Opinnäytetyö (AMK)

Tietotekniikka

Peliteknologia

2017

Santeri Halkivaha

WWW-POHJAISEN TOIMINNANOHJAUSJÄRJESTELMÄN KIELIVERSIOINTI

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

2017 | 54

Yliopettaja Mika Luimula, dosentti

Santeri Halkivaha

WWW-POHJAISEN TOIMINNANOHJAUSJÄRJESTELMÄN KIELIVERSIOINTI

Opinnäytetyön tarkoituksena oli tutkia niitä teknisiä vaihtoehtoja, joilla KuljetusVelho-ohjelmistoon voidaan toteuttaa kieliversiointi ja löytää tarkoitukseen parhaiten sopiva ratkaisu käytännön sovelluksineen. Päämääränä oli, että KuljetusVelho voidaan jatkossa laajentaa käyttämään useita eri kieliä ja tavoitteeksi asetettiin lokalisoinnin toteutuminen.

Lokalisointia ja kieliversiointia tehdään ohjelmiston käytettävyyden parantamiseksi siirryttäessä uusille markkina-alueille. Lokalisointi ei rajoitu pelkästään kielen kääntämiseen vaan kyse on kokonaisuudesta, joka ottaa kielen lisäksi huomioon myös kulttuurin vaikutuksen. Lokalisoinnin tuloksia pystytään arvioimaan parhaiten tutkimalla ohjelmiston käytettävyyttä. Kieliversioinnin toteutus rakennettiinkin käytettävyyttä silmällä pitäen.

Kieliversiointiin on olemassa lukuisia erilaisia teknisiä toteutuksia eikä ole olemassa mitään yleistä toimintatapaa kieliversioinnin tekniseen toteutukseen. Jokaiselle ohjelmistolle on lähtökohtaisesti rakennettava tarpeisiin parhaiten sopiva ratkaisu. Työtä helpottaa, jos lokalisoinnin mahdollisuus on otettu huomioon jo ohjelmistoa suunniteltaessa. Lokalisointi vaatii paljon aikaa ja resursseja, sillä sitä on hankala automatisoida. Lokalisointi on kuitenkin kansainvälisille ohjelmistoille elinehto.

Ohjelmistojen käytettävyyttä ja teknisiä toteutusvaihtoehtoja kieliversioinnin toteuttamiseen tutkittiin tätä työtä varten parhaan mahdollisen toteutuksen rakentamiseksi KuljetusVelhoon. Lisäksi työssä tutkittiin käytettävyyteen ja kulttuurin vaikutukseen liittyviä näkökulmia, joiden toteuttamiseen osana tätä työtä ei jäänyt riittävästi aikaa. Käytettävyyteen liittyvää tutkimusta hyödynnetään kuitenkin jatkokehityksessä tämän työn jälkeen. Myös tietokantojen suunnittelua tutkittiin parhaan mahdollisen tietokantatoteutuksen toteuttamiseksi kieliversiointiin. Kieliversiointi toteutettiin niin, että käännökset haetaan tietokannasta ja sijoitetaan käyttöliittymään. Tietokannasta haettu sanasto talletetaan palvelimelle tietokannan rasituksen minimoimiseksi.

Opinnäytetyönä toteutettu kieliversiointi ja lokalisaatio oli lähtökohdat huomioon ottaen iso projekti, joka jatkuu vielä tämän opinnäytetyön jälkeenkin. Opinnäytetyön avulla toteutettiin selvitystyö käytettävyyden, lokalisoinnin ja kulttuurivaikutuksen osalta, johon rakennettiin kieliversioinnin tulevaisuuden kehitystä varten vahva tekninen toteutus.

ASIASANAT:

Kieliversiointi, lokalisaatio, käytettävyys, toiminnanohjausjärjestelmä, tietokanta, PHP

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2017 | 54

Principal lecturer Mika Luimula, adj.prof.

Santeri Halkivaha

DEVELOPING LANGUAGE VERSIONS FOR WEB-BASED ERP SYSTEM

The aim of this thesis was to research technical options for implementing different language versions to the KuljetusVelho ERP system and to find the best solution with technical implementation. The aim was that KuljetusVelho could be used with multiple languages in the future and be fully localized.

There are multiple technical solutions available in localization and translating the software. There is no common procedure available when it comes to software translations. Each solution must be designed so that it is suitable to the target software. The process is easier when the possibility of localization is taken into account when designing the software. Localization is time and resource intensive because it is hard to automatize. However, localization is essential to international software.

Localization is implemented to improve software's usability when moving to new market areas. Localization is not only limited to translating the language but is a whole that takes into account the impact of culture as well. The results of localization process can be evaluated best by researching the usability of the software.

The language versions implemented in this thesis were built with usability in mind. Software usability theory and different technical implementations were researched for this thesis to find out what would be the best way to develop language versions for KuljetusVelho. Databases were also researched to be able to build the best possible database solution for language versions.

Language versions were implemented so that the translations are retrieved from the database and then placed in the user interface. The dictionary retrieved from the database is saved in the server to minimize database load. In this thesis, a demo version for language versions for KuljetusVelho was built. Changes were made to the small portion of the software to be able to study the subject further and run some tests in the future. The localization and language versions developed during this thesis was a part of a huge project which continues after this thesis and, therefore, further development is required to finalize the project.

KEYWORDS:

Software translation, localization, usability, ERP, database, PHP

SISÄLTÖ

KUVAT

TAULUKOT

SANASTO

CSS	Cascading Style Sheets. Www-sivujen tyyliohjeet määrittävä tiedosto.
ERP	Enterprise Resource Planning eli toiminnanohjausjärjestelmä. Yrityksen ohjaamiseen tarkoitettu tietojärjestelmä.
HTML	Hypertext Markup Language. Kuvauskieli, jolla voidaan kuvata www-sivujen tekstin rakennetta tai esitystapaa.
Lokalisointi	Lokalisoinnilla tarkoitetaan tuotteen sovittamista vieraaseen käyttöympäristöön, erityisesti vieraiden maiden kulttuuriin, muuttamalla tuotetta sopivammaksi määrätyssä kohteessa tapahtuvaa käyttöä varten. Lokalisointiin sisältyy niin kääntäminen, kulttuurin vaikutus kuin paikalliset säännöt.
PHP	PHP: Hypertext Preprocessor tai Personal Home Page. Ohjelmointikieli, jota käytetään www-palvelinympäristössä.
SaaS	Software as a Service. Yleensä www-pohjainen ohjelmistopalvelu, jossa asiakkaat käyttävät samaa tuotantoympäristöä ja käytöstä maksetaan käytön laajuuden mukaan.
Sisällönhallintajärjestelmä	Tietojärjestelmä, jonka avulla organisaatio voi keskitetysti hallita ja kehittää verkkopalveluitaan, www-sisällönhallinta.
SQL	Structured Query Language. IBM:n kehittämä relaatiotietokantojen kyselykieli, jolla tietokantaan voidaan tehdä hakuja, muutoksia ja lisäyksiä.
Unicode	Merkistöstandardi, joka kuvaa yksilöivällä koodiarvolla jokaista kirjoitusmerkkiä.
UTF-8	Unicoden yleisin koodaustapa.

1 JOHDANTO

Tämän päivän ohjelmistoala on kansainvälisesti kilpailtua. Tämä asettaa omat haasteensa yrityksille ja ohjelmistoille, sillä käytännössä vain kyseiselle markkina-alueelle suunnitellut ja lokalisoidut ohjelmistot pärjäävät kilpailussa. Pienet yritykset toimivat hyvin kapealla sektorilla pienillä markkina-alueilla. Näin pienillä yrityksillä on mahdollisuus pärjätä kilpailussa isompiaan vastaan toteuttamalla juuri asiakkaiden tarvitsemia räätälöityjä ratkaisuja pienelle joukolla käyttäjiä. Pienet yritykset eivät pysty kilpailemaan hinnalla vaan ne kilpailevat laadulla. Pienet yritykset kuitenkin törmäävät ongelmiin siirryttäessä tutulta markkina-alueelta uudelle markkina-alueelle.

Uudelle markkina-alueelle siirryttäessä lokalisaatiolla on keskeinen merkitys, ja koska lokalisaatio on hidas ja kallis prosessi, suuret yritykset saavat kilpailuetua puolelleen. Siinä missä suuret kansainväliset pörssiyhtiöt voivat tuoda ohjelmistonsa useille markkina-alueille samaan aikaan välittämättä juuri kustannuksista taikka potentiaalista, on pienten yritysten harkittava ja analysoitava tarkkaan kohdemarkkina-alueen riskit, potentiaali, kulttuurin vaikutus jne. Juuri tämän vuoksi on ensiarvoisen tärkeää keskittyä ohjelmiston lokalisaatioon ja käytettävyyteen. Ottaen huomioon kieliversioinnin ja lokalisaation monimutkaisuuden ja tärkeyden kansainvälisille ohjelmistoille ja yrityksille ohjelmistokehittäjien on hyvä tietää ainakin perusasiat aiheesta. Näitä taitoja omaavat työntekijät ja yritykset saavat edun kilpailijoihinsa nähden.

Tämän opinnäytetyön tarkoituksena oli tutkia ja toteuttaa kieliversiointi KuljetusVelho-toiminnanohjausjärjestelmään. Tarkoituksena oli tutkia eri ratkaisuvaihtoehtoja ja niiden perusteella rakentaa kyseiseen järjestelmään parhaiten sopeutuva ratkaisu. Tavoitteena oli mahdollistaa KuljetusVelho-ohjelmiston lokalisointi. Ratkaisun oli tarkoitus olla kokonaisvaltainen järjestelmän rakenteen muutos sisältäen tietokantaan ja selainpuoleen tehtävät muutokset useiden eri kielten tukemisen mahdollistamiseksi. KuljetusVelho on tarkoitettu erityisesti kuljetusliikkeiden toimintojen hallintaan. Järjestelmä on www-pohjainen ohjelmistopalvelu, joka on toteutettu yksikielisenä omana työnä. Näin ollen ei ole olemassa mitään valmista mallia kieliversioinnin toteuttamiseen, vaan opinnäytetyönä kehitettiin toteutus, jonka avulla on tulevaisuudessa mahdollista helposti lisätä uusia kieliä.

Kieliversiointi ja lokalisointi on elinehto siirryttäessä uusille markkina-alueille. On tärkeää, että ohjelmisto lokalisoidaan paikalliselle kielelle ja kulttuurille sopivaksi. Tämä parantaa ohjelmiston käytettävyyttä ja kilpailukykyä. Aihe on tärkeä niin kansainvälisessä kilpailussa menestymisen kuin myös yritysten kansainvälistymisen ja kasvun kannalta. Ohjelmistojen käytettävyyttä on tutkittu jo vuosikymmenien ajan ja käytettävyydestä on tarjolla runsaasti materiaalia ja tutkimustuloksia. Kieliversioinnin tekniseen toteutukseen ei ole olemassa yleistä toimintatapaa, mikä ilmenee kieliversioinnin toteutuksesta tarjolla olevan materiaalin epäkoherenttiudella.

Opinnäytetyön tavoitteena oli tutkia ja toteuttaa kieliversiointi KuljetusVelho-toiminnanohjausjärjestelmään. Työtä varten tutkittiin käytettävyyttä erityisesti ohjelmistojen, lokalisaation ja käyttäjätestauksen kannalta sekä perehdyttiin tietokantojen suunnitteluun. Teknisen toteutuksen pohjaksi valikoitui käännöstekstit sisältävä tietokantaratkaisu ja tarvittavat käyttöliittymän muutokset niin, että käyttöliittymä tukee sisältöjen hakua tietokannasta. Lisäksi tarvittiin keino lisätä uusia kieliä tietokantaan. Tarkoitus oli myös suunnitella ja toteuttaa kieliversioinnille käyttöliittymä, joka on käyttäjän kannalta mahdollisimman toimiva ja helppokäyttöinen. Työmäärään laajuudesta johtuen päätettiin painottaa kieliversioinnin teknistä puolta ja jättää käytettävyys ja käyttöliittymä näkökulmat vähemmälle huomiolle ja keskittyä niihin jatkokehityksen aikana.

2 LOGISYSTEMS OY

LogiSystems Oy on vuonna 2016 perustettu turkulainen yritys, joka on erikoistunut tilaus-toimitusketjun hallintaan niin logistiikkapalveluja ostavien kuin kuljetuspalveluja tarjoavien yritysten näkökulmasta. LogiSystems työllistää neljä henkilöä. LogiSystemsillä on kolme ohjelmistoa: TilausVelho, KuljetusVelho sekä PoikkeamaVelho. Ohjelmistot on toteutettu SaaS (Software as a Service) ratkaisuina eivätkä ne edellytä asiakkaalta käyttöönottoon liittyviä asennuksia ja ovat asiakkaiden käytettävissä verkkoselaimella alustasta riippumatta. Velho-ohjelmistojen käytöllä lisätään automaatiota, toimitusketjun läpinäkyvyyttä sekä reaaliaikaisuutta ja siten mahdollistetaan kustannustehokkaat sekä laadukkaat prosessit.

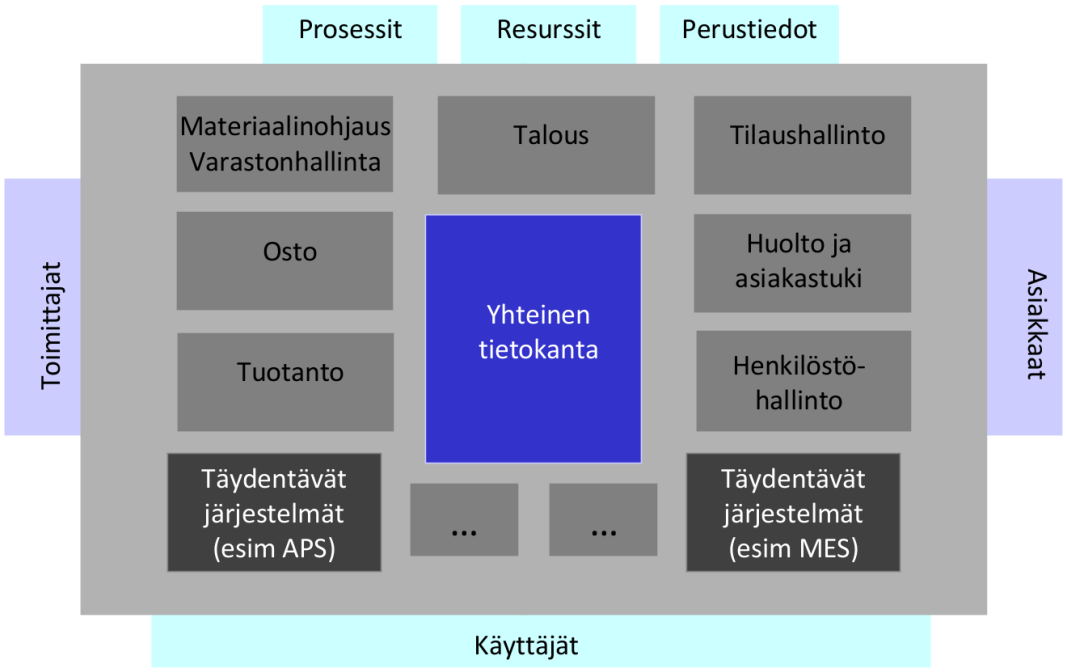
2.1 KuljetusVelho

KuljetusVelho on internet-pohjainen SaaS-ohjelmisto, jonka käyttöön ottaminen ei edellytä mitään asennuksia ja sitä voidaan käyttää verkkoselaimella niin tietokoneella, puhelimella kuin taulutietokoneellakin. KuljetusVelhoa käyttämällä voidaan

- hallinnoida asiakkaille, logistiikkakumppaneille sekä asiakkaiden asiakkaille tarjottavia tuotteita ja palveluja
- muodostaa tuotteista ja/tai palveluista valikoimia
- hallinnoida tuotteiden ja palvelujen tilaamista
- syöttää tilauksia nettilomakkeesta
- tuoda järjestelmään tilauksia erilaisilla vakioiduilla siirtotiedostoilla tai räätälöitynä tiedonsiirtona
- hinnoitella tarjolla olevia tuotteita tai palveluja
    - asiakaskohtaisesti
    - yleisillä hinnastoilla
    - tapauskohtaisesti
- järjestellä kuljetustilaukset manuaalisesti tai automaattisesti
- tiedottaa tilausten statuksista ja kuljetustehtävistä SMS-viesteillä tai sähköpostilla
- määritellä suoritteet valmiiksi laskutusta varten.

