Ohjelmistojen lokalisointi ja kansainvälistäminen

Matti Riikonen

24.8.2006

Joensuun yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

Tiivistelmä

Tämän tutkielman aiheena on kansainvälisen ohjelmiston tuottaminen. Kansainvälisen ohjelmiston tuottaminen sisältää kaiken tietokoneilla käytettävän materiaalin Internet-sivuista ohjelmistoihin sekä myös oheismateriaalin näille tuotteille. Tutkielman tarkoituksena on selvittää lukijalle aiheeseen liittyviä ongelmia sekä menetelmiä näiden ratkaisemiseen. Näiden menetelmien tarkoitus on mahdollistaa suuren ja heterogeenisen kohdeyleisön palveleminen niin kielellisten kuin muidenkin kulttuuririippuvaisten vaatimusten osalta.

ACM-luokat (ACM Computing Classification System, 1998 version): D.1, D.2

Avainsanat: lokalisointi, kansainvälistäminen

Sisältö

1 Johdanto

Perinteisessä ohjelmistotuotannossa keskitytään yleensä ohjelmointiteknisiin haasteisiin, kuten tiedon tallentamiseen, sen hakemiseen, käsittelyyn ja erinäisten asiakkaan vaatimusten täyttämiseen. Tuote valmistuu lopulta ja se täyttää sille asetetut vaatimukset jossain määrin. Tavoitteena on myös saada tyytyväinen asiakas ja palkka ajallaan ja hyvin tehdystä työstä. Ohjelmistosta luodaan myös yleensä komponentteja, joita voidaan tarvittaessa käyttää uudelleen.

1.1 Miksi lokalisointia ja kansainvälistämistä tarvitaan

Ajatellaan tilannetta, jossa ohjelmistoyritys saa uuden tilauksen samankaltaisesta ohjelmistosta, jonka se on jo toteuttanut, mutta tällä kertaa erikieliseltä asiakkaalta. On luonnollista, että valmistaja hyödyntää jo luotua ohjelmistoa. Ohjelmisto ei kuitenkaan sellaisenaan täytä uuden asiakkaan tarpeita. Ohjelmistoa täytyy muokata toimimaan uudella kielellä sekä mahdollisesti muuttaa sitä uuden asiakkaan vaatimusten mukaiseksi.

Muuttuneet vaatimukset ovat varmasti tuttuja suurimmalle osalle ohjelmistotuottajista, mutta käyttöliittymän kielen vaihtaminen voi osoittautua yllättävän työlääksi operaatioksi. Suoraviivaisesti ajateltuna koko ohjelmakoodista on käytävä läpi kaikki virheilmoitukset, kehotteet, tulostukset ja muut käyttäjälle näkyvät tekstit ja käännettävä ne uudelle kielelle. Tämän lisäksi asiakas voi haluta ohjelmistonsa noudattavan käyttäjien kotimaan ja äidinkielen käytäntöjä. Tämä tarkoittaa esimerkiksi kellonajan, valuutan ja kalenterin muuttamista saman näköisiksi kuin kyseisessä maassa asuva tai kyseistä kieltä puhuva on tottunut ne näkemään.

Toisaalta voidaan ajatella yritystä, joka myy tai mainostaa tavaraa Internetissä. Asiakkaita tulee yrityksen kotisivuille hakukoneiden kautta, poimimalla osoitteen mainoksista, mainosviirejä seuraamalla tai kirjoittamalla osoitteen suoraan selaimeen. Jos sivut on toteutettu vain yhdellä kielellä, eivät ne voi palvella kuin henkilöitä, jotka ymmärtävät kyseistä kieltä. Muut mahdolliset asiakkaat siirtyvät kilpailijan sivuille, jotka ovat mahdollisesti paremmin toteutetut. Esimerkiksi japanilainen autonvalmistaja voisi tehdä sivunsa japaniksi, joka on täysin järkevää, sillä onhan tehdas Japanissa ja yrityksellä luultavasti iso määrä asiakkaita siellä. Jos kuitenkin kyseisen yrityksen autoja

myydään muuallakin, kuten vaikkapa Suomessa, ei japaninkielisistä sivuista ole paljoa apua suurimmalle osalle näiden maiden kansalaisista. Yhtälailla yrityksen kilpailijatkin voivat myydä autoja samoissa maissa, ja jos heidän kotisivunsa on tehty näiden maiden kielillä, saavat he enemmän asiakkaita sivuillensa.

Kansainvälistäminen (internationalization) ja lokalisointi (localization) ovat prosesseja joiden tarkoituksena on helpottaa ja suurimmassa osassa tapauksista ylipäätänsä mahdollistaa useassa eri maassa asuvien ja eri kieliä puhuvien käyttäjien palveleminen järkevällä ja kustannustehokkaalla tavalla. Kielen lisäksi erilaisia taustoja omaavia käyttäjiä erottaa toisistaan monet seikat, kuten esimerkiksi kulttuuri. Esimerkiksi erilaiset kuvat ja symbolit herättävät eri käyttäjissä erilaisia tunteita, jolloin haluttu sanoma ei välttämättä ole sama eri kulttuurien edustajille. Kansainvälistäminen ja lokalisointi sisältävät myös näiden seikkojen huomioonottamisen. Sen sijaan että edellä mainitussa tapauksessa ohjelmiston tuottaja olisi kääntänyt manuaalisesti koko ohjelmiston toiselle kielelle, olisi ohjelmisto alunperin voitu luoda kansainvälistämistä ja lokalisointia käyttäen, jolloin olisi päästy helpommalla. Toisaalta Internet-sivujen omistaja voisi luoda sivustonsa usealla eri kielellä niin, että se olisi vielä helppo ylläpitää ja laajentaa kansainvälistämisen ja lokalisoinnin avulla. Lokalisoinnista käytetään suomen kielessä myös usein termejä kotoistaminen tai paikallistaminen.

1.2 Tutkielman sisältö

Tutkielmassa käydään aluksi läpi mitä kaikkea tulee ottaa huomioon, kun tehdään sovelluksia monille eri kielille ja kansallisuuksille. Lisäksi pohjustetaan kaikkia niitä haasteita, jotka odottavat tähän tehtävään ryhtyviä ohjelmistojen tuottajia. Seuraavaksi käydään läpi ohjelmointiin ja muihin teknisiin osa-alueisiin liittyviä ratkaisuja. Sitten tutustutaan lokalisointi- ja kansainvälistämisprosessiin, eli siihen kuinka annetut tekniset ratkaisut on hyvä järjestää hallinnollisella tasolla. Seuraavaksi paneudutaan tarkemmin kulttuurin vaikutukseen lokalisoinnissa. Lopuksi tehdään näistä yhteenveto.

2 Käyttäjien erot

Tässä luvussa tutustutaan lokalisointiin ja kansainvälistämiseen. Lisäksi käydään läpi niiden hyötyjä tavanomaiseen ohjelmistotuotantoon verrattuna. Luvussa on käytetty pitkälti O'Donnellin (1994) kirjaa.

2.1 Monikulttuurinen ohjelmistomaailma

Ohjelmistoja, olivat ne sitten Internet-sivuja tai käyttäjän koneelle asennettavia sovelluksia, voi tutkia ainakin kahdesta eri näkökulmasta: käyttäjän ja valmistajan. Käyttäjä näkee ohjelmiston ulospäin näkyvät osat, opastetekstit, värit, kuvat, graafisen ulkoasun ja niiden kautta mahdolliset toimenpiteet, joita ohjelmistolla voi tehdä.

Kaikki nämä ovat kuitenkin enemmän tai vähemmän riippuvaisia käyttäjästä ja tämän kielestä ja kulttuurista. Esimerkiksi suomalainen käyttäjä ei välttämättä ymmärrä yhtään mitään espanjankielisestä tekstistä ja äänestä. Värit ja kuvatkin voivat olla harhaanjohtavia, jos ne ovat espanjalaisia tottumuksia vastaavia. Erot ja ongelmat lisääntyvät, kun kyseessä on esimerkiksi kiinalainen käyttäjä ja vaikkapa israelilainen sovellus. Pahimmassa tapauksessa käyttäjä ei yksinkertaisesti voi käyttää sovellusta. Varsinkin Internetissä on varsin yleistä, että vaikkapa suomalainen käyttäjä löytää sivun, jossa on hänelle kiinnostavaa tietoa, mutta ei voi hyötyä siitä, koska ei ymmärrä kieltä. Toisaalta mahdollinen asiakas voi kävellä kauppaan ja löytää sieltä haluamansa ohjelmiston tarpeisiinsa. Luonnollisestikaan kyseinen asiakas ei voi ostaa tuotetta ellei hän osaa kieltä, jolla tuote on tehty.

Englantia käytetään tavallisesti ohjelmistojen luonnollisena kielenä, mutta sitäkään eivät kaikki käyttäjät osaa. Internetin käyttäjistä äidinkielenään englantia puhuvien suhteellinen määrä pienenee jatkuvasti. Taulukko 1 havainnollistaa tilanteen kehitystä. Siinä luvut ovat miljoonia ihmisiä, jotka käyttävät kyseistä kieltä. SHM-sarake (suhteellinen henkilömäärä) on prosenttimäärä kaikista ihmisistä, jotka puhuvat kyseistä kieltä, ja käyttävät Internettiä vuonna 2005. HM-sarake (henkilömäärä) sisältää tiedon siitä, kuinka paljon kyseistä kieltä puhuvia ihmisiä oli vuonna 2005. Taulukosta voidaan nähdä varsinkin englanninkielisten, ja muita kieliä puhuvien käyttäjämäärien kehitys viimeisiltä riveiltä. Vuonna 1996 Internetissä englantia kielenään käytti suurin osa käyttäjistä, suhteellinen ero on kuitenkin kaventunut koko ajan, ja vuonna 2005

muita kieliä käyttäviä oli jo lähes kolme kertaa enemmän kuin englantia käyttäviä. Tämä tarkoittaa sitä, että jos haluaa palvella suurta joukkoa ihmisiä Internet-sivuillaan, täytyy ne toteuttaa useammalla kuin yhdellä kielellä.

Tulevaisuudessa kehityksen jatkuessa samanlaisena englanninkieliset Internet-sivut palvelevat suhteellisesti yhä pienempää joukkoa. Jopa Yhdysvaltojen sisällä puhutaan useaa eri kieltä, espanja tulee englannin jälkeen seuraavana ja monet eivät edes ymmärrä englantia. Käyttäjälle oikealla kielellä toteutettu ohjelmisto on erityisen tärkeää, kun on kyse maasta, jossa englantia ei opeteta tai yleisesti osata. Samoin tietyt käyttäjäryhmät, kuten lapset, harvemmin osaavat muuta kieltä kuin äidinkieltään.

Taulukko 1: Internetin käyttäjämäärien kehitys kielien mukaan. (Global Reach, 2005)

Kieli	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	SHM	HM
Espanja	0	1	2	13	21	35	50	66	70	80	24%	332
Japani	2	7	9	20	39	48	61	70	85	105	84%	125
Saksa	1	4	6	14	22	37	43	53	62	71	72%	98
Ranska	0	2	3	10	17	18	23	28	40	49	68%	72
Kiina	0	1	2	10	31	48	78	103	160	220	25%	885
Skandinavia	2	2	3	8	9	11	14	15	16	17	88%	19.3
Italia	0	1	2	10	12	20	24	24	35	42	74%	57
Hollanti	0	1	2	6	7	11	13	12	14	15	73%	20
Korea	0	0	1	5	17	25	28	30	35	40	53%	75
Portugali	0	0	1	4	11	14	19	26	32	38	22%	170
Muut:		11	15	6.4	29	41	64	89	129	142		
Englanti	40	72	91	148	192	231	234	288	280	300	59%	508
Yhteensä:	50	117	151	245	391	529	627	729	941	1100	19%	6400
Muut kuin englanti:	10	45	71	109	211	307	418	517	680	820	13%	5792

Ohjelmiston valmistajan näkökulmasta tärkeintä on luonnollisesti tehdä ohjelmistoja, jotka tyydyttävät asiakkaita ja myyvät siten paremmin. Tehtäessä sovellusta erikoistilauksena tietylle asiakkaalle tiedetään yleensä mitä luonnollista kieltä ohjelmiston tulee olla, joten toteutus voidaan tehdä normaalisti. Tilanne hankaloituu, jos ohjelmiston tilaaja onkin esimerkiksi monikielinen yritys, joka haluaa saman ohjelmiston käyttöön vaikkapa kolmella eri kielellä, jotta sitä voidaan käyttää kaikissa yrityksen toimipisteissä. Suoraviivainen ratkaisu ongelmaan olisi tehdä kolme eri versiota ohjelmasta, jokaiselle kielelle omansa. Tämä tarkoittaa kuitenkin lähes kolminkertaista työmäärää. Alkuperäinen ohjelmistoversio täytyy ensinnäkin käydä läpi koko lähdekoodin osalta ja kääntää uudelle kielelle. Sitten täytyy muuttaa kaikki kulttuurisidonnaiset asetukset kuten päivämäärien esitykset. Seuraavaksi tulee käydä vielä läpi kaikki kuvat, äänet ja muut vastaavat, joissa on mahdollisesti luonnollista kieltä tai muuta, joka täytyy muuttaa. Tämä kaikki täytyy tehdä vielä toisen kerran, jotta kaikki kolme kieltä saadaan tuetuksi. Lopputuloksena saadaan kolme erillistä versiota samasta ohjelmasta, joista kukin palvelee yhtä asiakkaan käyttäjäryhmää.

Lisää ongelmia seuraa, jos ohjelmistoa täytyy muuttaa joko lisäämällä siihen toiminnallisuutta tai korjaamalla testauksessa tai käytössä huomattuja virheitä. Jokaiseen versioon täytyy yleensä tehdä samat muutokset. Toisaalta, jos virhe löytyi vaikkapa ohjelmiston saksalaisesta versiosta, ei voida aina tietää oliko se itseasiassa käännöksen ja muokkauksen yhteydessä tapahtunut virhe, vai löytyykö se myös kahdesta muusta versiosta. Muut versiot on siis pakko testata, ja ainakin versio, jossa virhe havaittiin on korjattava. Jos virhe löytyy muistakin versioista, täytyy sama korjaus suorittaa niillekin. Taas kerran joudutaan tekemään lähes kolminkertainen työ. Jos nyt haluttaisiin lisätä neljäs kieli, asiakkaan laajennettua jälleen yhdelle uudelle toimialueelle, pitäisi tehdä neljäs versio olemassaolevien versioiden pohjalta ja seuraava päivitys aiheuttaisi yhä enemmän lisätyötä.

Mahdollinen muunnelma edellisestä ratkaisusta olisi tehdä yksi iso ohjelma, jossa olisi kaikki eri versiot yhdessä. Ohjelman saisi sitten käännettyä mille hyvänsä tuetulle kielelle asettamalla jonkun muuttujan arvon lähdekoodista sopivaksi. Hyvä puoli tässä ratkaisussa olisi se, että ne ohjelmat ja muut ohjelman kokonaisuudet, joita ei tarvitse muuttaa eri kielille sopivaksi, esiintyisivät vain kerran eikä niihin tarvitsisi sen enempää koskea uusia kieliä lisättäessä. Haittapuolena ohjelman koko tosin kasvaa, ja sitä kehitettäessä joudutaan pyörittelemään mukana muiden kielien aliohjelmia, joilla ei välttämättä ole mitään merkitystä senhetkisen tehtävän suhteen. Myöskään erillisten versioiden ongelmista ei ole päästy eroon. Edelleen ohjelman muokkaaminen tai toiminnallisuuden lisääminen vaatii lähdekoodin läpikäymistä kaikilta kieliriippuvaisilta osiltaan ja tarvittaessa muutoksien tekemistä kaikkiin eri kieliversioihin.

Molemmista edellä esitetyistä ratkaisuista seuraa useita haittoja. Ensinnäkin varsinaisen tuotteen valmistuminen kaikilla kielillä viivästyy entisestään. Alkuperäisellä kielellä tehdyn version täytyy olla valmis ennen kuin uusia versioita voidaan alkaa kääntää. Jos valmista versiota ei odotettaisi, täytyisi mahdolliset toteuttamisen aikana huomatut virheet ja muutokset tehdä myös erikielisiin versioihin, jolloin työn määrä moninkertaistuisi. Ohjelmistot eivät koskaan voi olla valmiita tarpeeksi nopeasti, koska ohjelmiston valmistumisen pitkittyminen vähentää sen myyntiarvoa. Tämä taas johtuu siitä, että ohjelman toteuttamisen aikana alalla tapahtuu väistämättä kehitystä. Kehityksen myötä kilpailevat, uudemmat versiot ohjelmistosta saavat uusia, parempia piirteitä. Lisäksi ohjelmistosta voidaan alkaa tehdä uutta päivitystä tai kokonaan uutta versiota edellisen valmistuttua. Tällöin alkaa myös uusien kieliversioiden tekeminen, joten pahimmillaan ohjelmiston uusi versio julkaistaan samoihin aikoihin kuin uudet kieliver-

siot. Tällöin asiakas odottaisi mieluummin esimerkiksi uusimman ohjelmistoversion saksankielistä versiota kuin ottaisi vanhemman ohjelmiston saksankielisen vastikkeen.

Lisäksi ohjelmistojen kääntäminen luonnolliselta kieleltä toiselle on hankalampaa kuin pelkän tekstin kääntäminen. Jotta ylipäätänsä voidaan ottaa vaikkapa suomenkielinen versio ohjelmistosta ja kääntää se ruotsiksi, tarvitaan ohjelmoija, joka osaa ruotsia niin hyvin että osaa kirjoittaa ohjelman luonnollisen kielen ruotsiksi. Lisäksi ohjelmoijan täytyy osata suomea niin hyvin että ymmärtää tarkalleen mitä mikäkin ohjelman lause missäkin kohdassa tarkoittaa. Tilannetta voidaan hieman helpottaa, jos lähdekoodi kirjoitetaan vaikkapa englanniksi, sillä se on melko hyvin ymmärretty kieli ohjelmoijien keskuudessa. Pelkkä kääntäjä ei riitä, koska työn tekijän täytyy osata tulkata myös ohjelmakoodia.

