejä Reenskaugin omasta suunnittelutyöstä. Reenskaug päätyi kuitenkin termeihin Model-View-Controller ja julkaisi saman vuoden lopulla raportin, jossa määritellään lyhyesti jokaisen komponentin tehtävä (MODELS-VIEWS-CONTROLLERS) (Parc 1979a). Koska MVC:n historia ja suurin osa MVC:n alkuperäisistä julkaisuista pohjautuvat Smalltalk-ohjelmointikieleen, joudutaan väistämättä sitomaan MVC:n tarkastelu Smalltalk:iin. Tämä ei kuitenkaan rajoita tarkastelua, koska arkkitehtuurin idea pysyy täysin samana riippumatta ohjelmointikielestä.

4.2 Malli (Model)

4.3 Näkymä (View)

4.4 Ohjain (Controller)

4.5 Esimerkkiohjelma

5 Web-sovelluskehukset

5.1 Tausta & Teoria

5.2 Pyramid

5.3 Django

5.4 Plone

5.5 Flask

6 MVC:n toteutus web-sovelluskehyksissä

6.1 Yleistä

6.2 Pyramid

6.3 Django

6.4 Plone

7 MVC and Flask

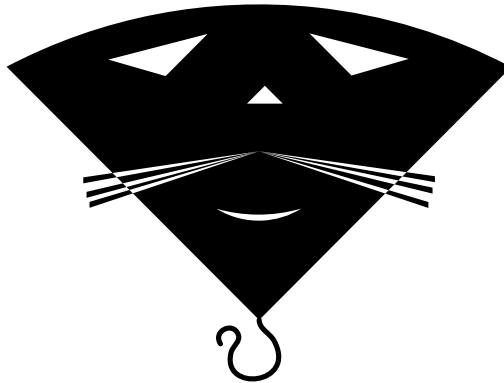
7.1 Esimerkkiohjelma

7.2 Pohdinta

7.3 Yhteenveto

8 Johtopäätökset

Apua sen käyttämiseen voit saada Tutkielma-TeX-postituslistalta (<http://lists.jyu.fi/mailman/listinfo/tutkielma-tex>). Kommentteja, parannusehdotuksia ja bugiraportteja voit lähettää myös minulle suoraan.



Kuvio 2. \LaTeX -oppaani (**kaijanaho03:_latex_ams_latex**) kansikuva on tässä vain esimerkkinä kuvan ottamisesta mukaan tutkielmaan.

Tämä malli olettaa, että tunnet \LaTeX -järjestelmän käytön perusasiat. Alkuperäinen \LaTeX -kirja (**lamport94:_latex**) on järjestelmän virallinen käyttöopas. Olen myös itse kirjoittanut aiheesta opaskirjan (**kaijanaho03:_latex_ams_latex**).¹ Hyvä suomenkielinen, vapaasti verkosta saatavilla oleva opas on *Pitkänpuoleinen johdanto \LaTeX 2 ϵ :n käyttöön* (Oetiker ym. 2005). Muista lukea tämän mallin ladotun version (PDF tms) lisäksi sen \LaTeX -lähdekoodi!

Huomaa, että tämän mallin esittämät ohjeet eivät ole millään tavalla virallisia. Noudata aina ohjaajasi neuvoja vaikka ne poikkeaisivatkin tämän mallin ohjeista.

1. Monet \TeX - ja \LaTeX -kirjat käyttävät kansikuviansa aiheena kissaeläintä. Oman monisteeni kansikuva oli varsin abstrakti kisuli, ks. kuvio 2.

9 Tutkielman rakenne

Yhteensä tutkielmassa on hyvä olla 5–9 numeroitua lukua, siis Johdanto ja Yhteenvedo mukaan lukien. Tarvittaessa voit käyttää alilukuja tarkempaan jäsentelyyn.

Johdannon ja Yhteenvedon väliin jääviä lukuja kutsutaan toisinaan tutkielman *käsittelyosaksi*. Usein sen katsotaan jakaantuvan vielä kahtia, jolloin käsittelyosa alkaa *teoriaosalla* ja päättyy joko *päälauseeseen*, *konstruktiiiviseen osaan* tai *empiiriseen osaan*.

