Verkkosivuston uudistaminen ja sisällönhallintajärjestelmän käyttöönotto

Jussi Saine

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Vuorovaikutteinen teknologia
Pro gradu -tutkielma
Toukokuu 2008

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Vuorovaikutteinen teknologia
Jussi Saine: Verkkosivuston uudistaminen ja sisällönhallintajärjestelmän käyttöönotto
Pro gradu -tutkielma, 58 sivua
Toukokuu 2008

Laajan verkkosivuston ylläpito on suunniteltava huolellisesti, jotta sivusto olisi hallittavissa käyttöönoton jälkeen. Aina välillä sivusto joudutaan kuitenkin uudistamaan riippumatta siitä kuinka hyvin edellinen sivusto on toteutettu. Sivuston rakenteessa tai tarvittavissa ominaisuuksissa voi esimerkiksi olla sellaisia vaatimuksia, joita ei vanhaan sivustoon ole enää järkevää toteuttaa. Uudistusprojekti voidaan viedä läpi eri tavoin. Onnistuneen uudistuksen tekijöitä ovat sivuston ja sen sisällön huolellinen määrittely ja kartoitus, muutosten ennakointi ja käyttäjien huomioonottaminen. Sivuston tyypistä riippuen yksi ratkaisu voi olla sisällönhallintajärjestelmän käyttöönotto.

Tutkimuksen tavoitteena oli luoda katsaus verkkosivuston uudistusprojektin vaiheisiin ja kuvailla erään projektin osana tehdyn esimerkkisivuston toteutusta. Uudistusprojektin eteneminen esitellään vaiheittain. Esimerkkisivuston toteutuksesta poimitaan joitakin ongelmatilanteita ja niiden ratkaisuehdotuksia. Lopuksi arvioidaan muutamia muita välineitä ja toteutustapoja sekä niiden soveltuvuutta esimerkkiprojektin toteutukseen.

Esimerkkiprojektista saatujen kokemusten perusteella voidaan todeta, että oikealla sisällönhallintajärjestelmän valinnalla voi olla vaikutusta sivustouudistuksen vaatimaan työmäärään. Lopputuloksen laatu on kuitenkin enemmän kiinni perusteellisesta alkusuunnittelusta ja toteutuksen huolellisuudesta kuin käytetystä sisällönhallintajärjestelmästä.

Avainsanat ja -sanonnat: verkkosivuston uudistusprojekti, sisällönhallintajärjestelmät, Drupal.

Sisällys

1. Johdanto

Verkkosivustojen sisällönhallinnan voi toteuttaa monella tavalla riippuen sivuston laadusta, laajuudesta, ylläpitäjistä ja käytettävissä olevista resursseista. Trendien vaihtuessa, tekniikan kehittyessä ja tarpeiden muuttuessa sivustoon kohdistuu uudistuspaineita. Joskus selvitään kevyellä ehostuksella, mutta aina välillä edessä on suurempi, kokonaisvaltainen remontti. Tällaisen uudistusprojektin yhteydessä tarjoutuu tilaisuus arvioida erilaisia sisällönhallintajärjestelmiä sekä niiden tarjoamia mahdollisuuksia sivuston käyttöön ja ylläpitoon.

Tässä tutkimuksessa käsitellään verkkosivujen uudistamista yleisellä tasolla sekä esitellään erään sivuston uudistusprojektin vaiheita. Osana kyseistä projektia tehtiin esimerkkitoteutus uudesta sivustosta projektiryhmän valitsemaa järjestelmää käyttäen. Siitä saatujen kokemusten perusteella on tarkoitus ottaa käyttöön uusi sivusto, jolla päästäisiin eroon vanhan sivuston sisällönhallintaan liittyvistä ongelmista. Toteutuksessa tuli vastaan monenlaisia tarkempaa perehtymistä vaativia tilanteita. Niiden esittelystä ja ratkaisuehdotuksista on toivottavasti hyötyä myös muille verkkosivuston uudistusprojektia aloitteleville ylläpitäjille.

Tutkimuksen toisessa luvussa esitellään tekstin ymmärrettävyyden kannalta keskeiset käsitteet. Aiheena on myös verkkosivustojen evoluutio suhteessa tekniikan ja sisällön kehittymiseen. Web-ohjelmointimenetelmien ja laitteiden kehittyessä sivustoille on keksitty uutta toiminnallisuutta, jonka toteuttaminen olisi aiemmin ollut mahdotonta. Vielä vuosikymmen sitten olisi ollut vaikea kuvitella maailmanlaajuista 3D-virtuaalimaailmaa joka on kokonaan käyttäjiensä luoma [Second Life, 2008]. Tällä hetkellä vallitseva suuntaus on yhteisöllisyys sekä käyttäjän roolin muuttuminen lukijasta tai asiakkaasta yhä enemmän aktiiviseksi osallistujaksi ja sisällöntuottajaksi. Tämä asettaa haasteita myös sisällönhallintajärjestelmälle, jonka on pystyttävä vastaamaan käyttäjien vaatimuksiin.

Kolmannessa luvussa käsitellään verkkosivuston uudistusprojektia kokonaisuutena. Uudistusprojekti jaetaan muutamiin eri vaiheisiin, jotka luvussa käydään läpi. Tärkeä vaihe projektin alussa on lähtökartoitus, jossa otetaan huomioon myös vanhan järjestelmän ominaisuudet. Myös käyttäjien huomioiminen heti alusta lähtien on tärkeää. Näin sivusto koostuu sellaisesta sisällöstä, jota sieltä halutaan, ja joka on myös riittävän helposti löydettävissä.

Neljännessä luvussa pohditaan sisällönhallintajärjestelmän valintaan liittyviä kysymyksiä ja esitellään muutamia vaihtoehtoisia toteutustapoja. Erilaisia sivustotyyppejä käydään läpi sen verran, että saadaan kuva niiden sisällönhallintajärjestelmälle asettamista erityisvaatimuksista. Tietyntyyppisellä sivustolla toimivaksi todettu järjestelmä ei välttämättä sovellu lainkaan toisentyyppisen sivuston ylläpitoon. Myös päinvastaisia esimerkkejä on, esimerkiksi Moodle-oppimisympäristöä on onnistuneesti käytetty ta-

vallisen verkkosivuston toteuttamiseen [Buckinghamshire, 2008]. Luvussa käsitellään myös vaihtoehtoisia lähestymistapoja ja muita järjestelmää valittaessa arvioitavia asioita.

Viidennessä luvussa tutustutaan tutkimuksessa esimerkkinä käytettävään verkkosivuston uudistusprojektiin. Projekti oli osa Tampereen yliopiston vuorovaikutteisen teknologian yksikön sivustouudistusta. Siinä toteutettiin sivustosta annettujen suunnitelmien perusteella malli, jonka perusteella lopullista toteutusta olisi helpompi lähteä tekemään. Tässä luvussa esitellään myös projektissa käytetty sisällönhallintajärjestelmä Drupal.

Kuudennessa luvussa käydään läpi esimerkkiprojektin esimerkkisivuston toteutusvaiheet. Käyttäjätestausta ei esimerkkisivustoa luodessa tehty, mutta käytettävyyteen pyrittiin muuten kiinnittämään huomiota. Luvussa käydään perusteluineen läpi sisällönhallintajärjestelmään tehtyjä muutoksia, erityisesti niitä joilla pyrittiin parempaan käyttäjäkokemukseen.

Seitsemännessä luvussa esitellään muutamia esimerkkisivuston toteutuksessa kohdattuja ongelmia. Esillä ovat pääasiallisesti toiminnallisuuden toteuttamiseen liittyvät kysymykset ja ohjelmointivirheet jätetään vähemmälle huomiolle. Tässä luvussa pohditaan myös sitä, miten ongelmanratkonta olisi onnistunut muutamilla vaihtoehtoisilla toteutustavoilla. Vertailukohdiksi on otettu yksi suoraan Drupalin kanssa kilpaileva järjestelmä sekä yksi Web-sovelluskehys, joka edustaa hieman toisenlaista lähestymistapaa verkkosivuston toteutukseen.

Kahdeksas luku sisältää yhteenvedon ja pohdintaa mahdollisista jatkotutkimuksen aiheista.

2. Verkkosivujen ylläpito ja evoluutio

Verkkosivuston ylläpito on ongelma, jota vaatii tarkempaa määrittelyä ennen kuin sitä voidaan ryhtyä tutkimaan. Erillisiä kokonaisuuksia ovat esimerkiksi sivuston tekninen ylläpito ja sisällöntuotanto. Ei kuitenkaan aina ole selvää, kumpaan edellisistä esimerkiksi ulkoasun ja käyttöliittymän ylläpito kuuluu. Verkkosivujen olemukseen kuuluu, että myös navigointiliittymä ja sivun ulkoasu ovat tyypillisesti osa sen sisältöä. Yhä useammin sisällön ylläpitäjällä pitää olla mahdollisuus muokata näitä tarvitsematta kutsua pääkäyttäjää tai teknistä asiantuntijaa apuun. Tällöin apu saadaan sisällönhallintajärjestelmästä, jonka avulla pystytään määrittelemään eritasoisia ylläpitäjärooleja.

Jos sivustolla on käytössä sisällönhallintajärjestelmä, käyttöliittymiä on itse asiassa kaksi erilaista: itse suunniteltu vierailijoille ja peruskäyttäjille näkyvä, sekä ylläpitäjän käyttämä sisällönhallintajärjestelmän käyttöliittymä. Sivuston käytettävyyttä arvioitaessa näitä osia pitäisi analysoida erillisinä kokonaisuuksina. Koska myös ylläpitopuolta käytetään jatkuvasti, ei ole aivan yhdentekevää miten se on toteutettu. Ylläpitokäyttöliittymän muokkaamismahdollisuudet ovat yleensä rajalliset. Sen paranteluun on myös vähemmän mielenkiintoa, koska se ei näy asiakkaille. Ylläpidon toimivuuteen on kiinnitettävä siis huomiota jo sisällönhallintajärjestelmää valittaessa.

Verkkosivuston ylläpidosta on siis löydettävissä muutamia erillisiä osa-alueita. Sivustojen toteutustavasta johtuen ylläpitäjäroolien rajanveto voi osoittautua hankalaksi. Lisäksi sivustolla voi olla erilaisia käyttöliittymiä, joita kaikkia ei edes pystytä muokkaamaan. Tässä luvussa käsitellään keskeisten käsitteiden lisäksi verkkosivustojen kehittymistä ja niiden ylläpitoon yleisesti vaikuttavia tekijöitä.