2.2 Toiminnanohjausjärjestelmät

Toiminnanohjausjärjestelmillä eli ERP-järjestelmillä (Enterprise Resource Planning) tarkoitetaan laajoja yrityksen ohjaamiseen tarkoitettuja, kokonaisvaltaisia tietojärjestelmiä. Toiminnanohjausjärjestelmä on yrityksen tietojärjestelmä, joka integroi yhteen eri toimintoja, kuten esimerkiksi tuotantoa, jakelua, varastonhallintaa, laskutusta taikka kirjanpitoa. Tyypillistä tällaiselle järjestelmälle on, että sen ytimessä on yksi yhteinen tietokanta, jota kaikki eri toiminnot käyttävät (Kuva 1). Kaikki toiminnot hyödyntävät samaa, ajantasaista tietoa. Toisaalta tämä asettaa myös vaatimuksia tiedon oikeellisuudelle: on erityisen tärkeää, että esimerkiksi perustiedot, kuten materiaalien ja tuotantoresurssien tiedot, ovat oikein ja ajan tasalla. (Logistiikan Maailma 2017).



Kuva 1. Toiminnanohjausjärjestelmän rakenne (Logistiikan Maailma 2017).

ERP-järjestelmillä pyritään parantamaan yrityksen tehokkuutta. ERP mahdollistaa reaaliaikaisen tiedonsiirron eri yritysten välillä. Reaaliaikaisen tiedonsiirron avulla pyritään vähentämään päällekkäistä työtä ja nopeuttamaan asioiden käsittelyä ja päätöksentekoa. Reaaliaikainen tiedon jako mahdollistaa koko yrityksen toiminnan parantamisen osastokohtaisen toiminnan optimoimisen sijasta. Perinteisesti yrityksen

jokaisella osastolla oli omat ohjelmistonsa, jotka eivät kommunikoineet keskenään. Tällaisia eristettyjä rakenteita ei voida synkronoida keskenään, mikä hidasti organisaation toimintaa ja laski tehokkuutta. (Spencer 2016).

Nykyajan liike-elämässä on kaikilla käytössään jokin järjestelmä liiketoiminnan ohjaukseen. ERP onkin syntynyt korvaamaan manuaalista kirjanpitoa ja toimintoja ja siten nopeuttamaan jokapäiväistä työskentelyä. Tietojärjestelmien kehityksen myötä on selvää, että myös yritysten kaikki toiminnot linkittyvät johonkin tietojärjestelmään.

ERP-järjestelmät ovat usein monimutkaisia kokonaisuuksia, joiden ylläpitäminen vaatii runsaasti asiantuntemusta. Perinteisesti monilla yrityksillä on ollut tapana tehdä ja ylläpitää toiminnanohjaukseen käytetyt järjestelmät itse, mutta nykyään monet ovat siirtyneet käyttämään valmiita ohjelmistoja, muun muassa järjestelmien monimutkaisuudesta johtuen. (Spencer 2016).

3 KÄYTETTÄVYYS

Käytettävyys on menetelmä- ja teoriakenttä, jonka kautta käyttäjän ja laitteen yhteistoimintaa pyritään saamaan tehokkaammaksi ja käyttäjän kannalta miellyttävämmäksi (Sinkkonen ym. 2009a, 12). Käytettävyys käyttää hyväksi psykologian sekä ihmisen ja koneen vuorovaikutuksen tutkimusta. Jacob Nielsen määrittelee käytettävyyden osaksi tuotteen käyttökelpoisuutta. Tuotteen käyttökelpoisuuteen vaikuttavia tekijöitä on monia, ja käytettävyys on niistä vain yksi. Toisaalta, käytettävyydenkin on oltava kunnossa, jotta tuote olisi käyttökelpoinen. Nielsenin mukaan hyvän käytettävyyden muodostavat käyttötilanteen opittavuus, virheettömyys, muistettavuus, tehokkuus ja miellyttävyys. (Nielsen 1995.)

Ihmisen toiminnasta tiedetään joitain asioita, jotka näyttävät pätevän enemmän tai vähemmän kaikkiin ihmisiin. Ihmisiä on kuitenkin niin paljon, että melkein kaikki säännöt sisältävät poikkeuksia. Ihmisten fysiologiset ja psykologiset ominaisuudet vaihtelevat ja myöskin kulttuurilla on oma merkityksensä siihen, miten toimimme. Ihmisiltä odotetaan myös erilaisiin toimintaympäristöihin liittyviä valmiuksia. (Sinkkonen ym. 2009a, 16–19.)

Lisäksi ihmisen toimintaan vaikuttavat vaihtelevat kulttuurielementit, kuten muoti, alakulttuurit, yrityskohtaiset toimintatavat, käyttäjän tehtävät ja käyttötilanne. Lisäksi tila, jossa toimitaan ja sen olosuhteet, kuin myös yksilön toimintarajoitukset ja –kyvyt. Nämä ovat seikkoja, jotka on selvitettävä jokaisessa tuotteen suunnitteluprojektissa erikseen. Kuva 2 kuvaa ihmisen toimintaa ja käyttöympäristöä. (Sinkkonen ym. 2009a, 16–19.)

Kuva 2. Ihmisen toiminta ja tuotteen käyttöympäristö (Sinkkonen ym. 2009a, 18).

Ihmiset ovat tuotteen käyttäjinä erilaisia. Ihmisillä on kuitenkin ominaisuuksia, jotka ovat yhteisiä kaikille ja jotka eivät muutu. Punaisen osan ominaisuuksia voidaan pitää tilanteesta riippumattomina ihmiseen ja tuotteeseen liittyvänä yleistietona. Ihmisen toiminnan perustana ovat synnynnäiset ominaisuudet, jotka jokainen terveenä syntynyt ihminen oppii, esimerkiksi käveleminen ja puhuminen. Puhekieli, käsitteiden muodostuminen ja asioiden merkitykset ja opitut asiat kuuluvat kulttuuriin. Kulttuurin sisällä on joukko alakulttuureja, kuten toimintakulttuurit, esimerkiksi web-kulttuuri, jotka asettavat suunnittelijalle tietynlaisen odotuspohjan. Näiden asioiden voidaan katsoa pysyvän pitkälti samoina hyvinkin laajassa joukossa ihmisiä. (Sinkkonen ym. 2009a, 18–19.)

Turkoosin osan asioita ei voi päätellä, vaan ne on selvitettävä jokaisessa projektissa erikseen. Nämä ominaisuudet riippuvat tilanteesta, jossa tiettyä laitetta tai tuotetta käytetään, ihmisistä, jotka sitä käyttävät, ja tehtävistä, joihin se on tarkoitettu. Tehtävien tulisi sujua paremmin tai miellyttävämmin, kuin ilman tuotetta, ja tuotteen

tulisi tukea mahdollisimman hyvin tehtäviä, joiden tueksi se on tarkoitettu. (Sinkkonen ym. 2009a, 18–19.)

3.1 Käyttäjien toiminnan ymmärtäminen

Käyttäjien toimintaa voidaan yrittää ymmärtää monien eri käsitejärjestelmien avulla. Usein lähtökohtana on itsensä tarkasteluun perustuva ihmiskäsitys. Harvalla suunnittelijalla on ihmistieteellistä koulutusta. Arkitiedolla on keskeinen asema käyttäjien vuorovaikutuksen tutkinnassa, jossa joudutaan teknisen koulutuksen pohjalta pohtimaan ihmisen toimintaa. (Saariluoma & Oulasvirta 2011, 48–51.)

Arkitietoon perustuva käytettävyysosaaminen ei kuitenkaan ole riskitön suunnittelutapa. Arkitieto ei tarjoa riittävää tietoa käyttäjistä, koska arkitieto ei ole riittävän analyyttistä, se ei perustu testituloksiin eikä tutkimuksiin eikä sen perusteella voida tehdä perusteltua suunnittelua. Arkitieto ei ole yhtä täsmällistä, kattavaa ja oikeaksi todistettua kuin tutkimuksiin perustuva tieto. Se perustuu kuitenkin yhden ihmisen kokemuksiin ja tarkastelun ulkopuolelle jää usein ihmistä koskevat tieteelliset näkökulmat, kuten alitajunta, aivojen toiminta ja kulttuurierot. (Saariluoma & Oulasvirta 2011, 48–51.)

3.1.1 Käyttäjäpsykologia

Kuten todettua, itsehavainnointiin perustuva käytettävyyssuunnittelu ei useinkaan ole riittävän tarkka, saati kattava tapa tehdä käytettävyyssuunnittelua. Suunnittelijoiden on usein vaikea ymmärtää tavallisten käyttäjien ongelmia asiantuntija-asemastaan johtuen. Lisäksi ihmisten toiminta perustuu usein tiedostamattomiin ja alitajuisiin mekanismeihin. Käyttäjien toimintaa voidaan kuitenkin tutkia psykologisilla tutkimusmenetelmillä. (Saariluoma 2004, 29–31.)

Käyttäjäpsykologiassa tutkimusmenetelmät jaetaan yleensä laadullisiin eli kvalitatiivisiin ja määrällisiin eli kvantitatiivisiin mittareihin. Kvalitatiivinen mittaaminen perustuu ilmiön määrittämiseen. Kvalitatiivinen havainto voi olla mikä vaan määriteltävissä ja eriteltävissä oleva ilmiö, esimerkiksi käyttäjän vaikeus löytää kielivaihtoehtoja ohjelmasta taikka ohjelman eri käyttötavat. Kvantitatiivinen mittaaminen perustuu puolestaan mitattavan ilmiön määrälliseen ominaisuuteen ja on numeerinen,

esimerkiksi kuinka monta sekuntia käyttäjällä kestää ohjelmiston kielen vaihtaminen. Yleisesti kallistutaan kvantitatiiviseen tutkimukseen kvalitatiivisen tutkimuksen kustannuksella. On mahdollista suorittaa nopeasti paljon numeerisia testejä, jotka antavat tietoa siitä, miten helposti käytettävinä tutkittavat henkilöt pitävät käyttöliittymää. Tämä numeerinen yleisarvio ei kuitenkaan auta vastaamaan siihen, miten käyttöliittymää pitää kehittää ja mitkä ominaisuudet aiheuttavat ongelmia ja miten näitä voidaan parantaa. Siksi onkin syytä suorittaa myös kvalitatiivista tutkimusta. (Saariluoma 2004, 34–37.)

Havainnointi ja osallistuva havainnointi on yksi käyttäjäpsykologian tutkimusmenetelmistä. Havainnoivassa tutkimuksessa tutkija seuraa tutkimuksen kannalta kiinnostavia toimintoja puuttumatta kuitenkaan asioiden kulkuun, kun taas puolestaan osallistuvassa havainnoinnissa tutkija osallistuu aktiivisesti tutkittavien toimintaan huomioita tehdessään. Kaikkiin havainnointiin perustuviin tutkimusmenetelmiin liittyy riskinsä. Tutkijan voi olla vaikea tulkita tilanteita objektiivisesti ja riippumattomasti, vaan hahmottaa tilanteet käsitteellisesti omiin käsitystapoihinsa perustaen. Jos esimerkiksi tutkija (ohjelmistokehittäjä), joka ei ole tutkittavan ohjelmiston käyttökohteen (logistiikka) ammattilainen, seuraa käyttökohteen ammattilaisen työskentelyä ohjelmiston parissa, on vaarana se, että tutkija tekee virheellisiä johtopäätöksiä siitä, miten ohjelmistoa käytetään. Riskejä ei pidä kuitenkaan liioitella. Tuloksien luotettavuutta voidaan parantaa tekemällä tarkentavia haastatteluja ja käyttämällä useita tutkijoita. Tutkijan on myös hyvä miettiä, mitä hän pystyy luotettavasti arvioimaan ja hakea apua niihin kohtiin, jotka ylittävät hänen oman osaamisalueensa. Havainnoinnin eri muodoilla saatuja tuloksia voidaan luotettavasti soveltaa käytännön ongelmien ratkaisuun, koska tarkkailtavat tapahtumat ovat luonnollisessa ympäristössään ja havaintojen ja todellisuuden ero on varsin pieni. (Saariluoma 2004, 38–40; Sinkkonen ym. 2009b, 100–103; Saariluoma ym. 2010, 190–192.)

Suurin osa käytettävyystutkimuksesta on käyttötestausta. Käyttötestaus on usein luonteeltaan psykologisia kokeita. Kokeen taustalla on usein käyttäjän toiminnan tutkimus. Kokeessa manipuloidaan ja muunnellaan tilanteita ja mitataan muutosten vaikutusta koehenkilöön. Tulokset ovat koetilanteen muuntelun ja koehenkilön reaktioiden välisiä yhteyksiä. Saatujen tulosten pohjalta tehdään päätelmiä taustalla olevista ilmiöistä. Usein käytetään myös kontrolliryhmää, johon saatuja tuloksia verrataan. Kokeellisessa tutkimuksessa mittauksia pystytään kontrolloimaan hyvin,

mutta toisaalta kokeissa voidaan tutkia vain paria asiaa kerrallaan, jolloin vaaditaan useita kokeita luotettavan kuvan saamiseksi tutkimuskohteesta. (Saariluoma 2004, 40–42; Saariluoma ym. 2010, 192–197.)

Käyttäjätutkimuksessa on yleisesti käytössä erilaiset kyselyt. Kyselyt jaetaan avoimiin ja suljettuihin kyselyihin. Ensimmäisessä määritellään kyselyn tarkoitus ja annetaan vastaajan vapaasti kirjoittaa mielipiteensä. Jälkimmäisessä tutkijat määrittelevät kysymykset ja antavat niihin vastausvaihtoehdot. Usein avoimia ja suljettuja kyselyitä sekoitetaan keskenään ja riippuu ongelmasta, miten kysymykset esitetään. Avoimet kysymykset antavat vastaajalle vapautta ilmaista mielipidettään, kun taas suljetut kysymykset antavat täsmällisempää tietoa siitä, mistä tutkijat ovat kiinnostuneita. Kyselyihin liittyy monia riskejä. Kysymykset voivat olla epätarkoituksenmukaisia, harhaanjohtavia, asenteellisia taikka vaikeasti ymmärrettäviä. Näin syntyy helposti mittausvirheitä. Toisaalta kyselyn avulla voidaan saada nopeasti ja pienin kustannuksin paljon tietoa. (Saariluoma 2004, 42–45; Sinkkonen ym. 2009b, 107–114; Saariluoma ym. 2010, 197–199.)

Käyttäjähaastattelut kuuluvat käyttäjätutkimuksen perusmenetelmiin. Haastattelussa tutkija haastattelee tutkittavaa henkilöä ja selvittää kyselemällä ja keskustelemalla tämän tutkittavaa asiaa koskevia näkemyksiä. Haastattelut voivat olla joko strukturoituja tai strukturoimattomia. Ensimmäinen sisältää ennalta määriteltyjä kysymyksiä ja jälkimmäisessä haastattelija jättää painotettavien näkökulmien valinnan haastateltavalle. Välimuotona haastattelija määrittää aiheet ennalta, mutta haastateltavalle annetaan mahdollisuus painottaa omia näkökulmiaan. Haastattelun avulla voidaan saada vastaavia tuloksia kuin kyselyillä, mutta haastattelussa on mahdollista tarvittaessa syventyä tarkemmin johonkin aiheeseen. Haastattelututkimuksen tulosten laatuun liittyy samoja ongelmia kuin kyselyissäkin, kuten kysymysten laatu. Haastattelut ovat myös kallis ja aikaa vievä käyttäjätutkimusmetodi. Toisaalta haastattelut yhdistettynä muihin edellä esitettyihin tutkimusmetodeihin on hyvin joustava ja tehokas tapa saada tietoa käyttäjiltä. (Saariluoma 2004, 45–47; Sinkkonen ym. 2009b, 83–96; Saariluoma ym. 2010, 199–201.)