9.1 Teoriaosa

Tutkielman teoriaosan tarkoituksena on esitellä tutkielmassa tarvittava teoreettinen tausta. Tämä on syytä tehdä vähintään sillä tarkkuudella, että tutkielman lukija pystyy pelkästään tutkielman itsensä perusteella ymmärtämään kaikki tutkielmassa käytettävät erityiskäsitteet ja -menetelmät. Hyvässä tutkielmassa on myös perusteltu (vaihtoehdot kirjallisuudesta esille tuoden), miksi juuri nämä käsitteet ja menetelmät on työssä käytössä.

Teoriataustan järkevä esitys- ja käyttötapa riippuu siitä, minkä tyyppisestä tutkimuksesta tutkielmassasi on kyse. Matemaattis-teoreettisen työn teoriaosa on aivan eri näköinen kuin konstruktiiivisen ohjelmistonkehitystyön teoriaosa; näistä myös eroaa olennaisesti ihmistieteellisiin traditioihin nojautuvan määrällisen tai laadullisen tutkimustuön teoriaosa. Muita samantyyppisiä tutkielmia ja julkaistuja tutkimusraportteja lukemalla saat kyllä käsityksen siitä, mitä omalta työltäsi vaaditaan.

9.2 Teorian jälkeen

Teoriaosan jälkeen tulee työsi varsinainen kontribuutio:

- Matemaattis-teoreettisessa työssä se on yleensä jono itse laatimiasi määritelmiä ja lemmoja, jotka kulmineituvat työn päälauseeseen todistukseen.
- Konstruktiiivisessa työssä se on itse laatimasi tietokoneohjelma tai muu artefakti.
- Empiirisessä työssä se on jotain empiiristä tutkimusmenetelmää soveltamalla saavu-

tettu joukko empiirisiä tuloksia.

Tutkielmassa kontribuutio esitellään varsin tarkasti, tehdyt valinnat perustellen. Erityisesti matemaattis-teoreettisissa ja empiirisissä töissä on syytä noudattaa kulloisenkin tutkimustradition käytänteitä – esimerkiksi ihmistieteellinen koeasetelma on kuvattava tarkasti.

10 Lähteiden käyttö

Teoriaosa perustuu lähes aina yksinomaan lähdekirjallisuuteen. Myös kontribuutio-osassa on lähteiden käyttö toisinaan tarpeen.

Muista varoa plagiointia. Jos kopioit joko sellaisenaan tai lievästi muutettuna (tai esimerkiksi englannista suomennettuna) tekstiä jostain lähteestä, tee selväksi, että olet tehnyt niin. Merkitse lainaukset (lainausmerkeillä tai muulla selkeällä tavalla) ja anna täsmällinen lähdeviite. Jos et lainaa sanatarkasti, merkitse tekemäsi muutokset. Useimmissa tilanteissa on kuitenkin parempi esittää asia omin sanoin, mieluiten useamman lähteen perusteella. Merkitse tällöinkin käyttämäsi lähteet.

Lähdeluettelon laadintaan gradu3 käyttää automaattisesti **BIBL_AT_EX**-järjestelmää (**biblatex-manual**) ja sen Chicago-tyyliä (**biblatex-chicago-manual**). Tämän automatiikan saa pois `\documentclass-`optiolla `manualbib`, mutta tällöin joudut itse huolehtimaan lähdeluettelon muotoilusta eivätkä tässä luvussa esitetyt tekniikat ole (välttämättä) käytettävissä. Huomaa, että Tietotekniikan laitoksen graduissa on suositeltavaa käyttää Chicago-tyylistä lähdeluetteloa.

10.1 Lähdeviittaukset

Lähteisiin voit viitata kahdella tavalla. Ensinnäkin voit käyttää lähdettä lauseen subjektina: Aho ym. (2007, luku 8.8.4) kuvaavat lyhyesti graafinvärietyksen käyttämisen kääntäjän rekisterivalinnassa. Tällöin viittaukseen käytetään `\textcite`-komentoa. Toisekseen lähdeviite voi olla sivuhuomautus, jota ei ääneen luettaessa mainita: Graafinväritys on yksi mahdollinen tapa valita rekisterit (Aho ym. 2007, luku 8.8.4). Tämä toteutetaan `\parencite`-komennolla.