2.1. Keskeisiä käsitteitä

Alla olevassa luettelossa esitellään tutkimuksen sisällön kannalta keskeisiä termejä.

- **Blogi (Weblog).** Päiväkirjatyyppinen verkkosivu tai -sivusto. Blogikirjoituksissa on henkilökohtainen näkökulma asioihin ja niiden sisältö on aikasidonnaista kuten päiväkirjoissa yleensä. Kun blogiin tulee uusia merkintöjä, ne lisätään listaan ja vanhat merkinnät säilyvät muuttumattomina.
- **CSS (Cascading Style Sheets).** Tyyliohjeet joiden avulla HTML-sivun ulkoasu muotoillaan. Tyyliohjeet voidaan tallentaa omaan tiedostoonsa ja niihin voidaan viitata kaikilta sivuston sivuilta. Näin ulkoasumuutokset tarvitsee tehdä vain yhteen tiedostoon ja ne näkyvät heti koko sivuston laajuisesti. Tyyleillä määritellään elementtien ominaisuudet kuten tekstin ja kuvien väri, sijoittelu ja koko.
- **Dynaaminen sivusto.** Verkkosivusto, jossa yksittäisen sivun sisältö voi muuttua käyttäjän toimien seurauksena. Yksinkertainen dynaamisuuden muoto voi ilmetä

esimerkiksi verkkosivulla olevassa lomakkeessa: kun lomakkeessa tehdään tietty valinta, sivulle voi aueta lisäkysymys. Dynaamisuudella voidaan tarkoittaa myös sitä, että sivuston varsinainen sisältö on tallennettu tietokantaan. Tällöin tietokannasta haetaan käyttäjän pyytämä sisältö, josta sivu sitten muodostetaan aina tapauskohtaisesti. Vrt. staattinen sivusto.

- **Eväste.** Pieni käyttäjän tietokoneelle tallentuva tekstitiedosto, jonka avulla tietoja käyttäjän asetuksista voidaan kuljettaa sivuston eri osien välillä. Esimerkiksi monikielisellä sivustolla evästeeseen voidaan tallentaa käyttäjän kielivalinta. Tällöin palvelin osaa evästeeltä tietoja kysymällä tarjota oikean kieliversion haetusta sisällöstä.
- **Flash.** Joukko teknologioita, joilla voidaan levittää kaikenlaista sisältöä paketoituna hyvin yhteensopivaan säiliöön. Erityisesti liikkuvan kuvan ja äänen sekä interaktiivisen multimedian levityksessä Flash on erittäin suosittu. Kun videoleike upotetaan osaksi Flash-tiedostoa, voidaan olla varmoja että se näkyy oikein vastaanottajilla. Vaatimuksena on ainoastaan, että selaimeen on asennettu tuen tarjoava Flash Player -lisäosa. Flashia kehittävän Adoben mukaan Flash Player on asennettu yli 97%:iin maailman tietokoneista [Adobe, 2008].
- **HTML (Hypertext Markup Language).** Sivunkuvauskieli, jolla kuvataan hypertekstiä eli tekstiä joka sisältää hyperlinkkejä. Internet-selain tulkitsee HTML-kielisen sivun ja näyttää sen käyttäjälle. HTML-kielellä on standardeja joita sitä käyttävien pitäisi noudattaa. Käytännössä useimmat selaimet osaavat kuitenkin näyttää myös puutteellisesti kuvaillut sivut lähes halutun näköisinä.
- **Java.** Ohjelmointikieli, käytetään monenlaisten ohjelmistojen toteutukseen. Javasovelluksia voidaan upottaa WWW-sivuille ns. appletteina jotka suoritetaan käyttäjän omalla tietokoneella.
- **Javascript.** Ohjelmointikieli, jota käytetään selainkäyttöisten ohjelmistojen toteutukseen. Javascript-koodi suoritetaan käyttäjän omalla koneella WWW-sivun latauksen yhteydessä. Javascriptin avulla verkkosivuille on mahdollista toteuttaa dynaamista toiminnallisuutta.
- **LDAP.** Hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla. LDAPia voidaan käyttää mm. käyttäjätunnistukseen. Hakemistopalvelussa käyttäjästä on tallennettuna käyttäjätunnus ja salasana sekä mahdollisesti muita tietoja kuten osoite, puhelinnumero ja sähköpostiosoite. Sisällönhallintajärjestelmä voi kirjautumishetkellä ottaa yhteyden LDAP-palvelimeen ja varmistaa siltä, että kirjautumista yrittävä käyttäjätunnus on olemassa ja salasana on oikein.
- **Mashup.** WWW-sovellustyyppi, jossa verkkosivusto yhdistelee tietoja eri lähteistä ja muodostaa omanlaisensa palvelun jota yksikään alkuperäisistä lähteistä ei tarjonnut. Tyypillinen esimerkkisovellus voi hakea yritysluettelosta kaupungin tietokoneliikkeet ja karttapalvelusta kaupungin kartan. Mashup-sivustolla liikkeet sijoitetaan kartalle josta liikkeen nimen valitsemalla aukeaa sen kotisivu.

- **PHP.** Ohjelmointikieli, jota käytetään erityisesti selainkäyttöisten sovellusten toteutuksessa. PHP-koodi suoritetaan WWW-palvelimella ja tuloksena saatu WWW-sivu lähetetään käyttäjän selaimelle.
- **Python.** Ohjelmointikieli, joka soveltuu kaikenlaisten ohjelmistojen toteutukseen, myös PHP:n korvaajaksi. Python-koodi suoritetaan WWW-palvelimella ja tuloksena saatu WWW-sivu lähetetään käyttäjän selaimelle.
- **Sisällönhallintajärjestelmä (CMS, Content Management System).** Tietojärjestelmä, jolla pyritään käsittelemään organisaation koko sisällönhallintaa. Tässä tutkimuksessa termiä sisällönhallintajärjestelmä käytetään synonyyminä WWW-sisällönhallintajärjestelmälle, joka nimensä mukaisesti keskittyy internetissä julkaistavana sisällön hallintaan.
- **Sivupohja.** Malli, johon verkkosivun sisältö voidaan upottaa. Sisältää yleisesti toistuvat elementit kuten sivun otsikkotiedot, logon ja navigoinnin sekä ohjeet siitä, miten tietokannasta haettu sisältö näytetään.
- **Staattinen sivusto.** Sivun sisältö pysyy samana käyttäjän toimista riippumatta. Jokainen verkkosivu on staattinen kokonaisuus joka näyttää kaikille käyttäjille täsmälleen samalta. Vrt. dynaaminen sivusto.
- **Tagi.** Avainsana tai termi joita käytetään sisällön kuvailussa. Esimerkiksi blogimerkintään voidaan liittää sitä kuvailevia avainsanoja. Näiden avulla sisältöä pystytään sitten lajittelemaan. Esimerkiksi tagin "jalkapallo" valitsemalla saadaan listattua sivuston ne blogimerkinnät joissa kyseinen tagi on.
- **Taksonomia.** Luokittelu, tapa jolla asioita luokitellaan jonkin yhdistävän tekijän kautta.
- **URL (Uniform Resource Locator).** Merkkijono jolla kerrotaan jonkin tiedon sijainti, siis WWW-sivun osoite.
- **URL-skeema.** Tapa, jolla kuvaillaan URL-osoitteiden muodostuminen. Sisällönhallintajärjestelmillä ja sovelluskehyksillä voi olla oma järjestelmänsä osoitteiden muodostamiseen. Esimerkiksi http://osoite.fi/uutiset/24122008/1/ tai http://osoite.fi/sisallot/uutiset/2008/joulukuu/24/kinkkua_tarjolla/ voisivat molemmat osoittaa vuoden 2008 jouluaaton ensimmäiseen uutiseen. Joillakin järjestelmillä nämä ovat muokattavissa, joskus joudutaan tyytymään järjestelmän oletuksena tarjoamaan tapaan. Tärkeää on, että osoitteet ovat pysyviä, jotta sisältöön tehdyt viittaukset ovat ehjiä myös vuosien kuluttua.
- **W3C (World Wide Web Consortium).** Kansainvälinen yritysten ja yhteisöjen yhteenliittymä, joka ylläpitää ja kehittää WWW:n standardeja, esimerkiksi CSS- ja HTML-standardit.
- **WYSIWYG (What You See is What You Get).** Sisällönmuokkaustapa, jossa jo muokkausvaiheessa nähdään miltä lopputulos näyttää. Esimerkiksi samanlaiseen lopputulokseen pääsee kirjoittamalla HTML-sivun sekä Notepad-tekstieditorilla että

Dreamweaver-WYSIWYG-editorilla [Adobe Dreamweaver,2008]. Notepadilla kirjoitetaan HTML-kuvauskieltä ja ulkoasu tarkistetaan lataamalla sivu selaimeen, Dreamweaverilla sivulle sijoitetaan elementtejä ja lopputulos on koko ajan nähtävillä.

- **Web 2.0.** Väljästi käytetty käsite internetin evoluution uudesta (nykyisestä) vaiheesta. Web 2.0 -sovelluksissa WWW-sivuille on toteutettu uudenlaista interaktiivisuutta esimerkiksi javascriptin avulla. WWW-sivut toimivat enemmän tavallisten ohjelmien tapaan. Web 2.0:lla voidaan tarkoittaa myös uuden sosiaalisen internetin aikakautta, jossa sisällöntuotanto siirtyy yrityksiltä sivuston asiakkaille ja käyttäjille. Tyypillisiä Web 2.0 -sovelluksia ovat esimerkiksi Facebook, Google Maps ja Youtube.

Verkkosivujen ylläpitomenetelmät ovat kehittyneet tekniikan tarjoamien mahdollisuuksien tahdissa. Internetin alkuaikoina kehitys oli nopeaa ja työkalut keskeneräisiä. Tällöin Web-suunnittelijoiden työstä suuri osa kului selaimien yhteensopivuusongelmien ja puutteellisten standardien aiheuttamien ongelmien ratkomiseen. Standardien kehittämisessä tärkeä rooli on ollut World Wide Web Consortiumilla [W3C, 2008] jonka HTML- ja CSS-standardit ovat osaltaan mahdollistaneet nykyaikaisen WWW:n olemassaolon.