3.1.2 Kulttuurin vaikutus

Kulttuuri on yhteisön käyttäytymispiirteiden ja ihmisen muokkaaman fyysisen maailman muodostama moniulotteinen kokonaisuus. Myös käyttäjän toiminta on paljolti kulttuurisesti määräytynyttä. Sellaiset biologisesti määräytyvät ominaisuudet, kuten muisti ja värinäkö, ovat useimmille ihmisille hyvin samanlaisia, mutta esimerkiksi muistisisällöt ja värien havaitseminen riippuvat kulttuurista. Kulttuurierot vaikuttavat havaitsemiseen ja toimintaan ja muodostavat suuren osan toimintaympäristöstä. Esimerkiksi länsimainen käyttäjä etsii valikoita mieluiten näytön vasemmasta reunasta, kun taas kiinalainen puolestaan oikealta. Osa käytettävyysseikoista riippuu pääasiassa kulttuurista, kuten se luetaanko tekstiä vasemmalta oikealle tai päinvastoin, mutta osa on geneettisesti määräytynyttä, kuten värien erottelukyky. (Sinkkonen ym. 2009a, 31–32; Saariluoma ym. 2010, 78–80.)

Eri alueilla elävien ja historiallisesti eri aikoina eläneiden ihmisten käyttäytymiserot selittyvät pääasiassa kulttuurieroilla, eivät geneettisillä eroilla. Kun puhutaan kulttuurieroista, ei tarkoiteta pelkästään eri alueilla asuvien ihmisten eroja. Myös saman kulttuurin eri ryhmien välillä on kulttuurieroja. Eri alakulttuureja voi määrittää niin kaupunginosa, sosiaaliluokat kuin musiikkimaku. Nykyaikaisessa kaupunkikulttuurissa ihmiset eivät yleensä kuulu vain johonkin tiettyyn alakulttuuriin, vaan useampaan alakulttuuriin yhtä aikaa. Lisäksi useilla yrityksillä ja ammattiryhmillä on omat kulttuurinsa. Samaan ammattiryhmään kuuluvat jakavat koulutuksensa kautta samat arvot ja tietämyksen. Kulttuuri ei ole enää paikkaan sidottua. Internetin aikakaudella ihmiset eri puolelta maailmaa voivat olla yhteyksissä toisiinsa. Tämä lisää yksilöllistä vaihtelua ja vaikeuttaa usein täsmällisten käyttäjäryhmien määrittelyä. (Sinkkonen ym. 2009a, 32–35; Saariluoma ym. 2010, 78–80.)

Myös eri sukupolvet eroavat toisistaan kulttuurisesti. Asia, joka on edelliselle sukupolvelle ollut vieras, on tuttu seuraavalle sukupolvelle. Kun puhutaan ihmisten ikäkausista, niin kysymys ei ole puhtaasti ihmisen iästä, vaan myös kulttuurisukupolvesta. Ikääntyneinäkin nykyiset keski-ikäiset käyttävät elektronisia tuotteita ja tietojärjestelmiä huomattavasti paremmin kuin nykyisten ikääntyneiden enemmistö. (Sinkkonen ym. 2009a, 32–35.)

Vaikka terveillä eri ikäisillä ihmisillä saattaa olla jossain määrin toisistaan poikkeavat kyvyt, niin ikä kertoo ihmisestä vain vähän. Erilaisissa suoritusaikaa ja -kykyä

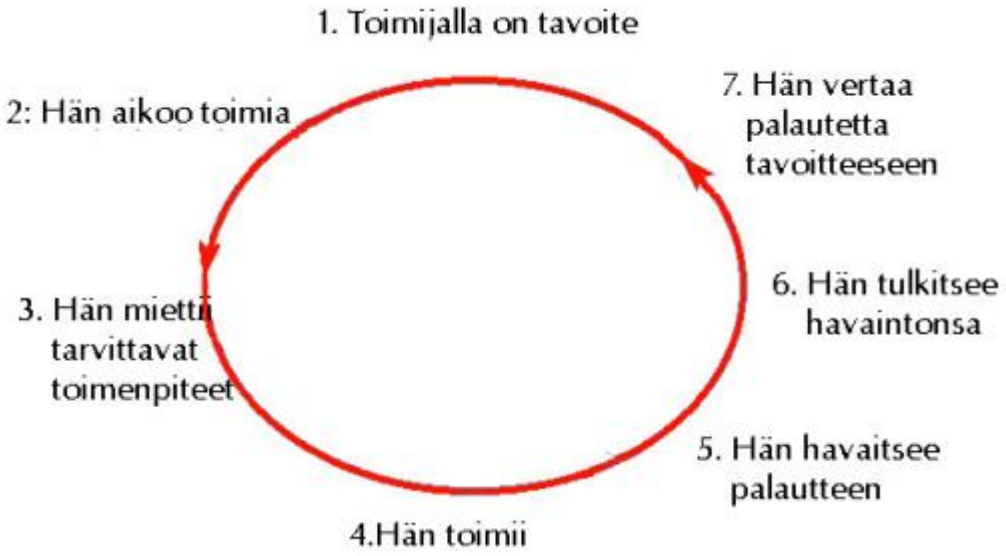
mittaavissa testeissä ikääntyneiden henkilöiden suoritusajat vaihtelevat ikäryhmän sisällä enemmän kuin nuorempien henkilöiden. Tästä voidaan päätellä, että tyypillinen vanhus on vielä harvinaisempi kuin tyypillinen nuori. Muun muassa tästä syystä iän vaikutuksia tutkittaessa poikittaistestit, joissa tutkitaan samanaikaisesti eri-ikäisiä ihmisiä, ovat epäluotettavia. Pitkittäistestit, joissa samaa ihmistä tutkitaan eri ikäkausina, ovat huomattavasti luotettavampia, mutta hitaampia. (Sinkkonen ym. 2009a, 32–35.)

Kaikki kulttuurielementit eivät kuitenkaan vaihtele taikka muutu. Esimerkiksi kieli, lait ja standardit ovat suhteellisen muuttumattomia toisin kuin vaikkapa muoti. Ohjelmiston navigointi on väline päästä sisällön luokse, ja siksi sen pitäisi olla mahdollisimman helppoa ja luonnollista. Navigointiin liittyvät elementit pitäisi suunnitella käytettävyysnäkökulmasta. Kieliopista ja sanastosta kannattaa pitää kiinni, koska ne helpottavat sisällön ymmärrettävyyttä. (Sinkkonen ym. 2009a, 35–37.)

3.2 Ohjelmiston käyttäminen

Käyttäessään ohjelmistoa ihminen pyrkii johonkin päämäärään, esimerkiksi tallettamaan kuljetustilauksen järjestelmään. Kaikessa tavoitteellisessa toiminnassa on kolme perusvaihetta: tavoitteen asettaminen, toiminnon tai toimenpiteen tekeminen ja vaikutuksen tarkastaminen. Esimerkiksi kuljetustilausta tallettaessaan käyttäjä asettaa tavoitteeksi tilauksen tallentamisen järjestelmään, tämän jälkeen suorittaa kyseisen toimenpiteen ja lopuksi tarkistaa/havainnoi, että kuljetustilaus tallentui järjestelmään oikein. (Sinkkonen ym. 2009a, 40.)

Psykologi Donald A. Norman on tutkinut ihmisen toimintaa ja kehittänyt seitsenvaiheisen toiminnan mallin (Kuva 3).



Kuva 3. Normanin toiminnan malli (Sinkkonen ym. 2009a, 48).

Suurin osa toiminnasta ei vaadi kaikkien vaiheiden läpikäyntiä ja enemmistö toimista ei valmistu yhdellä toimenpiteellä. Vaaditaan useita toimenpiteitä ja toiminta voi kestää useita tunteja taikka päiviä. Tällöin sekvenssiä toistetaan useita kertoja useita toimenpiteitä suoritettaessa, kunnes toiminta on saatu suoritettua. Sekvenssi tarjoaa jatkuvan palautteen saamisen, missä yhdestä toimenpiteestä saatu palaute ohjaa tulevia toimenpiteitä, tavoite johtaa alitavoitteisiin ja aikomukset johtavat aliaikomuksiin. Välillä toiminnan tavoite unohtuu taikka se uudelleen muotoillaan tai hylätään. (Norman 1990, 45–49.)

3.2.1 Havaitseminen

Käyttääkseen ohjelmistoa käyttäjän pitää pystyä havaitsemaan tuotteessa kaikki tehtävän suorittamisen kannalta oleellinen. Käyttäjän pitää pystyä seuraamaan toimenpiteidensä vaikutusta ohjelmiston tilaan. Jos käyttäjä ei näe kaikkea mitä pitäisi, syynä on joko se, että väärä asia käyttöliittymässä vie hänen huomionsa tai se, että asiat eivät hahmotu hänelle tai hahmottuvat väärin. Havaitseminen ei ole pelkkää aistimista. Ei riitä, että asiat ovat käyttöliittymässä, vaan käyttäjän pitää pystyä tunnistamaan ne. Käyttäjällä on yleensä ennakkokäsityksiä ohjelmasta. Käyttäessään ohjelmistoa, käyttäjä odottaa löytävänsä vastaavuuksia tehtävänsä ja tuotteen mahdollisuuksien välissä. Käyttäjä tulkitsee näkemänsä tai kuulemansa käyttäen

ennakkokäsityksiään asiasta. Muistiin jää tulkinta, ei objektiivinen totuus. Tulkintaan vaikuttavat käyttäjän toiminnan aiemmat kokemukset, opit ja ennakkoluulot. Käyttäjä hakee tyypillisesti selitysmalleja havaitsemilleen asioille, ja havainnot vääristyvät todennäköisempään suuntaan. Esimerkiksi näytöllä oleva valkoinen pitkulainen muoto nähdään kentäksi ikkunassa vain, jos käyttäjä tietää, millainen kenttä on. (Wiio 2004, 129–150; Sinkkonen ym. 2009a, 58–60.)

Käytettävyyden tutkija Jacob Nielsenin on kehittänyt kymmenen heuristiikkaa hyvän käyttöliittymän käytettävyyden arviointiin. Näistä Nielsenin heuristiikoista osa koskettaa myös havaitsemista. Ensinnäkin ohjelmiston tilan tulee aina olla havaittavissa. Ohjelmiston pitäisi aina pitää käyttäjät informoituina siitä, mitä on tapahtumassa asiaankuuluvalla palautteella kohtuullisessa ajassa. Toiseksi käytön vapaus ja käyttäjän kontrolli ohjelmasta voidaan laskea kuuluvaksi havainnointiin. Usein käyttäjät tekevät virheitä ohjelmassa, kuten napsauttavat väärää toimintoa. Tällöin vaaditaan mahdollisuutta palata takaisin taikka poistua kyseisestä ei-toivotusta tilasta. Tämä pitäisikin olla hyvin selkeästi esillä ja helposti havainnoitavissa. Kolmanneksi esteettinen ja minimalistinen suunnittelu edustaa havainnoitavuutta. Tekstisisältöjen ei pitäisi sisältää asiaankuulumattomia taikka tarpeettomia sisältöjä. Kaikki ylimääräinen informaatio vie huomion pois relevantilta informaatiolta ja heikentää sen havainnoitavuutta. (Nielsen 1995.)

Donald A. Norman on tutkinut käytettävyyttä ja kehittänyt neljä käyttöliittymän suunnitteluperiaatetta. Näistä ensimmäinen, näkyvyys, koskee havainnointia. Käyttöliittymässä oikeiden osien on oltava näkyviä ja niiden on välitettävä oikea viesti. Esimerkiksi valkoinen laatikko mustilla reunoilla kertoo, että siihen voi kirjoittaa. Myös toinen Normanin suunnitteluperiaatteista, käyttömahdollisuudet, liittyy havainnointiin. Käyttömahdollisuus tarkoittaa ominaisuuksia, joita esineellä on tai joita sillä näyttää olevan. Ominaisuudet määräävät, miten esinettä on mahdollista käyttää. Esimerkiksi osoitinlaitetta on mahdollista käyttää useaan eri tehtävään ja sitä liikutettaessa kohdistin kertoo, mitä juuri kyseisessä paikassa osoitinlaitteella voi tehdä. (Norman 1990, 8–28.)

3.2.2 Vuorovaikutus

Hyvässä käyttöliittymässä ulkoasu tuo esiin tuotteen käyttömahdollisuudet ja tekee tuotteesta yhtenäisen kokonaisuuden. Käyttäjän vuorovaikutus ohjelmiston kanssa

perustuu siihen, että käyttäjä osaa lukea suunnittelijan merkkikieltä. Jos kysymyksessä on komentopohjainen järjestelmä, se antaa käyttäjälle kehotteen ja käyttäjä vastaa asiaankuuluvalla komennolla, jos osaa eli muistaa komennot ulkoa. Jos kysymyksessä on graafinen käyttöliittymä, käyttäjät ovat oppineet tiettyjen symboleiden tarkoittavan tietynlaista toimintamahdollisuutta, jonka käyttäjä katsoessaan ikkunaa tunnistaa, ja osaa toimia. Vuorovaikutuksen rakentamisessa on ohjelmiston visuaalisella suunnittelulla tärkeä merkitys. Visuaalisessa suunnittelussa on tärkeää selkeä ja yksikäsitteinen suunnittelu, joka auttaa hahmottamaan kokonaisuuksia ja vastaa käyttäjän käsitystä todellisuudesta sekä helpottaa tärkeiden signaalien näkyvyyttä. (Sinkkonen ym. 2009a, 97–98; Berg ym. 2011, 155–157.)

Informaation visualisointi on tärkeää vuorovaikutuksen kannalta, koska visuaalinen esitys on ajattelun työväline. Informaation visualisoinnin tarve syntyy informaation tai valintavaihtoehtojen määrästä. Visualisoinnin laatua on syytä arvioida käyttäjäkokeilla ja sen käyttöä rajoittaa ihmisen tarkan näön alueen kapeus. Visualisointi havainnollistamismuotona tarjoaa mahdollisuuden käsitellä aistien ulottumattomissa olevaa tietoa. Kuvat ja diagrammit tarjoavat suuria määriä tietoa nopeasti ja vaivattomasti. Visualisointi edistää ohjelmiston käytettävyyttä tarjoamalla mahdollisuuden yksinkertaisempaan tiedon esitysmuotoon. Hyvin suunniteltu visualisointi paljastaa poikkeaman tai säännönmukaisuuden yksityiskohdista heti koko näkökentän laajuudelta, kun taas huonosti suunniteltu visualisointi vaatii erityistä tarkkaavaisuutta. Tärkeimmät syyt visualisointiin ovat informaation suuri määrä ja moniulotteisuus. Pyrkimyksenä on vähentää yksityiskohtien välistä vertailua. (Berg ym. 2011, 155–157.)

Nielsenin heuristiikoista joustavuus ja käytön tehokkuus sekä käyttäjien auttaminen tunnistamaan, määrittämään ja korjaamaan virheet perustuvat käyttäjän ja ohjelman väliseen vuorovaikutukseen. Joustava ja tehokas käyttö toteutuu ainoastaan, jos sekä aloittelevat että kokeneet käyttäjät pystyvät käyttämään ohjelmaa haluamallaan tavalla. Tämä tarkoittaa usein kokeneiden käyttäjien kohdalla pikanäppäinten käyttämistä usein toistuvien toimintojen kohdalla, mutta myös toisaalta mahdollisuutta muokata komentoja haluamikseen. Virheviestin sisältö tulisi olla käyttäjälle ymmärrettävä eikä sisältää pelkästään virhekoodia. Virheviestin pitäisi ilmaista käyttäjälle mistä ongelma johtuu ja miten se on mahdollista korjata. (Nielsen 1995.)