2.2 Lokalisointi ja kansainvälistäminen

Kun on tarve tehdä ohjelmisto usealla eri käyttäjäryhmälle, voidaan laatia tarvittava määrä versioita alkuperäisestä ohjelmistosta, jotta saadaan jokaiselle käyttäjäryhmälle sopiva versio. Toisaalta voidaan tehdä yksi ainoa ohjelma, joka sisältää kaikki eri versiot. Kummassakin näistä ratkaisuista seuraa kuitenkin useita haittapuolia, jotka hankaloittavat ohjelmiston kehittämistä. Eräs vaihtoehto näiden kahden lähestymistavan lisäksi on kansainvälistäminen.

Kansainvälistäminen on prosessi, jossa kieliriippuvainen materiaali erotetaan lähdekoodista, jolloin se on helppo korvata erikielisellä materiaalilla. Sen sijaan, että esimerkiksi käyttäjälle näkyvät luonnollisen kielen lauseet sijoitetaan itse koodiin, tehdään esimerkiksi resurssitiedosto, jonne merkkijonot varastoidaan. Tiedostosta sitten ladataan teksti kääntämisen aikana, jolloin ohjelmalle ei ole väliä mitä lauseet itseasiassa sisältävät. Ulkoisena kieliriippuvainen materiaali on helppo kääntää ilman että on pelkoa ohjelmakoodin muuttumisesta eikä kääntäjän tarvitse tietää mitään ohjelman rakenteesta. Yhdestä versiosta saadaan helposti minkä tahansa tuetun kielen mukainen, kunhan vain toimitetaan sen mukana oikeat kieliresurssit. Mahdolliset päivitykset ja korjaukset koskevat myös vain yhtä versiota, joten työtä ei sinällään tehdä sen enempää. Uuden kielen lisääminen vaatii yleensä vain kieliresurssien kääntämistä uudelle kielelle, eikä varsinaista lähdekoodin kopioimista tai muuntelua ko. versiolle sopivaksi.

Summauksena kansainvälistämisen hyödyt kielikohtaisten versioiden tekemiseen nähden ovat seuraavat:

- *Lyhyempi valmistumisaika:* Erillinen vaihe koodin valmistumisen ja kielikohtaisen version valmistumisen välillä jää pois.
- *Vähemmän versioita:* Koska jokaista kieltä kohden ei tarvita erillistä täydellistä lähdekoodiversiota, selvitään vähemmällä versionhallinnalla.
- *Kustannusten pieneneminen:* Kustannukset pienenevät projektissa useasta syystä. Kääntäjien ei tarvitse osata tulkata ohjelmakoodia poikkeustilanteita lukuunottamatta, ohjelmisto valmistuu nopeammin ja hallinnointi ja varsinkin ylläpito on helpompaa ja yksinkertaisempaa.
- *Laajentaminen helpottuu:* Uusien kielien lisääminen on helpompaa ja edullisempaa.

Kansainvälistämiseen liittyy kuitenkin myös tiettyjä haittoja. Jos ajatellaan normaalia ohjelmistoprojektin kehitystä, johon ei siis liity kansainvälistämistä, ohjelmisto valmistuu tietyllä tutuksi käyneellä tahdilla. Nyt kun samoille ohjelmoijille kerrotaan, että heidän täytyy koodata sama ohjelmisto täysin eri tavalla lopputuloksen silti ollessa täysin sama (niin kauan kuin uusia kieliä ei ole lisätty) on luonnollista, että vastustusta voi ilmetä. Sen lisäksi ohjelmoijat täytyy kouluttaa uuteen ohjelmointitapaan, ja tuotanto voi aluksi olla jopa hitaampaa muuttuneista vaatimuksista johtuen. Lisäksi jos ohjelmisto tehdään tukemaan vain kahta kieltä, voi ohjelmiston tuotanto kestää kauemmin kuin vain tekemällä kaksi eri versiota lähdekoodista (tai yhden version).

Kansainvälistämisen yhteydessä käytetään yleensä myös lokalisointia. *Lokalisointi* on prosessi jossa sovitetaan ohjelmisto jollekin kohderyhmälle, esimerkiksi tanskalaisille. Kansainvälistetyn sovelluksen yhteydessä tämä tarkoittaisi kulttuurisidonnaisten asetusten ja kielimateriaalin lisäämistä. Lokalisoitaessa täytyy ottaa huomioon ainakin tekstin kääntäminen, grafiikkojen muokkaaminen (varsinkin jos niissä on tekstiä, mutta myös tietynlaiset kuvat, värit jne. ovat toisissa kulttuureissa epäsoveliaita) ja uusien grafiikkojen luominen. Lokalisoinnissa ajatus lopulta on, että sovelluksen käyttäjä näkee sovelluksen niinkuin se olisi tehty käyttäjän maassa. Kuitenkin tulee harkita kuinka lähelle käyttäjää haluaa mennä, Suomestakin löytyy useita murteita, ja nettisivujen tekeminen näille eri murteille olisi ehkä hieman liioittelua. Toisaalta esimerkiksi yrityksellä, jolla on nettisivut, voi olla tiettyä hyötyä alkuperästään, jolloin sitä ei

kannatakaan peitellä. Moniin maihin ja kansallisuuksiin liitetään mielikuvia kuten että italialainen pizza on maailman parasta tai että saksalaiset osaavat tehdä hyviä autoja. Tällöin ei kannattaisikaan teeskennellä olevansa vaikkapa suomalainen yritys vaikka tarjoaakin palvelujansa suomeksi.

Kansainvälistämiselle on olemassa myös lyhenne, i18n, joka saadaan kansainvälistämisen englanninkielisestä vastineesta internationalization. Numero 18 lyhenteessä tulee 18:sta merkistä i:n ja n:n välissä. Myös lokalisoinnille on olemassa lyhenne L10n, joka saadaan samalla tavalla kuin kansainvälistämisen lyhenne i18n. Lokalisoinnin englanninkielinen vastike on localization, jossa siinäkin on l:n ja n:n välissä 10 merkkiä. Isoa alkukirjainta käytetään yleensä jotta erotetaan L-kirjain isosta I-kirjaimesta. Erityisesti markkinointipuolella käytetään myös termiä glokalisointi (glocalization, g11n), jolla tarkoitetaan kansainvälistämistä ja lokalisointia yhdessä.

2.3 Kohderyhmän vaikutus lokalisointiin

Koska lokalisointi on prosessi, jossa ohjelmisto sovitetaan jollekin kohderyhmälle, on kohderyhmällä luonnollisesti merkitystä. Tavallisimmassa tapauksessa lokalisointi koskee jotain kieli- tai kansallisuusryhmää, kuten suomalaisia tai ruotsalaisia. Kuitenkin kohderyhmä voi olla myös pienempi tällainen osajoukko. Esimerkiksi kohderyhmänä voi olla vaikkapa ruotsalainen matemaatikkoryhmä. Tällöin voidaan olettaa, että kaikki sovelluksen ko. lokalisoitua versiota käyttävät tuntevat matematiikan, eikä sitä tarvitse sen enempää selvittää niiltä osin. Tämä vaikuttaa lähinnä luonnollisen tekstin kääntämisprosessiin, jolloin voidaan olettaa että tietyt termit, joita alalla käytetään, ovat tiedossa kaikkilla eri kohderyhmiin kuuluvilla henkilöillä. Myöskin kyseinen käyttäjäryhmä voi odottaa tietynlaista kieliasua, jotta sovelluksen käyttö olisi luontevaa. Toisaalta kohderyhmä voi koostua esimerkiksi lapsista. Tällöin tulee varmistaa, että kyseistä kieltä osaava henkilö pystyy käyttämään sovellusta ilman tarpeettomia hankaluuksia. Ongelmia tulee, jos käännetty teksti on liian vaikeaa.

Usein kohderyhmää ei tosin voida määrittää. Jos lokalisoitava sovellus on websovellus, voi sitä käyttää periaatteessa kuka hyvänsä. Asiasisältö kuitenkin rajoittaa kohderyhmää jonkun verran, mutta kohderyhmä on helpompi määritellä sovellukselle, jonka käyttäjät ovat tarkasti tiedossa. Tällaisia sovelluksia ovat esimerkiksi yritysten sisäiseen käyttöön tulevat sovellukset, joiden käyttäjillä on kaikilla tietynlainen koke-

mus tai koulutus työskentelyyn.

Kohderyhmää ajateltaessa on hyvä miettiä kuka sovellusta tulee käyttämään. Onko kyseisellä henkilöllä jotain erikoistarpeita? Haluaako tämä tietynlaisen esitysasun, vieraannuttaako joku toinen ulkoasu käyttäjän, kun taas toinen saa tämän tuntemaan olonsa kotoisaksi? Onko jonkinlainen kieli käyttäjälle hankalaa, tai toinen mahdollisesti liian yksinkertaista? Kun kohderyhmä on tiedossa, kannattaa sen piirteet ottaa huomioon lokalisointia suoritettaessa.

3 Kieli- ja kulttuurierot

Tässä luvussa tutustutaan eri kielten ja kulttuurien välisiin eroihin. Palveltaessa käyttäjiä useista eri maista joudutaan melko usein kohtaamaan sellaisia tilanteita, joista ei selvitä pelkästään sovelluksen kieltä vaihtamalla. Esimerkiksi tietyillä symboleilla, kuvilla ja eleillä on erilaisia merkityksiä eri kulttuureissa. Tällaisten tilanteiden välttämiseksi on hyvä olla selvillä siitä kaikesta, mikä voi vaihdella eri kulttuurien välillä.

3.1 Kieli

Suomen kielessä, niinkuin monessa muussakin eurooppalaisessa kielessä, käytetään latinalaisia aakkosia. Aakkostoissa voi olla isojakin eroja, esimerkiksi unkarilaisessa aakkostossa ei ole q tai w kirjaimia, kun taas vietnamilaisessa aakkostossa ei ole f, j, w tai z kirjaimia. Tämän lisäksi monissa kielissä on ylä- tai alamerkkejä kuten ä:n ja ö:n pilkut suomen kielessä. Yhteistä kaikille latinalaista aakkostoa käyttäville kielille kuitenkin on se, että merkeistä yhdistelemällä saadaan sanoja. Kirjaimet toimivat ääntämisohjeina, eli merkit ä, i, t ja i eivät itsessään tarkoita mitään, mutta yhdistettynä muodostavat suomen kielen sanan "äiti".

On olemassa myös kieliä, jotka perustuvat muuhun aakkostoon kuin latinalaiseen. Kreikan kieli perustuu kreikkalaiseen aakkostoon, jonka merkit ovat tuttuja matemaatikoille. Latinalainen aakkosto pohjautuu itseasiassa alunperin kreikkalaiseen aakkostoon. Kyrillinen aakkosto on myös yksi laajalti käytössä oleva aakkosto, sitä käytetään mm. venäjän kielessä. Lisäksi on olemassa myös heprean, arabian, intian (ja muiden Aasian kielten) ja afrikan kielten aakkostot. Osassa kirjoitussuunta on oikealta vasemmalle, vastoin länsimaisille tuttua vasemmalta oikealle -kirjoitussuuntaa. Muitakin eroja löytyy. Esimerkiksi arabian kielessä merkit saavat eri muodon riippuen siitä missä osassa sanaa ne ovat, ja mitä merkkejä niiden vieressä on.

Myös latinalaisten aakkostojen kirjaimissa on erilaisia muotoja, esimerkiksi isot ja pienet kirjaimet. Useissa kielissä on myös ligatuureja, jotka ovat merkeistä koostuvia kokonaisuuksia, joita ajatellaan yhtenä merkkinä. Esimerkiksi hollannin kielessä on käytössä ij-ligatuuri. Jos virke tai sana, joka kirjoitetaan isolla alkukirjaimella, alkaa ij-ligatuurilla täytyy molemmat kirjaimet kirjoittaa isolla. Arabian ja hindin kielissä ligatuureja on useita, latinalaiseen aakkostoon perustuvissa kielissä vain harvoja. Arabian

ja hindin kielissä ligatuurit saavat myös yleensä aivan uuden muodon kuin niihin kuuluvat kirjaimet perättäin kirjoitettuna saisivat.

Näiden kirjoitustapojen lisäksi on olemassa myös ideografisia vaihtoehtoja. Näissä periaate on, että jokainen merkki tarkoittaa sanaa tai käsitettä. Tällainen kirjoitustapa on käytössä useissa Aasian maassa, kuten Japanissa ja Kiinassa. Eurooppalaisessa kulttuurissakin törmää tavallaan ideografeihin. Esimerkiksi liikennemerkeissä ja muissa opasteissa käytetään yleisesti kuvia sanojen sijaan. Kaikki ideografit niitä käyttävissä kielissä pohjautuvat kiinan kieleen, ja jokaisella kielellä on niille oma nimitys. Kiinan kielessä, niin yksinkertaistetussa kuin perinteisessäkin, niillä on nimi hanzi. Japanin kielessä nimi on Kanji ja korean kielessä Hanja. Ideografeja on huomattavasti enemmän kuin minkään aakkoston merkkejä. Esimerkiksi japanin kielessä on olemassa kymmeniä tuhansia ideografeja, jotka ovat siis verrannollisia sanoihin. Vertailukohtana kuitenkin esimerkiksi englannin kielessä on noin 600 000 sanaa (Oxford University Press, 2005), joten määrä ei kuitenkaan ole käytännössä kovin suuri.

Ideografeja käyttävissä kielissä käytetään jonkun verran myös foneettisia merkkejä. Kiinan kielessä tämä on harvinaisempaa, mutta japanissa yleistä. Foneettisten merkkien käyttö on yleistä japanin kielessä varsinkin sellaisten uusien sanojen kohdalla, joille ei ole olemassa sopivaa ideografia. Japanin kielen kaksi foneettista merkistöä ovat nimeltään Katakana ja Hiragana. Yleisesti ottaen Katakanaa käytetään vierasperäisten lainasanojen kirjoittamiseen, kun taas Hiraganaa käytetään Japanista peräisin olevien sanojen kirjoittamiseen. Yhdessä näitä kutsutaan nimellä Kana. Kummassakin merkistössä on noin 50 merkkiä, ja niitä voitaisiin myös verrata latinalaisten aakkostojen isoihin ja pieniin merkkeihin, Katakana ja Hiragana ovat periaatteessa kaksi eri tapaa kirjoittaa samat foneettiset merkit. Korean kielessä foneettinen merkistö on nimeltään Hangul. Se sisältää 24 merkkiä, joita yhdistelemällä saadaan tavuja, joista sanat muodostetaan. Kiinan kielessä yksinkertaistetussa versiossa yleisimmin käytetty foneettinen merkistö on nimeltään Pinyin. Perinteisessä kiinan kielessä ei ole foneettista merkistöä.

3.2 Päivämäärät

Päivämäärä voidaan ilmaista monella tavalla yhdenkin kielen sisällä. Esimerkiksi suomen kielessä 1. tammikuuta 2003, 1.1.2003, 1.1.03, tammikuun 1. 2003 ja keskiviikko

1. tammikuuta 2003 tarkoittavat kaikki samaa päivää ja jokainen suomalainen myös ymmärtää tämän samaksi päiväksi. Kansainvälisessä sovelluksessa voisi kuvitella, että asian saisi ratkaistua jättämällä luonnollisen kielen pois päivämäärästä jolloin jäljellä olisi vain kaikkien ymmärtämiä numeroita. Asia ei kuitenkaan ole näin yksinkertainen, sillä eri maissa on totuttu laittamaan päivämäärän vuosi, kuukausi ja päivä eri järjestykseen. Tällöin esimerkiksi päiväys 1.2.03 olisi Yhdysvalloissa 2. tammikuuta 2003 ja Japanissa 3. helmikuuta 2001. Tämän lisäksi amerikkalaiset ja englantilaiset käyttävät numeroiden erottimina päivämäärissä kauttaviivaa (/), kun taas italialaiset ovat tottuneet viivaan (-). Lisäksi italialaiset kirjoittavat kuukauden roomalaisilla numeroilla. Näitä erilaisia vaihtoehtoja voi olla samassa maassa käytössä useitakin, mikä tietysti lisää monimutkaisuutta. Joissakin kielissä, kuten englannissa, on tapana kirjoittaa kuukaudet ja viikonpäivät isolla. Esimerkiksi Suomessa ne kuitenkin kirjoitetaan pienellä. Viikonpäivistä ja kuukausista käytetään lyhenteitä päivämäärissä, mutta lyhenteiden pituudet vaihtelevat. Esimerkiksi englannissa kuukaudet lyhennetään kolmeen kirjaimeen, mutta kaikissa kielissä kolme ei riitä, toisissa kielissä kuukauden nimessä ei ole kolmea merkkiä.

Muitakin päivämääriin liittyviä eroja maiden väliltä löytyy. Yhdysvalloissa viikko alkaa sunnuntaista ja päättyy lauantaihin, kun taas Euroopassa viikko alkaa maanantaista ja päättyy sunnuntaihin. Tämä vaikuttaa ainakin kalenterien ulkoasuun, joissa ensimmäisenä kullakin viikolla tulee olla viikon ensimmäinen päivä. Kalenterivuodet ovat myös erilaisia eri maissa. Gregoriaanista kalenteria käyttävillä mailla (kuten Suomella) on vuodessa 365 päivää. Poikkeuksena on karkausvuosi, jossa on 366 päivää. Karkausvuosi on neljällä jaollisina vuosina lukuunottamatta sadalla jaollisia vuosia, jollei vuosi ole neljällä sadalla jaollinen. Monissa muslimimaissa käytössä on kalenteri, jonka vuodessa on 354 tai 355 päivää, vuodesta riippuen. Koska päiviä on vähemmän kuin gregoriaanisessa kalenterissa, vaihtuu vuoden ensimmäinen päivä suhteessa gregoriaaniseen kalenteriin joka vuosi.

Israelissa on käytössä kalenteri, jossa on vuodesta riippuen joko 12 tai 13 kuukautta, ja päiviä 353 - 355 normaaleina vuosina. Karkausvuosina käytetään karkauskuukautta, joten heidän kalenterinsa siirtyy tällöin samaan aikaan gregoriaanisen kalenterin kanssa. Vuosien laskemisessa on myös eroja eri maiden välillä. Gregoriaaninen kalenteri laskee vuosia Jeesuksen syntymästä. Japanilaiset käyttävät gregoriaanista kalenteria, mutta heillä on käytössä myös järjestelmä, jossa vuodet lasketaan aikakausien mukaan. Aina kun uusi keisari valitaan, alkaa uusi aikakausi ja vuodeksi tulee 1 kyseisen keisa-

rin aikakautta. Muslimien kalenteri aloittaa vuosien laskemisen hetkestä, jolloin Muhammed pakeni Mekasta Medinaan, joka tapahtui 16.7.662 gregoriaanista ajanlaskua. Buddhalaisten kalenteri aloittaa ajanlaskun Buddhan syntymästä, heprealainen kalenteri taas maailman alkamisesta, joka heidän mukaansa tapahtui vuonna 3761 eKr.