CSS-tyylitiedostot pyrkivät erottamaan sivuston ulkoasun sisällöstä. CSS:n versio 2 on jo 10 vuotta vanha ja uusinkin versio 2.1 on ollut käytössä jo vuodesta 2004 [W3C CSS, 2008]. Siksi useimmat valtavirran WWW-selaimet tukevat sen ominaisuuksia riittävän hyvin ja Web-suunnittelijat voivat käyttää CSS-tyylejä sovelluksissaan. Vastaava tilanne on HTML-kuvauskielessä, jonka nykyisin käytössä oleva versio 4.01 on jo vuodelta 1999 [W3C HTML, 2008]. CSS-versio 3.0 ja HTML 5.0 ovat luonnosasteella, mutta tällä hetkellä Web-suunnittelijalla on käsissään hyväksi havaittu ja laajasti tuettu perusta, jonka päälle sivuja voi lähteä työstämään. Kun perusteet ovat kunnossa, voidaan keskittyä entistä enemmän myös saavutettavuuteen ja esimerkiksi mobiililaitteiden huomioimiseen.

Aiemmin hyvin ristiriitaisia tunteita herättänyt javascript on noussut keskeiseen rooliin erityisesti Web 2.0 -sovelluksien myötä. Javascript on nykyään hyvin yleisessä käytössä sellaisilla verkkosivustoilla, joilla tarvitaan muutakin interaktiivisuutta kuin perinteistä linkkien valitsemista ja lomakkeiden täyttämistä. Eräitä syitä javascriptin aiempaan epäsuosioon on ollut sen monimutkaisuus (nimestään huolimatta sillä ei ole tekemistä Java-ohjelmointikielen kanssa vaan se on oma kielensä), huono toimivuus vanhoissa selaimissa ja hitaus. Nykyaikaiset selaimet selviävät javascriptin suorittamisesta huomattavasti edeltäjiään paremmin. Javascriptin suoritusnopeus onkin erityisesti viime aikoina kasvanut harppauksin. Esimerkiksi uusi Firefox 3 -selain on javascript-koodin suorittamisessa 5 kertaa edeltäjäänsä nopeampi ja lähes 10 kertaa nopeampi kuin Internet Explorer 7 [Kingsley-Hughes , 2008]. Tämä mahdollistaa monipuolisem-

mat verkkosovellukset ja avaa uusia mahdollisuuksia myös sisällönhallintajärjestelmien kehittäjille.

Yksi WWW-sivustoja muokannut tekijä on verkkoinfrastruktuurin kehittyminen. Kotiin saatavan internet-liittymän nopeus on vajaassa parissakymmenessä vuodessa kasvanut 33,6 kbps modeeminopeuksista satoja kertoja nopeampiin 24 Mbps tai jopa 100 Mbps nopeuksiin [Welho, 2008]. Tämän päivän Web-suunnittelijalla on siis mahdollisuuksia huomattavasti graafisempien ja monimutkaisempien sivustojen toteuttamiseen. Yhteyksien nopeutumista ei voi kuitenkaan pitää tekosyynä jättää sivuston optimointi puolitiehen.

2.2. Verkkosivuston evoluutio

Sekä laitteiden että ohjelmistojen kehitys on ollut nopeaa WWW:n lyhyen historian aikana (ensimmäinen suuren yleisön tietoisuuteen tullut WWW-selain Mosaic julkistettiin vuonna 1993 [NCSA Mosaic, 2008]). Tekniikan kehittyminen on tarjonnut uusia mahdollisuuksia myös sisällön tuottamiseen. Alkuaikojen staattisista Web-sivuista on edetty tietokantapohjaisiin kokonaisuuksiin ja uudenlaista interaktiivisuutta tarjoaviin palveluihin. Sivustojen kasvaessa ja sisällön päivitystahdin nopeutuessa ylläpitovastuuta on jouduttu jakamaan yhä suuremmalle käyttäjäjoukolle.

Kun WWW-sivusto perustetaan, se ei suinkaan ole pysyvä kokonaisuus vaan vaatii jatkuvaa ylläpitoa ja ajanmukaistamista. Tyypillinen sivuston kehityskaari voisi edetä esimerkiksi seuraavasti:

1. Aluksi yrityksen muutamaa staattista esittelysivua ylläpidetään tekstieditorilla. Ylläpidosta vastaa yrityksen IT-osasto muiden töidensä ohella.
2. Seuraavassa vaiheessa sisällön ylläpitovastuuta jaetaan ja käyttöön otetaan jokin WYSIWYG-editori, esimerkiksi Dreamweaver. Näin sisällön ylläpito onnistuu myös vähemmän teknisesti suuntautuneilta käyttäjiltä.
3. Sisällön määrän kasvaessa staattisten sivujen määrä kasvaa liian suureksi. Jonkin aikaa pärjätään, kun WYSIWYG-editorissa otetaan käyttöön sivupohjat (template) ja erilliset tyylitiedostot. Näin ulkoasun muutokset pystytään hoitamaan keskitetysti, mutta kaikista ongelmista ei vielä päästä eroon.
4. Sivusto muutetaan tietokantapohjaiseksi, jolloin rakenne, ulkoasu ja sisältö saadaan erotettua toisistaan. Ulkoasu voidaan hankkia tilaustyönä suunnittelutoimistosta ja sisällön ylläpitoa varten rakennetaan esimerkiksi PHP-pohjainen järjestelmä. Tällöin sivun sisältö haetaan tietokannasta ja sijoitetaan sivupohjaan oikeaan kohtaan.
5. Sisällön ylläpitovastuun jakamista halutaan määritellä yhä tarkemmin. Myös uudet Web-tekniikat kiinnostavat ja sivustolle halutaan lisää interaktiivisuutta. Tässä vaiheessa huomataan, että vanha itse rakennettu järjestelmä ei enää taivu-

kaan uusiin vaatimuksiin ilman työlästä uudelleensuunnittelua. Lisäksi järjestelmän aikoinaan ohjelmoinut henkilö on jo toisen yrityksen palveluksessa. Ratkaisu löytyy valmiista sisällönhallintajärjestelmästä, joka tarjoaa hyväksi todetut työkalut kaikkiin yleisiin käyttötarpeisiin. Käyttäjäoikeudet, erilaiset sisältötyypit ja ulkoasun hallinta onnistuvat helposti selainkäyttöisen käyttöliittymän kautta.

Yllä esitetty kuvaus on esimerkki yhdentyyppisen sivuston mahdollisesta kehityskaaresta. Vaikka sisällönhallintajärjestelmät ovat jo arkipäivää, on aivan mahdollista ja hyväksyttävää, että uusi sivusto aloitetaan edellä kuvatusta vaiheesta 1. Yksinkertaista ja harvoin päivittyvää sivustoa on turha rakentaa monimutkaisella sisällönhallintajärjestelmällä, jos sen ominaisuuksia ei tarvita.

Täysin uudenlaiset sivustot täytyy luonnollisesti toteuttaa itse alusta lähtien. Tällöin toteutusta ohjaa sivuston kantava idea. Valmiista sisällönhallintajärjestelmistä ei siis ole hyötyä, jos halutaan luoda uusi Facebook tai Youtube. Suurin osa verkkosivustoista on kuitenkin luonteeltaan sellaisia, että ne ovat toteutettavissa jollakin valmiilla järjestelmällä.

2.2.1. Staattiset ja dynaamiset sivustot

Edellisen kohdan esimerkissä verkkosivuston evoluutiosta suurin yksittäinen muutos tapahtui siirryttäessä staattisesta sivustosta dynaamiseen. Se merkitsee yleensä sivuston ja työskentelytapojen täydellistä remonttia. Aiemmin sivuja muokattiin työasemalle asennetulla editorilla, nyt siirrytään selaimella käytettävään työkaluun. Ennen sivusto oli kokoelma erillisiä HTML-sivuja. Dynaamisella sivustolla sisältö tallennetaan tietokantaan, josta se noudetaan ja muotoillaan kyseistä sisältötyyppiä varten sovitetulle sivupohjalle.

Staattisen sivuston rakenteelliset rajoitukset muodostuvat ennemmin tai myöhemmin ylivoimaiseksi ongelmaksi. Aikoinaan tehty sisältöhierarkia ei enää toimi ja uudelleenorganisointi on työlästä. Sisällön kategorisoinnin ongelmat sivuston laajentuessa saattavat jopa johtaa tilanteeseen, jossa edes sivuston ylläpitäjät eivät tiedä paljonko sivuilla on tietoa ja mistä sitä pitäisi etsiä [Chou, 2002]. Vierailijoille oikean tiedon löytäminen voi olla lähes mahdotonta.

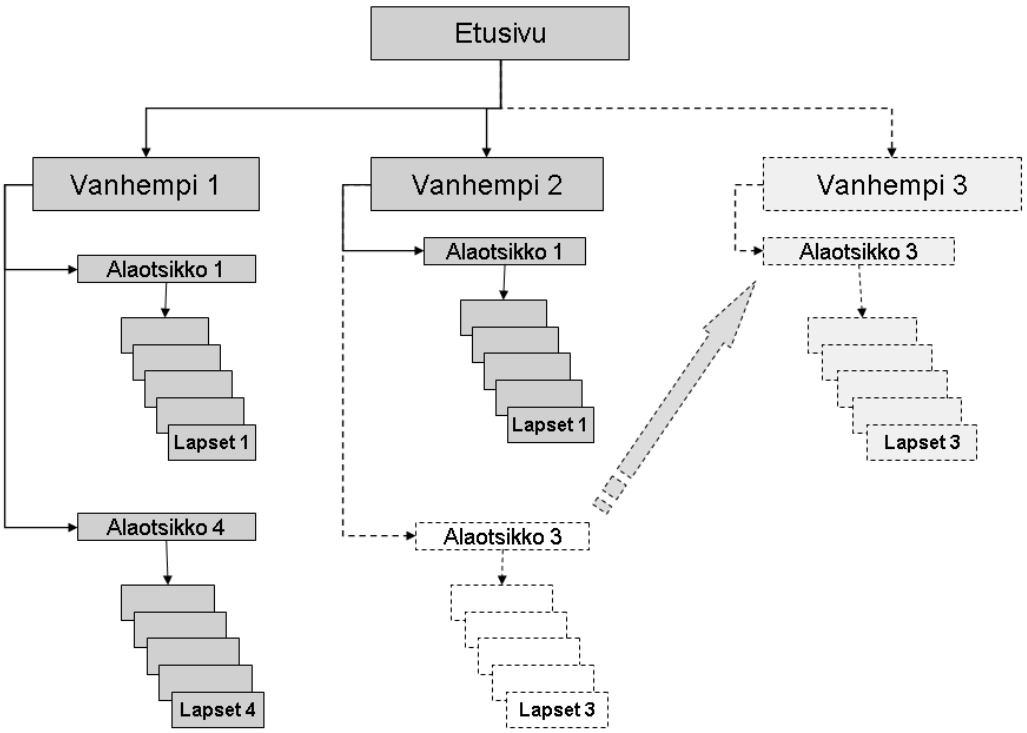
Staattisen sivuston rakenne voidaan järjestää uudelleen pitäen sivuston toteutus silti entisenlaisena. Tällainen uudelleenorganisointi ei auta kuin hetkeksi sisällön määrän kasvaessa. On vain ajan kysymys milloin tulee tarvetta joko uusille sisältötyypeille tai uudenlaiselle jaottelulle.