3.2.3 Muisti ja oppiminen

Aktiivisen toiminnan edellytys on, että aikaisemmat havainnot ja kokemukset ovat uudelleen käytettävissä. Uudelleenkäytön perustan rakentaa muisti. Muisti mahdollistaa myös pitkäjänteisen ja suunnitelmallisen toiminnan, erilaisten asioiden tekemisen suunnittelun, suunnitelmien edelleen kehittämisen ja niiden mukaan toimimisen. Tyypillisesti muisti jaetaan kolmeen osaan: työmuistiin, pitkäkestoiseen työmuistiin ja pitkäkestoiseen muistiin. Työmuisti voidaan jakaa keskusmuistiin sekä visuaaliseen ja auditiiviseen varastoon. Työmuistin varastot ovat toisistaan riippumattomia siinä mielessä, että yhden varaston häiritseminen tehtävän avulla ei vaikuta toisen prosessointiin. Esimerkiksi konekirjoittaja voi kirjoittaa ja seurata puhetta saman aikaisesti. Toisaalta, jos ihminen pyrkii tekemään kahta visuaalista tai auditiivista tehtävää samanaikaisesti, tämä ei yleensä onnistu. Työmuistin laajuudeksi on arvioitu noin 4–7 yksikköä. Työmuistin kapasiteetti on niin pieni, että se aiheuttaa ongelmia kaikissa työmuistia kuormittavissa tehtävissä. Ihminen kykenee koodaamaan yksittäisiä asioita suuremmiksi kokonaisuuksiksi. Työmuisti rajoittaa ainoastaan yksiköiden määrää, ei itse yksikön kokoa. Esimerkiksi kirjain, sana tai lause voivat kaikki kuormittaa lähes yhtä paljon muistia. Ihminen kykenee ikään kuin kiertämään muistinsa rajoituksia. (Saariluoma 2004, 82–88; Sinkkonen ym. 2009a, 142–148; Saariluoma ym. 2010, 66–67.)

Pitkäkestoinen työmuisti säilyttää tietoa työmuistia vakaammassa tilassa. Siihen voidaan varastoida hyvin suuria rakenteita, kuten esimerkiksi reittikarttoja. Pitkäkestoisen työmuistin tehtävä on aktiivisen tehtävän vaatiman tiedon tallentaminen työmuistia pysyvämpään tilaan, jossa se on turvassa häiriötekijöiltä. Se sallii tapahtumien pitämisen mielessä, vaikka tarkkaavaisuus ei kiinnittyisikään niihin. (Saariluoma 2004, 82–88.)

Oppiminen voidaan määritellä siten, että se on suhteellisen pysyvä muutos oppijan tiedoissa ja käytöksessä. Muutoksen aiheuttaja on kokemus, joka syntyy vuorovaikutuksessa ympäristön kanssa. Oppimista pitää osata soveltaa uusissakin tilanteissa. Oppiminen voidaan määritellä myös prosessiksi, jossa oppija joko muodostaa itselleen jonkinlaisen mentaalimallin tai mielikuvan opeteltavasta taidosta tai asiasta ja pystyy soveltamaan tätä mallia tai mielikuvaa uusissa tilanteissa tai hän harjoittelee jonkin suorituksen, kunnes pystyy toistamaan sen puhtaasti. Oppiminen on tiedon tallettamista muistiin, kokemusten karttumista, taitojen kehittymistä, asenteiden

muutoksia tai uudenlaista ymmärtämistä. Uusi tieto varastoidaan pitkäkestoiseen muistiin, joten oppiminen ja pitkäkestoinen muisti ovat toisiaan lähellä olevia käsitteitä. (Saariluoma 2004, 88–91; Sinkkonen ym. 2009a, 149–151.)

Oppimisen tulokset varastoidaan pitkäkestoiseen muistiin. Jos tieto ei ole tallentunut pitkäkestoiseen muistiin, ei voida puhua oppimisesta. Oppimisessa kaikki tapahtuu askeleittain. Ihminen ei opi monimutkaisia toimintosarjoja kerralla, vaan ne jäävät mieleen vasta useiden harjoitusten jälkeen. Pitkäkestoisessa muistissa oleva informaatio on aina organisoitunutta. Satunnaiset ja toisiinsa liittymättömät asiat eivät jää muistiin. Pitkäkestoisella muistilla on kaksi oppimisen kannalta oleellista ominaisuutta. Ensinnäkin oppiminen riippuu aina siitä, mitä on jo opittu. Toiseksi oppiminen tapahtuu vähitellen ajan kuluessa. Oppiminen ei tapahdu hetkessä, vaan uudet asiat integroituvat aina pitkäkestoisessa muistissa jo oleviin asioihin. (Saariluoma 2004, 88–91.)

Ajattelu, pohtiminen ja ihmettely on tärkeää asioiden ymmärtämisen ja sitä kautta oppimisen kannalta. Päätöksenteko on valitsemista useasta eri mahdollisuudesta, joista päätöksentekijän on oltava tietoinen, eli hänen on pidettävä ne mielessään. Koska tämä kuormittaa työmuistia, ei kunnolla käsiteltäviä vaihtoehtoja voi olla kovin paljon. Monimutkaisessa päätöksenteossa virheet ja päätöksentekoon kuluva aika ovat suoraan verrannollisia päätökseen vaikuttavien vaihtoehtojen määrään. Perehtyessään uuden ohjelman käyttöön, käyttäjä voi käyttää yleisiä ongelmanratkaisumenetelmiä, kuten sattumanvaraista kokeilua tai analogista päättelyä. Hän saattaa yrittää käyttää mallina jotain aiemmin käyttämäänsä ohjelmistoa. Työmuistin kapasiteettirajoituksesta johtuen monimutkaiset ongelmat on vaikea ratkaista ilman ulkoista apua. Ongelmanratkaisu etenee niin, että henkilö tekee ensin hypoteesin, testaa sitten hypoteesin käyttämällä ohjelmistoa hypoteesin mukaan ja tarkistaa lopuksi toiminnan vaikutuksen. Onnistuneen ongelmanratkaisun ehto on, että ongelmanratkaisija näkee tai pystyy helposti päättelemään koko ongelma-avaruuden. Ohjelmistojen kohdalla tämä tarkoittaa, että ohjelman käyttöliittymän elementtien asettelulla on suuri merkitys siihen, miten ohjelmistoa yritetään käyttää. (Saariluoma 2004, 82–91; Sinkkonen ym. 2009a, 167–173.)

Opittavuutta on yleisesti pidetty tehokkuuden vastakohtana, mutta selkeä, hyvin tehty ja helposti opittava järjestelmä parantaa myös ohjelmiston tehokkuutta. Virheiden korjaamiseen menee paljon aikaa, ja jos ohjelmisto ei ole johdonmukainen ja yhdenmukainen termistöltään ja toimintatavaltaan sekä selkeä rakenteeltaan, kärsivät

sekä tehokkuus että opittavuus. Jos ohjelmisto on rakennettu niin, että se vastaa käyttäjien tarpeita ja tavoitteita ja siihen on rakennettu kokeneita käyttäjiä varten oikoteitä, mutta aloittelija voi edetä vaihe vaiheelta, saa tuotteesta sekä helppokäyttöisen että tehokkaan. (Sinkkonen ym. 2009a, 194–195.)

Jakob Nielsenin heuristiikkojen mukaan käyttöliittymän tulisi aina olla johdonmukainen ja noudattaa standardeja. Käyttäjän ei pitäisi joutua arvailemaan tarkoittavatko eri sanat, tilanteet ja toiminnot samaa asiaa. Ohjelmiston käytön pitäisi pohjautua tunnistamiseen muistamisen sijaan. Käyttäjän muistin kuormittamista pitäisi pyrkiä minimoimaan tekemällä päämääristä, toiminnoista ja vaihtoehdoista näkyviä. Käyttäjän ei pitäisi joutua muistelemaan käyttömahdollisuuksia. Käyttöohjeiden tulisi olla käyttöliittymässä näkyvällä paikalla ja niiden käyttö mahdollista tarvittaessa. Vaikka onkin parempi, että ohjelmistoa pystyisi käyttämään ilman erillistä ohjeistusta, on kuitenkin syytä tarjota erillistä ohjeistusta ja dokumentaatiota. Ohjeiden pitäisi olla helposti etsittävissä käyttöohjeesta, keskittynyt käyttäjän näkökulmaan, listata konkreettisia toimia eikä olla liian pitkiä. (Nielsen 1995.)

Normanin suunnitteluperiaatteiden mukaan ohjaimilla ja niiden välisillä toiminnoilla pitää olla selvä yhteys, kytkentä. Esimerkiksi kun pystysuoran sivupalkin alanuolta napsauttaa hiirellä, niin näytöllä oleva sisältö liikkuu ylöspäin ja alhaalta tulee lisää sisältöä näkyviin. Samalla sivupalkin palkki liikkuu alaspäin. Luontevalla kytkennällä tarkoitetaan fyysisten vastaavuuksien ja kulttuurinormien hyväksikäyttöä kuten kohdistimen liikkumista ylöspäin, kun osoitinlaitetta liikutetaan ylöspäin. Ohjelmistoa on helppo käyttää silloin, kun mahdolliset toimenpiteet ovat näkyviä sekä säätimet ja näytöt perustuvat luonteviin kytkentöihin. (Norman 1990, 8–28.)

Normanin suunnitteluperiaatteisiin kuuluu myös käyttäjän toimintamahdollisuuksien rajoittaminen. Rajoituksilla suunnittelija rakentaa tuotteen käyttölogiikan vähentämällä käyttäjän toimintamahdollisuuksia. Looginen rajoitus tarkoittaa sitä, että käyttäjä päättelee, mitkä asiat ovat käytössä ja mitkä eivät. Esimerkiksi painikkeen himmeästä tekstistä käyttäjä tietää, että se ei ole käytettävissä. Kulttuurinen rajoitus on jotain, mitä on opittu tekemään tai olemaan tekemättä. Rajoituksia käytetään apuvälineinä uuden ohjelmiston käytössä tai järjestelmän navigoinnissa. Rajoitus voidaan käsittää myös käyttömahdollisuuden vastakohdaksi. (Norman 1990, 8–28.)

3.2.4 Tunteet ja käyttökokemus

Tunteet ovat osittain tajunnasta riippumaton ihmisen toiminnanohjausjärjestelmä. Tunteet kertovat meille, mitä asiat merkitsevät. Havaintoja ei epäillä samalla tavalla kuin tunteita. Tunteisiin perustuva päätöksenteko voi jättää asiat huomiotta ja johtaa virheratkaisuihin. Kaikilla toiminnoilla on sekä kognitiivinen että emotionaalinen merkitys. Negatiiviset tunteet ja uskomukset heikentävät käyttäjän kykyä sietää ohjelmiston ongelmia ja vastaavasti positiiviset tunteet ohjelmistoa kohtaan tarkoittavat viitseliäisyyttä ja sinnikkyyttä yrittää uudelleen. (Saariluoma 2004, 95–97; Sinkkonen ym. 2009a, 212–216.)

Tunne eli emootio on koettu elämys, tietoisuuden tila, joka syntyy sisäisen tai ulkoisen tapahtuman seurauksena. Ihminen muistaa, kuulee tai kokee jotain, jonka olettaa edistävän tai haittaavan omia pyrkimyksiä, ja seurauksena on tunnereaktio. Tunne syntyy automaattisesti, ja näin ihminen pystyy reagoimaan nopeasti tapahtumiin. Tunnereaktio syrjäyttää helposti muut asiat ja suuntaa voimavarat tavalla, joka voi olla joko hyödyksi tai haitaksi kyseessä olevassa tilanteessa. Tunteet syntyvät tyypillisesti käyttäjän tavoitteista ja tarpeista, suunnitelmista ja odotuksista. Asiat menevät joko odotusten mukaan tai odotusten vastaisesti. (Sinkkonen ym. 2009a, 216–218.)

Mieliala on pidempiaikainen kuin tunne. Tunteella on aina jokin kohde, mielialalla ei. Esimerkiksi voidaan olla surullisia jostakin (tunne) ja masentuneita (mieliala). Kun tunnetila kestää muutamia sekunteja, niin mieliala voi kestää tunteja ja jopa päiviä. Mieliala voi syntyä tunteesta. Mieliala saattaa vääristää arviointeja ohjelmistosta mielialan suuntaan. Ihmiset muistavat asioita ja kiinnittävät huomiota asioihin, jotka vastaavat heidän mielialaansa. Parhaiten muistetaan nykyhetken kanssa samanlaisessa mielialassa syntyneet muistot. Mieliala vaikuttaa käytökseen ja ajatteluun kuten tunnekin. Mieliala vaikuttaa käyttäjän ongelmanratkaisu-, ajattelu-, havaitsemis- ja muistamistapaan ja toiminnan tehokkuuteen. Lievästikin positiivinen mieliala parantaa käyttäjän joustavuutta, luovuutta, ajattelua ja ongelmanratkaisukykyä. Positiivisessa mielentilassa käyttäjät löytävät helpommin uusia ratkaisuja ja sietävät paremmin pieniä vastoinkäymisiä. (Sinkkonen ym. 2009a, 221–222.)

Nielsenin heuristiikoiden mukaan ohjelmiston ja oikean maailman välillä pitäisi olla yhteys. Näin ollen ohjelmiston pitäisi ”puhua” käyttäjän kieltä ja käyttää sanoja, termejä ja lauseita, jotka ovat käyttäjälle tuttuja mieluummin kuin teknisesti orientoitunutta

termistöä. Informaatio pitäisi näyttää luonnollisessa ja loogisessa järjestyksessä ja näin ollen noudattaa oikean maailman käytäntöjä. Nielsen painottaa myös virheiden ehkäisemisen tärkeyttä. Hyvää virheilmoitustakin tärkeämpää on estää virhettä tapahtumasta alkuunkaan, joko poistamalla virhealtis tila tai korjaamalla se, taikka pyytämällä käyttäjältä vahvistusta ennen toimenpiteen suorittamista. (Nielsen 1995.)

3.3 Lokalisointi osana käytettävyyttä

3.3.1 Ohjelmiston lokalisointi

Lokalisointia ja kieliversiointia tehdään ohjelmiston käytettävyyden parantamiseksi. Ohjelmiston lokalisoinnilla tarkoitetaan yleensä ohjelmiston kääntämistä toiselle kielelle. Ohjelmiston lokalisointi on kuitenkin myös paljon muutakin kuin vain käyttöliittymän kääntämistä toiselle kielelle, vaikka käännöstyö onkin eittämättä tärkein aspekti puhuttaessa ohjelmiston lokalisoinnista. Laajemmin ohjelmiston lokalisointi voidaan ymmärtää ohjelmiston muokkaamisena kohdemarkkina-alueen lingvististen, kulttuurillisten ja teknisten vaatimusten mukaisiksi. Ohjelmiston lokalisointi vaatii lähtökohtaisesti paljon resursseja ja aikaa, sillä sitä on vaikea automatisoida. (SDL 2017.)

Hyvin lokalisoitu ohjelmisto antaa vaikutuksen siitä, kuin se olisi suunniteltu kyseiselle markkina-alueelle. Lokalisoinnissa on syytä kiinnittää huomiota käännösten lisäksi ainakin seuraaviin seikkoihin: mittayksiköt; numeroiden, osoitteiden, ajan ja päivämäärän muotoilu; kirjasinlajit ja tekstin muotoilut; paikalliset säädökset ja lait; tekijänoikeudet; tietoturva; rahayksiköt sekä verot. Ohjelmiston lokalisointiprosessi sisältää yleensä seuraavat vaiheet:

- materiaalin, työkalujen ja resurssien analysointi
- kulttuurin, tekniikan ja kielen arviointi
- asiasanaston luonti ja ylläpito
- sisällön kääntäminen kohde kielelle
- käyttöliittymän muokkaus uusia käännöksiä vastaavaksi
- kuvien, symbolien ja muun median/grafiikan lokalisointi
- ohjelmiston lokalisoidun version testaus
- käännösten ja ohjelmiston toiminnallisuuden laadunvarmistus

- lokalisoidun ohjelmiston julkaiseminen.