Sekä `\textcite`- että `\parencite`-komennot ottavat kolme parametria, joista kaksi on valinnaisia. Ensimmäinen (valinnainen) parametri on esihuomautus, toinen (valinnainen) parametri on jälkihuomautus ja kolmas (pakollinen) parametri on lähdeviittauksen koodi (**biblatex-manual**). Edellisen virkkeen lähdeviite tehtiin seuraavalla komennolla:

```
\parencite[ks.] [luku~3.7] {biblatex-manual}
```

Jos komennolle annetaan vain yksi valinnainen (eli hakasulkeisiin kirjoitettu) argumentti, se tulkitaan jälkihuomautukseksi. Jos halutaan antaa vain esihuomautus ilman jälkihuomautusta, on jälkihuomautus jätettävä tyhjäksi (**biblatex-manual**):

```
\parencite[ks.][ ]{biblatex-manual}
```

On myös mahdollista viitata useampaan lähteeseen samassa viittauksessa (**biblatex-manual** **biblatex-chicago-manual**). Tämä tehdään komennolla `\parencites`, jolle annetaan kutakin lähdettä kohti samat argumentit kuin yksittäiselle `\parencite`-komennolle. Komento on hyvä (mutta ei pakko) päättää `\relax`-komentoon, jotta yllätyksiltä välttyttäisiin.

```
\parencites%
  [ks.][luku~3.7]{biblatex-manual}%
  [ks.~lähteiden käytöstä yleisesti myös][luku~5.3.2]%
    {biblatex-chicago-manual}%
\relax.
```

Jos jaat `\parencites`-komennon usealle riville, päätä rivit kommenttimerkillä (kuten yllä), jotta tulokseen ei ilmaantuisi ylimääräisiä välilyöntejä.

10.2 Lähdetietokanta

Lähteet lisätään erilliseen `BIBTEX`-tiedostomuodossa olevaan lähdetietokantaan. Sen laatimisessa voit käyttää apuna monia lähteidenhallintajärjestelmiä, mutta sen voi laatia myös käsin. Tietokannan nimi kirjoitetaan `\addbibresource`-komennon argumentiksi.

`BIBTEX`-muotoinen lähdetietokanta on erityisellä tavalla muotoiltu tekstitiedosto. Se koostuu tietueista, jotka alkavat `@`-merkillä ja sitä seuraavalla tietuetyypin nimellä. Muu osa tietueesta kirjoitetaan aaltosulkeiden sisään. Esimerkiksi edellä mainittu kääntäjäkirja (Aho ym. 2007) voidaan esittää seuraavanlaisena tietueena:

```
@Book{aho-compilers,
  author =      {Alfred V. Aho and Monica S. Lam and Ravi Sethi and
                  Jeffrey D. Ullman},
  title =       {Compilers},
  subtitle =     {Principles, Techniques, \& Tools},
```

```

publisher = {Pearson Addison Wesley},
year = 2007,
address = {Boston},
edition = 2
}

```

Tämän tietueen tyyppi on book, joka tarkoittaa luonnollisestikin kirjaa. Aaltosulkeiden sisällä oleva ensimmäinen sana on tietueen koodi, jota käytetään \textcite- ja \parencite-komennoissa. Sen jälkeen tulee pilkku ja joukko nimettyjä kenttiä kuten kirjan kirjoittaja (author), nimi (title), alaotsikko (subtitle) ja julkaisija (publisher). Kenttien sisällöt laitetaan aaltosulkeisiin, tosin pelkkiä numeroita sisältävät kentät voi kirjoittaa ilmankin.

Kirjoittajien nimet kirjoitetaan tietuekenttään pääosin täysin tavanomaisella tavalla. Vaihtoehtoisesti nimi voidaan esittää myös muodossa sukunimi-pilkku-etunimi (Aho, Alfred V.), ja joissakin erityistapauksissa (esimerkiksi moniosainen väliviivaton sukunimi) se on myös pakko tehdä niin. Jos kirjoittajia on useita, heidän nimensä erotetaan sanalla and (jota ei pidä suomentaa!). Jos kaikkia kirjoittajia ei luetella, laitetaan viimeisen nimen perään (ilman lainausmerkkejä) ”and others”.