Joitakin staattisuuden haittoja on osittain mahdollista kiertää huolellisella suunnittelulla, vaikka sivustoa ei muutettaisikaan dynaamiseksi. Esimerkiksi linkit sivuston muihin osiin joudutaan silti korjaamaan staattista sivustoa järjestellessä. Dynaamiseen si-

vustoon vaihtamista onkin syytä ryhtyä pohtimaan, jos edessä näyttää joka tapauksessa olevan suuritöinen uudelleenorganisointi.

Sivuston sisällöstä riippuen nopein tapa vaihtaa dynaamiseen sivustoon voi olla jonkin sisällönhallintajärjestelmän käyttöönotto. Tietokantapohjainen järjestelmä mahdollistaa perinteisen navigoinnin lisäksi hakutoiminnon ja taksonomioihin perustuvan tiedon etsimisen [Brown et al., 2005]. Näin sisältö saadaan ryhmiteltyä uudestaan aina kulloisenkin käyttötilanteen mukaan.

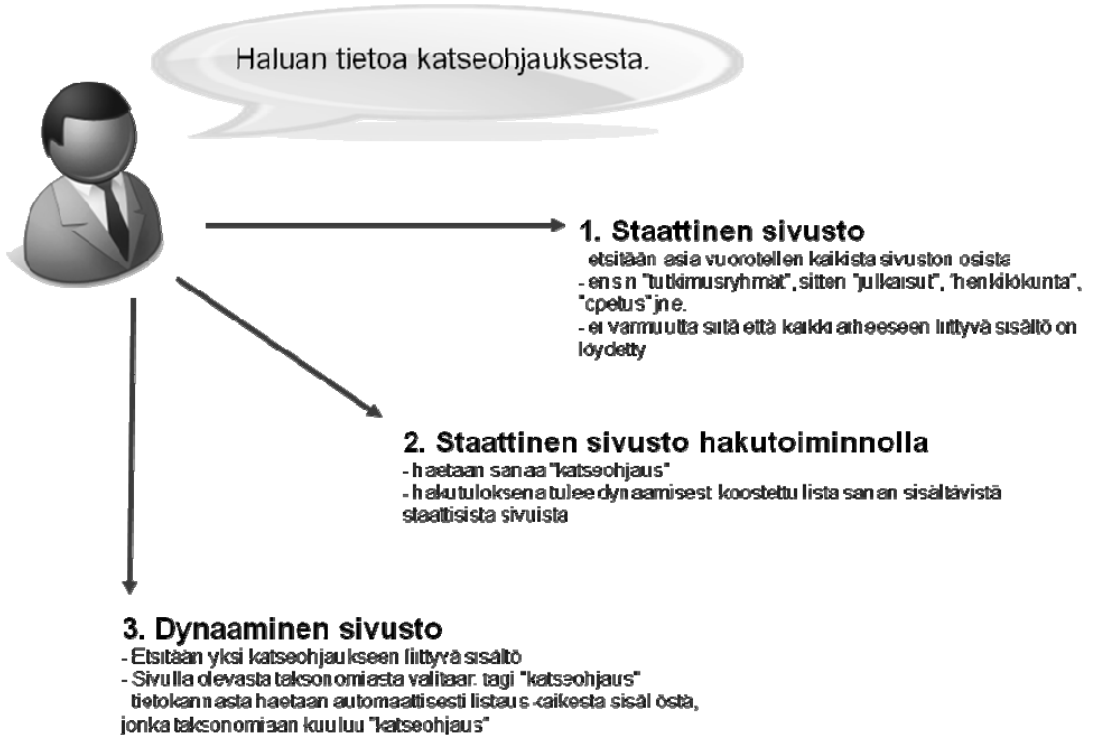
Myös dynaamiselle sivustolle voidaan luoda rakenteita, jotka edesauttavat sisällön pysymistä järjestyksessä. Samalla saatetaan ainakin näennäisesti menettää osa dynaamisuuden mahdollistamasta joustavuudesta. Toimintamalli on kuitenkin perusteltu silloin, kun sivusto uhkaa rönsyillä tai sisältö on oikeasti järjestettävissä loogisiin kokonaisuuksiin. Kuvassa 1 esitetään yksi mahdollinen vaihtoehto dynaamisen verkkosivuston sisällön organisointiin.



Kuva 1 Puumaisen sivurakenteen uudelleenorganisointi

Kuvassa 1 nähdään puurakenne, jossa jokaisella sivulla etusivua lukuun ottamatta on yläsivu (vanhempi) ja mahdollisesti alasivuja (lapsia) [Niederacher, 1999]. Staattiseen sivustoon verrattuna tämä malli on selvästi joustavampi. Jos esimerkiksi huomataan, että jokin osa sivustosta (”Alaotsikko 3”) olisi hyvä irrottaa omaksi kokonaisuudekseen, riittää kun osion ”vanhimman” sivun vanhempi vaihdetaan. Kuvassa 1 on luotu uusi

osio, ”Vanhempi 3”, ja kaikki alasivut siirtyvät automaattisesti oikeaan kohtaan vanhempansa (”Alaotsikko 3”) mukana. Joissakin järjestelmissä puumalli on automaattisesti käytössä. Toisissa taas sisältösivut luodaan täysin irrallisina yksikköinä, joihin ei välttämättä ole mitään pääsyä muuten kuin kirjoittamalla sivun URL-osoite selaimen osoiteriville.



Kuva 2. Tiedon etsintä staattisesta ja dynaamisesta sivustosta.

Kuvassa 2 on kuvattu yksinkertainen käyttötapaus, jossa käyttäjä tulee sivustolle etsiäkseen tietoa katseohjauksesta. Staattisen sivuston tapauksessa sivuston valmiiksi määritelty rakenne asettaa rajoituksia sille, miten tietoa voidaan hakea. Dynaamisessa sivustossa sisältö voidaan listata myös esimerkiksi taksonomian mukaan. Dynaamiselle sivustolle tyypilliseen tapaan navigointivalikossa näkyvä sisällön järjestelytapa ei rajoita tai sulje pois muita lajittelutapoja. Täysin staattisen ja dynaamisen sivuston välistä löytyy vielä staattinen sivusto, johon on toteutettu hakutoiminto. Tällöin hakutuloksissa näytetään linkit hakusanan sisältäviin staattisiin sivuihin. Varsinaiset sisältösivut kärsivät edelleen staattisuuden rajoitteista.

Yksi esimerkki dynaamisuuden tarjoamista navigointimahdollisuuksista on ns. ”tagipilvi” (tag cloud) joka visualisoi sivustolla käytetyt tagit esimerkiksi niiden suosion mukaan.

Kuva 3. Esimerkki tagipilvestä (kuva haettu osoitteesta http://en.wikipedia.org/wiki/Image:Web_2.0_Map.svg).

Kuvassa 3 on esimerkki tagipilvestä jossa useimmin esiintyvät tagit ovat isommalla ja vaaleammalla tekstillä kuin harvemmin käytetyt. Sen avulla pystyy nopeasti muodostamaan kokonaiskuvan sivuston tarjoamasta sisällöstä [Rivadeneira et al., 2007]. Arvio ei tietenkään ole ehdottoman luotettava, mutta siitä on hyötyä vieraan sivuston sisällön hahmottamisessa. Tagipilven visualisointia ei ole tutkittu vielä paljoa, mutta tekstikoko näyttäisi olevan merkittävin tekijä tagien havaitsemisessa, kunhan kasvattamisessa ei mennä liiallisuuksiin [Halvey ja Keane, 2007]. Tagipilvi on voi olla sisällönhallintajärjestelmissä vakiovarusteena tai hankittavissa lisämoduulina [Tagadelic, 2008].

Sisällönhallintajärjestelmät perustuvat dynaamisiin WWW-sivuihin. Täysin staattisten ja dynaamisten sivujen välissä on vielä yksi toteutustapa, jossa staattinen sisältö on erotettu staattisista sivupohjista. Jos esimerkiksi jokin sivu halutaan antaa muokattavaksi kolmannelle osapuolelle, riittää kun sille toimitetaan kyseisen sivun sisältö. Kun tämä tiedosto tulee täydennettynä takaisin, se on heti valmis julkaistavaksi. Jos taas sivu olisi toteutettu sisällönhallintajärjestelmällä, sisältö täytyisi ensin kopioida tekstieditoriin, toimittaa muokkaajalle ja palautuksen jälkeen leikata ja liittää takaisin sisällönhallintajärjestelmään [Inplace CMS, 2008]. Jos sivusto koostuu vain staattisista sivuista ja se halutaan pitää helpommin hallittavana, tällainen ratkaisu voi olla perusteltu. Kyseessä ei kuitenkaan ole oikea sisällönhallintajärjestelmä, joten esimerkiksi käyttäjäoikeuksien määrittely ei ole mahdollista.

3. Verkkosivuston uudistusprojekti

Verkkosivusto ei ole koskaan "valmis", kuten esimerkiksi painotuote. Se elää ja muuttuu koko ajan. Kun vierailijan halutaan tulevan sivustolle uudestaankin, sisällön ajantasaisuus ja oikeellisuus on tärkeää. Vaikka muuta sisältöä sivuille ei tulisikaan, pitäisi ainakin uutisosastoa tai tapahtumakalenteria päivittää säännöllisesti. Tällöin vierailija näkee, että sivustoa edelleen ylläpidetään ja sinne kannattaa tulla uudestaankin.

Verkkosivuston pitäminen nykyaikaisena ja käyttäjille houkuttelevana voi olla yksi uudistusprojektin aloitukseen johtava syy. Sisällöstä riippumatta uusi kävijä kokee ulkoasun tärkeimmäksi yksittäiseksi verkkosivuston uskottavuuteen vaikuttavaksi tekijäksi [Fogg et al., 2003]. Ensivaikutelma kestää pitkään ja siihen vaikuttaa sisällön visuaalinen suunnittelu.