Ohjelmistoa lokalisoitaessa huomioon otettavia asioita on runsaasti, mikä johtaa väistämättä siihen, että lokalisointi vaatii paljon resursseja ja aikaa. Lisäksi varsinkin eri kulttuurialueille siirryttäessä virheitä syntyy helposti. (SDL 2017; TechRepublic 2013.)

Ohjelmistot sisältävät käyttäjältä piilossa olevaa koodia ja käyttäjälle näkyvää tekstiä. Yleisesti tapana on kääntää ainoastaan käyttäjälle näkyvä sisältö eikä koodista mahdollisesti löytyviä kommentteja ja muita tekstisisältöjä, mitkä eivät suoraan näy käyttäjälle. Lokalisointi on yhdistelmä kääntämistä ja uudelleen kirjoittamista. Kääntäjän näkökulmasta normaaliin lokalisointiin verrattuna ohjelmiston lokalisointi vaatii kielen hallinnan ja kulttuurin tuntemuksen lisäksi ymmärrystä ohjelmistoista ja niiden rakenteesta. Tämä ei kuitenkaan tarkoita sitä, että kääntäjän täytyisi olla IT-alan ammattilainen, vaan pikemminkin sitä, että kääntäjä ymmärtää missä ympäristössä operoidaan ja mihin tarkoitukseen ja kohtaan käyttöliittymässä käännökset tulevat. Asian merkitys korostuu, jos kääntäjällä on pääsy itse ohjelmiston koodiin. Tällöin kääntäjältä vaaditaan ymmärrystä myös koodin rakenteesta ja siitä, mistä kohtaa koodia käännettävä sisältö löytyy. Normaaliin käännöstyöhön verrattuna ohjelmistot ovat yleensä jatkuvassa muutoksessa ja niitä jatkokehitetään jatkuvasti. Näin ollen siinä missä perinteisessä käännöstyössä käännetään valmiita tekstejä, saattavat ohjelmiston sisällöt muuttua kehityksen aikana, mikä on syytä ottaa huomioon ohjelmistoa lokalisoidessa. (Iina 2011; SDL 2017.)

3.3.2 Yleiset virheet

Lokalisointi on kohtuullisen helposti toteutettavissa, jos ohjelmisto on alusta alkaen suunniteltu ja toteutettu lokalisaation mahdollisuus mielessä pitäen. Aina näin ei kuitenkaan ole mikä vaikeuttaa lokalisointi prosessia, sillä tällöin myös itse ohjelmiston rakennetta on muutettava tukemaan lokalisointia. Tästä syystä alusta alkaen on syytä kiinnittää huomiota lokalisointiin ja noudattaa muutamaa yleistä ohjesääntöä ja näin ollen välttyä ylimääräiseltä työltä.

Tekstin upottaminen suoraan koodiin ei ole suositeltavaa, sillä tällöin tekstisisältöjen ylläpito on huomattavan vaikeaa ja käännettävien sisältöjen hakeminen työlästä. Tekstisisällöt pitäisikin löytyä erillisestä tiedostosta taikka tietokannasta, jolloin niiden ylläpito ja niiden hakeminen on huomattavasti helpompaa. Jokaisella termillä taikka

lauseella pitäisi olla oma uniikki ID, minkä perusteella ne voidaan hakea taulukosta ja upottaa koodiin. Kun tekstisisällöt löytyvät erillisestä tiedostosta, voidaan kyseinen tiedosto lähettää suoraan kääntäjälle ilman, että käännettäviä sisältöjä tarvitsee etsiä koodin seasta. (Grotendorst 2017.)

Käyttöliittämän elementtien pitäisi olla joustavia eikä pikseleihin sidottu, sillä tekstisisältöjen pituus vaihtelee kielen mukaan. Näin ollen pikseleihin sidottujen elementtien kokoa joudutaan muuttamaan kieliversiosta toiseen, mikä lisää työn määrää. Joskus kieli vaihtelee riippuen maasta, missä sitä puhutaan, esimerkiksi brittienglanti ja amerikanenglanti. Tällöin on syytä kiinnittää huomiota kielen lisäksi maahan ja lokalisaatiossa pyrkiä muokkaamaan kieltä paikallisen esitystavan mukaiseksi. Näin ollen on aina pyrittävä lokalisoimaan ohjelmisto kokonaan pelkän kääntämisen sijaan. Usein myös erilaisia viestejä automatisoidaan niin, että ne rakennetaan pienistä paloista kuhunkin tilanteeseen sopiviksi, esimerkiksi tervetuloviestit, vikailmoitukset, vastaukset jne. Tämä ei kuitenkaan ole suositeltavaa, sillä eri kielissä on erilaiset lauserakenteet ja sanajärjestykset, jolloin paloista rakennetut viestit eivät tulosta kieliopillisesti oikeanlaista tekstiä kaikilla kielillä. Ohjelmiston tulostamat viestit onkin syytä kirjoittaa aina kokonaisina lauseina eikä yksittäisinä sanoina, joista viesti sitten rakennetaan. (Grotendorst 2017.)

Myös erikoismerkkeihin on syytä kiinnittää huomiota, sillä eri kielissä käytetään erilaisia merkkejä ja länsimaiset kielet eroavat täysin esimerkiksi itämaisista kielistä merkistöltään. Jollei asiaan kiinnitä huomiota, ohjelmisto ei useinkaan pysty tulkitsemaan vieraita merkkejä ja näyttämään niitä oikein. Yleisesti käytössä on Unicoden merkistöstandardi, jonka avulla myös erikoismerkit pystytään näyttämään oikein. Numeroiden, yksiköiden, päivämäärien ja ajan esitystapa vaihtelee kulttuureittain, joten ei ole järkevää upottaa näitä sellaisenaan käyttöliittymään ilman mahdollisuutta muuttaa näiden esitysmuotoa. Ongelmaan on olemassa useita eri koodikirjastoja, jotka muuttavat esitystavan tarvittavaan muotoon. Näin ollen numerot, yksiköt, päivämäärät ja ajan voi tallentaa haluamassaan muodossa ja koodikirjasto määrittää esitystavan kussakin tilanteessa. (Grotendorst 2017.)

Länsimaissa on totuttu lukemaan vasemmalta oikealla, mutta esimerkiksi Lähi-idässä luetaan oikealta vasemmalle ja joissakin Aasian maissa teksti on pystysuorassa ja sitä luetaan ylhäältä alas. Tähän on syytä kiinnittää huomiota käyttöliittymän kannalta, jos tarkoitus on lokalisoida ohjelmistoa markkina-alueille, jossa tekstiä luetaan eri suuntaan kuin länsimaissa. Myös kontekstilla on väliä. Kääntäjän on vaikea kääntää

tekstisisältöjä parhaalla mahdollisella tavalla, jollei ole selvää mikä on konteksti ja mihin kohtaan käyttöliittymää käännös sijoittuu. Näin ollen pitäisikin pyrkiä tarjoamaan kääntäjälle mahdollisuuksien mukaan tarkatkin muistiinpanot kontekstista ja käyttöympäristöstä. Kuvankaappaukset käyttöliittymästä ja joissain tilanteissa myös koodin tarjoaminen kääntäjälle helpottaa kääntäjän työtä ja auttaa saavuttamaan parhaat mahdolliset käännökset. (Grotendorst 2017.)

Useissa käyttöliittymissä on kuvia, jotka sisältävät tekstiä. Näissä tilanteissa myös kuvien sisältämät tekstit pitäisi kääntää, mikä voi osoittautua varsin vaikeaksi ja aikaa vieväksi prosessiksi. Onkin syytä pitää kuvat ja teksti erillään ohjelman rakenteessa, jolloin kuvista löytyvä tekstisisältö on helposti käännettävissä. Kulttuurien väliset erot on hyvä pitää mielessä käyttöliittymän graafisia elementtejä tarkasteltaessa, sillä kaikki kuvat ja symbolit eivät välttämättä sovellu kaikille markkina-alueille. Kuvien ja symbolien merkitys voi vaihdella kulttuurista riippuen. (Grotendorst 2017.)

Pienet suunnitteluvirheet voivat vaikeuttaa ohjelmiston lokalisointia suunnattomasti, mikä puolestaan voi koitua yritykselle kalliiksi. Pidemmällä aikavälillä on mahdollista säästyä paljolta työltä ottamalla lokalisaation mahdollisuuden huomioon jo ohjelmistoa suunniteltaessa. Lokalisoitua ohjelmistoa on myös syytä testata jatkuvasti ohjelmiston suunnittelusta jatkokehitykseen. Huomiota kannattaa kiinnittää kirjoitusvirheisiin, kielioppiin, epäjohdonmukaisuuksiin ja lokalisoitavuuteen. Kun lokalisaation mahdollisuuden huomioi jo suunnittelussa, on mahdollista välttyä ylimääräiseltä työltä.

4 TIETOKANTOJEN SUUNNITTELU

Tietokanta on loogisesti yhteenkuuluvien, tallennettujen tietojen joukko, jota voidaan helposti käsitellä tietokantakielellä (Hovi ym. 2005, 4). Tietokannat ovat nykyään välttämättömiä liiketoiminnan kannalta. Lähes jokainen digitaalinen palvelu, jota käytämme, pyörii tietokannan päällä. Yritykset pitävät ja ylläpitävät tärkeitä tietojaan tietokannoissa. Tietokannoista tekee erityisen voimakkaita niiden käyttämiseen ja ylläpitoon kehitetyt tietokannan hallintajärjestelmät, joiden avulla on mahdollista hallinnoida suuria määriä dataa turvallisesti. (Garcia-Molina ym. 2009, 1.)

Tietokantojen suunnittelu on tärkeää, koska yhä suurempi osa tiedoista tallennetaan tietokantoihin. On myös tärkeää, että tarvittavat tiedot löytyvät tietokannoista ja että tiedot voidaan esittää tiedon hakijalle nopeasti ja varmasti. Lisäksi tietokannan rakennetta joudutaan usein muuttamaan. Tällöin hyvä suunnittelu säästää sekä aikaa että rahaa. (Hovi ym. 2005, 2.)

Tietokanta muodostaa nykyaikaisten sovellusten perustan. Mitä monimutkaisemmasta ja laajemmasta kokonaisuudesta on kyse, sitä tärkeämpää hyvä suunnittelu on. Tietokannan suunnittelu käsittää laajan kirjon asioita vaatimusmäärittelystä tietokannan mallinnukseen ja fyysiseen suunnitteluun. Hyvän tietokannan rakenteen keskeisiä ominaisuuksia ovat:

- kattavuus: sisältää kaikki järjestelmissä tai kyselyissä tarvittavat tiedot ja yhteydet
- selkeys ja ymmärrettävyys: yksinkertainen rakenne, helpot kyselyt
- muutosjoustavuus: laajennettavuus minimoiden nykyisten ohjelmien muutokset
- yleiskäyttöisyys: soveltuvuus erilaisiin ympäristöihin ja eri asiakkaille tarvitsematta muuttaa tietokannan rakennetta
- eheys: toisteisuuden välttäminen, oikeellisuus, sisäinen ristiriidattomuus
- ohjelmointimukavuus: selkeät tietorakenteet, sarakkeilla kiinteä merkitys
- suorituskyky eli tehokkuus: riittävä vastausaika tapahtumille ja riittävän tehokkaat eräajot.

Nykypäivänä levytilan säästö ei ole enää keskeinen tavoite, sillä levyjen hinnat ovat huomattavasti matalammat mitä vielä muutamia vuosia sitten. (Hovi ym. 2005, 20–23.)

Muita hyvän tietokannan ominaisuuksia saattavat tapauskohtaisesti olla:

- yhteensopivuus olemassa olevien tietojärjestelmien tai tietokannan hallintajärjestelmien kanssa
- siirrettävyys laiteympäristöstä tai tietokannan hallintajärjestelmästä toiseen
- turvallisuus: tietoihin päästään käsiksi vain myönnettyjen käyttöoikeuksien mukaan.

Hyvä tietokannanhallintajärjestelmä huolehtii automaattisesti joistakin hyvän tietokannan ominaisuuksista, kuten turvallisuudesta. (Hovi ym. 2005, 23.)

4.1 Taulujen muodostaminen

Jokaista asiaa tietokannassa edustaa taulu. Jokainen taulu koostuu kentistä, jotka edustavat elementtejä, jotka määrittelevät tai kuvailevat asiaa, jota taulu koskee. Taulut muodostavat tietokannan perustan. Esimerkiksi jokaiselle kielelle voidaan luoda oma sanastotaulu, jonka kenttiä termit ovat. (Hernandez 2000, 151–152.)

Tietokantoja suunnitellessa tapana on ensin muodostaa käsitemalli käsiteanalyysin avulla. Käsiteanalyysin tavoitteena on määrittää ja kuvata havainnollisella kaaviolla tietokantaan talletettavia tietoja. Käsiteanalyysissä kuvataan sitä reaalimaailmasta rajattua osaa kohdealuetta, jota on tarkoitus kuvata tietokannassa. Lopputuloksena syntyvä käsitemalli kuvaa kohdealuetta ja määrittelee pohjan tietokannan fyysiselle rakenteelle. Käsitemallia tarkennetaan asteittain, kunnes lopulta syntyy tarkat piirustukset, joiden avulla voidaan toteuttaa tietokanta tauluineen. (Hovi ym. 2005, 32–34.)

Tietokannan tauluja muodostettaessa aluksi määritellään alustava taululista, jonka avulla selvitetään ja luodaan tietokannan taulut. Aluksi määritellään alustava kenttälista. Tarkoituksena on selvittää, mihin asioihin listan kohdat viittaavat epäsuorasti. Alustava kenttälista pitää sisällään organisaation perustavat tietovaatimukset ja tietokantaan määriteltävien kenttien ydinjoukon. Alustavasta kenttälistasta etsitään yhteyksiä kenttien välillä ja tutkitaan, määritteleekö tai kuvaako jokin kenttäjoukko jonkin tietyn asian. Jos listan kentistä voi johtaa jonkin asian, lisätään tämä asia uuteen alustavaan taululistaan. (Hernandez 2000, 150–154.)

Alustavan taululistan määrittelyyn on syytä käyttää aikaa ja tarkastella taululistaa monelta eri kantilta. Kun alustavasta taululistasta ollaan saatu niin täydellinen kuin se

voi olla, niin se muutetaan lopulliseksi taululistaksi lisäämällä siihen kaksi elementtiä: taulun tyyppi ja taulun kuvaus. Taulut luokitellaan tyypin mukaan. Taulutyyppejä on neljä:

- data: taulu säilyttää dataa, jonka avulla tuotetaan informaatiota, ja se edustaa asiaa, joka on organisaatiolle tärkeä.
- linkitys: taulun avulla luodaan monesta moneen –yhteyksissä linkki kahden taulun välille.
- osajoukko: taulu sisältää täydentäviä kenttiä, jotka liittyvät johonkin tiettyyn datatauluun ja jotka kuvaavat kyseisen taulun asiaa hyvin tarkasti.
- vahvistus: taulun avulla toteutetaan datan eheys.

Taulun kuvaus määrittelee selvästi taulun edustaman asian ja kertoo, miksi asia on yritykselle tärkeä. Taulujen kuvaukset ovat tärkeitä, jos kaikkien halutaan ymmärtävän, miksi kukin taulu on olemassa ja miksi organisaatio on kiinnostunut tietojen keräämisestä kuhunkin tauluun. Kuvauksen pitää määritellä taulu selvästi ja ilmoittaa, kuinka tärkeä se organisaatiolle on. Kuvauksen on oltava tarkka, tiivis, yksiselitteinen ja selkeä. (Hernandez 2000, 154–167.)

Kun lopullinen taululista ollaan muodostettu, jokaiselle taululle valitaan kentät. Käytettävät kentät otetaan alustavasta kenttälistasta. Ne kentät, mitkä kuvaavat parhaiten taulun asian ominaisuuksia, valitaan kyseiseen tauluun. Jos kenttä tai kenttien joukko kuvaa useamman kuin yhden taulun ominaisuuksia, ne sijoitetaan sen mukaisesti. (Hernandez 2000, 174–177.)