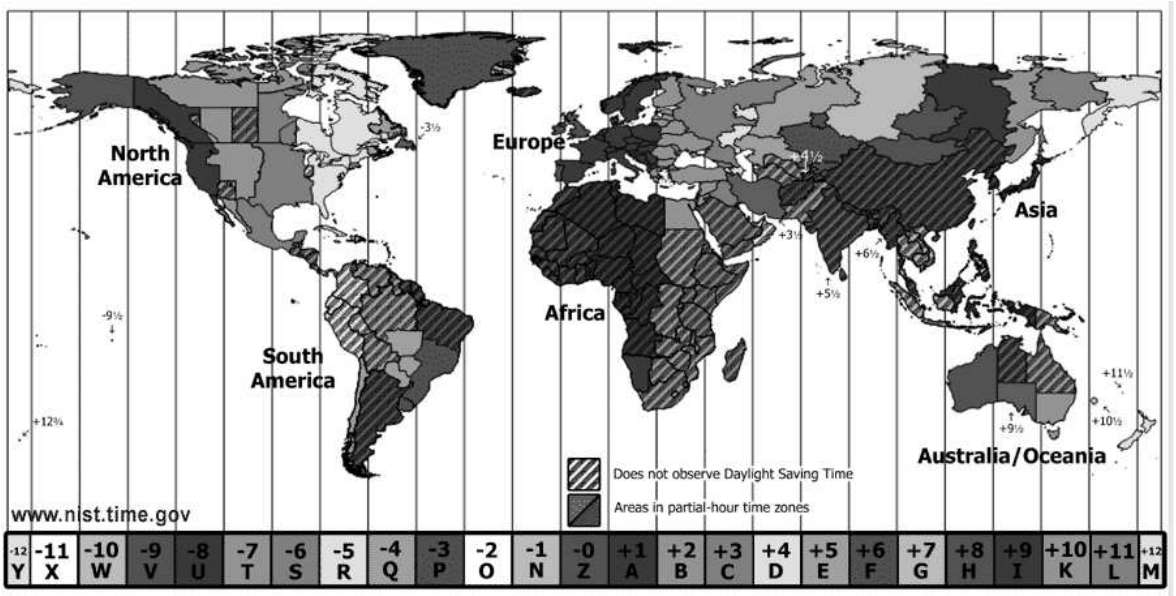
Tilannetta on helpotettu luomalla ISO-standardi päivämäärän esittämiseen. Kyseinen standardi on ISO 8601, joka määrittelee päivämäärälle gregoriaanisessa kalenterissa yksioikoisen VVVV-KK-PP -esitysmuodon. Mutta eroja siis riittää. Ei voidakaan olettaa, että mikään yksi kalenteriratkaisu täyttäisi kaikkien maailman käyttäjien tarpeet. Jos sovellus esimerkiksi vaatii käyttäjän antamaan syntymäpäivänsä, mutta olettaa, että nykyinen päivämäärä on gregoriaanisen ajanlaskun mukainen, saadaan käyttäjän jotain aivan muuta kuin tarkoitus olisi, jos tämä antaa syntymäpäivänsä vaikkapa heprealaisen kalenterin mukaan. Vain numeroilla annetuissa päivämäärissä voivat kuukaudet ja päivät mennä sekaisin, asiakas voi olettaa esimerkiksi laskun eräpäiväksi jotain aivan muuta kuin se oikeasti on. Kansainvälisiä sovelluksia laatiessa onkin tärkeä olla tietoinen näistä eroista.

3.3 Aika

Kaikkialla maailmassa on käytössä sama järjestelmä ajan mittaamiseen ja ilmaisemiseen, vuorokaudessa on 24 tuntia, tunnissa 60 minuuttia ja minuutissa 60 sekuntia. Maapallo on ajettu 24:ään aikavyöhykkeeseen, yksi jokaiselle tunnille. Jokaisella aikavyöhykkeellä on eri aika, ja se on suhteutettu GMT:hen (Greenwich Mean Time, nimetty Englannissa sijaitsevan Greenwich -kaupungin mukaan). Esimerkiksi GMT+3 tarkoittaisi aikavyöhykettä joka on kolmas Greenwichistä itään. Aikavyöhykkeillä on myös suhteellisen GMT-ajan lisäksi nimiä, esimerkiksi GMT+2 -vyöhyke tunnetaan myös nimellä EET (Eastern European Time).

Aikavyöhykkeet eivät kuitenkaan kulje suorina viivoina maapallon poikki, vaan valtiot voivat itse valita mihin aikavyöhykkeeseen kuuluvat. On myös mahdollista, että yksi valtio kuuluu useampaan kuin yhteen aikavyöhykkeeseen, mikä on tavallista varsinkin isoilla valtioilla. Kuva 1 näyttää aikavyöhykkeiden sijoittumisen valtioittain ja alueittain. Kuvasta nähdäänkin, että varsinaisia aikavyöhykkeiden rajoja noudatetaan melko vähän, valtiot käyttävät yleensä itselleen sopivaa aikaa, esimerkiksi Ranska ja Espanja ovat molemmat GMT+1 -aikavyöhykkeellä, vaikka maantieteellisesti kuului-

sivat GMT-0 -vyöhykkeelle. On kuitenkin käytännöllistä olla samassa ajassa muiden Länsi-Euroopan maiden kanssa. Monissa maissa myös siirretään kelloja vuodenaikojen mukaan. Suomi on yksi näistä maista. Kelloja siirretään tunnilla eteenpäin tai taaksepäin, vuodenajasta riippuen. Tästä johtuen maapallon eteläpuolella siirretään kelloja eri suuntaan kuin pohjoispuolella samaan aikaan. Lisäksi kaikki maat eivät näin tee, mikä tietysti hankaloittaa tilannetta entisestään.



Kuva 1: Aikavyöhykkeet (National Institute of Standards and Technology, 2005).

Aika voidaan myös esittää eri tavoilla. Yhdysvalloissa on käytössä 12 tunnin kello, jossa käytetään a.m. -merkintää ajasta ennen puolta päivää ja p.m. merkintää puolen päivän jälkeisestä ajasta. Esimerkiksi kello 14 olisi 2 p.m. Yhdysvalloissa. Tämän lisäksi minuuttien ja tuntien erottimena voidaan käyttää pistettä, kaksoispistettä, h-kirjainta tai ei mitään. Myös ajan esittämiseen voi käyttää ISO 8601 -standardia, jossa aika esitetään muodossa TT:MM:SS.

3.4 Valuutta ja numerot

Rahasummien esittämisessä on maiden välillä suuriakin eroja. Olennaisin on luonnollisesti maiden eriävät valuutat. Suomessa, ja suurimmassa osassa Euroopan Unionin maita, käytetään euroa. Tämä helpottaa sovellusten luomista EU:n alueelle, vaikka tällöinkin täytyy ottaa huomioon maat, jotka eivät euroa käytä. Tilanne monimutkaistuu, kun siirrytään EU:n ulkopuolelle, jossa käytössä on useita eri valuuttoja. Monien mai-

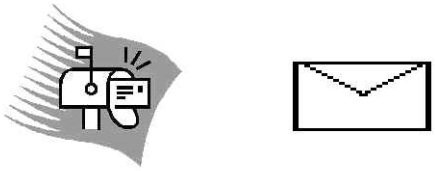
den kansalaiset haluavat tuotteita ostaessaan nähdä hinnan omalla valuutallaan. Tämä pätee niin Internet-sivuihin, jotka myyvät tuotteita kuin sovelluksiin, jotka esittävät rahasummia. Jos sovelluksen saa tukemaan uutta rahasummaa vain vaihtamalla valuuttasymbolin (ja ehkä lisäämällä tilaa rahasummille tarkoitettuihin kenttiin), on tilanne vielä helppo. Internetsivuilla tilanne on hieman monimutkaisempi, koska tuotteiden hinnat täytyy saada näkymään eri asiakkaille eri valuutoilla niin, että yritys saa rahaa yhtä paljon joka asiakkaalta (maksettuaan tarvittavat verot yms.) Tällöin tuotteiden hinnat täytyy tarkistaa ajoittain, jotta pysytään mukana valuuttakurssien heilahteluissa.

Myös valuuttojen esitysasuissa on eroja. Suomessa on totuttu esittämään valuutat, ja muutkin numerot, siten että kokonaislukuja ja desimaalilukuja erottaa pilkku ja tuhansia tyhje. Esimerkiksi Saksassa niin tuhansia kuin desimaalilukuja erottaakin piste, kun taas Venäjällä tuhansia erottaa piste, mutta desimaaleja tyhje. Valuuttasymboli voidaan laittaa joko rahasumman eteen, kokonaislukujen ja desimaalien väliin tai luvun jälkeen (esim. $100.52, 100$52 tai 100,52 €). Desimaalien määrä vaihtelee, esimerkiksi Suomessa käytetään kahta desimaalilukua valuutoissa, kun taas Japanissa käytetään kolmea. Valuutoille on laadittu ISO-standardi helpottamaan kunkin maan valuutan erottamista samanlaisella tavalla. Kyseinen standardi on ISO 4217, jossa kullekin valuutalle on annettu kolmikirjaiminen koodi kuten USD, EUR ja JPY.

3.5 Värit ja symbolit

Väreillä on erilaisia merkityksiä eri kulttuureissa. Taulukossa 2 on otettu muutamia värejä ja niiden merkityksiä vertailuun. Esimerkiksi valkoinen tarkoittaa Yhdysvalloissa, Ranskassa ja Egyptissä yleisesti ottaen vain positiivisia asioita. Intiassa, Japanissa ja Kiinassa taas valkoinen yhdistetään kuolemaan. Samoin Yhdysvalloissa punainen yhdistetään vaaraan, kun taas Kiinassa se yhdistetään iloisuuteen. Erot ovat siis varsin merkittäviä, vaikka niitä ei tule ajateltua, jos ei ole asiaan perehtynyt. Värit vaikuttavat lähinnä sovellusten käyttöliittymien suunnitteluun, sillä käyttäjä tekee niistä omat mielleyhtymänsä. Kuitenkin tietyillä väreillä on joissakin tilanteissa universaali merkitys, kuten liikennevaloissa (Wikipedia, 2006). Niissä punainen valo tarkoittaa aina pysähtymistä, ja vihreä sitä, että saa mennä. Usein käytetään myös keltaista valoa kertomaan siitä, että siirrytään pian punaisesta vihreään, tai päinvastoin. Punaisen ja vihreän sävyt voivat myös vaihdella hieman, punaiseen voidaan lisätä hieman oranssia ja vihreään hieman sinistä, jotta punavihervärisokeat erottavat ne helpommin.

Symboleilla on myös omat merkityksensä eri kulttuureissa. Käyttöliittymiä suunniteltaessa saattaa tulla ongelmia, kun käytetään symbolia, jota ei tunneta kohdekulttuureissa. Esimerkiksi sähköpostin lähettämistä varten sovelluksessa tai Internet-sivustolla voitaisiin käyttää symbolia (Russo ja Boor, 1993). Kuvassa 2 on kaksi symbolia, jotka molemmat tarkoittavat postin tai sähköpostin lähettämistä. Vasemmanpuoleinen symboli kuvaa Yhdysvalloissa käytössä olevaa postilaatikkoa, joka voi olla monelle tuntematon. Kyseinen postilaatikko toimii siten, että sen kyljessä oleva lippu nousee pystyyn kun postilaatikkoon tulee postia. Varsinkin jos kuvassa ei olisi edes kirjettä, tarkoitus voisi jäädä ymmärtämättä. Kulttuurikohtaisia symboleja ja kuvia kannattaakin välttää jo alkuperäisen sovelluksen tekemisvaiheessa sen sijaan että lokalisointivaiheessa korvaisi ne erilaisilla symboleilla. Oikeanpuoleinen kirjekuoren symboli on maailmanlaajuisesti tunnettu ja sitä kannattaisikin tässä yhteydessä käyttää.



Kuva 2: Postisymboleja.

3.6 Osoitteet ja puhelinnumerot

Jos sovellus tarvitsee osoitetietoja, ei voida olettaa, että kaikki maailman ihmiset haluavat niiden näkyvän samalla tavalla. Suomalaisille normaali tapa olisi ilmaista osoite muodossa: koko nimi, osoite, postinumero ja postitoimipaikka, sekä tarvittaessa maa. Lisäksi jos käytetään yrityksen nimeä, se voi tulla joko ennen vastaanottajan nimeä tai sen jälkeen. Osoitteeseen voidaan käyttää kaksi riviä, jos tarve näin vaatii. Näin ei

Taulukko 2: Värien merkityksiä eri kulttuureissa (Russo ja Boor, 1993)

Kulttuuri	Punainen	Sininen	Vihreä	Keltainen	Valkoinen
Yhdysvallat	Vaara	Maskuliinisuus	Turvallisuus	Pelokkuus	Puhtaus
Ranska	Ylhäisyys	Vapaus, Rauha	Rikollisuus	Väliaikaisuus	Neutraalius
Egypti	Kuolema	Hyveellisyys, usko, totuus	Hedelmällisyys, voima	Iloisuus	Riemu
Intia	Elämä, luovuus		Varakkuus, hedelmällisyys	Menestys	Kuolema, puhtaus
Japani	Vaara, kiukku	Pahanteko	Tulevaisuus, nuoruus	Hienovaraisuus, ylemmyys	Kuolema
Kiina	Iloisuus	Taivas, pilvet	Ming -Dynastia, taivas, pilvet	Syntymä, varakkuus, voima	Kuolema, puhtaus

kuitenkaan toimita kaikissa maissa, ja tämä täytyy ottaa huomioon mikäli käsitellään osoitteita.

Taulukko 3: Eri tapoja osoitteiden esittämiseen (Software Marketing Resource, 2005)

Maa	Osoite
Saksa	Yrityksen nimi Etunimi Sukunimi Osoiterivi 1 Osoiterivi 2 Tyhjä rivi Maanumero Postinumero Kaupunki
Englanti	Etunimi Sukunimi Yrityksen nimi Osoiterivi 1 (yleensä talon nimi tai numero ja katu) Osoiterivi 2 (yleensä lähiö kuten kylä) Osoiterivi 3 (yleensä postitoimipaikka) Maa (valinnainen) Postinumero Maa
Kiina	Maa Provinssi Kaupunki Osoiterivi 1 Sukunimi Etunimi Maa
Venäjä	Maa Postinumero Osavaltio tai tasavalta Alue Kaupunki Osoiterivi 1 Osoiterivi 2 Yrityksen nimi Sukunimi Etunimi Toinen nimi

Taulukko 3 havainnollistaa erilaisia tapoja ilmaista osoitteita eri maissa. Huomioitavaa taulukosta on sen verran, että jos käytetään puhuttelutitteliä, kuten herra tai rouva, se

lisätään taulukon maissa nimien eteen, paitsi Kiinassa nimien jälkeen. Myös joissakin maissa käytetään arvonimikettä kuten johtaja, jolloin se yleensä tulee puhuttelutittelin jälkeen, jos se on nimien edessä, tai nimien eteen, jos puhuttelutitteliä ei käytetä. Venäjällä sukunimi ja etunimet tulevat eri riveille, mutta jos etunimistä käytetään vain ensimmäisiä kirjaimia (esim. M.A.) tulevat ne samalle riville sukunimen kanssa. Myös englannin kielessä yritysten nimet kirjoitetaan siten, että jokainen erillinen sana kirjoitetaan isolla, sanoja kuten "ja"ja "tai"lukuunottamatta. Yhdysvalloissa osoitteeseen täytyy myös sisällyttää osavaltio. Ohjelmoija ei voi tehdä tässäkään asiassa oletuksia, vaan hänen tulee varautua todella erilaisiin osoitteisiin.

Puhelinnumeroiden esityasut eroavat myös maittain, jokaisella maalla on oma maakoodinsa joka tulee laittaa puhelinnumeron eteen mikäli ei ole ko. maassa. Sekä suuntanumerot maiden sisällä että numeroiden pituudet vaihtelevat. Esitysasuissakin on eroja. Joissakin maissa numeroiden osat erotetaan esimerkiksi viivalla, joissakin ne kirjoitetaan yhteen tai erotetaan välillä.

3.7 Mittajärjestelmät

Euroopassa on pääosin käytössä metrinen SI-järjestelmä, jossa metri on perusyksikkö. Metriä jaetaan tai kerrotaan kymmenillä ja näin saadaan uusia mittoja. Massaa SI-järjestelmässä mitataan grammoilla, joita taas kerrotaan jokapäiväisessä käytössä lähinnä ylöspäin tuhansilla, että saadaan kilogrammoja tai tonneja. Kuitenkin esimerkiksi Yhdysvalloissa käytössä on valtaosin Imperiaalinen systeemi, joka perustuu tuumaan (2,54 cm), jalkaan (12 tuumaa eli 0,3048 metriä), jaardiin (3 jalkaa eli 0.9144 metriä) ja mailiin (1760 jaardia eli 1.6093 km). Näihin tulee kiinnittää huomiota mittoja esitettäessä, sillä toiseen systeemiin tottunut henkilö ei yleensä heti ymmärrä toisella esitettyjä mittoja.

3.8 Muita eroja

Maiden ja kulttuurien väliltä löytyy paljon muitakin eroja joita käydään tässä kappaleessa läpi. Lainsäädännöt vaihtelevat maittain. Toisessa maassa täysi-ikäinen henkilö ei olekaan jossain toisessa maassa täysi-ikäinen. Jotkut tuotteet saattavat myös olla laittomia joissain maissa vaikka ovatkin laillisia muissa maissa. Tärkeintä kulttuurien ja

maiden välisten erojen ymmärtämisessä on se, että näkee eroja olevan muuallakin kuin käytetyssä kielessä.