Mittavaan uudistukseen tarvitaan myös muita syitä, kuten sivuston rakenteelliset tai sisällön ylläpitoon liittyvät ongelmat. Rakenteellisia ongelmia voidaan ratkoa esimerkiksi tekemällä sivustosta dynaaminen. Se ei yksinään riitä, jos rakennetta joudutaan silti korjailemaan usein. On pyrittävä löytämään ne syyt, jotka aiheuttavat uudelleenorganisoinnin tarpeen. Sisällön ylläpitoon liittyviä ongelmia voi olla erityyppisiä. Syy sisällön huonoon tilaan voi olla esimerkiksi hankalassa ylläpitokäyttöliittymässä tai siinä, että ylläpitovastuuta ei ole voitu kohdentaa oikeille tahoille. Mikäli sisällön päivittäminen vaatii kattavat oikeudet koko sivustoon, ylläpitovastuuta ei voida antaa kaikille niille, joille se muuten luonnollisesti kuuluisi. Tällöin esimerkiksi IT-osasto voi joutua huolehtimaan sisällön päivittämisestä.

3.1. Alkukartoitus

Uudistusprojektin alussa on ensin tehtävä selvitys, jossa käsitellään nykyisen sivuston tila, uusimisen syyt, kohdeyleisö, sisältö, teknologia ja päivittäminen. On löydettävä vastaukset esimerkiksi seuraaviin kysymyksiin [Goto ja Cotler, 2003, s. 39-40]:

- **Nykyinen sivusto.** Mitkä sivuston osa-alueet ovat erityisen onnistuneita? Miksi? Mitä puutteita nykyisessä sivustossa on? Jos olisi mahdollista muuttaa kolme asiaa sivustossa jo tänään, mitkä ne olisivat?
- **Uusimisen syyt.** Miksi sivusto uusitaan (vanhentunut sivusto, laajennetut palvelut, erilainen yleisö)? Mitkä ovat ensisijaiset toiminnalliset tavoitteet joihin sivuston uusimisella pyritään? Entä toissijaiset? Mihin ongelmiin sivuston uusimisella haetaan ratkaisua?
- **Yleisö.** Millainen on sivuston tyypillinen käyttäjä? Mitkä ovat ensisijaisia toimintoja joihin käyttäjä ryhtyy sivustolle tultuaan? Millaiset käyttäjämäärät sivustolla on? Kuinka niitä mitataan? Miten huomioidaan saavutettavuus tai mobiililaitteet?

- **Sisältö.** Käytetäänkö uudessa sivustossa jo olemassa olevaa sisältöä nykyisiltä sivuilta? Mikä on sivuston perusrakenne ja miten se on organisoitu? Millaisia visuaalisia elementtejä halutaan ottaa mukaan nykyisestä sivustosta (logo, väriteema, navigointi, nimeämiskäytännöt)? Miten uuden sivuston sisältö ja toiminnallisuus eroavat nykyisestä sivustosta?
- **Teknologia ja resurssit.** Onko kohdealusta tai -selain tiedossa? Mitä tekniikoita sivustolla halutaan käyttää (Flash, javascript, multimedia)? Miten ne parantavat käyttäjäkokemusta? Mitä ohjelmistoja sivusto käyttää? Mitkä ovat tietoturvavaatimukset? Mitä erikoistoimintoja tarvitaan? Löytyykö organisaation sisältä toteuttajia vai halutaanko mieluummin mahdollisimman valmis avaimet käteen -ratkaisu?
- **Markkinointi ja päivittäminen.** Miten uudistetusta sivustosta on tarkoitus tiedottaa? Miten sivustoa aiotaan jatkossa päivittää? Kuka on vastuussa sivuston päivittämisestä ja sisällön toimittamisesta?

Yksi tapa sivuston laadun varmistamiseen on auditointi. Sen vaiheet lyhyesti lueteltuna ovat tiedon keruu, tilanneanalyysi ja raportti suosituksineen. Auditointi on tehokas tapa aloittaa uudistusprojekti [Deshpande et al., 2002]. Alkukartoituksesta se eroaa erityisesti siten, että vaatimukset ja sivuston ominaisuudet määritellään yksityiskohtaisemmin, samoin kriteerit niiden saavuttamiselle. Säännöllisen auditoinnin avulla verkkosivusto saadaan pidettyä vaatimusten mukaisena myös uudistusprojektin päättymisen jälkeen.

Uudistusprojektin alussa on hyvä suorittaa myös tehtäväanalyysi. Käyttäjän rooliin asettuminen paljastaa sivuston merkitykselliset osat ja auttaa näin myös kehitysresurssien kohdentamisessa. Tehtäväanalyysi voidaan aloittaa arvioimalla nykyistä sivustoa. Sen tukemia tehtäviä luetteloidessa tulee mieleen myös tehtäviä joita sivusto ei tue. Yrityksen tai yhteisön tuottamaan painettuun materiaaliin on myös syytä tutustua. Esitteitä varten on todennäköisesti tehty tehtäväanalyysi siitä, mitä tietoja organisaatiota aiemmin tuntematon asiakas siitä haluaa saada. Vanhan sivuston sivuhaussa käytettyjen hakusanojen analysointi auttaa niinikään yleisimpien tehtävien määrittelyssä [Ritter et al., 2005]. Lisäksi on mahdollista selvittää millaisista osoitteista sivustolle on saavuttu.

WWW-palvelimen lokeihin tallentuu ns. HTTP referer -kenttä. Siitä nähdään sen sivun URL-osoite, jolla olevan linkin valitsemalla käyttäjä on saapunut sivustolle. Näin voidaan saada tietoa esimerkiksi yleisiin hakukoneisiin kirjoitetuista hakusanoista jotka johtavat vierailuun organisaation sivuilla. Jo tuettujen tehtävien ja hakusanojen analysoinnin lisäksi voidaan luonnollisesti haastatella käyttäjiä eri kohderyhmistä. Käyttäjille voidaan myös ehdottaa jo suunniteltuja tehtäviä. Joukossa voi mahdollisesti olla tehtäviä, joita kukaan ei käytännössä tekisikään [Ritter et al., 2005]. Haastattelujen kattavuudesta ja perusteellisuudesta riippuen on syytä tarkistaa tällaisen harvoin suoritetun tehtävän tärkeysastetta sivuston toteutusta suunniteltaessa.

Ulkoasun päivittäminen on yksi tapa osoittaa, että ylläpitäjä panostaa sivustoon. Mikäli dynaaminen verkkosivusto on suunniteltu huolella, uuden visuaalisen ilmeen toteuttaminen on melko helppoa. Staattisilla sivustoilla muutosten teko voi osoittautua ongelmalliseksi ja työlääksikin. Tällöin voi olla syytä tarkastella sivuston toteutustapaa ja miettiä, olisiko työskentely- ja toteutustapojen muuttamisesta hyötyä.

Ulkoasua uudistettaessa on kuitenkin varmistettava, että organisaation jo olemassa olevaa brändiä ei samalla hukata. Sivustolle palaavan vierailijan on pystyttävä yhdistämään vanha ja uusi sivusto saman organisaation kotisivuksi. Tämä vaikuttaa myös tuntemukseen sisällön luotettavuudesta. Tunnettujen organisaatioiden sivustot koetaan miellyttävämpinä ja luotettavampina kuin vähemmän tunnettujen [Roberts et al., 2003].

Lähtökartoituksessa on syytä huomioida myös sivuston käytettävyysnäkökohdat. Verkkosivuston käyttökokemus on lopulta se, mihin kannattaa kiinnittää huomiota. Jos jokin erikoiselta tuntuva sisällön ryhmittely on havaittu toimivaksi, sitä kannattaa käyttää vaikkei se suunnittelijasta tuntuisikaan erityisen loogiselta [Chou, 2002].

3.2. Suunnittelu

Suunnitteluvaiheessa ei keskitytä vain sivuston visuaalisuuteen. Itse asiassa visuaalinen suunnittelu on kokonaisuus, joka voidaan jopa antaa ulkopuolisen graafikon huoleksi silloin, kun sivuston olemus alkaa muuten hahmottua. Visuaalisen suunnittelun tueksi luodaan sivuluonnoksia ja luonnostellaan mm. mahdollinen värimaailma, tausta, käytettävät kirjasimet, kuvakkeet ja muut grafiikat. Sivuston suunnitteluun kuuluu oleellisena osana myös käyttöliittymän ja arkkitehtuurin kehittäminen. Tällöin päätetään mm.

- nimeämiskäytännöt,
- navigaatio,
- URL-suunnittelu ja
- tietoturva.

Nimeämiskäytännöillä tarkoitetaan tässä sitä, että sivustolle määritellään yhtenäinen tapa eri elementtien (sivuotsikot, navigointivalikot, lyhenteet) nimeämiseksi. Suunnittelutyöstä tulee sujuvampaa kun tällaiset hidastavat asiat on ensin sovittu eikä niistä tarvitse kiistellä muissa työvaiheissa. Johdonmukainen ja huolellinen nimeämiskäytäntö lisää sivuston uskottavuutta. Se myös parantaa hakutoiminnon toimivuutta ja vähentää väärinkäsityksien vaaraa erityisesti silloin, kun aihepiiri on lukijalle vieras [Chou, 2002].

Navigointia suunniteltaessa on huomioitava, että tulevan sivuston käyttäjä viettää suurimman osan verkossa käyttämästään ajasta muilla sivustoilla. On siis tarpeen, että navigointi tuntuu heti ”tutulta” eikä sen opetteluun tarvitse käyttää aikaa. Suunnittelussa on huomioitava ainakin se, että käyttäjä tietää missä päin sivustoa ollaan, mitä on lähistöllä ja miten päästään etusivulle. Jos sivu ei ole juuri sitä mitä haetaan, voidaan

silti nopealla vilkaisulla nähdä, että lähistöltä (esimerkiksi saman pääotsikon alta) löytyy haettu tieto [Nielsen, 2000].

Kun suunnitelmista on saatu tarpeeksi kattava käsitys sivustosta, kokonaisuus hahmotellaan esimerkiksi kuvakäsikirjoituksena [Griffiths, 2002]. Näin saadaan ensimmäinen kunnollinen näkemys siitä, miltä uusi sivusto tulee näyttämään ja mitä se tulee sisältämään.