4.2 Indeksointi ja optimointi

Taulujen lisäksi tietokantaan voidaan perustaa indeksejä. Indeksien päätarkoitus on nopeuttaa hakuja tietokannan tauluista. Indeksi on kuin kirjan hakemisto. Koska hakemisto on järjestyksestä, sieltä voi nopeasti hakea halutun asian ja hypätä sitten oikealle sivulle. Ei tarvitse lukea kirjan joka sivua läpi eikä vastaavasti tarvitse lukea koko taulua läpi. Taulujen koon kasvaessa, kuten esimerkiksi laajojen sanastojen tapauksessa, ei ole järkevää lukea koko taulua läpi löytääkseen vain muutaman haetun termin. Indeksien avulla voidaan määrittää esimerkiksi, että termejä haetaan ainoastaan niistä taulun riveistä, joissa kielikoodina on suomi. Näin ollen taulun muut rivit jätetään tarkastelun ulkopuolelle ja tietokantahaku keskitetään ainoastaan niihin

riveihin, jotka täyttävät määritellyn indeksin ehdot. Näin ollen tietokantahaku nopeutuu, kun läpikäytävien rivien määrää saadaan pienennettyä ennen haun suorittamista. (Hovi ym. 2005, 12; Garcia-Molina ym. 2009, 350–358.)

Kaikkiin kunnon relaatiotietokantatuotteisiin kuuluu optimointiohjelma, joka ottaa vastaan SQL-kyselyn ja pyrkii löytämään sille nopeimman saantipuun. Optimointiohjelma päättää esimerkiksi, onko saantipolkuna taulun läpiluku vai indeksin kautta haku. SQL-kielen yksinkertaisuus perustuu paljolti siihen, että tietokantakutsussa ei tarvitse määritellä saantipolkua, esimerkiksi sitä, minkä indeksin kautta taulua luetaan ja miten indeksiä hyödynnetään. Optimointiohjelma näkee joukon mahdollisia saantipolkuja tietylle SQL-lauseelle. Se ennustaa näiden vaihtoehtojen keston ja valitsee sen vaihtoehdon, jonka ennustettu kesto on lyhyin. (Hovi ym. 2005, 12, 149–152.)

Taululle voidaan perustaa indeksejä heti kun se on luotu tai milloin tahansa myöhemmin. Ison taulun kohdalla indeksin perustaminen vie jonkin verran aikaa, sillä tietokantajärjestelmän on luettava taulu läpi, lajiteltava ja sitten perustettava indeksirakenne. SQL-käskyissä ei normaalisti viitata indekseihin. Optimoija tekee päätöksen siitä, tapahtuuko luku indeksin kautta vai läpilukuna. Jo käytössä olevan tietokannan suorituskykyä voidaan parantaa lisäämällä sopivia indeksejä ilman, että tarvitsee muuttaa ohjelmia. (Hovi ym. 2005, 158.)

Vakiintuneiden nykytuotteiden optimointiohjelmat ovat kustannuspohjaisia. Ne näkevät joukon vaihtoehtoja ja laskevat niille kustannusarvion, joka on yleensä CPU-ajan ja levyajan painotettu summa. Kustannusarvio perustuu muun muassa tilastotietoihin taulujen ja indeksien koosta sekä sarakkeiden arvojakaumista. Parhaiden optimointiohjelmien ennustekaavat perustuvat kymmeniin eri muuttujiin. Tästä huolimatta optimointiohjelmat eivät ole täydellisiä. Optimointiohjelmat eivät aina näe parasta vaihtoehtoa ja joskus kustannusarviot ovat liian kaukana totuudesta. Tällöin optimointiohjelmaa voidaan auttaa muotoilemalla SQL-lause optimointiystävällisemmin taikka parantamalla indeksointia. Mitä parempi indeksi, sitä todennäköisemmin optimointiohjelma sen valitsee, vaikka kustannusarviot olisivat kaukana totuudesta. Optimointiohjelma saadaan näkemään ja valitsemaan paras saantipolku kullekin SQL-lauseelle. Taulujen suunnittelussa tehdyt virheet voivat puolestaan estää tehokkaan indeksoinnin. (Hovi ym. 2005, 296–306.)

5 PROJEKTIN TOTEUTUS

Ohjelmistojen kieliversioinnissa ei noudateta mitään tiettyä yleistä toimintatapaa, vaikka monet toteutuksista ovatkin osittain yhteneviä. Näin ollen kieliversiointi rakennetaan aina tapauskohtaisesti ohjelmiston erityispiirteet huomioon ottaen. Web-pohjaisissa ohjelmistoissa ja verkkosivuissa tilanne on hieman helpompi, sillä sisällönhallintajärjestelmät, kuten WordPress ja Drupal, tarjoavat mahdollisuutta kieliversiointiin, mutta koska KuljetusVelhoa ei ole rakennettu minkään järjestelmän päälle vaan toteutettu alusta alkaen itse, oli myös kieliversioinnin suhteen rakennettava parhaiten soveltuva ratkaisu itse hyödyntämättä valmiita sisällönhallintajärjestelmiä.

Opinnäytetyön tarkoituksena oli rakentaa helppokäyttöinen, yksinkertainen ja helposti laajennettava ratkaisu KuljetusVelhon kieliversioinnin toteuttamiseen. Tavoitteena oli muuttaa ohjelmistoa siten, että tulevaisuudessa uusien kielien lisääminen olisi vaivatonta. Tämän toteuttamiseksi annettiin vapaat kädet, mutta alusta alkaen oli selvää, että tarvittaisiin käännöstekstit sisältävä tietokantaratkaisu ja selainpuolen muutokset niin, että selainpuoli tukee sisältöjen hakua tietokannasta. Lisäksi tarvittaisiin keino lisätä uusia kieliä tietokantaan. Kaikkia suunnitelmia ei pystytty lyhyessä ajassa toteuttamaan, mutta työ jatkuu tämän opinnäytetyön jälkeenkin.

5.1 Käyttöliittymä

Käyttöliittymä on ohjelmiston käyttäjän kannalta keskeisin tekijä. Käyttäjä ei ole kiinnostunut siitä, mitä taustalla tapahtuu ja miten joku ominaisuus on toteutettu, vaan siitä, miten ominaisuudet ja tieto käyttäjälle esitetään. Käytettävyyden tutkija Jacob Nielsen on erikoistunut erityisesti www-käyttöliittymiin ja listaakin käyttäjän oman äidinkielen olevan yksi tärkeimmistä käytettävyyteen vaikuttavista tekijöistä. Käyttöliittymässä käytetty kieli tulisi olla käyttäjän oma äidinkieli ja sovellusalueen ammattitermistöä. Käytettävä sanasto tulisi tarkistaa käyttäjien kanssa, jotta sovelluksen ja käyttäjän käsitteet vastaavat toisiaan. Vertauskuvat eli metaforat auttavat käyttäjää hyödyntämään olemassa olevaa tietoaan. Metaforien käyttö on tehokas tapa siirtää merkitsevää informaatiota lähettäjältä vastaanottajalle. (Nielsen 1995.)

Näin ollen on pyrittävä käyttämään logistiikka-alalle vakiintuneita käsitteitä ja termistöä ymmärrettävyyden parantamiseksi. Logistiikka-alalle vakiintunutta termistöä löytyy useista eri lähteistä internetistä ja logistiikan sanasto -kirjoista, mutta mitään varsinaista standardia terminologiaan ei ole olemassa eikä mikään taho ylläpidä mitään virallista termien tietopankkia. Eri lähteiden tarjoamat sanastot ovat pääosin yhteneviä, mutta pieniä eroavaisuuksiakin löytyy. Hyvinä termipankkeina toimii Pekka Rytilän kirjoittama Logistiikan sanasto –kirja sekä Logistiikanmaailma.fi-verkkosivuston sanasto.

Tarkoituksena oli tehdä universaali ratkaisu kieliversiointiin, jolloin pitää olla mahdollista lisätä järjestelmään mikä tahansa kieli ongelmitta. Eri kielissä on toisistaan eroavia merkkejä, kuten suomen kielessä ääkköset ja saksankielessä ß-merkki, jolloin on syytä varmistua siitä, että kaikki tekstisisältö näkyy oikein, myös erikoismerkit. Tähän tarkoitukseen on kehitetty Unicode-standardi, joka kuvaa yksilöivällä koodiarvolla jokaista merkkiä, jolloin tietokoneen on helppo tulkita ja näyttää myös erikoismerkit ongelmitta. Unicoden avulla on mahdollista näyttää yli satatuhatta eri kirjoitusmerkkiä, joka vastaa yleismaailmallisen merkistön määrittävää kansainvälistä standardia ISO/IEC 10646. UTF-8 on yleisin Unicoden koodaustapa, ja se soveltuu parhaiten käytettäväksi sen vanhempien järjestelmien yhteensopivuuden vuoksi vaikkakin UTF-8 vie hieman enemmän tilaa kuin muut koodaustavat. (Arno 2017.)

5.1.1 Kielivaihtoehdot

Kielivaihtoehtojen näyttäminen on käyttöliittymän suunnittelun kannalta keskeistä. Mahdollisuus vaihtaa ohjelmiston kieltä pitää olla näkyvällä paikalla, usein jommassakummassa yläreunassa. Pääsääntö on, että kielivaihtoehtojen pitää olla helposti saatavissa. Kielivaihtoehdot esitetään perinteisesti omankielisinä esim. suomi, svenska, english jne., jolloin oman kielen valinta helpottuu verrattuna siihen, että kaikki kielivaihtoehdot esitettäisiin jollakin vieraalla kielellä. (Vertommen 2015.)

Erittäin yleinen tapa kielivaihtoehtojen esittämiseen on lippujen käyttö. Kyseinen lähestymistapa ei ole suositeltavaa lukuisista eri syistä. Ensinnäkin liput edustavat maata eivätkä kieltä. Useissa maissa puhutaan useita eri kieliä virallisina kielinä esim. Belgiassa puhutaan hollantia, ranskaa ja saksaa. Lisäksi esim. portugalia ja espanjaa puhutaan virallisena kielenä useissa eri maissa, ei ainoastaan Portugalissa ja Espanjassa. Toisekseen liput ovat ongelmallinen lähestymistapa myös esteettömyyden näkökulmasta. Värisokean voi olla vaikea erottaa Ranskan ja Italian lippuja toisistaan ja

joissain tapauksissa Hollannin lippu voi muistuttaa huomattavasti jonkin muun maan lippua. (Bittersmann 2011.)

Näiden syiden vuoksi kielivaihtoehdot tulisi näyttää tekstimuodossa omankielisinä. Pudotusvalikot ovat ongelmallisia siitä syystä, että tällöin kielivaihtoehdot ovat piilossa käyttäjältä. Vaikka kielivaihtoehtojen esittämiselle ei ole olemassa mitään yleismaailmallista standardia, Kuvan 4 lähestymistapa on kaikki seikat huomioon ottaen toimiva.

Kuva 4. Kielivaihtoehtojen esittäminen (Bittersmann 2011).

On mahdollista automaattisesti tunnistaa käyttäjän kieli selaimen avulla, kun käytössä on www-pohjainen ohjelma. Käyttäjän suosimaa kieltä voi yrittää selvittää Accept-Language HTTP headerin avulla taikka JavaScriptin window.navigator.language-ominaisuuden avulla. Molemmat perustuvat siihen, että käyttäjä on asettanut käyttöjärjestelmälleen ensisijaisen kielen. Mahdollista on myös käyttäjän paikantaminen käyttämällä HTML5:n Geolocation ominaisuutta, mutta tällöin käyttäjän on sallittava paikannus erikseen. Vaihtoehtona on myös käyttäjän IP-osoitteen paikannus palvelimen puolella. Tällöin käyttäjältä ei tarvita erikseen lupaa sijainnin selvittämiseen, joskin IP-osoitteen paikannus ei aina anna kovinkaan tarkkaa tietoa käyttäjän sijainnista. Vaikka käyttäjän sijaintia ja suosimaan kieltä yritettäisiin määritellä automaattisesti ei kielivaihtoehtoja kuitenkaan pidä piilottaa vaan antaa käyttäjälle mahdollisuus valita haluamansa kieli myös manuaalisesti. (Carnes 2013.)

5.1.2 Navigointi

Jotkin verkkosivut ja ohjelmat ohjaavat käyttäjän takaisin päävalikkoon kieltä vaihdettaessa, jolloin käyttäjän on uudestaan navigoitava takaisin haluamaansa kohtaan. Käyttömukavuuden kannalta onkin tärkeää, että kieltä vaihtaessa näkyvissä on sama sivu missä käyttäjä oli jo aikaisemminkin. (Vertommen 2015.)

Jotkin ohjelmistot pyytävät valitsemaan kielen ensimmäisellä käyttökerralla, jotta vältytään näyttämästä vieraskielistä käyttöliittymää. Tämä on käytössä lähinnä

laajemmissa ja suurimmissa ohjelmistoissa, eikä ole opinnäytetyön kohteena olevan KuljetusVelho-ohjelmiston kannalta ajankohtainen. (Arno 2017.)

5.2 Tietokanta

5.2.1 Tietokannan rakenne

Puhuttaessa kieliversioinnin tietokantaratkaisusta on vaihtoehtoisia toteutustapoja useita, vaikka ne muistuttavatkin paljon toisiaan pienillä eroilla. KuljetusVelhon tietokanta pyörii Microsoftin SQL-serverillä ja on relaatiotietokanta. Kolmen yleisesti käytössä olevan ratkaisun pohjalta rakennettiin oma versio tietokannasta noudattaen helppokäyttöisyyden, yksinkertaisuuden ja laajennettavuuden tavoitteita.

Taulukon 1 ratkaisu on yksinkertaisin. Siinä luodaan aina uusi sarake uutta kieltä kohti samaan tauluun. Yhdellä rivillä on aina yksi termi käännettynä eri kielille.

Taulukko 1. Yksinkertaisin tietokantaratkaisu kieliversiointiin.

ID	Title_fi	Title_en	Title_se
1	otsikko	header	rubrik
2			
3			

Tässä ratkaisussa hyvinä puolina ovat yksinkertaisuus, helpot kyselyt ja se, että kaikki käännökset saadaan samalle riville. Huonoina puolina taas voidaan pitää vaikeutta ylläpitää useita kieliä sekä uusien kielien lisäämisen monimutkaisuutta. Lisäksi tietokantaan jää tyhjiä kenttiä, jos jokaista termiä ei käännetä kaikille kielille. (Naypoka 2017.)

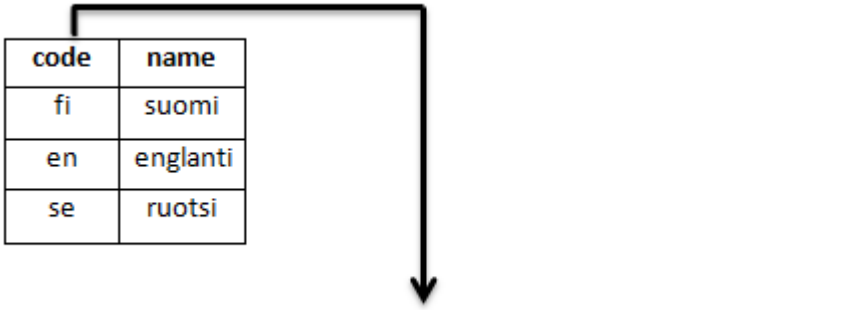
Taulukon 2 ratkaisu on muuten vastaava taulukon 1 ratkaisun kanssa, mutta taulukon 2 ratkaisussa jokainen käännös on omalla rivillään.

Taulukko 2. Tietokantaratkaisu, jossa jokainen käännös on omalla rivillään.

ID	Language_id	Title
1	fi	otsikko
2	en	header
3	se	rubrik
4		
5		
6		

Ratkaisun hyvinä puolina on yksinkertaisuus ja helpot kyselyt. Huonoina puolina ylläpitämisen vaikeus ja uusien kielien lisäämisen monimutkaisuus. (Naypoka 2017.)