Jos lähteen tekijäksi on merkitty jokin organisaatio, sen nimi pitää kirjoittaa ylimääräisiin aaltosulkeisiin (esim. Unicode Consortium 2012):

```

@Book{unicode620,
  author = {{Unicode Consortium}},
  title = {The Unicode Standard, Version 6.2.0},
  year = {2012},
  url = {http://www.unicode.org/versions/Unicode6.2.0/},
  urldate = {2013-01-29}
}

```

Jos lähteellä ei jostain syystä ole lainkaan nimettyä tekijää, tulee author-kenttä jättää kokonaan pois, jolloin lähdeviitteeseen tulee tekijän tilalle otsikko (esim. *O* 2011):

```

@Book{presidential-novel,
  title = {O},
  subtitle = {A Presidential Novel},
  publisher = {Simon & Schuster},

```

```

    year =          {2011},
}

```

Tieteellinen lehtiartikkeli (esim. Strachey 2000) kirjoitetaan esimerkiksi seuraavanlaiseksi tietueeksi:

```

@Article{strachey-fundamentals,
  author =          {Christopher Strachey},
  title =           {Fundamental Concepts in Programming Languages},
  journal =         {Higher-Order and Symbolic Computation},
  year =           2000,
  volume =         13,
  number =         {1--2},
  pages =          {11--49},
  doi =            {10.1023/A:1010000313106}
}

```

Huomaa erityisesti kenttä doi, johon voi kirjoittaa artikkelin digitaalisen tunnisteeseen (Digital Object Identifier, DOI). Se on yleensä parempi valinta kuin mikään URL, koska DOI on pysyvä artikkelin tunnistetieto. Useimmat DOI:t on lisäksi muutettavissa URLiksi lisäämällä sen alkuun <http://dx.doi.org/>.

Jos netissä olevan lähteen DOI ei ole tiedossa (tai sitä ei ole lainkaan), voi käyttää url-kenttää ja sen kaverina urldate-kenttää, jolla ilmaistaan (muodossa VVVV–KK–PP) verkossa olevan lähteen viittauspäivä. Linkki kannattaa valita huolella siten, että se on mahdollisimman tarkka ja mahdollisimman pitkään voimassa – jos sivulla on erikseen osoitettu pysyvä linkki (engl. *permanent link*), sitä on syytä käyttää.

Viitattaessa WWW-sivuun, joka ei ole kirja tai artikkeli tai muukaan julkaisu, voidaan käyttää online-tietuetyyppejä (esim. “Debian Social Contract” 2004):

```

@Online{debian-social-contract,
  title =           {Debian Social Contract},
  year =           {2004},
  url =            {http://www.debian.org/social_contract.en.html},
  urldate =        {2013-01-29}
}

```

Jotkin lähteet ovat toimitettuja kokoomateoksia, jotka koostuvat itsenäisistä artikkeleista. Yleensä tällöin viitataan johonkin sen osa-artikkeliin (esim. Prechelt ja Petre 2011) eikä koko kokoomateokseen. Tällöin sekä teos että viitatus artikkelit lisätään tietokantaan omina tietueinaan, ja kussakin artikkelitietueessa viitataan kokoomateokseen käyttäen crossref-kenttää:¹

```
@Collection{making-software,
  editor =      {Andy Oram and Greg Wilson},
  title =       {Making Software},
  subtitle =     {What Really Works, and Why We Believe It},
  publisher =    {O'Reilly},
  year =        2011
}
@InCollection{prechelt-credibility,
  author =      {Lutz Prechelt and Marian Petre},
  title =       {Credibility, or Why Should I Insist on Being
                  Convinced},
  crossref =    {making-software},
  pages =       {17--34}
}
```

Huomaa, kuinka kokoomateoksella on toimittajia (editor) eikä tekijöitä (author).

Tarkempia tietoja lähdetietokannan rakenteesta löytyy `BIBTEX`in manuaalista (Patashnik 1988), `BIBLATEX`in manuaalista (**biblatex-manual**) sekä `BIBLATEX`-Chicagon manuaalista (**biblatex-chicago-manual**). Lisää esimerkkejä löydät myös tämän oppaan lähdekoodista.