3.9 Kieli- ja kulttuurierojen käsittely tietojärjestelmän eri tasoilla

Tässä luvussa on edellä lueteltu erilaisia kieli- ja kulttuurieroja, jotka tulee ottaa huomioon sovellusta kansainvälistettäessä ja lokalisoitaessa. Tässä kappaleessa käydään kootusti läpi kaikki nämä erot ja niiden merkitykset tietojärjestelmän eri tasoilla.

Tietojärjestelmän tasot ovat tässä tutkielmassa: käyttöjärjestelmä, ikkunointijärjestelmä, sovellus ja web-sovellus. Tasoja voidaan ajatella päällekkäisinä. Käyttöjärjestelmä on kaikista pohjimmaisena, kauimpana käyttäjästä. Ikkunointijärjestelmä tulee seuraavaksi, ja päällimmäisenä on joko sovellus tai web-sovellus. Varsinaisesti käyttäjä kokee näistä käyttävänsä itse sovellusta. Normaalisti sovellus hoitaa tapauskohtaisen tiedon käsittelyn, joka on ko. sovellukselle ominaista. Ikkunointijärjestelmä hoitaa tiedon esittämisen ruudulla, ja käyttöjärjestelmä tekee varsinaisen laskennallisen ja muun työn.

Taulukossa 4 kieli- ja kulttuurierot on esitetty kootusti eri tasoilla merkityksineen. Siitä nähdään, että käytettävä merkistö on käyttöjärjestelmän vastuulla, samoin ajan mittaaminen ja tiedon varastointi aina tapauksesta riippuen. Ikkunointijärjestelmä taas huolehtii tiedon esittämisestä ruudulla kussakin tapauksessa. Sovellukselle, oli se sitten web-sovellus tai tavallinen sovellus, taas jää syötteiden käsittely ja varsinkin niiden ottaminen. Itse sovelluksen lokalisoitava teksti on myös sovellustasolla. Sovellus eroaa web-sovelluksesta tässä tapauksessa siinä mielessä, että web-sovellusta käytetään selaimella, kun taas normaalia sovellusta käytetään sellaisenaan paikallisesti.

Taulukko 4: Erojen merkitys tietojärjestelmän eri tasoilla.

Lokalisoitava elementti	Sovellus	Käyttöjärjestelmä	Ikkunointijärjestelmä	Web-sovellus
Kieli (sana, lause)	Lokalisoitava merkkijono	Merkistö	Locale	Lokalisoitava merkkijono
Päivämäärä	Syötteen lukeminen oikein	Ajan mittaaminen	Päivämäärien esitysmuodot	Syötteen ottaminen oikein
Aika	Syötteen lukeminen oikein	Ajan mittaaminen	Esitysmuoto	Syötteen ottaminen oikein
Valuutta ja numerot	Valuuttamuunnokset	Laskeminen	Esitysmuoto	Valuuttamuunnokset
Värit ja symbolit	Erottelu ohjelmassa	Resurssien varastointi	Tilannekohtainen valinta	Värien ja symboleiden valinta eri tilanteisiin
Osoitteet ja puhelinnumerot	Syötteen lukeminen oikein	Merkistö	-	Syötteen ottaminen oikein, esitysmuoto
Mittayksiköt	Muunnokset	Merkistö	Esitysmuoto	Muunnokset

4 Tekniset ratkaisut

Tässä luvussa selvitetään lokalisointiin ja kansainvälistämiseen liittyviä seikkoja teknisestä näkökulmasta. Tarkoituksena on tutustua lokalisoinnin ja kansainvälistämisen toteuttamiseen käytännöllisellä tasolla ja käydä läpi merkkien koodaamiseen käytettyjen merkistöjen kehitystä. Luvussa esitetään myös muutamin esimerkein ohjelmointipohjaisia ratkaisuja käyttäen Java-ohjelmointikieltä.

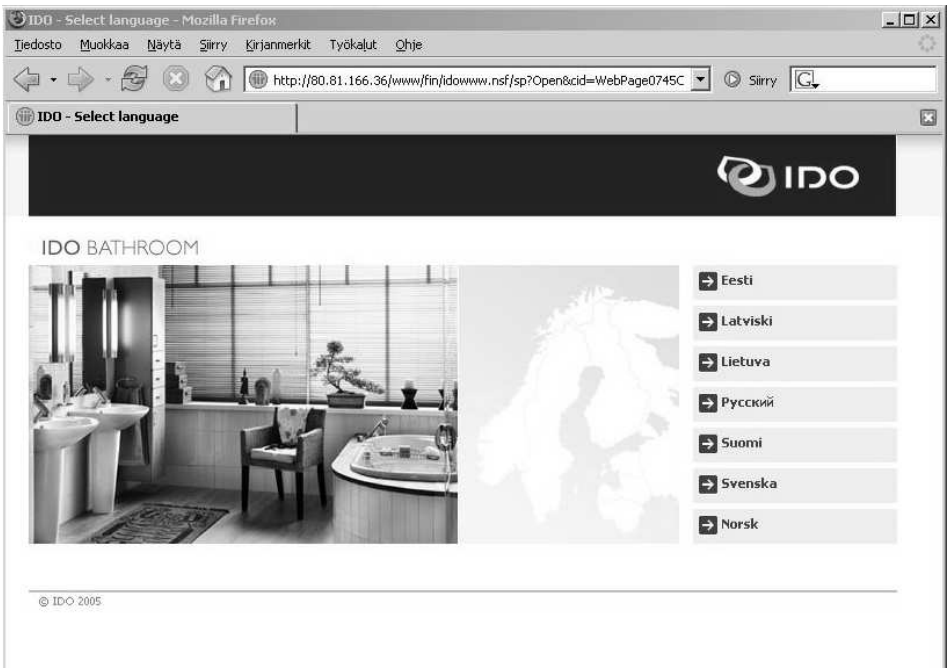
4.1 Kieli- ja kulttuurisidonnaisten asetusten valinta

Jos luodaan sovellusta vaikka yrityksen sisäiseen käyttöön, tiedetään yleensä millä kielellä sitä halutaan käyttää. Tällöin riittää, että tehdään kyseiselle yritykselle yksi lokalisoitu versio sovelluksesta. Sovellus on luonnollisesti järkevää kansainvälistää ja sitten vasta lokalisoida, jotta voidaan tarjota mahdolliselle seuraavalle asiakkaalle erikielinen lokalisoitu versio. Jos asiakas haluaa sovelluksen toimivan kahdella tai useammalla kielellä, voidaan tehdä halutuille kielille lokalisoidut versiot. Toisaalta voidaan tehdä sovelluksesta versio, joka kysyy käynnistettäessä millä kielellä käyttäjä haluaa sovelluksen toimivan ja lataa sitten halutun lokalisoidun version.

Toisaalta Internetissä käyttäjän vierailu Internet-sivustolle alkaa siitä että tämä siirtyy Internet-sivustolle joko kirjoittamalla osoitteen selaimeen, linkin kautta tai hakemalla osoitteen hakukoneella. Tässä vaiheessa täytyy saada selville millä kielellä asiakas haluaa sivuston näkyvän. Yksi tavallinen vaihtoehto on luoda Internet-sivuston ensimmäinen sivu ns. sisäänheittosivuksi. Sisäänheittosivu on eräänlainen portaali lokalisoiduille Internet-sivuille. Se ei yleensä sisällä muuta kuin mahdollisimman yksinkertaisen valikon tai vastaavan, jossa asiakasta pyydetään valitsemaan tälle sopiva kieli annetuista vaihtoehdoista. Eri sivut ovat yleensä eri kielillä tehtyjä versioita samasta alkuperäiskielisestä sivustosta. Koska ei tiedetä mitä kieltä asiakas käyttää, ei sisäänheittosivulla voida käyttää vain yhtä kieltä, kuten englantia. Asiakas ei välttämättä tätä ymmärrä ja saattaa epäonnistua siirtymisessä hänen kielellään luodulle sivustolle.

Sinällään tuntuisi järkevältä laittaa erikielisten sivujen linkit kuviin, esimerkiksi valtioiden, lipuiksi koska tällöin ei tarvitsisi käyttää luonnollista kieltä. Tämä ratkaisu ei kuitenkaan ole hyvä, sillä samoja kieliä puhutaan useassa eri valtiossa. Minkä valtion lippu olisi sopiva esimerkiksi englannin kielelle? Alkujaan kieltä puhuttiin Isossa-

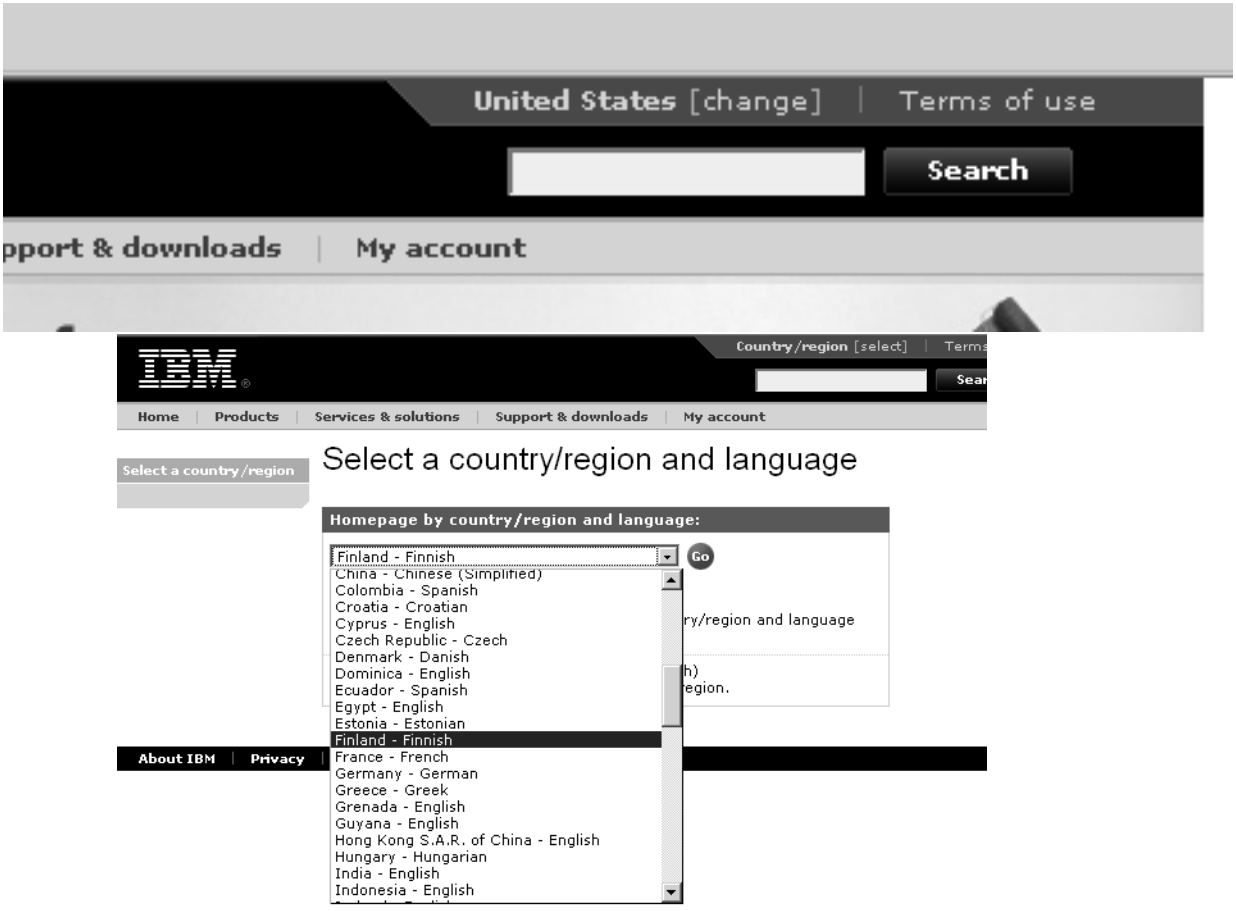
Britanniassa, nykyisin suurin osa puhujista löytyy Yhdysvalloista. Ison-Britannian ja varsinkin Englannin lippu on monelle tuntematon vaikka näin ei ehkä äkkiseltään ajattelisikaan. Parempi ratkaisu on kirjoittaa kullekin eri kielellä toteutetulle sivustolle ohjaava linkki kyseisellä kielellä. Siis suomenkieliselle sivustolle pääsisi esimerkiksi linkistä "Suomeksi"ja ruotsinkieliselle sivustolle linkistä "Svenska". Kuvassa 3 on esimerkki tällaisesta sisäänheittosivusta. Kun asiakas on kerran tehnyt valintansa, voidaan tämän jälkeen koneelle tallentaa keksi (mikäli asiakas on selaimeen sen mahdollistanut), jolloin jatkossa käyttäjä voidaan siirtää sisäänheittosivulta suoraan oikeankieliselle sivulle kysymättä uudelleen. Tässä tilanteessa täytyy muistaa, että asiakas voi vahingossa painaa väärää kieltä, jolloin hänelle täytyy aina antaa mahdollisuus vaihtaa erikieliselle sivulle valinnan jälkeenkin. Tämä kannattaa toteuttaa tekemällä valintamahdollisuus myös lokalisoiduille sivuille.



Kuva 3: Sisäänheittosivu (Ido, 2005)

Toinen vaihtoehto on valita yksi lokalisoiduista sivuista oletussivuksi ja lisätä sivulle mahdollisuus vaihtaa lokalisoituihin versioihin. Järkeväksi tämän ratkaisun tekee tilanne, jossa suurin osa sivustojen asiakkaista haluaisi nähdä sivuston kyseisellä oletuskielellä, ja loput ymmärtävät kieltä sen verran että löytävät kohdan josta saa vaihdettua toisiin sivuihin. Internet-sivujen käyttäjiä voi tutkia sivuston lokitiedostoita analysoimalla. On myös hyvä tutkia lokitiedostoja sisäänheittosivun laatimisen jälkeen. Jos jokin lokalisoitu versio sivustosta on selvästi suositumpi kuin muut, voidaan harkita sen

siirtämistä oletussivuksi ja sisäänheittosivusta luopumista. Kuva 4 näyttää yhden tavan hoitaa kielen vaihtaminen lokalisoidulta sivulta. Esimerkiksi IBM:n osoite `http://www.ibm.com` vie oletuksena yrityksen amerikkalaisille sivuille, joista sitten saa vaihdettua sopivampaan lokalisoituun versioon, jos näin haluaa. Käyttäjän täytyy kuitenkin osata englantia sen verran, että löytää sivuston vaihtoon johtavan linkin ja sitten vielä oman kielensä linkin takaa löytyvästä pudotuslistasta. Näihin tapoihin voidaan yhdistää myös lokalisoitujen sivujen linkittäminen järkeviin osoitteisiin. Hyvänä esimerkkinä toimii vaikkapa `http://www.sivut.fi` ja `http://www.sivut.se` tai `http://www.sivut.com/fi` ja `http://www.sivut.com/se`. Näin asiakas voi suoraan kirjoittaa osoitteen selaimeensa ja päästä heti lokalisoidulle sivulle. Taulukko 5 listaa muutamia maita ja niiden maalyhenteitä.



Kuva 4: Kielen vaihtaminen (IBM, 2005)

Kolmas vaihtoehto asiakkaan käyttämän kielen päättelemiseen on käyttää lokaaleja (locale). Lokaali on eräänlainen tunniste, joka asetetaan selaimeen tai käyttöjärjestelmään. Se sisältää kaksi tietoa, maan ja kielen. Pelkkä maa ei riitä, sillä jo Suomessakin

on kaksi virallista kieltä; suomi ja ruotsi. Esimerkiksi Sveitsissä niitä on neljä, saksa, italia, ranska ja retoromaani. Pelkkä kielikään ei toisaalta riitä, sillä esim. jo eri maiden englannin kielissä on eroja. Maavalinta tuo siis mukanaan kyseiselle maalle ominaiset asetukset kuten valuutan, aika-asetukset ja valittuun kieleen liittyvät erityispiirteet. Kielivalinta taas kertoo halutun kielen ja yhdessä näillä saadaankin tarpeeksi tietoa. Jos lokaali on valittu oikein, voidaan se lukea kun selain lataa sivuston asiakkaan koneelle. Tämän tiedon avulla voidaan suoraan valita asiakkaalle oikea materiaali näkyviin ilman erillistä kielivalinnan tiedustelua. On kuitenkin muistettava, että asiakas voi käyttää esimerkiksi yleistä tietokonetta tai voi olla asettanut lokaalin joksikin muuksi kuin millä oikeasti haluaisi sivun näkyvän. Siksi kannattaakin antaa lokaalejakin käytettäessä asiakkaalle vielä mahdollisuus vaihtaa kieltä ja muistaa tämä asetus.

Käyttäjä voi tulla Internet-sivuille myös hakukoneen kautta. Tällöin käyttäjä siirtyy yleensä suoraan tietylle alisivulle käymättä pääsivun kautta, jossa sopiva kieli voitaisiin valita. Tällaisessa tilanteessa on hyvä antaa käyttäjälle mahdollisuus vaihtaa sivuston kieltä esimerkiksi sivun yläreunassa olevalla kielenvaihtopainikkeella. Tosin käyttäjä saattaa hyvinkin haluta selata juuri kyseistä sivua, joten kieltä ei sinäänsä tarvitsekaan aina vaihtaa.