Sivuston tulevat ominaisuudet ovat nyt pääpiirteissään selvillä, joten tiedetään mihin valittavan sisällönhallintajärjestelmän on kyettävä. URL-osoitteiden suunnittelu ja tietoturvan määrittely ovat vaiheita, joiden ratkaisuihin vaikuttaa käytettäväksi valittava sisällönhallintajärjestelmä. Viimeistään nyt on siis syytä ryhtyä järjestelmien evaluointiin. Ominaisuuksien vertailu paperilla auttaa alkuun, mutta käytännössä tällainen vertailu ei paljasta läheskään kaikkia järjestelmien eroja. Kunnollisen käsityksen saa vasta kun kehittäjät voivat tutustua eri vaihtoehtojen toimintaan käytännössä. Kaikkien evaluoitavien järjestelmien asentaminen ja toimintakuntoon saattaminen on hidasta. Siksi ensimmäisellä karsintakierroksella voidaan käyttää esimerkiksi sivustoja, jonne on asennettu useita järjestelmiä evaluointia ja harjoittelua varten [Open source CMS, 2008]. Kaupallisten järjestelmien toimittajat rakentavat pyydettäessä demoympäristön ja tarjoavat suunnitelman sivuston toteutuksesta annettujen määritelmien pohjalta.

Sisällönhallintajärjestelmän valinta voi osoittautua ongelmalliseksi osaksi koko verkkosivuston uudistusprojektia. Voisi jopa olla parempi suorittaa uudistus (tai ainakin sen suunnittelu kokonaisuudessaan) ennen järjestelmän valintaa. Valittavan järjestelmän ei pitäisi antaa vaikuttaa suunnittelutyöhön. Uuden sivuston lopullinen suunnitelma on itse asiassa yksi sisällönhallintajärjestelmän valintakriteereistä. Sivuston toteutusta ei saa rajoittaa valittavan sisällönhallintajärjestelmän rajoitusten vuoksi, vaan järjestelmällä on pystyttävä luomaan suunnitelman mukainen sivusto [Robertson, 2006a]. Sisällönhallintajärjestelmän valintaan keskitytään tarkemmin luvussa 4.

3.3. Kehitystyö

Tämän vaiheen alussa käytettävä sisällönhallintajärjestelmä on valittu ja sivuston ominaisuudet määritelty. Käytössä on ”hiekkalaatikkosivusto”, jossa tehdyt muutokset testataan. Varsinaista sivustoa voidaan alkaa toteuttaa mielellään vasta kun kaikki ominaisuudet ja toiminnot ovat käytössä hiekkalaatikossa. Muussa tapauksessa julkaistavaksi aiotusta sivustosta tuleekin toinen testisivusto. Tämä ei sinänsä ole pahasta kunhan maltetaan toteuttaa myös kolmas, lopullinen sivusto huolimatta mahdollisista aikatauluongelmista.

3.4. Käyttöönotto

Käyttöönotto tapahtuu kun vanhan sivuston säilytettävä materiaali on siirretty ja uusi järjestelmä on toimintakuntoinen. Uutta sisältöä voidaan lisätä myös vasta käyttöönoton jälkeen. Samaten voidaan varautua siihen, että järjestelmän toiminnallisuuttakin hienosäädetään kun toimivuus käytännössä alkaa selvitä.

Jotta sivuston käyttöönotto ei lykkäytyisi tarpeettomasti, on syytä päättää virallinen käyttöönottopäivä, jolloin vanhasta sivustosta luovutaan ja uusi sivusto avataan. Uuden sivuston toteutuksesta vastaava työryhmä ei kuitenkaan aina voi vaikuttaa projektin onnistumiseen. Mukana on myös muita osapuolia kuten organisaation johto, jonka toimet voivat olla tarpeen esimerkiksi palkkaus- tai vastuukysymyksissä. Tuloksena voikin olla tilanne, jossa uusi sivusto on kyllä julkaistu, mutta sen ylläpidosta ja sisällön oikeellisuudesta ei vastaa kukaan [Boucher and Smith, 2000].

4. Sisällönhallintajärjestelmän valinta

Edellisessä luvussa käytiin läpi verkkosivuston uudistamisprojektin eri vaiheet. Tässä luvussa tarkastellaan niitä tekijöitä, joita on syytä huomioida sisällönhallintajärjestelmää valittaessa. Järjestelmän valinta käynnistyy uudistamisprojektin suunnitteluvaiheen aikana. Tällöin on kartoitettu vanhan sivuston säilytettävä sisältö ja ollaan selvillä uudistetun sivuston olemuksesta, ominaisuuksista, kohderyhmästä sekä käytettävissä olevista resursseista.

Sisällönhallintajärjestelmän valinnassa on tiedostettava sivuston keskeiset tehtävät. Useat valmiit järjestelmät hallitsevat monen tyyppisen sisällön hallinnan, mutta on myös sellaisia järjestelmiä jotka keskittyvät vain yhteen aihepiiriin, esimerkiksi blogeihin. Monitaitoisilla järjestelmillä on luonnollisesti omat vahvuutensa ja heikkoutensa. Esimerkiksi keskinkertaisen keskustelufoorumin tarjoava järjestelmä ei ehkä ole perusteltu valinta sivustolle, jonka keskeisiä toimintoja on jäsenten välinen viestintä. Tarjolla olevilla järjestelmillä on myös oma lähestymistapansa sisällönhallintaan siitä riippuen, millaiset lähtökohdat niillä on. Samanlaiseen lopputulokseen voidaan päästä hyvin erilaisilla toteutuksilla.

Sisällönhallintajärjestelmän valintaa tehtäessä on huolellisesti mietittävä, mitkä itse asiassa ovat niitä kysymyksiä joita kannattaa esittää. Yleinen ominaisuusluettelo voi osoittautua turhaksi, koska kaikki vaatimukset täyttäviä järjestelmiä löytyy todennäköisesti runsaasti. Sen sijaan kannattaa keskittyä löytämään oikeat laadulliset kriteerit, jotka järjestelmän on täytettävä. Voi myös olla hyödyllistä arvioida, miten tärkeimmät ominaisuudet järjestelmässä on toteutettu [Robertson, 2006b]. Esimerkiksi aikanaan keskustelufoorumista laajenemaan lähtenyt sisällönhallintajärjestelmä voi toimia hyvin, mutta jotkin toiminnot voivat olla hyvin vaikeita toteuttaa alustassa olevien rajoitusten vuoksi.

4.1. Toteutettavan sivuston keskeiset tehtävät

Toteutettavan sivuston tyyppi vaikuttaa siihen, mitkä sisällönhallintajärjestelmän ominaisuudet ovat erityisen tärkeitä ja mitkä voi jättää vähemmälle huomiolle. Tämä on hyödyllistä pitää mielessä sisällönhallintajärjestelmää valittaessa. Seuraavassa on muutamista lähteistä ([Brain, 2005], [Lipman, 2003], [Xavier Uni. Libray], [Alexandrou, 2004]) johdettu lista erityyppisistä verkkosivustoista.

1. **Henkilökohtainen blogisivusto.** Päiväkirjamaisen blogin perustaminen ja ylläpito on tarvittaessa erittäin helppoa. Esimerkiksi Blogger tarjoaa mahdollisuuden luoda oma blogi parissa minuutissa [Blogger, 2008]. Myös sisällönhallinta-

järjestelmissä on työkalut blogin luomiseen, mutta pelkästään siihen käyttöön ne ovat ylimitoitettuja.

2. **Esittelysivusto.** Esittelysivustoa voidaan käyttää esimerkiksi yrityksen mainostamiseen. Sivustolla löytyy ainakin yhteystiedot ja lyhyt esittely yrityksen toiminnasta. Yksinkertaisimmillaan tällainen sivusto on joukko staattisia HTML-sivuja. Nopein tapa saada kevyt esittelysivusto aikaiseksi on käyttää jotakin WYSIWYG-editoria, esimerkiksi Dreamweaveria. Nykyaikaisilla editoreilla on mahdollista käyttää sivustopohjia ja CSS-tyylejä, eikä niitä käyttävän päivittäjän tarvitse hallita HTML-kieltä.
3. **Yhteisön sivusto.** Tyypillinen yhteisön kokoontumispaikka internetissä on yleensä rakentunut keskustelufoorumin ympärille. Yksi tunnetuimmista keskustelufoorumijärjestelmistä on vBulletin [vBulletin, 2008].
4. **Tietokantasivusto.** Tietokantatyyppisten sivustojen kategoriaan kuuluvat esimerkiksi tietosanakirjat, hakukoneet ja karttapalvelut. Tällainen sivusto vaatii omaa ohjelmointityötä, mutta aivan kaikkea ei tarvitse tehdä itse. Esimerkiksi asunnonvälittäjän asunnonhakupalvelussa voidaan käyttää toisten toimittajien palveluita mashup-tyyppisesti. Omaan sivustoon voidaan upottaa toisaalta haettava kartta ja liittää siihen myytävien asuntojen osoitteet.
5. **Kauppapaikka.** Verkossa toimivan kaupan pystyttäminen ei sekään vaadi enää ohjelmointityötä, vaan tarjolla on monenlaisia järjestelmiä, niin ilmaisia kuin maksullisiakin. Valmiin kaupan pystyttämisessä suurimmaksi työvaiheeksi jää helpoimmillaan sivuston ulkoasusuunnittelu [Tähtinen, 2007].
6. **Portaali.** Portaali on ”aloituspaikka” tai portti muiden tahojen palveluihin. Linkkilistojen lisäksi portaalissa voi olla esimerkiksi päivän uutiset, säätiedot, linkit palveluntarjoajan palveluihin ja sähköpostiin. Siinä voi olla myös itse tuotettua sisältöä kuten uutisia tai artikkeleita. Yhteisöjen käytössä portaali ja keskustelufoorumit voivat täydentää toisiaan ja monessa tapauksessa niiden välinen rajanveto voikin olla hankalaa.
7. **Oppimisympäristö.** Verkko-oppimisympäristö on pitkälle erikoistunut sisällönhallintajärjestelmä. Siinä on piirteitä niin yhteisöstä kuin tietokannastakin, mutta sisällön organisointi ja toiminnallisuus eroavat silti niistä suuresti. Oppimisympäristöille ominaista on kursseihin perustuva sisällön luokittelu ja erikoistoiminnot kuten selainkäyttöiset testit, kyselyt ja kokeet. Myös oppimisympäristöjä on saatavana eri valmiusasteilla. Nykyään tunnetuin oppimisympäristö on Moodle [Moodle, 2008], jota voidaan käyttää myös sisällönhallintajärjestelmänä [Buckinghamshire, 2008].
8. **Wiki.** Wiki on sivusto, jonka sisällön tekevät käyttäjät itse. Käyttäjät voivat luoda uusia sivuja ja muokata jo olemassaolevia sivuja. Monesti käytössä on myös versionhallinta, jolloin sivuihin tehtyjä muutoksia voidaan seurata. Koska monia

wikejä voidaan muokata myös ilman kirjautumista, saadaan versionhallinnan kautta palautettua mahdollisen ilkivallan kohteeksi joutuneet sivut. Valmiita wiki-ohjelmistoja on tarjolla erilaisiin tarpeisiin. Wikin käyttöönotto onkin yksinkertaisimmillaan hyvin helppoa.