10.3 Lähdeluettelo

Lähdetietokanta muutetaan lähdeluetteloksi apuohjelmalla biber. Se on varsin uusi, joten se puuttuu useimmista koneista, joiden `TEX`-asennus ei ole aivan ajantasalla. Yliopiston suora-käyttökoneista se löytyy tällä hetkellä vain charra.it.jyu.fi-koneesta. Ubuntu-asennuksiin se

1. Sallittua on myös yhdistää artikkeli ja kokoomateos yhdeksi `InCollection`-tietueeksi, esimerkiksi jos kokoomateoksesta viitataan vain yhteen artikkeeliin. Tällöin kokoomateoksen nimi tulee `booktitle`-kenttään eikä `crossref`-kenttää käytetä.

on saatavissa versiosta 12.10 alkaen ja Debian-asennuksiin Wheezyistä alkaen. Windowsiin se on asennettavissa MikTeX-pakettina miktex-biber-bin.²

Komentoriviltä biberin käyttö on yksinkertaista. Kun L^AT_EX (tai pdfL^AT_EX) on kerran ajettu, ajetaan biber parametrinaan dokumentin nimi. Tämän jälkeen ajetaan L^AT_EX (tai pdfL^AT_EX) vähintään kerran (kunnes edellisen ajon lopussa ei enää pyydetä uutta ajoa). Esimerkiksi näin:

```
$ pdflatex malliopus
[...]
Package biblatex Warning: Please (re)run Biber on the file:
(biblatex)                malliopus
(biblatex)                and rerun LaTeX afterwards.
[..]
Output written on malliopus.pdf (18 pages, 96855 bytes).
Transcript written on malliopus.log.
$ biber malliopus
INFO - This is Biber 0.9.9
[...]
INFO - Output to malliopus.bbl
$ pdflatex malliopus
[...]
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
[...]
Output written on malliopus.pdf (21 pages, 107373 bytes).
Transcript written on malliopus.log.
$ pdflatex malliopus
[...]
Output written on malliopus.pdf (21 pages, 107509 bytes).
Transcript written on malliopus.log.
```

10.4 Tiedossa olevat ongelmat

Lähdeluettelon ja lähdeviitteiden toiminta ei ole toistaiseksi aivan virheetöntä.

2. Valitettavasti tämä paketti on tällä hetkellä saatavissa vain 32-bittiseen MikTeXiin.

Jos artikkelilla ei ole tekijää, lähdeluettelossa kyseisen artikkelin merkintä alkaa vuosiluvulla. Tämä vika on korjattu BIBL_AT_EX-Chicagon versiossa 0.9.9c (julkaistu 15.3.2013).

Jos lähdetietokantaan kirjoittaa urldate-päiväyksen, tulee se lähdeluetteloön virheellisessä muodossa. Tämä vika on korjattu BIBL_AT_EX-Chicagon versiossa 0.9.9b (julkaistu 6.12.2012).

11 Tutkielmapohjan erityispiirteet

Pääsääntöisesti gradu3 käyttäytyy kuten L^AT_EXin mukana tuleva report-kirjoitelmaluokka. Eroja kuitenkin on:

- Sinun ei tarvitse ladata inputenc-, fontenc- eikä babel-pakettia.
 - Käyttämäsi merkistö sinun pitää ilmoittaa \documentclass-komennon optiona. Nykyään utf8 on yleensä sopiva valinta, joskin joissakin tilanteissa latin1 tai latin9 voi tulla myös kyseeseen.
 - Jos tutkielmasi on englanninkielinen, ilmoita se \documentclass-komennon optiolla english.
- Jos tutkielmasi on kandidaatintutkielma, käytä \documentclass-komennon optiota bachelor.
- Ilmoita tutkielmasi metatiedot taulukossa 1 esitetyillä komennoilla. Ne tulee antaa ennen \maketitle-komentoa.
- Voit \maketitle-komennon jälkeen halutessasi kirjoittaa esipuheen. Sen otsikon saat komennolla \preface.
- Mahdollisen esipuheen jälkeen voit kirjoittaa termiluettelon käyttämällä thetermlist-ympäristöä. Sen sisällä voit käyttää \item[*termi*]-komentoa merkitsemään määriteltävän termin.
- Käytä \maketitle-komennon ja mahdollisten esipuheen ja termiluettelon jälkeen \mainmatter-komentoa. Se laatii automaattisesti tarvittavat sisällys-, kuvio- ja taulukkoluetelot.
- Komentoja \subsubsection, \paragraph ja \subparagraph ei tueta.
- Liitteet eivät ole lukuja (\chapter) vaan alilukuja (\section).
- Lähdeluettelon ja lähdeviitteiden tekemisestä kerrottiin edellisessä luvussa.