4.1.1 Lokaalin käyttäminen

Tässä kappaleessa tutustutaan lokaalin käyttämiseen Java -ohjelmointikielessä. Lokaaleja käytetään muissakin ohjelmointikielissä ja tämän kappaleen tarkoitus on lähinnä

Taulukko 5: Maalyhenteitä (Checkdomain.com, 2005)

Maa	Lyhenne
Suomi	fi
Ruotsi	se
Venäjä	ru
Kanada	ca
Espanja	es
Japani	jp
Ranska	fr
Kiina	cn

```
Locale fi = new Locale("fi","FI");
Locale se = new Locale("sv","SE");

SimpleDateFormat sdf =
  new SimpleDateFormat(‘‘dd MMMM yyyy’’,fi);
System.out.println(sdf.format(new java.util.Date()));

sdf = new SimpleDateFormat(‘‘dd MMMM yyyy’’,se);
System.out.println(sdf.format(new java.util.Date()));

DateFormat df =
  DateFormat.getDateInstance(DateFormat.LONG, fi);
System.out.println(df.format(new java.util.Date()));

df = DateFormat.getDateTimeInstance(
  DateFormat.SHORT, DateFormat.LONG, fi);
System.out.println(df.format(new java.util.Date()));
```

Kuva 5: Lokaalin käyttö.

havainnollistaa esimerkkikielen kautta kuinka lokaalien toteutus voidaan hoitaa. Javassa sitä voidaan käyttää ohjelman sisäisesti esimerkiksi muuttujana, joka kertoo mitä asetuksia käytetään. Tämä on järkevää, kun ollaan tekemässä sovellusta esimerkiksi yrityksen sisäiseen käyttöön. Kuva 5 on osa ohjelmasta, jossa käytetään lokaale- ja päivämäärien tulostamiseen eri tavoin. Aluksi luodaan suomen- ja ruotsinkielinen lokaali *fi* ja *se*. Sitten luodaan ja tulostetaan päivämääriä luomalla aluksi olio *sdf* tai *df*. Luotaessa oliota kerrotaan sille missä muodossa päivämäärä tulee tulostaa, esimerkiksi "dd MMMM yyy"tai *DateFormat.LONG* ja käytetävä lokaali (fi tai se). Sitten voidaan kutsua olion metodia *format* joka tulostaa parametrinaan saadun päivämäärän annettujen ohjeiden ja lokaalin mukaan. Tulee huomata, että tulostuslause *System.out.println(sdf.format(new java.util.Date()));*on sama joka kerta, lukuunottamatta muuttujaa *sdf* tai *df*. Taulukko 6 taas näyttää mitä kuvan 5 koodirivit tulostavat.

Web-ympäristössä lokaalien hyödyt tulevat esiin vielä paremmin. Sivujen katseleminen selaimen avulla toimii siten, että selain lähettää sivuja ylläpitävälle palvelimelle pyynnön saada ladata sivut käyttäjälle näkyviin. Tämän pyynnön mukana lähetetään myös lokaalitieto. Käyttäjälle ei yleensä mainita lokaaleista selaimen asetuksia valittaessa, vaan pyydetään lähinnä valitsemaan kieli. Kuvassa 6 näkyy Mozilla Firefox -selaimen kielen valintaan käytetty valikko. Kyseinen selain antaa myös mahdollisuuden kokeilla useampaa lokaalia, jos listan kärkipään lokaaleja ei tueta. Internet-sivuja luotaessa kannattaakin tehdä oletusarvoksi jokin lokaali, jota käytetään jos käyttäjällä on valittuna lokaali, jota ei tueta tai useita lokaaleja joista yhtäkään ei tueta.

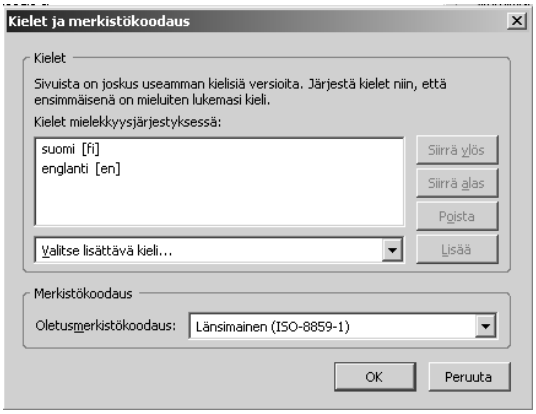
Yleensä selaimissa on jokin lokaali valittuna oletusarvoisesti selaimen lokalisoidusta versiosta riippuen. Kun käyttäjä sitten pyytää Internet-sivua selaimellaan, saadaan valittu lokaali pyynnöstä kuvan 7 koodipätkän osoittamalla tavalla. Javan metodin *doGet* tarkoitus on nimen omaan käsitellä selaimen pyyntö ja tehdä siitä riippuen annettuja käskyjä. Luokan *HttpServletRequest* olio sisältää selaimen palvelimelle lähettämän tiedon kuten valitun lokaalin tai lokaalit. Tämän luokan olion *request* metodi *getLocale* palauttaisi lokaalin. Esimerkissä haetaan kuitenkin suoraan valittu kieli joka onnistuu metodilla *getLocale().getLanguage()*. Tarkemmat määrittelyt Java-kielen metodeille löytyvät sen spesifikaatioista osoitteesta `http://java.sun.com/j2se/1.5.0/docs/api/` (versiolle 1.5.0).

4.2 Kieli- ja kulttuuririippuvaisen materiaalin erottaminen ohjelmakoodista

Kansainvälistämisessä on erittäin tärkeää, että kulttuuririippuvainen materiaali saadaan erotettua ohjelman lähdekoodista. Jos näin ei tehtäisi, olisi uusien kielien lisääminen ja vanhojen muokkaaminen hankalaa. Kun virheilmoitusten ja muiden tulosteiden tekstit on varastoitu vaikkapa erillisiin tiedostoihin lähdekoodista, on niiden kääntäminen

Taulukko 6: Tulostus.

20 helmikuu 2005
20 februari 2005
20. helmikuuta 2005
20.2.2005 11:54



Kuva 6: Lokaalin valinta selaimesta.

```
public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException
    {
...
Locale def = request.getLocale();
...
}
```

Kuva 7: Lokaalin hakeminen.

uusille kielille helppoa, koska ohjelmakoodi ei ole tiellä. Kuitenkin eristettynä kontekstistaan luonnollinen kieli voidaan ymmärtää väärin. Siksi tuleekin varmistaa, että kääntäjät tietävät mihin tarkoitukseen mikin lause on tarkoitettu.

Kielen eristäminen lähdekoodista tapahtuu Javassa resurssinippujen avulla. Resurssinippu on tiedosto, jossa on muuttujia ja niille arvoja. Kuva 8 on esimerkki Javan resurssitiedoston sisällöstä suomeksi. Kuvassa 9 taas on vastaavan resurssinipun englanninkielinen versio. Yhtäsuuruusmerkin vasemmalla puolella on muuttuja, ja sen oikealla puolella taas muuttujalle annettu arvo. Arvo haetaan ohjelmakoodissa resurssinipusta ja saadaan lokaalista riippuvainen versio. Resurssinipusta voitaisiin tehdä myös vaikkapa ruotsinkielinen versio. Tällöin resurssinipusta muuttujan arvon hakeavalle lausekkeelle annettu lokaali päättäisi sen mistä resurssinipusta arvo haetaan. Ruotsinkielinen lokaali valitsisi ruotsinkielisen resurssinipun mikäli sellainen on toteutettu. Resurssiniput ovat siis Javassa joko tekstimuotoisessa resurssitiedostossa, käännettävässä tiedostossa. Toisaalta esimerkiksi Windows-ohjelmoinnissa resursseja käsitellään hieman erilailla. Siellä resurssit ovat ennen kääntämistä resurssitiedostoissa, mutta kääntäessä ne liitetään suoritettavaan, binäärimuotoiseen tiedostoon.

Kuvassa 10 taas on havainnollistettu edellä mainittujen Javan resurssinippujen käyttämistä. Aluksi haetaan käytössä oleva lokaali komennolla *Locale def = request.getLocale();*. Sitten haetaan sopiva resurssinippu komennolla *ResourceBundle rb = ResourceBundle.getBundle("mriikon.terve",def);*, jonka jälkeen tulostetaan resurssinipusta riippuvaisella kielellä selvitys käyttäjän haluamasta kielestä. Lopputulos on joko "Tällä selaimella on valittu kieleksi suomi."tai "You have chosen english for the language for this browser.".

```
tervehdys.alku = Tällä selaimella on valittu kieleksi
tervehdys.kieli = suomi
tervehdys.loppu = .
tervehdys.title = Testiä
```

Kuva 8: Resurssinippu suomeksi.

4.3 Merkistöt

Tässä kappaleessa käydään läpi merkkien koodaamiseen käytettyjen merkistöjen kehitystä niiden nykyiseen olomuotoonsa lokalisoinnin ja kansainvälistämisen kannalta.

4.3.1 Kuudesta bitistä tavuun

Alkujaan tietokoneita käytettiin lähinnä monimutkaisten laskusuoritusten tekemiseen. Tällöin tietokoneen käyttäminen oli hyvin lähellä itse tietokoneen tapaa suorittaa tehtäviä jolloin käyttöliittymä oli melko olematon. Ensimmäiset tietokoneet kehitettiin Yhdysvalloissa, joten on luonnollista, että niitä ei suunniteltu vaikkapa kiinalaisten käytettäviksi. Jos luonnollista kieltä käytettiin, se oli englanti.

Varsinaisten merkkien koodaamiseen tarvittiin kuitenkin keinot (O'Donnel, 1994). Tietokone näkee tiedon binäärimuodossa, joten kirjaimelle ei suoranaisesti ole mitään valmista esitysmuotoa. Tietokoneiden valmistajat käyttivätkin 1950-luvulla hyvin yksinkertaista merkistöä kirjainten esittämiseen. Vain englantilaisen standardiaakkoston kirjaimet A-Z (suurella kirjoitettuna), numerot 0-9 sekä muutama muu merkki olivat tässä merkistössä mukana ja kaikki saatiin mahtumaan kuuteen bittiin per merkki (64 mahdollista erilaista merkkiä). Tällöin ei tunnettu tavun (8 bittiä) käsitettä vaan dataa ajateltiin sanoina, jotka mitattiin bitteinä. Myöhemmin kuitenkin standardiksi muodostui tavun kokoinen merkin esitysmuoto jolloin mahdollisten merkkien määrä nousi 2^8=256:een. Alunperin merkistöjä oli useita, mutta lopulta jäljelle jäi oikeastaan vain kaksi: American Standard Code for Information Interchange (ASCII) ja IBM:n Extended Binary Coded Decimal Interchange Code (U.S. EBCDIC). Molemmat sisältävät englannin kielen aakkoset A-Z (sekä pienet että suuret kirjaimet), numerot 0-9 sekä

```
tervehdys.alku = You have chosen
tervehdys.kieli = english
tervehdys.loppu = for the language for this browser.
tervehdys.title = Testing
```

Kuva 9: Resurssinippu englanniksi.

```
...
Locale def = request.getLocale();
      ResourceBundle rb =
 ResourceBundle.getBundle("mriikon.terve",def);
...
 out.println(rb.getString("tervehdys.alku")
 + rb.getString("tervehdys.kieli")
 + rb.getString("tervehdys.loppu")
 + "<BR>");
...
```

Kuva 10: Resurssinipun käyttäminen.

useita muita merkkejä (! "# $ % & ´ ( ) * + , - . : ; < = > ? @) sekä kontrollimerkkejä tekstinkäsittelyyn kuten LF, NULL, CR (Carriage Return) ja DEL. Eroavaisuuksiakin näiden kahden väliltä löytyy. Kaikki ASCII-merkistön merkit sopivat seitsemään bittiin kun taas U.S. EBCDIC käyttää kaikki 8 bittiä ja merkkien koodausjärjestys on erilainen. ASCII:n ylimääräiselle bitille keksittiinkin kaikenlaisia käyttötarkoituksia, ja pian olikin mahdotonta kertoa mihin sitä sitä oikeasti tuli käyttää.

Koska ASCII suunniteltiin alunperin englannin kirjoittamiseen, oli selvää että ongelmia seurasi kun haluttiin esittää vaikkapa ranskan kieltä tietokoneella (O'Donnel, 1994). Jopa niinkin läheltä Yhdysvaltoja kuin Kanadasta löytyi ranskaa puhuvia tietokoneenkäyttäjiä, joten hankaluuksia ei voitu välttää. Sinällään jos tietokoneita olisi vieläkin käytetty lähinnä laskutoimituksiin, ei englanninkielisyys olisi ollut suuri ongelma. Mutta vuosien mittaan tietokoneet olivat siirtyneet myös liiketoimintaan jolloin halutun kielen käyttö nousi niin suureen arvoon että asialle oli tehtävä jotain. ASCII:lle kehitettiinkin useita vaihtoehtoja ja standardi ISO 646 määrittelee säännöt näiden vaihtoehtojen käyttämiseen. ISO 646 määrittelee tietyt muuttumattomat merkit joita käytetään useassa kielessä, kuten englannin kielen aakkoset A-Z (suurella ja pienellä) sekä muutettavia merkkejä, joita voi vaihtaa kunkin kielen tarpeisiin sopiviksi. Näin saatiin englannin kielessä käytetyt, mutta muissa kielissä yleensä tarpeettomat merkit pois, ja tilalle ko. kielessä käytettyjä merkkejä.

Suomen kieltä ei tuettu vielä tuohon aikaan, eikä suomen kielelle ole olemassa ISO 646:n versiota. Lisäksi monissa muissakin kielissä käytetään merkkejä jotka ovat muunnelmia muista kirjaimista kuten ranskan kielen é ja è. Hankaluuksia tietysti olisi voinut seurata jos halusi käyttää vaihdettuja merkkejä esimerkiksi ohjelmakoodissa, mutta koska monet kääntäjät käyttivät vain puhdasta ASCII:ta, näkivät ne vaihdettujen merkkien kohdalla vain niiden alkuperäiset vastikkeet. Tällöin jos jokin merkki, vaikkapa kaarisulkeet { ja } oli korvattu jollain muilla merkeillä, jotta ko. kielellä voisi kirjoittaa järkevästi, tuli käyttää näitä korvaavia merkkejä kaarisulkeiden tilalla koodissa.

4.3.2 ASCII:n rajoitukset

Kun tietokoneet yleistyivät, ISO 646 ei enää tukenut kaikkia kieliä riittävän tehokkaasti. Mm. kyrilliset aakkostot ja lähi-idän kielet jäivät toteuttamatta, ja ongelmia tuli jos halusi montaa eri kieltä samaan dokumenttiin. Niinpä seuraava kehitysaskel olikin aidosti 8-bittiset kirjaimistot joista suosituimmaksi päätyivät ISO 8859-sarjan kirjaimistot, kuten ISO 8859-1 ja 8859-2. Yhteensä ISO 8859-sarjassa on kirjoitushetkellä 16 merkistöä. ASCII:n suosiosta johtuen, ISO 8859:n sarjan merkistöt tehtiin tukemaan myös ASCII:ta, sen merkit koodattiin samassa järjestyksessä alkuun, ja loppupuolisko käytettiin ylimääristen merkkien sisällyttämiseen. ISO 8859:n lisäksi on olemassa myös muita 8-bittisiä merkistöjä, kuten LTD 37(1619)-1988 (intialainen standardi) ja TIS 620 (thaimaalainen vastaava, sama kuin ISO 8859-11). Koska 8-bittiä ei kuitenkaan riitä esimerkiksi kaikkien Euroopan latinalaista aakkostoa käyttävien kielien merkkien esittämiseen, on ISO 8859:lle vaihtoehtoinenkin ratkaisu. ISO 6937 toimii siten, että jokainen kirjain on kylläkin erillisenä numeroarvona merkistössä, mutta ylä- ja alamerkit kirjaimille ovat yksinään koodattuna. Kirjaimia ja ylä- tai alamerkkejä voi sitten yhdistellä saaden näin eri kielien vaatimia merkkejä kuten meille tutut å, ä ja ö. Haittapuoli tässä ratkaisussa on se, että merkit ovat nyt vaihtelevan pituisia bitteinä laskettuna. Normaali a on yhden tavun kokoinen, mutta ä onkin kahden tavun kokoinen. ISO 6937:n lisäksi on olemassa myös muita 8-bittisiä merkistöjä kuten IBM:n pc850 ja Hewlett-Packardin ROMAN8 tai ARABIC8. Nämä ovat enemmän ja vähemmän verrannollisia ISO 8859:n merkistöihin.

4.3.3 Monitavuiset merkistöt

Koska ASCII, ISO 8859-sarja, tai juuri mikään muukaan latinalaista aakkostoa käyttävää kieltä tukeva merkistö ei tukenut mm. ideograafisia merkkejä, täytyi näitä varten kehittää oma merkistönsä (Dr. International, 2003). Kahdeksan bittiä ei muutenkaan millään riittäisi esimerkiksi kiinan kielen tuhansiin merkkeihin, jotka pitäisi teoriassa kaikki koodata merkistöön. Merkkiä kohden tarvitaan siis enemmän tilaa, ja tavallinen ratkaisu onkin käyttää kaksi tavua yhden sijaan. Näin saadaan huomattavasti enemmän tilaa merkkien esittämiseen.

Osa näiden kielien merkeistä kuitenkin sopii yhteen tavuun, kuten japanin kielen katakanan merkit. Siksipä merkit ovatkin vaihtelevan pituisia, joka hankaloittaa tilannetta. Kun tekstiä tutkitaan tavu tavulta, täytyy erottaa onko luettu tavu itsenäinen merkki vai kaksitavuisen merkin ensimmäinen tavu. Osa tavun 256:sta merkistä on varattu erityisesti kaksitavuisten merkkien ensimmäiseksi tavuksi, ja tietty osa ei voi olla jälkimmäinen tavu. Tästä johtuen merkkejä voidaan ilmaista maksimissaan vain noin 16 000, teoreettisen maksimin 2^{16}=65 536 sijaan. Tämä on kuitenkin huomattavasti enemmän kuin tavuun mahtuva 256 merkkiä. Ensimmäinen tällainen merkistö oli japanilainen JIS X 0208, sen uusin versio kirjoitushetkellä on JIS X 0212. Muita ovat mm. Big-5 (perinteinen kiina, merkistö on käytössä yleisimmin Taiwanissa), Shift-JIS (Microsoftin versio Japanin JIS X 0208 -merkistöstä), GB2312 (Kiinan merkit yksinkertaistetulle kiinalle) ja KS C 5601 (Korea).

4.3.4 Unicode

Unicode on tällä hetkellä ainoa vaihtoehto kaikkien maailman kielien ja aakkostojen tukemiseen (Unicode Inc., 2005). Ennen sitä pelkästään Euroopassa tarvittiin useita eri merkistöjä, esimerkiksi ISO 8859 -sarjan eri versioita. Merkistöt eivät kuitenkaan toimineet yhdessä kovin hyvin, ja eri kielten yhdistely oli monimutkaista. Ajatus olikin luoda yksi merkistö, joka tukisi jokaista kieltä antamalla jokaiselle merkille oman numeronsa. Näin voitaisiin kirjoittaa siis mitä hyvänsä kieltä tai kieliä samaan dokumenttiin ilman turhia hankaluuksia merkistöjen kanssa. Ennen vuotta 1991 tähän oli kaksi erilaista ratkaisua, Unicode ja ISO DIS 10646. Näistä yhdistettiin ISO/IEC 10646-1 (tunnetaan myös nimellä ISO 10646) joka muistutti kuitenkin hyvin paljon alkuperäistä Unicodea. Unicode standardi on yhtäläinen ISO/IEC 10646-1:n kanssa ja

myös virallinen tapa toteuttaa ko. ISO -standardi. Sitä pitää yllä taloudellista hyötyä tavoittelematon konsortio johon kuuluvat mm. Apple, HP, IBM ja Microsoft.