Vaikka sivuston keskeiset tehtävät ohjaavat sisällönhallintajärjestelmän valintaa, ei niiden pidä vaikuttaa ratkaisuun liikaa. Vaikka järjestelmän pohjana olisi blogisivusto ja muutkin sisältösivut koostettaisiin blogimerkintöjä muotoillen, ratkaisu voi olla aivan toimiva [Weitzman et al., 2006]. Etuna tällaisessa toteutustavassa on se, että yhteen sisältötyyppiin keskittyvät järjestelmät ovat selkeämpiä käyttää kuin laajat ja monipuoliset vaihtoehdot. Kuitenkin yksittäisiä sisältöjä on mahdollista muotoilla siten, että ne saadaan näyttämään toisentyyppisiltä. Näin esimerkiksi blogimerkinnästä voidaan oikeilla asetuksilla ja ulkoasulla muotoilla vaikkapa yrityksen esittely- tai yhteystietosivu.

4.2. Erityyppisiä ratkaisuja

Sisällönhallintajärjestelmän valmiusaste voidaan valita riippuen omista tarpeista ja resursseista. Valittavana on vaihtoehtoja valmiista järjestelmistä (”lisää vain sisältö”) Web-sovelluskehyksiin. Työtä pelkäämättömälle on myös vaihtoehtona ohjelmoida koko järjestelmä alusta lähtien itse.

Sivuston tyypistä riippuen tarjolla on myös ylläpitopalveluita, jotka tarjoavat valmiin alustan suosituimmille sisällönhallintajärjestelmille. Listasta vain valitaan haluttu järjestelmä, joka asentuu automaattisesti ja on heti käyttövalmis [Swarma, 2008]. Tämän jälkeen voidaan ryhtyä suoraan muokkaamaan järjestelmää tai tuottamaan sisältöä. Henkilökohtaista kotisivua varten ei yleensä myöskään tarvitse ryhtyä rakentamaan sivustoa alusta lähtien. Valittavissa on erilaisia yhteisöjä joihin liittymällä saa helppokäyttöiset välineet oman (joskin ominaisuuksiltaan rajatun) kotisivun koostamiseen [Myspace, 2008].

4.2.1. Valmis sisällönhallintajärjestelmä

Sisällönhallinta- ja julkaisujärjestelmät ovat perinteisesti olleet kalliita järjestelmiä. Avoimen lähdekoodin ja joustavien lisenssiehtojen yleistyttyä on kuitenkin mahdollista saada itse järjestelmä ilmaiseksi ja maksaa tarvittaessa ylläpidosta tai erikoistarpeisiin tehdystä kehitystyöstä. Kaupallisia järjestelmiä ovat esimerkiksi EpiServer ja EpiX. Yleisiä avoimen lähdekoodin järjestelmiä taas ovat Drupal, Joomla ja Plone. Sisällönhallintajärjestelmät voivat olla joustavia eri tavoin: joko ne tarjoavat heti monipuoliset työkalut erityyppisten sisältöjen tuottamiseen, tai ne ovat muokattavissa ja laajennettavissa tai ne keskittyvät tekemään tietyn osa-alueen mahdollisimman hyvin.

Kaupalliset sisällönhallintajärjestelmät voivat olla toteutukseltaan hyvin erilaisia kuin avoimen lähdekoodin järjestelmät. Esimerkiksi Episerver [Episerver, 2007] vaatii

toimiakseen myöskin kaupalliset Windows Server -käyttöjärjestelmän ja SQL Server -tietokannan. Vastineeksi se tarjoaa saumattoman integroinnin Active Directory -käyttäjätietokantaan ja hakemistopalveluun sekä esimerkiksi Microsoft Sharepointiin, joka on selainkäyttöinen ryhmätyötila ja dokumentinhallintajärjestelmä. Ratkaisu on luotettava ja sen takana on alalla jo pitkään toimineita suuryrityksiä. Näin toteutetun järjestelmän hinta on vähintään tuhansia euroja pelkkien lisenssimaksujen osalta, tosin käytetylle rahamäärälle saa myös vastinetta. Kaupallisia sisällönhallintajärjestelmiä ei tässä tutkimuksessa käsitellä sen tarkemmin, vaan painopiste on vapaan lähdekoodin järjestelmissä.

Yksi hyvä aloituspaikka vapaan lähdekoodin sisällönhallintajärjestelmää valitessa on OpenSourceCMS [Open source CMS, 2008], jossa on esitelty kymmeniä erilaisia vapaan lähdekoodin sisällönhallintajärjestelmiä. Käyttäjäarviot ja vinkit ovat avuksi järjestelmän valinnassa, mutta niiden lisäksi sivusto sisältää toimivat demoversiot esiteltävistä järjestelmistä. Sisällönhallintajärjestelmät ovat vapaasti käytettävissä ja niiden tila ”nollataan” aina muutaman tunnin välein. Tällaisessa palvelussa voidaan vertailla suurta määrää järjestelmiä lyhyessä ajassa, koska asennusta ei tarvitse tehdä. Toinen etu on siinä, että tarkempaan vertailuun valittavista järjestelmistä pystytään muodostamaan todenmukaisempi kuva verrattuna siihen, että vertailtaisiin ominaisuusluetteloita tai luettaisiin muiden tekemiä vertailuja.

4.2.2. Web-sovelluskehys

Markkinoilla on myös sovelluskehyksiä jotka tarjoavat valmiita komponentteja sovelluksen rakentamiseen. Tällöin sisällönhallintajärjestelmä voidaan suunnitella juuri omia tarpeita vastaavaksi, mutta jokaista perustoimintoa ei tarvitse tehdä itse. Sisällönhallintajärjestelmän pohjaksi sopii esimerkiksi Python-pohjainen Django [Django, 2008], jota käytetään mm. verkkolehtien julkaisuun.

Web-sovelluskehykset tarjoavat rajapintoja erilaisten toimintojen hoitamiseen. Esimerkiksi tietojen tallentaminen ja hakeminen tietokannasta toteutetaan sovelluskehyksen tarjoamaa rajapintaa käyttäen. Tällöin ohjelmoijan ei tarvitse välittää siitä, mikä tietokanta palvelimelle on asennettu. Hänen tarvitsee vain kirjoittaa tietokantaoperaatiot sovelluskehykselle, joka kääntää ne palvelimella käytettävän tietokantaohjelmiston ymmärtämään muotoon.

Tietokantarajapinnan lisäksi tyypillinen Web-sovelluskehys tarjoaa työkalut mm. käyttäjätunnistukseen, URL-osoitteiden suunnitteluun ja sisällön liittämiseen sivupohjiin. Djangossa on lisäksi valmis käyttöliittymä tietokannan, käyttäjien ja käyttäjäryhmien hallintaan. Web-sovelluskehyksiä on toteutettu erilaisilla ohjelmointikielillä kuten Java, PHP, Perl ja Ruby. Yksi sovelluskehyksen valintaa ohjaava tekijä onkin se, minkä ohjelmointikielen sisällönhallintajärjestelmän toteuttava kehittäjä hallitsee parhaiten.

4.2.3. Itse ohjelmoitava järjestelmä

Yksi äkkiseltään mielenkiintoinen vaihtoehto on toteuttaa koko järjestelmä alusta lähtien omin voimin. Tämä ratkaisu vaatii nykyään poikkeuksellisen hyvät perustelut, koska sovelluskehyksiä ja valmiita sisällönhallintajärjestelmiä on tarjolla lähes millaiseen käyttötarkoitukseen tahansa, kaikissa hintaluokassa. Suunnitteluvaihe on itse toteutetuissa järjestelmissä ratkaisevan tärkeä. On otettava tarkkaan huomioon kasvun tarpeet ja mietittävä miten järjestelmää on tarpeen ja mahdollista laajentaa kun tarvitaan uudentyyppistä sisältöä. Itse toteutetussa järjestelmässä on tärkeää huomioida hyvät vientiominaisuudet, joilla järjestelmän vaihto tulevaisuudessa onnistuu. Tällainen toiminnallisuus on tietysti toteutettavissa myöhemminkin, olettaen että järjestelmän suunnittelija on vielä käytettävissä.

4.2.4. Johtopäätökset

Yleensä järkevin työkalu verkkosivuston uudistukseen on joko valmis sisällönhallintajärjestelmä tai sovelluskehyksen päälle rakennettu järjestelmä. Tällöin ongelmatilanteessa ei olla pelkästään oman kehittäjätiimin varassa, vaan tarjolla on joko kaupallista tukea tai osaava kehittäjäyhteisö. Tässä tutkimuksessa tutustutaan muutamiin vapaan lähdekoodin vaihtoehtoihin. Varsinaisista sisällönhallintajärjestelmistä vertailussa ovat Drupal ja Joomla, Web-sovelluskehyksiä edustaa Django.

4.3. Ylläpito- ja tukipalvelut

Kaupallisten sisällönhallintajärjestelmien ylläpitoon ja muokkaukseen on saatavissa maksullista tukea valmistajalta ja monesti myös kolmansilta osapuolilta. Tällaisen järjestelmän etuna on mielikuva vakaudesta, koska sen takana on jokin selkeästi määriteltävä taho eli tuotteen valmistanut yritys. Myös suosituimmille avoimen lähdekoodin järjestelmille on tarjolla kaupallista tukea. Järjestelmän kaupallisuudesta tai avoimuudesta riippumatta niiden ympärille muodostuu yleensä virtuaalinen käyttäjä- tai kehittäjäyhteisö. Keskustelufoorumeilta voi hakea ja saada apua ongelmiin. Monet käyttäjät tuottavat dokumentaatiota ja ohjeistuksia ja julkaisevat ne yleisesti käytettäväksi. Alkujaan esimerkiksi sisäiseen käyttöön tarkoitetut ohjeet on havaittu hyviksi ja ne julkaistaan valmistajan tukifoorumeilla. Tämän jälkeen muut voivat tehdä niihin korjauksia ja lisäyksiä, jolloin ohjeen alkuperäinen tekijäkin hyötyy tilanteesta myös käytännössä. Motivaationa tälle vapaaehtoistyölle on useimmiten halu henkilökohtaiseen kehittymiseen sekä omien tietojen ja taitojen jakamiseen [Kuznetsov, 2006].