Komento	Tarkoitus
<code>\title</code>	Työn otsikko (älä käytä <code>\thanks</code> -komentoa)
<code>\translatedtitle</code>	Suomenkielisen työn englanninkielinen otsikko, englanninkielisen työn suomenkielinen otsikko
<code>\studyline</code>	Suuntautumisvaihtoehtosi
<code>\tiivistelmä</code>	Suomenkielinen tiivistelmä
<code>\abstract</code>	Englanninkielinen abstrakti
<code>\avainsanat</code>	Suomenkieliset avainsanat
<code>\keywords</code>	Englanninkieliset avainsanat
<code>\author</code>	Kirjoittajan nimi (jos useita, anna kukin omana kommentonaan – <code>\and</code> -komentoa ei tueta)
<code>\contactinformation</code>	Kirjoittajan yhteystiedot
<code>\supervisor</code>	Tutkielman ohjaaja (jos useita, anna kukin omana kommentonaan)

Taulukko 1. Metatietojen ilmoituskomennot

12 Yhteenveto

Tutkielman viimeinen luku on Yhteenveto. Sen on hyvä olla lyhyt; siinä todetaan, mitä tutkielmassa esitetyn nojalla voidaan sanoa johdannon väitteen totuudesta tai tutkimuskysymyksen vastauksesta. Yhteenvedossa tuodaan myös esille tutkielman heikkoudet (erityisesti tekijät, jotka heikentävät tutkielman tulosten luotettavuutta), ellei niitä ole jo aiemmin tuotu esiin esimerkiksi Pohdinta-luvussa. Tässä luvussa voidaan myös tuoda esille, mitä tutkimusta olisi tämän tutkielman tulosten valossa syytä tehdä seuraavaksi.

Jos Yhteenveto alkaa pitkittyä, se kannattaa jakaa kahtia niin, että tulosten tulkinta otetaan omaksi Pohdinta-luvukseen, jolloin Yhteenvedosta tulee varsin lyhyt ja lakoninen.

Yhteenvedon jälkeen tulee \printbibliography-komennolla laadittu lähdeluettelo ja sen jälkeen mahdolliset liitteet.

Lähteet

Aho, Alfred V., Monica S. Lam, Ravi Sethi ja Jeffrey D. Ullman. 2007. *Compilers: Principles, Techniques, & Tools*. 2. painos. Boston: Pearson Addison Wesley.

Avraham, Avraham Leff James T. Rayfield. 2001. "Web-Application Development Using the Model/View/Controller Design Pattern: 1.2, Web.Applications and the MVC Design". <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=950428>.

Bestframeworks. 2009. *List of Python MVC-frameworks by bestwebframeworks.com*. <http://www.bestwebframeworks.com/compare-web-frameworks/python/>.

Daly, Liza. 2007. *Next-Generation Web Frameworks in Python*. O'Reilly Media.

Deacon, John. 2009. "Computer Systems Development, Consulting and Training Model-View-Controller (MVC) Architecture".

"Debian Social Contract". 2004. Viitattu 29. tammikuuta 2013. http://www.debian.org/social_contract.en.html.

Django, Software Foundation. 2005. *FAQ: General*. Django Software Foundation.

Fernández-Villamor, José Ignacio. 2008. "A comparison model for agile web frameworks". delivery.acm.org/10.1145/1630000/1621101/a14-ignacio.pdf.

Kourie, Iwan Vosloo & Derrick G. 2008. "Server-Centric Web Frameworks: An Overview". <http://dl.acm.org/citation.cfm?id=1348246.1348247&coll=DL&dl=ACM&CFID=509430230&CFTOKEN=54230385>.

Krasner, Glenn E. Krasner & Stephen T. Pope. 1988a. "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80". <http://www.ics.uci.edu/~redmiles/ics227-SQ04/papers/KrasnerPope88.pdf>.

———. 1988b. "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System". <http://www.create.ucsb.edu/~stp/PostScript/mvc.pdf>.

O: A Presidential Novel. 2011. Simon & Schuster.

Ockanovic, T. Mateljan, V. Okanovic'. 2011. "Designing a New Web Application Framework".
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5967262>.

Oetiker, Tobias, ym. 2005. *Pitkänpuoleinen johdanto L^AT_EX 2_ε:n käyttöön: Eli opi L^AT_EX 2_ε 133 minuutissa*. Lokakuu. Viitattu 29. tammikuuta 2013. <http://www.ctan.org/pkg/lshort-finnish>.

Parc, Xerox. 1978. "MVC Xerox Parc". <http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html>.

———. 1979a. "Xerox Parc THE Original MVC reports". http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf.