Kuvan 5 vaihtoehto eroaa aikaisemmista siinä, että kielet sijaitsevat eri taulussa ja erillisessä Title_ID-sarakkeessa jokaiselle termille luodaan oma tunnus. Näin ollen saman termin kaikilla käännöksillä on sama tunnus, toisin kuin aikaisemmissa ratkaisuissa. Näin ollen on helpompi käsitellä termejä selainpuolella, kun termin eri käännöksillä ei ole eri tunnusta.



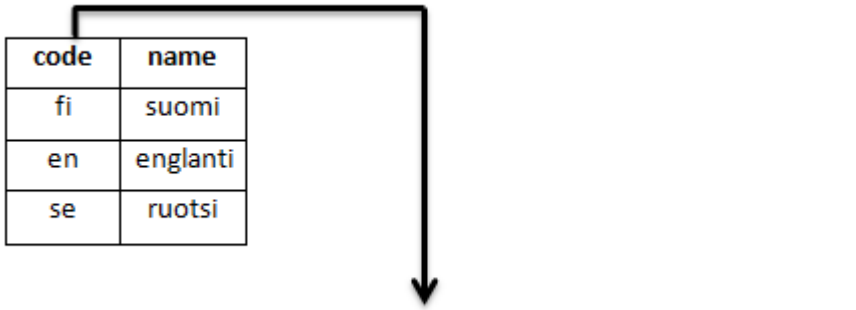
code	name
fi	suomi
en	englanti
se	ruotsi

ID	Title_ID	Language_code	Text
1	1	fi	otsikko
2	1	en	heading
3	1	se	rubrik
4	2	fi	
5	2	en	
6	2	se	

Kuva 5. Tietokantaratkaisu, jossa jokaiselle termille luodaan oma tunnus, joka pitää sisällään termin käännökset.

Tietokantarakenteen hyvinä puolina on selkeä lähestymistapa, kielten lisäämisen helppous ja helpot kyselyt. Huonoina puolina ymmärrettävyys selainpuolella. (Naypoka 2017.)

Kaikki edellä kuvatut tietokantaratkaisut ovat toimivia ja käytössä. Päädyttiin kuvan 5 mukaiseen tietokantaan pienillä muutoksilla (Kuva 6). Title_ID-sarakkeen numerot korvataan termin suomenkielisellä nimellä. Näin ollen selainpuolella on helpompaa käsitellä käännöksiä ja sijoittaa ne oikeisiin kohtiin.



code	name
fi	suomi
en	englanti
se	ruotsi

ID	Title_ID	Language_code	Text
1	otsikko	fi	otsikko
2	otsikko	en	heading
3	otsikko	se	rubrik
4	osoite	fi	osoite
5	osoite	en	address
6	osoite	se	adress

Kuva 6. Tietokantaratkaisu, jossa termin tunnus on sen suomenkielinen nimi.

5.2.2 Käännösten lisääminen tietokantaan

Oleellinen osa toimivaa kieliversiointia on mahdollisuus lisätä käännöksiä ja uusia kieliä helposti järjestelmään. Ei ole kenenkään edun mukaista käyttää paljon arvokkaita työtunteja käännöksiin ja niiden tietokantaan lisäämiseen. Tästä syystä hyvissä kieliversioinneissa pyritään mahdollisimman paljon automatisoimaan toimintoja ajan säästämiseksi.

Ohjelmiston tekstisisältöjä käännettäessä on tapana käyttää ammattikääntäjää käännösten laadun varmistamiseksi. Tekstisisällön voi kääntää myös itse taikka käyttää konekäännöstä. Ei ole kuitenkaan mahdollista täysin automatisoida käännöstyötä, vaikka käytettäisiin konekääntäjää. Parhaatkin konekääntäjät tekevät virheitä, jolloin ihmisen on kuitenkin manuaalisesti käytävä käännökset läpi ja korjattava virheet. Toki jos käytettävissä ei ole ammattikääntäjää, konekäännös nopeuttaa käännösprosessia, mutta se ei kuitenkaan poista manuaalista työtä kokonaan yhtälöstä. Google kääntäjää voi helposti hyödyntää käännöstyössä käyttämällä Google Driven taulukkotyökalua. Käännettävät termit ovat yhdessä

sarakkeessa allekkain ja viereiseen sarakkeeseen käännetään Google kääntäjän avulla termit halutulle kielelle seuraavaa komentoa hyväksi käyttäen:

=GoogleTranslate(A1, "fi", "en")

Komento kääntää solun A1 sisällön suomesta englanniksi komennon sisältävään soluun. Kun komennon laajentaa koskemaan koko saraketta periaatteella A-sarakkeen solu käännetään viereiseen sarakkeeseen, on mahdollista kääntää tuhansia termejä käsittäviä sanastoja helposti ja nopeasti toiselle kielelle. On kuitenkin muistettava, että konekäännöksen oikeellisuuteen ei voi luottaa, vaan aina on syytä tarkistaa käännökset manuaalisesti.

Käännöksiä lisättäessä tietokantaan kaikki tekstisisältö tallennetaan samaan tiedostoon. Tiedosto voi olla joko tekstitiedosto taikka Excel-taulukko, mutta Excel-taulukkoa on helpompi muokata haluttuun muotoon. Komentosarja käy tiedoston läpi rivi riviltä lisäten käännökset tietokantaan periaatteella yksi käännös per rivi (Kuva 7).

Käännös 1
Käännös 2
Käännös 3

ID	Title_ID	Language_code	Text
163	5	en	Käännös 1
164	6	en	Käännös 2
165	7	en	Käännös 3

Kuva 7. Käännösten lisääminen tietokantaan.

Käännökset voivat olla tietyssä järjestyksessä, jolloin ne menevät automaattisesti oikeille paikoilleen tietokannassa. Jos käännökset ovat sekalaisessa järjestyksessä, viereiseen soluun on lisättävä käännöksen tunnus esim. otsikko (Kuva 6), jolloin ne sijoittuvat oikeille paikoilleen tietokannassa.

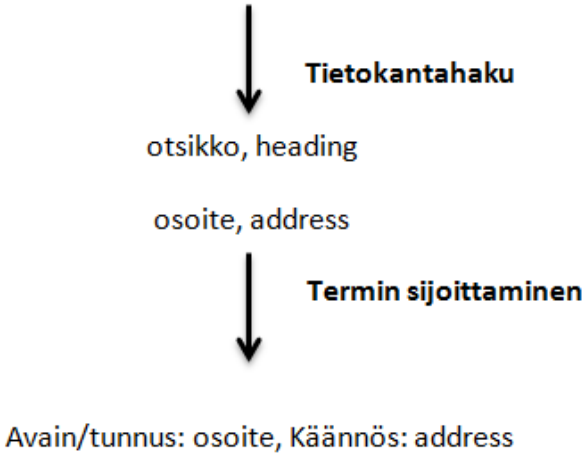
Microsoftin SQL Server Management Studio tarjoaa mahdollisuuden lisätä tietoja tietokantaan suoraan Excel-taulukosta. Tällöin Excel-tiedostosta on löydyttävä kaikki sarakkeet, ei pelkkiä käännöksiä, jotta ohjelma pystyy lisäämään tiedot haluttuun tauluun tietokannassa. Tämä on ehdottomasti tarjolla olevista vaihtoehdoista helpoin ja kätevin, sillä tietokantaa on helppo muokata ja ylläpitää Excel-tiedostossa ja lopuksi lisätä muutokset tietokantaan.

5.3 Kieliversioinnin toteutus

Koska tekstisisällöt KuljetusVelhossa oli upotettu staattisina HTML:n sekaan, vaadittiin selainpuolen muuttamista siten, että tekstisisällöt pystytään hakemaan tietokannasta ja sijoittelemaan oikeille paikoille käyttöliittymässä. Tietokannan kuormituksen kannalta ei ole järkevää hakea sanastoa useita kertoja uudelleen. Tietokannan kuormituksen minimoimiseksi hyödynnetään web-palvelimen levymuistia, jonne sanasto tallennetaan. Kun käyttäjä avaa sivun taikka lomakkeen, sanasto haetaan puskurimuistina toimivasta tiedostosta eikä tietokannasta. Tämä ei pelkästään vähennä tietokannan kuormitusta vaan myös nopeuttaa ohjelman toimintaa.

Palvelimelta löytyvät sanastot päivitetään tarpeen tullen komentosarjalla. Kun komentosarjan käynnistää, suoritetaan tietokantahaku, jossa haetaan sanastot valituille kielille. Tietokantahaku palauttaa termit, jotka tallennetaan taulukkoon tunnus käännös –pareina. Taulukko talletetaan palvelimelle myöhempää käyttöä varten. Taulukosta käännökset sijoitetaan oikeille kohdille käyttöliittymässä käyttämällä tunnusta avaimena. Näin ollen kun on tiedossa kohta, johon käännös sijoitetaan, haetaan taulukosta oikeaa käännöstä käyttäen tunnusta avaimena, jonka parina oikea käännös on (Kuva 8).

ID	Title_ID	Language_code	Text
1	otsikko	fi	otsikko
2	otsikko	en	heading
3	otsikko	se	rubrik
4	osoite	fi	osoite
5	osoite	en	address
6	osoite	se	adress



Kuva 8. Käännösten hakeminen tietokannasta.

Sanasto sisällytetään jokaiselle sivulle ja lomakkeelle include_once-komennolla. Kun sivu tai lomake avataan, käännökset haetaan globaalista taulukosta, johon sanasto tallennettiin, kun käännökset haettiin palvelimelta. Näin ollen jokaiselta sivulta taikka lomakkeelta viitataan globaaliin sanastotaulukkoon, joka löytyy palvelimelta.

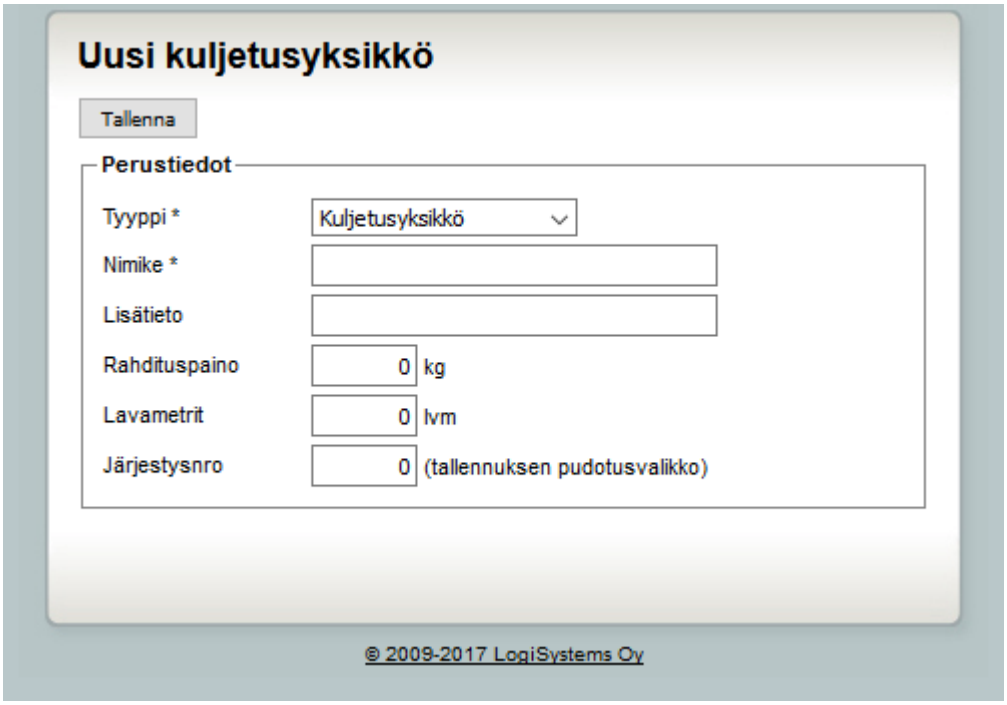
5.3.1 Front end eli selainpuoli

Kun käyttäjä kirjautuu sisään KuljetusVelhoon, hän valitsee itselleen ohjelmiston kielen pudotusvalikosta (Kuvat 9 ja 10). Näin ollen käyttäjät pääsevät välittömästi käyttämään ohjelmistoa haluamallaan kielellä. Verkkoselain muistaa kielivalinnan jatkossa, joten kielen valinnan joutuu suorittamaan ainoastaan kerran, jollei halua käyttää ohjelmistoa jollakin muulla kielellä, jolloin kielen vaihtaminen on helppoa sisäänkirjautumisen yhteydessä.

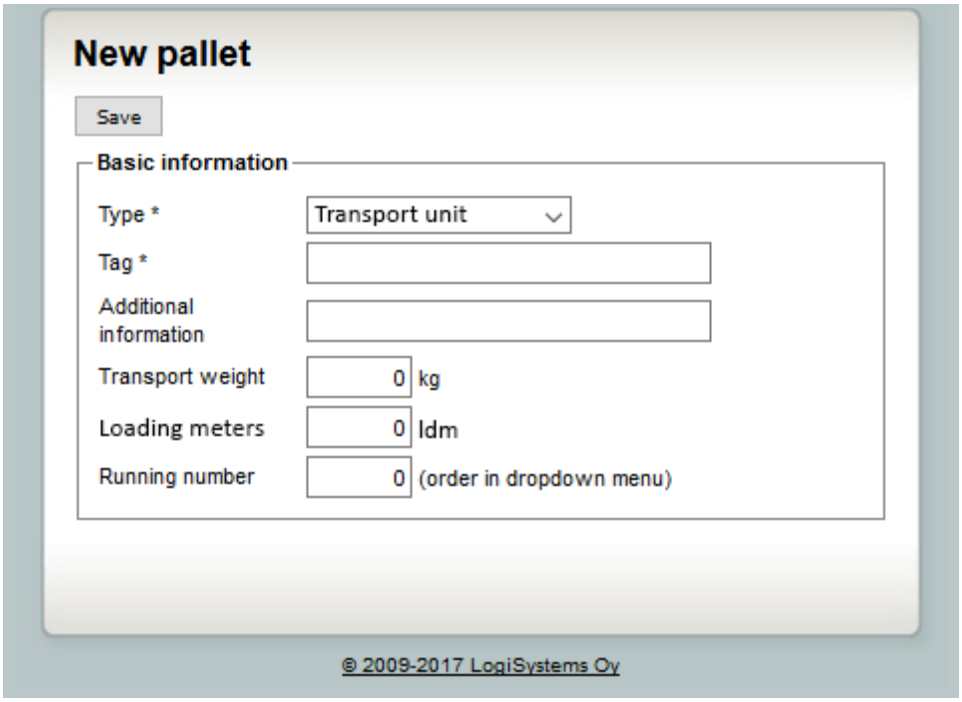
Kuva 9. Kirjautuminen KuljetusVelhoon.

Kuva 10. Kirjautumissivu englanniksi.

Näin käyttäjät voivat jo sisäänkirjautuessaan valita haluamansa kielen ja käyttää ohjelmistoa jatkossa sillä. Käyttöliittymä näkyy käyttäjälle halutulla kielellä (Kuvat 11 ja 12). Ohjelmiston kaikki tekstisisältö on käännetty valitulle kielelle. Näin ohjelmiston käyttö on käyttäjälle mahdollisimman luontevaa ja helppoa.



Kuva 11. Suomenkielinen lomake.



Kuva 12. Englanninkielinen lomake.

Käyttäjälle kielen valinta ja ohjelmiston käyttö on yksinkertaista ja tehokasta. Käyttäjän ei tarvitse miettiä termien ja lauseiden merkityksiä, koska ne tarjotaan käyttäjälle omalla äidinkielellä. Näin ohjelmiston käytön tehokkuus ja sujuvuus kasvavat, mikä tehostaa prosessien hallintaa yrityksessä.