Unicodesta on olemassa muutamia eri versiota: UTF-8, UTF-16 ja UTF-32. UTF-8 on tarkoitettu lähinnä ASCII:ta enemmän ja vähemmän käyttäviä järjestelmiä varten, kuten HTML:ää ja sähköpostia. UTF-16 sisältää kahden tavun kokoisia merkkejä, jolloin saadaan koko teoreettinen 16 bitin 65 536:n merkin maksimi käyttöön. UTF-32 taas käyttää 32 bittiä jokaista merkkiä kohden, jolloin tilaa kuluu enemmän, mutta merkkejä voidaan esittää useampia. Unicode sisältääkin niin latinalaisen aakkoston merkkit kuin ideograafiset merkit eri kielistä, numerot, Lähi-idän oikealta vasemmalle kirjoitettujen kielien merkit, teknisiä symboleja, matemaattisia merkkejä jne. Tilaakin on vielä kirjoitushetkellä jäljellä. Unicoden versio kirjoitushetkellä on 4.1.0 (julkaistu 31.3.2005, Unicode Inc., 2005). Todella kansainväliseen ohjelmointiin Unicode on erittäin hyvä ratkaisu, sillä se poistaa koodisettien väliset ongelmat ja mahdollistaa minkä hyvänsä kielen kirjoittamisen ilman sen suurempia ongelmia.

5 Lokalisointiprosessi

Tässä luvussa tutustutaan tapoihin hoitaa itse ohjelmiston lokalisointiprosessi hallinnollisella tasolla. Luvussa käydään läpi sekä yleisesti sovelluksen lokalisointi että web-sovelluksen, kuten Internet-sivuston, lokalisointi.

5.1 Yleinen lokalisointi

Tämän kappaleen toimintamalli perustuu pääosin Dr. Internationalin (2003) kirjaan. Kappaleessa käydään yleisesti ottaen läpi asiat, jotka tulee ottaa huomioon lokalisoitaessa ohjelmistosovellusta eri tavoilla, jotka käydään läpi. Esitetty lähestymistapa on esimerkinomainen yksi vaihtoehto toteuttaa sovelluksen lokalisointiprosessi.

5.1.1 Osa-alueet

Kun lokalisointityö alkaa, on olemassa alkuperäinen sovellus, joka on laadittu jollakin luonnollisella kielellä, esimerkiksi suomeksi. Vaikka lokalisointi olisi tiedossa ennen sovelluksen tekemisen aloittamista, tehdään silti aluksi vastaavanlainen alkuperäinen sovellus. Tämän sovelluksen tulee olla kansainvälistetty, jolloin lokalisointi helpottuu huomattavasti, kuten jo aiemmin on todettu. Tästä vaiheesta alkaa sovelluksen lokalisointi.

Tekstiä löytyy lähes jokaisesta sovelluksesta ja sen kääntäminen on olennaisin osa tuotteen lokalisoinnissa. Tekstin tulee olla yksiselitteistä, ja huumoria sekä kielellä leikkimistä tulee välttää. Vitsejä ja muuta vastaavaa on hankalaa, ellei jopa mahdotonta, kääntää toisille kielille (Russo ja Boor, 1993). Lisäksi kääntäessä tulee ottaa huomioon kohdekäyttäjän kulttuuri. Varsinaisen ohjelman kielen läpikäymisen lisäksi tulee ottaa myös huomioon aputiedostot, oheismateriaali ja tuotteen paketointi kuten CD-kotelo. Tekstin kääntämisessä, ja alkuperäisen tekstin kirjoittamisessa, tulee myös kiinnittää huomiota yhtenäisyyteen. Sama asia voidaan usein sanoa useammalla kuin yhdellä tavalla, ja jos sovelluksen käyttöoppaassa ja itse sovelluksessa käytetään eri sanoja samasta asiasta, voi käyttäjä mennä sekaisin. Joskus myös joissain kielissä käytetään eri maissa eri sanaa jollekin asialle. Tällöin tulee valita mahdollisimman universaali sana käytössä olevista vaihtoehdoista.

Tekstiin liittyy myös näppäinoikoteiden valinta. Tietyt yleisesti hyväksytyt yhdistelmät kuten ctrl + c tai ctrl + v kannattaa jättää sikseen vaikka niiden kirjaimet tulevatkin englannin kielen sanoista copy ja verbose. Kuitenkin esimerkiksi valikkojen pikanäppäimet muuttuvat kun valikko käännetään toiselle kielelle. Ideografisissa kielissä taas ei voida käyttää yhden näppäimen oikoteitä, tällöin tulee säilyttää alkuperäiset näppäinoikotiet.

Sovelluksen käyttöliittymän asettelu vaatii huomiota. Merkkijonon pituus muuttuu kun se käännetään. Muutoksen suuruus riippuu lähde- ja kohdekielestä. Esimerkiksi englanniksi jonkin toiminnon hyväksymiseen käytettävän näppäimen teksti on "Ok", suomen kielessä se on "Hyväksy". Taulussa 7 on muutamia termejä, joita käytetään usein käyttöliittymissä ja niiden koon muuttuminen käänettäessä suomesta englanniksi. Koon muuttuminen tuleekin ottaa huomioon jo alkuperäistä sovellusta laatiessa, jotta vältytään ongelmilta lokalisoitaessa. Muutokset käyttöliittymään täytyy kuitenkin tehdä mikäli niiltä ei voida välttyä. Tekstin kirjoitussuunnan muuttuminenkin vaikuttaa käyttöliittymään, jolloin se täytyy peilata. Peilaaminen tarkoittaa sananmukaisesti käyttöliittymän peilaamista sivusuunnassa. Tällöin oikealta vasemmalle kirjoitettava teksti sopii uuteen käyttöliittymään järkevästi. Myös kirjasinlajin koko voi vaikuttaa käyttöliittymän asetteluun. Esimerkiksi monien Aasian maiden ideografiset kirjasinlajit vaativat huomattavasti isomman tilan pystysuunnassa kuin latinalaista aakkostoa käyttävät kielet. Koska kirjoitussuuntia on useita, voi teksti viedä enemmän tilaa muutoinkin kuin leveyssuunnassa. Jos esimerkiksi vasemmalta oikealle ja ylhäältä alas kirjoitettu teksti korvataan ylhäältä alas ja oikealta vasemmalle kirjoitetulla tekstillä, se vie täysin eri tavalla tilaa käännettynä.

Taulukko 7: Sanojen pituuden muuttuminen.

Englanniksi	Suomeksi	Prosentuaalinen muutos
Ok	Hyväksy	71%
Cancel	Peruuta	14%
Next	Seuraava	50%
File	Tiedosto	50%
Save	Tallenna	50%
Close	Sulje	0%

Vaikka tekstin sisällyttämistä grafiikoihin tuleekin välttää, on ne silti käytävä läpi. Gra-

fiikoissa voi olla esimerkiksi symboleja, värejä tai henkilöhahmoja jotka eivät toimi samalla tavalla kohdekulttuurissa kuin lähdekulttuurissa (Russo ja Boor, 1993). Samoin jos käytetään musiikkia jonkun asian ilmaisemiseen, voi joku musiikki olla toisessa kulttuurissa täysin tuntematon, ja toisessa se yhdistetään heti haluttuun asiaan. Varsinaisen ohjelman lisäksi tulee huomioida myös ohjelmistosovelluksen oheismateriaali, kuten CD- tai DVD-levyn kotelo ja ohjekirja. Näistä löytyy kuvia, tekstiä ja mahdollisesti muitakin tässä luvussa mainittuja läpikäytäviä seikkoja.

5.1.2 Lokalisointiprosessi

Dr. International (2003) suosittelee, että ohjelmistosovelluksen lokalisoinnin tulisi olla osa sen kehitysprosessia. Kun tuotteen toiminnallisuus on vakaa, tulee varmistaa, että se voidaan lokalisoida järkevästi. Tuote on vakaa toiminnallisuudeltaan kun se on pääosin toteutettu eikä siihen enää ole tulossa muutoksia, jolleivat ne ole kriittisiä. Lokalisointi tulee aloittaa heti kun sovelluksen toiminnallisuus on vakaa, jolloin ehditään vielä tehdä muutoksia lähdekoodiin mikäli lokalisointi sitä vaatii, viivästyttämättä kuitenkaan alkuperäisen tuotteen julkaisupäivämäärää. Jos on tarkoitus lokalisoida usealle eri kielelle, kannattaa yksi kielistä valita ns. pilottikieleksi.

Pilottikielen tarkoitus on toimia esimerkkikielenä, joka toteutetaan ensiksi. Pilottikieleksi kannattaa valita jokin helposti lähdekielestä käännettävä kieli, jolloin voidaan keskittyä itse lokalisoinnin tuomiin ongelmiin. Kun tämä versio on toteutettu, voidaan loput tehdä keskittyen vain kielellisiin yksityiskohtiin koska itse lokalisointi on suurimmilta osin tehty. Tällöin lokalisointi muille kielille tulisi aloittaa kun tuotteen käyttöliittymä on vakaa.

Kaikkia muutoksia tulee välttää alkuperäiseen versioon kun lokalisointi on aloitettu, varsinkin jos ne eivät paranna tuotetta merkittävästi. Jokainen muutos täytyy ottaa huomioon jokaisessa lokalisoitavassa versiossa jolloin niiden julkaisu voi viivästyä. Tuotteen käyttöliittymä tulee myös jäädyttää ennen sen julkaisua. Tämä tarkoittaa sitä, että siihen ei enää saa tehdä muutoksia. Kun käyttöliittymä on jäädytetty, lokalisointiryhmä tietää että heillä on käsissään varmasti lopullinen versio, jolloin voidaan keskittyä sen lokalisointiin.

5.1.3 Lokalisointityökalut

Kannattaa valita oikea työkalu työn alla olevaan tehtävään. Lokalisointia voidaan tehdä joko niin, että muokataan lähdekoodia ennen kuin se on käännetty tai sitten kääntämisen jälkeen. Suurin osa lokalisointityökaluista tukee kumpaakin tapaa, ja ne helpottavat muutenkin työn määrää hallinnoimalla eri versioita automaattisesti. Työkalun valinta sitten taas riippuu käytössä olevasta ohjelmointikielestä ja resurssitiedostoista. Lähes kaikille yleisille ohjelmointikielille ja resurssitiedostoille, joilla lokalisointia voi tehdä, on olemassa apuvälineitä. Lokalisointityökalulla tulisi myös pystyä pitämään yllä sanastoa, jonne tallennetaan jokainen käännetty lause. Kun sitten käsitellään uutta tekstiä, työkalu voi tarkistaa onko kyseisiä lauseita jo käännetty. Jos näin on, voidaan käännös ladata muistista sen sijaan että se täytyisi kääntää uudelleen. Esimerkkeinä tällaisista lokalisointityökaluista käyvät Multilizer (`http://www.multilizer.com`) ja Passolo (`http://www.passolo.com`).

5.1.4 Lokalisointityöryhmä

Dr. International (2003) suosittelee, että lokalisointityöryhmä koostuisi lokalisointijohtajasta, yhdestä tai useammasta lokalisointi-insinööristä, ja useasta lokalisoijasta. Johtaja hoitaa lokalisointitiimin ja kehitystiimin välisen kommunikoinnin ja pitää huolta siitä, että lokalisoitavaksi tuleva lähdekoodi on kansainvälistetty sovitulla tavalla. Lokalisointi-insinöörit taas hoitavat ohjelmakoodiin tehtävät muutokset, jotka johtuvat lokalisoinnista. He myös auttavat lokalisoijia ymmärtämään resurssien liittymisen itse ohjelmaan.

Lokalisoijat ovat kääntäjiä, jotka myös tietävät taustat itse sovelluksesta. He muuttavat järjestelmän käyttöliittymää jos tarve vaatii, lokalisoivat grafiikat ja kääntävät tekstit. Heidän täytyy osata sekä ohjelmointia, että omata kielitaidot kohde- ja lähdekielestä. Kääntämisessä tulee muistaa, että oikeellisen käännöksen lisäksi täytyy valita käännöksistä tilanteeseen sopivin. Tässä avuksi tulevat lokalisointi-insinöörit, jotka tietävät tarkalleen mitä mikin merkkijono tarkoittaa ja mihin se liittyy. Lokalisoijat hoitavat myös tekemiensä käännöksien ja muutoksien sekä koko lokalisoidun ohjelmistoversion testaamisen.

5.2 Web-lokalisointi

Tämä ratkaisumalli on esitelty Yunkerin (2002) kirjassa. Teos keskittyy erityisesti Internet-sivujen lokalisointiin ja Internetissä tapahtuvaan monikansalliseen kaupankäyntiin. Malli onkin erityisesti suunniteltu Internet-sivustojen lokalisointiin. Kappaleessa käydään läpi lokalisointiprosessin viisi vaihetta: valmistelu, kansainvälistäminen, lokalisointi, sisällön hallinta ja arviointi. Tämän jälkeen tutustutaan Yunkerin (2002) ehdotukseen lokalisointityöryhmän kokoonpanosta.

5.2.1 Valmistelu

Kun Internet-sivusto päätetään tehdä tukemaan useita eri kieli- ja maa-asetuksia, valitaan aluksi halutut asetukset. Sinällään ei ole merkitystä, onko ohjelmisto jo olemassa vai vasta suunnitteilla. Työ luonnollisesti helpottuu jos tehdään uusi tuote, koska silloin ei tarvitse korjailla jo tehtyä työtä. Halutut kieli- ja maa-asetukset kannattaa valita ajatellen kustannuksia ja tuottoa. Jos esimerkiksi tuote on suomalainen ja siitä voitaisiin tehdä joko ruotsalainen tai japanilainen arvioidun tuoton ollessa sama, niin kannattaisi valita kustannuksiltaan edullisempi vaihtoehto. Kuitenkin jos yrityksestä löytyy ruotsin kielen osaajia jotka voivat tehdä kääntämistyön, olisi ruotsin kielelle lokalisointi halvempaa ja siten siis kannattavampaa. Kun halutut asetukset on päätetty, tulee varata sopivat Internet-osoitteet uusilta markkina-alueilta jos kyseessä on sovellus jota on tarkoitus mainostaa. Jos yritys on vaikkapa Suomalainen ja sen kotisivut ovat .fi -päätteiset ja siirrytään Japanin markkinoille tulee varata .jp -päätteinen tunnus. Tässä tilanteessa on myös hyvä varata maa-alueeton tunnus jonne voidaan tehdä sisäänheittosivu josta käyttäjät ohjataan sopiville lokalisoiduille sivuille. Valmistelujen yhteydessä on myös syytä laatia aikataulu, jakaa vastuut ja tehtävät ja suunnitella budjetti.

Valmisteluvaiheeseen Yunker (2002) esittää seuraavanlaisen tarkastuslistan.

- Arvioi yrityksen valmius siirtyä kansainväliseksi.
- Valitse kohdealue.
- Testaa yhtiön sekä tuotteiden nimet, tai luo uudet.
- Rekisteröi maakohtaiset web-osoitteet.

- Rekisteröi monikieliset web-osoitteet.
- Arvioi projektin laajuus, mukaanlukien sanojen sekä lokalisoitavien kuvien määrä.
- Valitse tai palkkaa projektin johtaja.
- Suunnittele budjetti.
- Aseta aikataulu.

5.2.2 Web-sovellusten kansainvälistäminen

Valmistelujen jälkeen voidaan siirtyä varsinaisen työn pariin. Kansainvälistäminen suoritetaan ennen lokalisointia, ja siinä täytyy painottaa huomiota varsinkin luvussa 3 esitettyihin seikkoihin kuten kalenteriin, valuuttaan ja kellonaikoihin. Nämä kaikki täytyy voida saada kullekin lokalisoidulle versiolle sopiviksi. Web-sovelluksen kansainvälistäminen ei sinällään eroa paljoa yleisesti ottaen sovelluksen kansainvälistämisestä, sillä se ei näy käyttäjälle suoranaisesti. Varsinainen ero näkyykin lokalisointivaiheessa.

Kansainvälistämiseen Yunker (2002) esittää seuraavanlaisen tarkastuslistan.

- Eristä ja muokkaa kaikki ohjelmiston toiminnallisuus, joka vaatii lokalisointia.
- Muokkaa rahan esitysasua niin, että se toimii erilaisilla valuutoilla.
- Tee hakeminen mahdolliseksi eri kielillä.
- Tue uusia aakkostamisjärjestyksiä.
- Muokkaa tietokantaa niin että se tukee uusia merkkejä ja merkkijonojen pituuksia.
- Suunnittele käyttöliittymä niin, että se on sekä kansainvälisesti hyvä, että helposti lokalisoitavissa.
- Poista tarpeettomat grafiikat ja vastaavat jotka voivat pidentää latausaikoja.
- Eristä kaikki merkkijonot, jotka vaativat kääntämistä web-skripteistä ja funktioista, ja siirrä ne yhteen resurssitiedostoon.

- Kirjoita alkuperäinen teksti uudelleen niin, että se on helpompi kääntää.
- Kehitä tyyliopas.
- Tee sanasto termeistä.
- Suunnittele nimeämisjärjestelmä lokalisoiduille web-tiedostoille ja organisointijärjestys lokalisoiduille hakemistoille.
- Valmistele lähdetiedostot, tekstit, grafiikat ja skriptit.

5.2.3 Lokalisointi

Kun alkuperäinen versio sovelluksesta on kansainvälistetty, voidaan aloittaa lokalisoitujen versioiden laatiminen. Tässä vaiheessa täytyy kaikki sovelluksessa käytetyt tekstit, jotka näkyvät käyttäjälle kääntää halutulle uudelle kielelle. Jos yrityksestä löytyy kohdekielen taitajia, voidaan käännöstyö tehdä talon sisälläkin. Isoilla yrityksillä tämä on helpompaa, pienemmillä yrityksillä todennäköisyys on vähäisempi.