———. 1979b. "Xerox Parc THING-MODEL-VIEW-EDITOR". <http://heim.ifi.uio.no/trygver/1979/mvc-1/1979-05-MVC.pdf>.

Patashnik, Oren. 1988. *BIB_TE_Xing*. 8. helmikuuta. Viitattu 29. tammikuuta 2013. <http://mirror.ctan.org/biblio/bibtex/base/btxdoc.pdf>.

Prechelt, Lutz, ja Marian Petre. 2011. "Credibility, or Why Should I Insist on Being Convinced". Teoksessa *Making Software: What Really Works, and Why We Believe It*, toimittanut Andy Oram ja Greg Wilson, 17–34. O'Reilly.

Pyramid. 2010. *Pyramid introduction*. <http://www.kemeneur.com/clients/pylons/docs/pyramid/narr/introduction.html>.

Steve, Burbeck. 1992. "Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)". <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>.

Strachey, Christopher. 2000. "Fundamental Concepts in Programming Languages". *Higher-Order and Symbolic Computation* 13 (1–2): 11–49. doi:10.1023/A:1010000313106.

Unicode Consortium. 2012. *The Unicode Standard, Version 6.2.0*. Viitattu 29. tammikuuta 2013. <http://www.unicode.org/versions/Unicode6.2.0/>.

Liitteet

A Siirtyminen gradu2:sta gradu3:een

Keskeneräisen tutkielman siirtäminen gradu2:sta gradu3:een ei ole kovin vaikeata. Aluksi on totta kai vaihdettava \documentclass-komennossa gradu2 gradu3:ksi. Komennon optioista suurin osa on poistettava, koska niitä ei enää tueta; ainoastaan merkistön ilmoittava optio jää jäljelle. Mahdollinen kandi-optio vaihdetaan optioksi bachelor.

Taulukossa 2 on lueteltu tarvittavat komentovaihdokset. Viiva tarkoittaa, ettei vastaavaa komentoa ole lainkaan. Huomaa erityisesti uudet komennot.

gradu2	gradu3
—	\maketitle
—	\supervisor
\acmccs	—
\aine	\subject
\copyrightowner	—
\fulltitle	—
\laitos	\department
\license	—
\linja	\studyline
\paikka	—
\setauthor	\author
\termlist	thetermlist-ympäristö
\tyyppi	\type
\yhteystiedot	\contactinformation
\yliopisto	\university
\ysa	—

Taulukko 2. Komentomuutokset gradu2:sta gradu3:een

Isoin työ voi aiheutua lähdeluettelon laatimistekniikan muuttumiseen sopeutumisesta.

B Harvemmin tarvittavat ominaisuudet

Aiemmin esiteltyjen lisäksi gradu3 tarjoaa seuraavat lisäominaisuudet:

- `LATEX 2ε`:n vakio-optiot `draft` ja `final` toimivat.
- Vaikka tutkielman suomenkielisyyttä ei tarvitse erikseen mainita, `finnish`-optio toimii.
- `\university`-komennolla voit ilmoittaa tutkielman kotiyliopistoksi jonkin muun kuin Jyväskylän yliopiston.
- `\department`-komennolla voit ilmoittaa tutkielman kotilaitokseksi jonkin muun kuin Tietotekniikan laitoksen.
- `\subject`-komennolla voit ilmoittaa tutkielman oppiaineeksi jonkin muun kuin tietotekniikan. Huomaa, että oppiaine tulisi suomenkielisissä tutkielmissa kirjoittaa genetiivimuodossa ja isolla alkukirjaimella ("`Tietotekniikan`"), englanninkielisissä tutkielmissa in-preposition kanssa ("`in Information Technology`").
- `\type`-komennolla voit ilmoittaa tutkielman tyypin, jos se on jokin muu kuin `pro gradu` (oletus) tai kandidaatintutkielma (optiolla `bachelor`).
- `\setdate`-komennolla voit asettaa päivämäärän haluamaksesi. Anna komennolle kolme parametria – päivä, kuukausi ja vuosi – numeerisessa muodossa.
- Ympäristöllä `chapterquote` voit laittaa luvun alkuun mietelauseen. Sillä on yksi pakollinen parametri (lainauksen attribuutio).
- Komento `\graducslsdate` sisältää käytössä olevan gradu3:n julkaisupäivämäärän ja `\graducslsversion` sen versionumeron.