5.3.2 Koodi

KuljetusVelhon kirjautumissivulla olevasta pudotusvalikosta valitaan haluttu kieli.

```
<form method="post" action="">
    <select name="kielet" onchange="window.location='login.php?language='+this.value">
        <option value="fi"<?php echo (isset($_COOKIE['kieliValinta'])
                        && $_COOKIE['kieliValinta'] === "fi" ? " selected" : ""); ?>
                        >Suomi</option>
        <option value="en"<?php echo (isset($_COOKIE['kieliValinta'])
                        && $_COOKIE['kieliValinta'] === "en" ? " selected" : ""); ?>
                        >English</option>
    </select>
</form>
```

Valittaessa kieltä kirjautumissivu ladataan uudelleen, jolloin kielivalinta talletetaan keksiin.

```
$kieli = $_GET['language'];
if (isset($kieli))
{
    setcookie('kieliValinta', $kieli, time() +60*60*24*30);
    $kieli = null;
    header("location: login.php");
}
```

Kielivalinta siirretään sitten PHP-tiedostoon, jossa valitun kielen sanasto haetaan palvelimelta ja talletetaan muuttujaan. Tähän tiedostoon viitataan joka sivulta include_once-komennolla.

```
$_SESSION['kieli'] = $_COOKIE['kieliValinta'];

$path = "\sanasto$_SESSION[kieli].txt";

if (file_exists($path))
{
    $sanasto = include($path);
}
```

Kielivalinta tallennetaan keksiin myös sissäänkirjauduttaessa.

```
if (isset($_POST['login']))
{
    setcookie('kieliValinta', $_POST['kielet'], time() +60*60*24*30);
}
```

Käyttöliittymässä käännökset haetaan taulusta etsimällä avaimen paria.

```
$_SESSION['sanasto']['uusi kuljetusyksikkö']
<?php echo $_SESSION['sanasto']['perustiedot'] ?>
```

Sanastoa päivitettäessä otetaan yhteys tietokantaan ja haetaan sanastot tietokannasta.

```
$link = MSSQL_CONNECT($hostname,$username,$password);
mssql_select_db($dbName);

$query = "SELECT title_id, text FROM $taulu
          WHERE language_code='$kieli'";
$result = mssql_query( $query );
```

Tämän jälkeen saatu sanasto talletetaan taulukkoon avain käännös –pareina.

```
$kaannosTaulu[] = $row;

while ($row = mssql_fetch_assoc($result))
{
    $kaannosTaulu[$row['title_id']] = $row['text'];
}
```

Seuraavaksi sanastotaulukko talletetaan palvelimelle.

```
$myfile =  fopen($path, "w");
file_put_contents($path, '<?php return ' .
                  var_export($kaannosTaulu, true) . ';');
fclose($myfile);
```

Palvelimella sanasto on tallennettuna tekstitiedostoon seuraavassa muodossa:

<?php return array (0 => false, 'otsikko' => 'heading', 'osoite' => 'address',…,);

5.4 Vaihtoehtoiset toteutustavat

Käytössä on myös lukuisia muita toteutustapoja kieliversioinnin toteuttamiseen. Näitä vaihtoehtoisia toteutustapoja tutkittiin tätä työtä varten, mutta edellä esitettyyn nähden ne olivat soveltumattomia KuljetusVelhon tarpeisiin.

Erityisesti useat pienet verkkosivut toteuttavat kieliversioinnin luomalla erilliset HTML-tiedostot eri kielille. Tämä on yksinkertainen ja toimiva ratkaisu, jos sisältöä ei ole paljoa. On kuitenkin huomioitava, että varsinkin jos sisältöjä on paljon ja kielivaihtoehtoja useita, eri HTML-tiedostojen ylläpidosta tulee tarpeettoman monimutkaista. Erillisten HTML-tiedostojen ylläpito on aikaa vievää, ja pienetkin muutokset tekstisisällöissä vaativat suuren työn.

Vaihtoehtona olisi ollut myös käännösten sijoittaminen erillisiin CSS-tiedostoihin. Kieltä vaihdettaessa ladataan aina eri tyylitiedosto. Kyseinen ratkaisu sopii lähinnä pienemmille ja yksinkertaisimmille sivustoille, kuten edellinenkin ratkaisu. Niin kuin käännösten sijoittamisessa erillisiin HTML-tiedostoihin, myös käännösten sijoittamisessa CSS-tiedostoihin on samat ongelmat. Sanaston kasvaessa ja kielten lisääntyessä, ylläpito käy työlääksi ja uusien sisältöjen lisääminen vaatii paljon ylimääräistä työtä.

On mahdollista automatisoida kieliversiointi täysin koneen käännettäväksi. Google tarjoaa vastaavaa sivujen automaattista käännöstä. Valitettavasti konekäännöksessä luotettavuus nousee esteeksi. Konekäännöksen jälki ei täytä laatuvaatimuksia millaisia odottaisi kaupalliselta ohjelmistolta. Luotettavin konekääntäjä on IBM:n Watson-tekoäly, mutta vielä vuonna 2017 ei olla tilanteessa, jossa käännöstyötä voisi ulkoistaa tekoälylle. (IBM 2017.)

6 JOHTOPÄÄTÖKSET

Projektin aikana saatiin toteutettua KuljetusVelhon kieliversiointi. Kieliversiointi ja lokalisaatio oli lähtökohdat huomioon ottaen iso projekti, joka jatkuu vielä tämän opinnäytetyön jälkeenkin. Opinnäytetyön avulla toteutettiin selvitystyö käytettävyyden, lokalisoinnin ja kulttuurivaikutuksen osalta, johon rakennettiin kieliversioinnin tulevaisuuden kehitystä varten vahva tekninen toteutus. Opinnäytetyön aikana tutkittiin ohjelmiston käytettävyyttä lokalisaatiota ja käyttäjätestausta silmällä pitäen. Lisäksi perehdyttiin tietokantojen suunnitteluun.

Lokalisointi on tärkeää uusille markkina-alueille siirryttäessä eikä rajoitu pelkästään kielen kääntämiseen, vaan kyse on kokonaisuudesta, joka ottaa kielen lisäksi huomioon myös kulttuurin vaikutuksen. Lokalisointia on tapana tehdä käytettävyyden parantamiseksi ja näin ollen viime kädessä ohjelmiston käytön helpottamiseksi. Lokalisoinnissa on kyse monimutkaisesta prosessista, joka on syytä ottaa huomioon jo aikaisessa vaiheessa. Lokalisointi on erittäin aikaa vievää ja vaatii paljon resursseja, sillä huomioon otettavia seikkoja on useita. Opinnäytetyön puitteissa kerittiin raapaisemaan vain pintaa ja työ jatkuukin tulevaisuudessa. Osittain lokalisointia voidaan pitää jopa päättymättömänä projektina, jota täytyy tarkastella ja hienosäätää aina aika ajoin.

Opinnäytetyön puitteissa saatiin rakennettua testattava versio KuljetusVelhon kieliversioinnista. Ohjelmistolle saatiin rakennettua juuri sille parhaiten sopiva tietokantaratkaisu. Selainpuolen koodin puolesta saatiin tehtyä tekninen toteutus, jonka avulla käännökset on mahdollista hakea tietokannasta ja sijoittaa käyttöliittymään. Kieliversioinnin käyttöliittymän ulkoasun suunnitelmien toteutukselle ei jäänyt aikaa, vaan opinnäytetyön puitteissa tyydyttiin yksinkertaisempaan käyttöliittymään, sillä toimivan käyttöliittymän toteuttaminen olisi vaatinut liian paljon aikaa ja testausta huomioon ottaen kieliversiointitoteutuksen etenemisen.

Lokalisoinnin lopputulosta pystytään parhaiten arvioimaan tutkimalla ohjelmiston käytettävyyttä. Lokalisoidun ohjelmiston on oltava myös käytettävyydeltään toimiva, ellei jopa parempi kuin ennen lokalisointia. Käyttäjätestaus on hyvä tapa arvioida lokalisoinnin lopputulosta ja saada tärkeää tietoa jatkokehityksen kannalta. Opinnäytetyön aikana ei päästy kieliversioinnissa vielä niin pitkälle, että olisi ollut järkevää lähteä suorittamaan laajoja ja kattavia käyttäjätestauksia. Käyttäjätestausta on

kuitenkin järkevää suorittaa tulevaisuudessa, sillä vain testauksen avulla on viime kädessä mahdollista varmistua lokalisoidun ohjelmiston käytettävyydestä. Opinnäytetyön aikana perehdyttiin käyttäjätestaukseen, mutta varsinaisia käyttäjätestejä ei päästy suunnittelemaan taikka toteuttamaan.

Kieliversioinnin teknisiä toteutuksia on yhtä paljon kuin ohjelmistojakin. Jokaiselle ohjelmistolle täytyy lähtökohtaisesti rakentaa oma tekninen ratkaisu kieliversioinnin toteuttamiseen, eikä ole olemassa mitään yleistä toimintatapaa kieliversioinnin tekniseen toteutukseen. On olemassa monia yleisesti käytössä olevia ratkaisumalleja, mutta viime kädessä ohjelmiston luonteenpiirteet ratkaisevat, mikä ratkaisu sopii parhaiten kyseiselle ohjelmistolle. Opinnäytetyössä perehdyttiin KuljetusVelhon rakenteeseen, jotta saatiin ymmärrys siitä, miten kieliversiointi olisi parhaiten toteutettavissa.

Lopullista ratkaisua hienosäädettiin pitkään ja alkuperäisen suunnitelman mukaan sanasto olisi tallennettu PHP-sessioniin, mutta lopulta päädyttiin globaalin taulukon käyttämiseen, sillä PHP-session kuormittaa liikaa palvelinta laajassa käytössä, koska kaikille käyttäjille luodaan palvelimelle oma session. Sen sijaan palvelimelta löytyvä sanastotaulukko sisällytettiin jokaiseen komentosarjaan ja näin ollen voidaan viitata suoraan siihen ilman, että sanastoa talletettaisiin sessioniin viemään tilaa palvelimelta.

Ohjelmiston lokalisaatiotyön laajuus yllätti opinnäytetyön aikana. Lisäksi käytettävyys asetti omat haasteensa, sillä loppukäyttäjä ei ole kiinnostunut teknisistä ratkaisuista vaan ohjelmiston käytettävyydestä. Näin ollen oli varottava muuttamasta ohjelmistoa liian tekniseen suuntaan. Työ opetti paljon lokalisoinnin ja käytettävyyden lisäksi www-kehityksestä. Palvelin- ja selainpuolen välinen kommunikaatio sekä hyvän tietokannan tehokkuus, laajennettavuus ja ylläpidettävyys olivat asioita, joita työn kautta oli mahdollista syventää entisestään.

Opinnäytetyön tavoitteet olivat alusta alkaen selkeät, ja suurimmilta vastoinkäymisiltä vältyttiin. Tulevaisuudessa suunnittelisin kenties hieman enemmän kieliversioinnin toteutusta ohjelmiston käyttäjän näkökulmasta. Käyttäjien toiminnan selvittäminen auttaa ymmärtämään käyttäjien tarpeita ja sitä, miten ohjelmistoa todellisuudessa käytetään, onko tarvetta vaihtaa kieltä kesken käytön, käytön välissä, yrityksen sisällä jne. Edellä mainitut seikat auttavat jo kieliversioinnin suunnitteluvaiheessa ottamaan askelia oikeaan suuntaan sen sijaan, että nojattaisiin yleiseen tietoon ja tilastoihin.

Ottaen huomioon kieliversioinnin ja lokalisaation monimutkaisuuden ja tärkeyden kansainvälisille ohjelmistoille ja yrityksille ohjelmistokehittäjien on hyvä tietää ainakin perusasiat aiheesta. Nykypäivän ohjelmistoala on kansainvälistä ja kilpailu on kovaa. Kovassa kilpailussa erottuakseen on ohjelmistojen oltava hyviä käyttää ja lokalisoitu juuri kyseiselle markkina-alueelle. Näitä taitoja omaavat työntekijät ja yritykset saavat edun kilpailijoihin nähden. Tämä etu voi olla ratkaisevassa asemassa työpaikkoja ja markkinaosuuksia jaettaessa.

LÄHTEET

Arno, C. 2017. Design Multilingual Website: A Beginner's Guide. Viitattu 15.3.2017 http://www.hongkiat.com/blog/design-multilingual-website-a-beginners-guide/.

Berg, M.; Kojo, I. & Oulasvirta, A. (toim.) 2011. Ihmisen ja tietokoneen vuorovaikutus. Helsinki: Gaudeamus.

Bittersmann, G. 2011. About languages and flags. Viitattu 15.3.2017 http://bittersmann.de/articles/no-flags/.

Carnes, N. 2013. Removing Interface Elements: Should You Ask The User Or Their Browser?. Viitattu 15.3.2017 https://www.smashingmagazine.com/2013/02/remove-interface-elements/.

Garcia-Molina, H.; Ullman, J. & Widon, J. 2009. Database systems. The complete book. 2. painos. Upper Saddle River: Pearson Education.

Grotendorst, T. 2017. 10 Common Mistakes In Software Localization And How To Avoid Them. Viitattu 20.4.2017 https://phraseapp.com/blog/posts/localization-10-common-mistakes-in-software-localization-and-how-to-avoid-them/.

Hernandez, M. 2000. Tietokannat. Suunnittelu ja toteutus. Helsinki: Edita.

Hovi, A.; Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu ja indeksointi. 1. painos. Jyväskylä: Docendo.

IBM 2017. Language Translator. Viitattu 1.3.2017 https://www.ibm.com/watson/developercloud/language-translator.html.

Iina. 2011. What Is Software Localization?. Viitattu 20.4.2017 http://translation-blog.multilizer.com/what-is-software-localization/.

Logistiikan Maailma 2017. Toiminnanohjausjärjestelmä. Viitattu 2.4.2017 http://www.logistiikanmaailma.fi/logistiikka/ohjausjarjestelmat/toiminnanohjausjarjestelma/.

Naypoka, L. 2017. Multilanguage Database Design in MySQL. Viitattu 23.2.2017 http://www.apphp.com/tutorials/index.php?page=multilanguage-database-design-in-mysql.

Nielsen, J. 1995. 10 Usability Heuristics for User Interface Design. Viitattu 14.3.2017 https://www.nngroup.com/articles/ten-usability-heuristics/.

Norman, D. 1990. The design of everyday things. New York: Doubleday.

Saariluoma, P. 2004. Käyttäjäpsykologia. Ihmisen ja koneen vuorovaikutuksen uusi ajattelutapa. 1. painos. Helsinki: WSOY.

Saariluoma, P. & Oulasvirta, A. (toim.) 2011. Ihmisen ja tietokoneen vuorovaikutus. Helsinki: Gaudeamus.

Saariluoma, P.; Kujala, T.; Kuuva, S.; Kymäläinen, T.; Leikas, J.; Liikkanen, L. & Oulasvirta, A. 2010. Ihminen ja teknologia. Hyvän vuorovaikutuksen suunnittelu. Helsinki: Teknologiateollisuus ry.

SDL 2017. Discover the benefits of software localization. Viitattu 20.4.2017 http://www.translationzone.com/solutions/software-localization/.

Sinkkonen, I.; Kuoppala, H.; Parkkinen, J. & Vastamäki, R. 2009a. Käytettävyyden psykologia. 1. versio. Helsinki: Adage.

Sinkkonen, I.; Nuutila, E. & Törmä, S. 2009b. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanoma.

Spencer, W. 2016. ERP (Enterprise Resource Planning). Viitattu 2.4.2017 http://www.tech-faq.com/erp.shtml.

TechRepublic 2013. 10 tips and best practices for software localization. Viitattu 20.4.2017 http://www.techrepublic.com/blog/10-things/10-tips-and-best-practices-for-software-localization/.

Vertommen, K. 2015. Tips for Designing and Building a Multilingual Website. Viitattu 15.3.2017 https://webdesign.tutsplus.com/articles/tips-for-designing-and-building-a-multilingual-website--cms-24708.

Wiio, A. 2004. Käyttäjäystävällisen sovelluksen suunnittelu. 1. painos. Helsinki: Edita.