Eräs mahdollinen vaihtoehto on ulkoistaa käännöstyö jollekin toiselle yritykselle. Tämä on kalliimpi ratkaisu, mutta vähentää projektin yrityksen sisäistä työmäärää ja takaa hyvän lopputuloksen, mikäli kääntäjäyritys on valittu huolella ja materiaali on toimitettu heille kokonaisuudessaan. Sovelluksen kommentit kannattaa myös kääntää uudelle kielelle, että sitä on helpompi tulkita ja muokata lähdekooditasolla.

Varsinaisen tekstin lisäksi myös kaikki sovelluksen grafiikat täytyy käydä läpi. Jos niissä on tekstiä, se täytyy kääntää. Jos käännöstyö tehdään ulkoisesti, täytyy kääntäjäyritykselle toimittaa kuvien lähdetiedostot. Kuvien lokalisointi on huomattavasti kalliimpaa kuin normaalin tekstin, koska kuvia on vaikeampi käsitellä ja täytyy huolehtia siitä, että uusi teksti myös sopii kuvaan. Lisäksi kuvat täytyy käydä läpi grafiikoidenkin osalta. Tästä syystä ainakin tekstiä kuvissa tulisi välttää lokalisointikustannusten pienentämiseksi. Jotkut kuvat, symbolit tai värit voivat joissain maissa olla loukkaavia.

Kääntämisessä täytyy kiinnittää kielen lisäksi huomiota myös kohdemaahan. Pelkkä tekstin kääntäminen englanniksi ei ole riittävää varsinkin jos kyse on Internet-sivusta tai ylipäätänsä tekstistä, joka on tarkoitettu markkinointiin. Tekstin voi kääntää usealla

eri tavalla, jotka kaikki voivat olla teknisesti oikeita. Näistä käännöksistä tulee valita aina tilanteeseen sopiva oikea. Ohjelmiston kehittäjän tulisikin luoda hyvät suhteet kohdealueen käyttäjiin (Russo ja Boor, 1993). Jos käytetään ulkoista kääntäjää kannattaa varmistaa ennen työn aloittamista mistä on kyse että lopputulos tyydyttää varmasti molempia osapuolia.

Kun varsinainen lokalisointi on tehty, pitää lopputulos vielä editoida, eli tarkistaa. Editoijana kannattaa käyttää kohdekieltä äidinkielenään puhuvaa henkilöä tai toista kääntäjää. Editoija kannattaa valita muusta kuin käännöksen tehneestä yrityksestä (jos käännös tehtiin ulkoisesti), ulkoisesta yrityksestä tai eri työryhmästä yrityksen sisällä (jos käännös tehtiin sisäisesti).

Yunker (2002) esittelee kirjassaan erilaisia lokalisointitapoja, joita voidaan Internet-sivuihin soveltaa. Vaihtoehdot ovat seuraavat:

- *Kattava lokalisointi:* Nimensä mukaisesti periaatteessa täydellinen lokalisointi. Kaikki alkuperäisen sivun materiaali lokalisoidaan.
- *Lisäävä lokalisointi:* Tässä ajatuksena on lisätä materiaalia eri lokalisoiduille versioille pikkuhiljaa, eikä suinkaan kaikkea kerralla. Lopputulos voi kuitenkin olla sama kuin kattavassa lokalisoinnissa.
- *Räätälöity lokalisointi:* Lokalisointiin liittyy täysi sivuston uudelleensuunnittelu. Tämä on tarpeen jos alkuperäistä sivua ei oikein mitenkään voida lokalisoida sellaisenaan vaan vaaditaan uusi lähestymistapa.

Lokalisointia varten taas käytetään seuraavanlaista tarkastuslistaa.

- Kääntäminen:
  - Testaa kääntäjiä.
  - Luo kääntämisen työnkulku.
  - Toteuta mahdolliset työkalut kääntämisen helpottamiseksi.
  - Luo yksityiskohtaiset ohjeet kääntäjille.
- Editointi:
  - Testaa editoijia.

  - Luo yksityiskohtaiset ohjeet editoijille.

- Grafiikoiden ja suunnittelun lokalisointi:
  - Suunnittele tekstin asettelu kulttuurille sopivaksi ja tehokkaaksi
  - Suunnittele sisäänheittosivu.
  - Testaa lokalisoidun sivuston käytettävyys.
- Arvioituta käännös kohdealueella.
- Kokoa Internet-sivut ja aseta ne valmiiksi testausta varten.
- Toiminnallisuuden ja käytettävyyden testaus yhtiön sisällä sekä kohdealueella.

5.2.4 Sisällön hallinta

Kun tuote on kansainvälistetty ja lokalisoitu, tarvitaan valmiudet tuotteen päivitykseen ja erillisten lokalisoitujen versioiden hallintaan. Jos alkuperäiseen (tai mihin hyvänsä lokalisoituun versioon) tehdään jotain teknisiä muutoksia, täytyy ne saada myös muihin versioihin mikäli tarve näin vaatii. Myöskin jos tuotteisiin halutaan lisätä uutta sisältöä, kuten Internet-sivuille uusia tuotteita myyntiin tai näytille, on hyvä olla järjestelmä, jolla muutokset saadaan helposti jokaiseen eri versioon. Yrityksen sisällä on myös tärkeää saada kaikki käyttämään samaa järjestelmää niin sisällön hallintaan kuin muuhunkin ylläpitoon.

Tässä vaiheessa on myös hyvä luoda käännösmuisti, eli tietokanta jossa on jokainen jo kertaalleen käännetty lause. Käännösmuistiin kirjataan siis jokainen kullekin kielelle käännetty termi tai kokonainen lause. Yksittäisia sanojahan ei pysty sinänsä kääntämään aina samalla tavalla toiselle kielelle, mutta kokonainen lause kääntyy joka kerta kutakuinkin samoin. Käännösmuistin avulla voidaan uutta sisältöä luodessa tarkistaa, josko joku uusista lauseista tai termeistä on jo käännetty. Tällöin voidaan käännös suorittaa heti. Näin varsinaiselle kääntäjälle annetaan vain uudet, kääntämättömät osat. Tässä vaiheessa on myös hyvä aloittaa tuotteen markkinointi sen kohdealueella. Tämä voidaan hoitaa esimerkiksi Internet-sivuilla mainosviirien avulla tai mainonnalla muissa medioissa kuten televisiossa.

Sisällönhallinnassa taas tulisi käyttää seuraavanlaista tarkastuslistaa.

- Kehitä sisällönhallintajärjestelmä:
  - Kehitä työjärjestys, jota kaikkien osallistujien tulee käyttää.
  - Mikäli mahdollista, osta ohjelmisto mahdollisimman monen tehtävän automatisoimiseksi.
- Toteuta käännösmuistiohjelmisto:
  - Varastoi kaikki tehdyt käännökset käännösmuistiin.
  - Varmista että muistia pystyy jatkossa hyödyntämään.
- Kouluta henkilökunta tukemaan lokalisoituja Internet-sivuja ja kansainvälisiä asiakkaita.
- Aloita jatkuva ylläpito:
  - Lokalisoi kaikki uusi sisältö.
  - Käytä käännösmuistia.
  - Kääntäminen.
  - Editointi.
  - Grafiikat.
  - Valmistelu ja testaaminen.
  - Toteuta muutokset kaikilla tai valituilla lokalisoiduilla sivustoilla.
  - Varmista että käännösmuistia voivat käyttää kaikki aliurakoijat mikäli niitä käytetään.
- Tue asiakkaita (puhelin, sähköposti ja faksi)
- Mainosta lokalisoituja Internet-sivustoja:
  - Rekisteröi sivut paikallisille ja kansainvälisille hakukoneille.
  - Mainosta kohdealueella.
  - Lähetä lehdistöilmoituksia paikalliselle medialle.

5.2.5 Arviointi

Viimeiseksi tulee suorittaa tehdyn työn arviointi. Kun tuote on ollut asiakkaan käytössä jo jonkun aikaa ja ylläpitoa on tehty tarpeen vaatiessa, on hyvä aika katsoa kuinka uusi tuote on parantanut myyntiä kyseisellä alueella ja onko mainonta parantanut tuotteen tunnettavuutta. Jos kyseessä on Internet-sivusto, on hyvä myös tarkistaa sivun lokitiedostot. Niistä näkee sivun liikenteen määrän ja yhteydenottojen laadun. Jos parannuksia tarvitaan, on tässä vaiheessa hyvä miettiä niiden toteuttamista.

Viimeisessä vaiheessa käytössä oleva tarkastuslista taas on seuraavanlainen.

- Analysoi myyntejä markkina-alueittain.
- Testaa tuotteiden tunnettavuutta.
- Testaa sivuston käytettävyys.
- Analysoi sivuston lokitiedostoja maittain.
- Laske sijoituksesta saatu voitto.
- Arvioi kilpailevia sivustoja.
- Suorita arviointeja käännöksen onnistumisesta.
- Suunnittele parannuksia kaikille lokalisoiduille sivustoille tuotteiden ja palvelujen osalta.
- Suunnittele uusia tai paranneltuja tuotteita.
- Suunnittele seuraavan sukupolven sivustoa.

5.2.6 Sivujen sukupolvet

Yunker (2002) esittelee myös tavan toteuttaa Internet-sivujen lokalisointia sukupolvissa. Tämä tarkoittaa sitä, että lähdettäessä liikkeelle alkuperäiskielisestä sivustosta tähdätään useaan muuhun lokalisoituun versioon. Tällöin ei kaikkia kannata tehdä heti, vaan aluksi tärkeimmät, helpoimmat tai molemmat. Hän esittelee myös esimerkin lokalisoitavan Internet-sivuston sukupolvista.

- *Ensimmäinen sukupolvi:* Lokalisointi kolmelle eri kielelle, tarjolla vain osa myytävistä tuotteista tai palveluista. Tarjotaan vain rajattu tuki asiakkaille, esimerkiksi tarjotaan puhelintukea ja sähköpostitukea vain alkuperäisellä kielellä ja kerrotaan tästä Internet-sivuilla.

- *Toinen sukupolvi:* Lisätään kaksi uutta kieltä, lisätään käytettävyyttä esimerkiksi lokalisoidulla hakumahdollisuudella. Tarjotaan sähköpostitukea myös lokalisoiduilla kielillä.

- *Kolmas sukupolvi:* Lisätään järjestelmä, jolla voidaan hallita kaikkien lokalisoitujen versioiden sisältöä. Lisätään maksuvaihtoehtoja ja valuutanmuunnin, joka muuntaa valuutat jatkuvasti oikeiksi. Järjestetään myös puhelintuki lokalisoiduille kielille.

Yksityiskohdat eivät tässä ole niin tärkeitä kuin perusajatus: kun lähdetään pienestä liikkeelle ei haukata kerralla liian suurta palaa. Alussa voidaan suurin osa työstä ulkoistaa ja samalla kouluttaa omaa työvoimaa ylläpitoon ja lopulta myös lokalisointiin.

5.2.7 Lokalisointityöryhmä

Yunker (2002) havainnollistaa myös lokalisointiin tarvittavan ryhmän kokoonpanoa. Ensimmäistä uutta kieltä varten tarvitaan kansainvälistämisen ja lokalisoinnin johtaja, kääntäjä, editoija, ohjelmoija, suunnittelija ja arvioija. Kansainvälistämisen ja lokalisoinnin johtaja hallinnoi ryhmää ja pitää huolta aikatauluista. Kääntäjä hoitaa varsinaisen luonnollisen kielen kääntämisen lähdekieleltä kohdekielelle. Editoija oikolukee ja tarkistaa lopullisen tuotteen. Ohjelmoija taas hoitaa varsinaiseen lähdekoodiin liittyvät asiat niin lokalisoinnin kuin kansainvälistämisenkin osalta. Arvioija taas arvioi kunkin työvaiheen sujumisen.

Jos halutaan lisätä useampi kieli, tarvitaan jokaista kohden uusi kääntäjä, editoija ja arvioija. Siis yhdellä kielellä tämä tekee 6 henkeä, kahdella 9, kolmella 12 ja niin edelleen. Työntekijöiltä ei sinällään vaadita mitään erikoisosaamista, kunhan suunnittelija ja ohjelmoija ovat yhtiön sisältä, jolloin he yleensä ovat tutustuneet Internet-sivuihin jo entuudestaan. Tällainen työryhmä toimiikin yrityksen sisällä, ja raportoi johtajan kautta muulle yritykselle.

Toinen vaihtoehto on ulkoistaa vain kääntäminen tai lokalisointi kokonaan. Kääntämisen tapauksessa tilanne on melko selvä, sillä yrityksessä ei tarvitse tällöin olla kohdekieltä osaavia työntekijöitä eikä kääntäjiä tarvitse yksitellen palkata. Jos koko lokalisointi ulkoistetaan, tarvitaan silti lokalisoinnin johtaja, joka kommunikoi ulkoisen yrityksen kanssa. Ulkoistaminen on varsinkin yrityksen kansainvälisen taipaleen alkupuolella suositeltava ja helpompi vaihtoehto. Tämäkään ei kuitenkaan kokonaan ratkaise lokalisointia, sillä ulkoiset yritykset voivat nekin tuottaa pettymyksen ja työtä riittää silti omankin yrityksenkin puolella.

6 Kulttuurierojen huomiominen käyttöliittymien suunnittelussa

Kun halutaan luoda ohjelmistoja vaihtelevalle käyttäjäjoukolle, on kansainvälistäminen ja lokalisointi toimiva ratkaisu. Käyttäjiä erottaa kuitenkin kielen lisäksi myös kulttuuritausta mm. erilaisen moraalin, minäkuvan ja maailmankuvan muodossa. Nämä asiat huomioimalla tavoitettaisiin käyttöliittymällä käyttäjä yhä paremmin. Tähän ongelmaan on keskustelua käyty erilaisista vaihtoehdoista. Ainakin Leventhalin *& al.* (1995) mukaan kaksi strategiaa on ilmestynyt, joko luodaan jokaiselle käyttäjäryhmälle oma käyttöliittymä, tai luodaan vain yksinkertaisesti yksi hyvä käyttöliittymä.

Day (1998) taas on sitä mieltä, että sovellusten tulee huomioida käyttäjien erot kielellisten seikkojen lisäksi myös syvemmällä tasolla. Myös Russon ja Boorin (1993) mielestä kulttuurieroihin tulee kiinnittää huomiota yksinkertaisten kielestä riippuvaisten piirteiden lisäksi. Yeo (1996) ehdottaa kulttuurierojen huomioimiseksi CUI:ta (Cultural User Interface), jota hän kuvailee korkkina pullolle. Pullo on sovellus, johon käy kullekin käyttäjäryhmälle sopiva korkki (CUI). Korkin voi sitten aina tarpeen vaatiessa vaihtaa. CUI:n sisällytettäisiin sitten kaikki mahdollinen, joka voi vaikuttaa sovelluksen käyttämiseen.

Yeo (1996) jatkaa vielä tätä ratkaisua PUI:n (Personal User Interface), jossa ajatus on, että, jokaiselle käyttäjälle voitaisiin kehittää oma käyttöliittymä kunkin mielen mukaan. Tässä luvussa käydään läpi kolme mallia kulttuurien luokitteluun, ja siirrytään sitten lokalisoinnin ja kansainvälistämisen yhdistämiseen näiden mallien lähestymistapaan.

6.1 Hoefsteden malli

Kuinka kulttuurieroja sitten tulisi kartoittaa? Hofstede (1991) on esitellyt kulttuurimallin, jossa määritellään eri kulttuuriryhmille arvoja, jotka sitten määrittelevät ryhmän tietynlaiseksi. Hofsteden (1991) mallissa asteikkoja on neljä: PD (Power Distance, valtaetäisyys), yksilöllisyys/yhteisöllisyys (Individualism/Collectivism), maskuliinisuus/feminiinisyys (Masculinity/Femininity) ja epävarmuuden välttely (Uncertainty Avoidance). Hoefstede (1991) esittelee eri asteikkojen ääripäille piirteitä, joilla ryhmiä voidaan arvioida. Esimerkiksi feminiinisissä yhteisöissä opettajat välttelevät

oppilaidensa avointa kehumista, kun taas maskuliinisissa yhteisöissä asia on päinvastoin.

Ford ja Gelderblom (2003) ovat tehneet tutkimuksen Hoefsteden (1991) esittelemän mallin paikkansapitävyydestä käyttöliittymien käyttämisen suhteen. Testissä luokiteltiin käyttäjiä eri ryhmiin, ja annettiin heille sitten käytettäväksi erityyppisille ryhmille ensisijaisesti luotuja Internet-sivuja. Tämän tarkoituksena oli selvittää, pystyykö tiettyyn ryhmään luokiteltu käyttäjä käyttämään samalle ryhmälle tarkoitettua sivua nopeammin ja paremmin kuin eri ryhmälle tarkoitettua sivua. Tutkimuksessa ei kuitenkaan pystytty todistamaan, että Hoefsteden (1991) mallin eri asteikoilla olevat käyttäjät olisivat hyötyneet vastaavan asteikon sivuista. Mallia ei myöskään pystytty todistamaan toimimattomaksi.

Myös Bryan *& al.* (1994) on tehnyt tutkimusta Hoefstedin mallin pohjalta. Heidän tutkimuksensa keskittyi tietojenkäsittelyn ammattilaisten erojen tutkimiseen. Vertailukohtana oli Hong Kongin ja Yhdysvaltojen työntekijät. Tutkimuksessa testattiin, onko heidän välillään eroja työn tekemisen suhteen. Eroja löytyi erityisesti yksilöllisyys/yhteisöllisyys -asteikolla. Yhdysvaltalaiset työntekijät ajattelivat, että reilu kohtelu ja valvonta on tärkeää työn turvaamisen kannalta, ja tiimityöskentely ja palaute ovat tärkeitä työn turvallisuuden lisäksi myös henkilökohtaisen kehityksen kannalta. Hong Kongin työntekijät taas ajattelivat, että nämä asiat ovat enemmän sosiaalista kanssakäymistä parantavia seikkoja.

6.2 Trompenaarsin malli

Myös Trompenaars (1994) on esitellyt vastaavanlaisen mallin kuin Hoefstede (1991). Hänen mallissaan kulttuuria määritteleviä asteikkoja on seitsemän:

- *Universalism and Particularism:* Keskittyykö yhteisö sääntöjen noudattamiseen vai henkilökohtaisiin suhteisiin.
- *Individualism and Communitarianism:* Onko yksilön etu tärkeämpää kuin yhteisön etu.
- *Specific versus Diffuse Relationships:* Suositaanko liiketoiminnassa tiukkoja määritelmiä kuten sopimuksissa, vai löysempiä henkilöiden välisiä suhteita kuten luottamusta.

- *Neutral versus Affective Communication Styles:* Näyttävätkö henkilöt tunteitansa.
- *Achievement and Ascription:* Statuksen saaminen saavuttaminen joko tekemällä (näyttämällä kyvykkyytensä esimerkiksi työpaikalla) tai olemalla (esimerkiksi koulutuksen tai iän avulla).
- *Time Orientation:* Ajatellaanko aikaa tulevina tapahtuvina, vai menneisyyden, nykyisyyden ja tulevaisuuden sekoituksena. Toisaalta onko menneisyys, tulevaisuus vai nykyhetki tärkein päätöksiä mietittäessä.
- *Nature Orientation:* Yrittävätkö ihmiset hallita luontoa, elää sen kanssa vai antaa sen hallita heitä.

Periaate Trompenaarsin mallissa on samankaltainen kuin Hoefstedilla, yhteisöt arvioidaan asteikolla ja näin niitä voidaan luokitella ja tätä luokittelua käyttää apuna esimerkiksi Internet-sivujen suunnittelussa. Ainakin Gould *& al.* (2000) ovat tehneet tutkimusta tästä mallista vertailemalla Internet-sivuja. Tutkimuksessa vertailtaviksi maiksi valittiin Yhdysvallat ja Malesia. Trompenaarsin asteikolla nämä maat eroavat lähinnä yksilön ja yhteisön edun asteikolla. Yhdysvalloissa yhteisön etu ei ole yhtä tärkeä kuin Malesiassa, jossa tilanne on päinvastainen. Tutkimuksessa valittiin kolme Internet-sivua kummastakin kohdemaasta samalta alalta niin, että kutakin sivua voitiin verrata vastaavaan toisen maan versioon. Tutkimuksessa analysoitiin Internet-sivuja kontekstin pohjalta, ja pääteltiin syitä miksi erilaiset sivut ovat toimivia niiden kohdemaassa, mutteivat toisessa. Lopuksi Gould *& al.* (2000) antavat vielä ohjeita Trompenaarsin eri asteikkojen ääripäiden tukemiseen Internet-sivujen suunnittelussa. Esimerkiksi Malesialaisten liiketoiminnassa suosimia henkilökohtaisia suhteita tulisi tukea korostamalla Internet-sivuilla myyjän laatua tuotteen laadun sijaan, ja käyttää yrityksestä ja tuotteista kiinnostavia symboleja ja logoja.

6.3 Kulttuurimerkkimalli

Barber ja Badre (1998) ovat esitelleet Internet-sivujen lokalisointiin kulttuurimerkkimallin. Heidän mukaan lokalisoitaessa tulee huomioida sivustojen elementit, jotka ovat tietynlaisia kullekin kulttuuriryhmälle. Näitä elementtejä Barber ja Badre kutsuvat kulttuurimerkeiksi (Cultural Marker). Kulttuurimerkit löytyvät käyttöliittymästä ja

sellaisia ovat mm. sivun asettelu, värien käyttö, fontti, kieli, äänet ja liikkeen käyttö. Teoriansa tueksi Barber ja Badre tekivät tutkimuksen, jossa he keräsivät kaiken kaikkiaan 168 Internet-sivua analysoitavaksi kulttuurimerkkien löytämiseksi. Sivut luokiteltiin maan, kielen ja alan mukaan. Kun merkkejä löydettiin, ne analysoitiin ja näin luotiin erilaisia kategorioita, joissa tietyt merkit ovat tavanomaisia. Esimerkiksi hallitusten sivuilla käytettiin usein maan lippua, arabian- ja hepreankielisillä sivuilla teksti, linkit ja grafiikat järjestettiin yleensä oikealta vasemmalle eikä keskelle tai vasemmalta oikealle ja Etelä-Amerikan maissa värejä käytettiin paljon kaikilla eri alojen sivuilla.

Sheppard ja Scholtz (1999) ovat tehneet tutkimusta Barberin ja Badren (1998) teorian pohjalta. Heidän tutkimuksensa tavoitteena oli selvittää, vaikuttavatko kulttuurimerkit käyttäjän suoriutumiseen tai mieltymykseen. Tutkimukseen he käyttivät kahta itse tekemäänsä versiota samasta Internet-sivusta (`http://www.edmunds.com`) ja kymmentä eri testihenkilöä. Testihenkilöiden tuli suorittaa annettuja tehtäviä sivuilla. Tehtävissä onnistumisen ja käyttäjän tyytyväisyys vaikuttivat sivuston laadun arviointiin Toinen versio sivusta oli alkuperäistä vastaava Amerikkalainen versio, josta ylimääräistä toiminnallisuutta oli poistettu. Toinen versio oli Lähi-Idän kulttuumerkkejä vastaava versio, josta toiminnallisuutta oli myös karsittu samalla tavalla. Testihenkilöistä viisi oli kotoisin Yhdysvalloista ja loput viisi Lähi-Idästä. Mieltymyseroja ei tutkimuksessa havaittu, mutta testihenkilöt suoriutuivat tehtävissä paremmin oman kulttuurinsa sivuilla.

Myös Sun (2001) on tutkinut kulttuurimerkkien merkitystä Internet-sivujen suunnittelussa. Hänen tutkimuksensa keskittyi käyttäjien mielipiteiden keräämiseen Internet-sivujen kulttuumerkkien osalta. Tutkimuksessa käytettiin oikeita, kokonaisia Internet-sivuja `http://www.adobe.com` ja `http://www.lotus.com`. Kumpikin Internet-sivuista oli monikielinen ja jokainen testihenkilö käytti sekä kansainvälistä, että ko. testihenkilön taustaan sopivaa lokalisoitua versiota testin aikana.

Lisäksi Lotuksen Internet-sivuista käytettiin myös tanskalaista versiota väärälle kulttuurille lokalisoituna testikappaleena. Lotuksen sivut valittiin, koska ne olivat hyvin lokalisoidut ja sisältivät kulttuurimerkkkejä. Adoben sivut taas valittiin koska ne eivät sisältäneet kulttuurimerkkejä käännetyn kielen lisäksi. Testihenkilöitä oli kolme: saksalainen, kiinalainen ja brazilialainen. Tavoitteena oli selvittää huomaavatko käyttäjät erilaiset kulttuurimerkit Lotuksen Internet-sivuilla, ja verrata sitten käyttäjän mieltymystä Lotuksen sivujen ja Adoben kulttuurimerkittömien sivujen välillä.

Tutkimuksessa kiinalainen ja brazilialainen käyttäjä kokivat olonsa mukavaksi Lotuksen heidän kulttuurilleen suunnatuilla sivuilla, saksalainen käyttäjä taas oli tyytyväinen aakkostettuun navigointipalkkiin Lotuksen saksankielisellä sivulla. Adoben lokalisoidut sivut taas eivät saaneet aikaan vastaavanlaista tyytymystä testihenkilöissä. Kaikki testattavat myös huomasivat, että Lotuksen tanskalainen sivu ei ollut suunniteltu heitä varten.

6.4 Mallien rajoitukset

Kulttuuria ei ole helppo määritellä. Sille ei ole olemassa mitään yksittäistä määritelmää, joten yksioikoisen mallin tekeminen voi olla hankalaa. Tällaisia malleja voidaankin kritisoida, ja löytää niistä ongelmia. Kulttuuriryhmiä muodostetaan yleensä kansallisuuksien mukaan, suomalaiset muodostavat yhden kulttuurin, ruotsalaiset toisen. Kuitenkin tämä voi olla merkityksetön ero joissain tapauksissa, ryhmiä eivät välttämättä erota valtioiden rajat ja yhden maan sisältä voi löytyä montakin ryhmää.

Vaihtoehtoinen lähestymistapa kulttuurin ymmärtämiseen voisikin löytyä kognitiivisesta antropologiasta. Se on tieteenala, joka keskittyy tutkimaan ihmisten kulttuurin ja ajatusten suhdetta. Strauss ja Quinn (1997) ovat tehneet tutkimusta asiasta, ja heidän mukaansa ihmiset tulkitsevat kokemiansa asioita ja tähän tulkintaan vaikuttavat niin ennakko-odotukset, menneet kokemukset kuin tunteetkin tapahtumahetkellä. Heidän mukaansa varsinaista eroa ei ole yksilön kokemuksien ja kulttuurisidonnaisten kokemusten välillä, siis henkilön saamiin käsityksiin asioista ja tapahtumista voi yhtä hyvin vaikuttaa hänen henkilökohtaiset näkemyksensä, kuin kulttuuriinkin sidotut asenteet ja odotukset. Siis varsinaista kulttuuria ei sinällään olisi, ainakaan kulttuuria mikä suoraan sanelisi yksilön toiminnan jokaisessa tilanteessa.

Straussin ja Quinnin (1997) teorian ajatus onkin selittää, miten saman kulttuurin sisällä voi olla ristiriitoja, ja kuinka ihmiset kokevat asioita joihin heidän taustansa vaikuttaa. Tämä ei sinällään välttämättä sulje pois muita teorioita, mutta voisi olla yksi ratkaisu niiden jättämiin aukkoihin ja ongelmiin.

6.5 Kulttuurimallien yhdistäminen kansainvälistämiseen ja lokalisointiin

Kansainvälistämisen ja lokalisoimisen avulla saadaan aikaiseksi sovelluksia, jotka on tehty usealla kielellä ja joita voivat käyttää useaan eri käyttäjäryhmään kuuluvat käyttäjät. Huomioimalla myös käyttäjäryhmän kulttuuritausta, saadaan sovelluksen käytettävyyttä parannettua entisestään. Karat ja Karat (1996) mukaan menestyvän sovelluksen täytyykin mukautua käyttäjien kulttuuriin, sen sijaan että käyttäjät mukautuisivat sovellukseen. Heidän mukaansa tämä on tärkeää varsinkin silloin, kun sovellus ei ole neutraali.

Sovellus on neutraali, jos se ei ota kantaa käyttäjän arkipäiväiseen elämään kuten koulutukseen tai työhön. Nykyisin sovellukset kuitenkin liittyvät yhä enemmän ja enemmän käyttäjien arkipäivään, jolloin niiden täytyy myös sopeutua kulttuuriin. Jos kyseessä kuitenkin on neutraali sovellus, lokalisointi ja kansainvälistäminen riittää usein.

Marcus *& al.* (1999) yhtyy myös tähän, heidän mielestään sovelluksen pohjan muodostaa kansainvälistäminen ja lokalisointi, mutta kulttuuri näyttelee tärkeää osaa käyttäjän syvällisemmässä tavoittamisessa. Kulttuurimallien käyttämistä kansainvälistämisen ja lokalisoinnin yhteydessä voitaisiinkin ajatella eräänlaisena lisukkeena, jota tulee käyttää tilanteissa, joissa pelkkä lokalisoiminen ei riitä.

7 Yhteenveto

Nykyään ohjelmistoja tehdään jo varsin monikieliselle käyttäjäjoukolle. Samaa ohjelmistoa voidaan myydä monessa maassa, mutta käyttäjät odottavat ja usein jopa vaativat voida käyttää ohjelmistoa omalla kielellään. Internet on hyvä väline tiedon levittämiseen ja sen etsimiseen, mutta tiedon täytyy olla sellaisessa muodossa, että käyttäjä voi siitä hyötyä.

Lokalisointi ja kansainvälistäminen on yksi tapa mahdollistaa erilaiset kieli- ja kulttuuritaustat omaavien käyttäjien palvelu niin, että kukin voi käyttää sovellusta. Eri kielet ja kulttuurit eroavat toisistaan huomattavasti, ja lokalisoitavissa versioissa tulee jo pinnallisissa eroissa paljon huomioitavaa. Valuutat, numeroista koostuvan tiedon esittäminen, uskonnon vaikutukset ja monet muut seikat tulee ottaa huomioon, että eri käyttäjät pystyvät käyttämään ohjelmistoa haluamallaan tavalla. Itse lokalisointiprosessi niin web-sovelluksen, kuin tavallisen sovelluksen osalta on verrattavissa tavalliseen ohjelmiston kehitysprosessiin.

Käyttäjien kulttuurin huomioiminen syvemmällä tasolla kuin erilaisen kielen ja esitysmuotojen osalta on tietyissä tilanteissa tärkeää. Varsinkin tilanteissa, joissa kilpailu on kova, voi käyttäjän kulttuurin syvällisempi tukeminen saada käyttäjän kääntymään paremmin toteutetun ohjelmiston puoleen.

8 Sanasto

ASCII	American Standard Code for Information Interchange, ensimmäisiä nimettyjä merkistöjä, 28
EBDIC	Extendend Binary Coded Decimal Interchange Code, IBM:n vastine ASCII:lle , 28
ISO 646	ASCII:n ISO-versio, 29
ISO 8859	8-bittinen ASCII-laajennus, merkistöperhe, 30
i18n	Lyhenne, joka tulee kansainvälistämisen englanninkielisestä nimestä internationalization
Kansainvälistäminen	Prosessi, jossa sovelluksesta tehdään helposti lokalisoitava, 6
L10n	Lyhenne, joka tulee lokalisoinnin englanninkielisestä nimestä localization
Locale	Lokaali, kertoo käytettävän maa- ja kielitiedon, 22
Lokalisointi	Prosessi, jossa sovellus sovitetaan jollekin kohderyhmälle, 7
Merkistö	Sovelluksessa käytettävä aakkosto, 28
Unicode	Lähes kaikki maailman merkit sisältävä merkistö, 31

Viitteet

Barber, W., Badre, A. (1998) Culturability: The Merging of Culture and Usability. *HFWeb ’98*, Ei kustantajaa.

Bryan, N., McLean, E., Smits, S., Burn, J. (1994) The structure of work perceptions among Hong Kong and United States IS professionals: a multidimensional scaling test of the Hofstede cultural paradigm. *SIGCPR ’94*, ACM Press, New York, NY, USA, 219–230.

Checkdomain.com (2005) *Checkdomain.com*. WWW-sivusto, `http://www.checkdomain.com/` 31.10.2005

Day, D. (1998) Shared values and shared interfaces: The role of culture in the globalisation of human-computer systems. *Interacting with Computers 9*, **1998**(9), 269–274.

Dr. International (2003) *Developing International Software*. Microsoft Press, USA.

Esselink, B. (1998) *A Practical Guide to Software Localization*. John Benjamins Publishing Company, Philadelphia.

Ford, G., Gelderblom, H. (2003) The Effects of Culture on Performance Achieved through the use of Human Computer Interaction. *SAICSIT ’03* (toim. Eloff, J., Engelbrecht, A., Kotzé, P., Eloff, M.), South African Institute for Computer Scientists and Information Technologists, Etelä-Afrikka, 218–230.

Global Reach (2005) *Global Online Marketing Services. Internet Marketing Agency - GLREACH*. WWW-sivusto, `http://global-reach.biz/` (19.10.2005).

Gould, W., Zakaria, N., Yusof, S. (2000) Applying culture to website design: a comparison of Malaysian and US websites. *IPCC/SIGDOC ’00*, IEEE Educational Activities Department, Piscataway, NJ, USA 161–171.

Hoefstede, G. (1991) *Cultures and organisations: software of the mind*. McGraw Hill, New York.

IBM (2005) *IBM United States* WWW-sivusto, `http://www.ibm.com/` (31.10.2005).

Ido (2005) *Ido Bathroom*. WWW-sivusto, `http://www.idobath.com`

(31.10.2005).

Karat J. ja Karat C. (1996) Perspectives on Design and Internationalization. *SIGCHI Bulletin* **1996**(1), 39–40.

Leventhal, L., Teasley, B., Stone, D., Lancaster, A., Nielsen, J., Marcus, A., Kurosu, M., Nardi, B., Heller, R. (1994) Designing for Diverse Users: Will Just a Better Interface Do?. *CHI '94*, ACM Press, New York, NY, USA, 191–192.

Marcus, A., Aykin, N., Chavan, A., Prabhu, G., Kurosu, M. (1999) SIG on one size fits all?: cultural diversity in user interface design. *CHI '99*, ACM Press, New York, NY, USA, 342–342.

National Institute of Standards and Technology (2005) *The official U.S. time* WWW-sivusto, `http://nist.time.gov/` (19.10.2005).

Oxford University Press (2005) *Oxford English Dictionary: the definitive record of the English language* WWW-sivusto, `http://www.oed.com/` (15.11.2005).

O'Donnel, S. (1994) *Programming for the World*. Prentice Hall, USA.

Russo, P., Boor, S. (1993) How Fluent is Your Interface? Designing for International Users. *INTERCHI '93* (toim. S. Ashlund, A. Henderson, E. Hollnagel, K. Mullet, T. White), ACM Press, New York, NY, USA, 342–347.

Sheppard, C., Scholtz, J. (1999) The Effects of Cultural Markers on Web Site Use. *HFWeb '99*, Ei kustantajaa.

Software Marketing Resource (2005) *Software Marketing Resource for software developers, authors and software publishers* WWW-sivusto, `http://www.softwaremarketingresource.com/` (24.10.2005).

Strauss, C., Quinn, N. (1997) *A Cognitive Theory of Cultural Meaning*. Cambridge University Press, Cambridge.

Sun, H. (2001) Building a culturally-competent corporate web site: an exploratory study of cultural markers in multilingual web design. *SIGDOC '01*, ACM Press, New York, NY, USA, 95–102.

Trompenaars, F. (1994) *Riding the Waves of Culture: Understanding Cultural Diversity*

in Global Business. Irwin Professional Publishing, USA.

Unicode Inc. (2005) *Unicode Home Page* WWW-sivusto, `http://www.unicode.org/` (26.10.2005).

Wikipedia (2006) *Wikipedia*. WWW-sivusto, `http://www.wikipedia.org` (3.5.2006).

Yeo, A. (1996) Cultural User Interfaces - A Silver Lining in Cultural Diversity. *SIGCHI Bulletin* **1996**(3), 4–7.

Yunker, J. (2002) *Beyond Borders: Web Globalization Strategies*. New Riders Publishing, USA.