Täysin omaan ohjelmakoodiin perustuva järjestelmä vaatii kaiken toiminnallisuuden suunnittelemista itse. Tähän ei yleensä ole syytä ryhtyä elleivät tulevan järjestelmän vaatimukset ole niin erikoiset, ettei valmiista järjestelmistä ole apua ilman mittavaa muokkaamista. Varsinkin omaan koodiin perustuvissa järjestelmissä dokumentoin-

nin osuus kasvaa tärkeäksi, koska ongelmatilanteissa ulkopuolista apua ei ole saatavissa ja ratkaisut on keksittävä itse.

4.4. Sisällönhallintajärjestelmien yleisiä ominaisuuksia

Tässä luvussa käsitellään joitakin yleisiä ominaisuuksia, joiden toteutusta valittavassa sisällönhallintajärjestelmässä kannattaa arvioida suhteutettuna omiin tarpeisiin. Yksi suomalaisittain tärkeä arvioinnin kohde on se, miten järjestelmässä on toteutettu monikielisen sisällön aiheuttamat haasteet.

4.4.1. Monikielisyys

Kansainvälinen verkkoyhteisö on toistaiseksi vahvasti englanninkielinen. Tämä ei kuitenkaan anna syytä väheksyä muiden kielten asemaa. Mikäli verkkosivustosta on tulossa yksikielinen, ei sisällönhallintajärjestelmän valinnalla yleensä ole merkitystä. Jos taas on tarkoitus tuottaa sisältöä useammalla kielellä, asia monimutkaistuu huomattavasti. Pelkkä tekstien kääntäminen toiselle kielelle ei riitä, vaan lisäksi on huomioitava esimerkiksi valuutan ja päivämäärien muotoilu [Parr, 2006]. Miten monikielinen sisältö ja sen ylläpito organisoidaan? Jos jostain sisältösivusta on olemassa versioita eri kielillä, on mietittävä miten ylläpito toteutetaan. Jossakin järjestelmässä toteutus voi olla sellainen, että jokainen kieliversio on muokattavissa samanaikaisesti yhdellä ainoalla sivulla. Jossain toisessa järjestelmässä saattaa näkyä vain sen kielinen versio kuin muokkaajalla itsellään on käytössä.

Staattisilla verkkosivustoilla erikielisten sivujen käsittely on loogisesti helppoa. Voidaan esimerkiksi määritellä, että suomenkielinen sivusto löytyy osoitteesta http://www.osoite.fi/ ja ruotsinkielinen osoitteesta http://www.osoite.se/. Nämä sivut voidaan määritellä omiksi kansioikseen www-palvelimella. Tällöin kyseessä on itse asiassa erilliset sivustot, joten kolmikielisen verkkosivuston ylläpidossa on lähes kolminkertainen työmäärä yksikielisen sivuston ylläpitoon verrattuna.

Sisällönhallintajärjestelmissä monikielisyydellä voidaan tarkoittaa muutamaa erilaista asiaa. Yksinkertaisimmillaan järjestelmästä käännetään vain käyttöliittymä, jolloin navigointi ja muut käyttöliittymäelementit saa vaihdettua haluamalleen kielelle. Tämä on helposti toteutettavissa ja kuuluu yleensä järjestelmien perusominaisuuksiin. Suositut sisällönhallintajärjestelmät on yleensä jo valmiiksi käännetty jopa kymmenille kielille. Monimutkaisemmaksi asian tekee se, että myös sisältö pitäisi saada käännettyä tarpeiden mukaan. Tämä voidaan toteuttaa muutamalla eri tavalla. Yksi vaihtoehto on määritellä kieliversio sivuston URL-osoitteessa, jolloin sisällönhallintajärjestelmä hoitaa oikean sivun näyttämisen osoitteen perusteella. Toisessa tavassa käyttäjän kielivalinta tallennetaan evästeeseen tai tietokantaan käyttäjätietoihin. Tällöin erikielisten sivujen osoite pysyy samana ja järjestelmä päättelee evästeen tai tietokantamerkinnän perusteella näytettävän sivun kielen.

Joissakin sisällönhallintajärjestelmissä saman sivun erikieliset vastineet ovat järjestelmän näkökulmasta täysin erillisiä sisältöjä. Tällöin käyttäjän on itse pidettävä huolta siitä, että erikieliset sivut linkittyvät oikein ja että sivuston rakenne pysyy hallittavissa. Toinen tapa on tallentaa sisällön kieliversiot samaan säiliöön, jolloin niitä pystytään tarkastelemaan muokkaustilassa vierekkäin [Episerver käsikirja, 2007, s. 89-95]. Tällöin järjestelmä huolehtii erikielisten sisältöjen organisoinnista, eikä käyttäjän tarvitse kantaa huolta rakenteen sekaantumisesta.

4.4.2. Tietoturva

Sisällönhallintajärjestelmälle asetettavat tietoturvavaatimukset ovat vaihtelevia, kuten myös tietoturvallisuuden toteutustavat. Yksinkertaisimmillaan sivuston koko sisältö on kaikille avointa. Toisessa ääripäässä sivustolta vaaditaan vahvaa tietoturvaa ja sisältöä varjellaan loppuun asti. Tällaisissa tapauksissa järjestelmän turvallisuuden arviointiin on käytettävissä tietoturvaohjeistus, joka määrittelee vaatimukset hyvinkin tarkasti.

Sisällönhallintajärjestelmällä on luonnollisesti suuri merkitys tietoturvan suhteen, mutta pelkästään sen asetuksilla ei vielä saavuteta turvallista sivustoa. Erikseen on harkittava, tarvitaanko esimerkiksi kirjautumista tai jopa koko sivustoa varten esimerkiksi SSL-tekniikalla salattu yhteys. Sisällönhallintajärjestelmän asetuksiin voidaan määritellä esimerkiksi, että sivuston muokkausoikeudet myönnetään vain tietyille käyttäjille silloin, kun kirjautuminen on tapahtunut organisaation sisäverkosta.

4.4.3. Käyttäjäkokemus

Sisällönhallintajärjestelmien ylläpitokäyttöliittymiä ei juuri ole mahdollista muokata. Vaikka ulkoasuun voitaisiinkin hieman vaikuttaa, järjestelmän toteutus ja toimintaperiaatteet asettavat rajoja käytettävyydelle. Järjestelmää valitessa onkin syytä määritellä, millaisille käyttäjille myönnetään järjestelmän ylläpito-oikeudet. Mikäli vain muutamalla käyttäjällä on oikeus toimia hankalan ylläpitokäyttöliittymän puolella, heidän osaamistaan on helpohkoa kontrolloida esimerkiksi koulutuksia järjestämällä. Tilanne muodostuu hankalaksi, jos ylläpitäjäjoukko on suuri ja osa joukosta käyttää ylläpitokäyttöliittymää harvoin. Tällöin ylläpidon käytettävyyteen on pakko kiinnittää enemmän huomiota.

Yksi käyttäjäkokemukseen vaikuttava tekijä on järjestelmän nopeus. Toimintaperiaatteesta riippuen yksittäinen käyttäjälle näytettävä sivu saattaa sisältää kymmeniä hakuja tietokantaan. Lisäksi sivulta ladattavat javascript-koodit hidastavat käyttöä selaimesta riippuen. Sivuston optimointia voidaan suorittaa monin tavoin, esimerkiksi WWW-palvelimen välimuistiasetuksia säätämällä ja lähdekoodin automaattisella optimoinnilla. Parhaassa tapauksessa esimerkiksi PHP-lähdekoodin optimoinnilla sivujen latausnopeus voi yli kaksinkertaistua [Carlin, 2006].

Monet sisällönhallintajärjestelmät ovat selainriippuvaisia erityisesti sivuja ylläpidettäessä. Käyttäjäkokemuksen kannalta on ongelmallista, jos sivujen muokkausta ei voi tehdä ensisijaisella selaimella. Jos ylläpitokäyttöliittymän toteutuksessa on käytetty esimerkiksi ActiveX-tekniikkaa, sivuja voidaan ylläpitää ainoastaan Internet Explorer -selaimella Windows-käyttöjärjestelmässä. Normaalisti sivujen näyttämisessä ei pitäisi olla ongelmia eri selainten kesken, vaan ongelma koskee nimenomaan ylläpitokäyttöliittymää.

4.5. Riskien arviointi

Sisällönhallintajärjestelmää valittaessa on myös pohdittava sitä, millaisia vaikutuksia sillä on tulevaisuudessa sivuston ylläpitoon ja jatkuvuuteen. Järjestelmiä arvioitaessa on huomioitava ainakin seuraavat kysymykset:

- miten isot muutokset ja päivitykset hoidetaan,
- millainen kehitystahti sovelluksella on ja
- mitä tapahtuu jos valitun järjestelmän kehitystyö päättyy?

Näihin kysymyksiin vaikuttaa mm. se, millaista tukea järjestelmän ylläpitoon on saatavilla ja millainen yritys tai yhteisö sen taustalla on. Tulevaisuuden kannalta tärkeä valintakriteeri on myös se, miten järjestelmään tallennettu sisältö on siirrettävissä muualle. Esimerkiksi XML-muotoisena saatava sisältö saadaan tuotua toiseen ympäristöön, jos järjestelmää joudutaan vaihtamaan esimerkiksi tuen tai kehityksen loppuessa tai ominaisuuksien jäädessä riittämättömiksi.

Voidaan myös ajatella niin, että nopeasti muuttuvalla kentällä valittu sisällönhallintajärjestelmä kelpaa joka tapauksessa vain muutamaksi vuodeksi. Tässä ajassa markkinoille voi tulla organisaation tarpeisiin paremmin vastaavia järjestelmiä. Ehkä onkin syytä tarkastella vain muutaman seuraavan vuoden tarpeita [Robertson, 2005]. Mikäli järjestelmä näyttää kattavan lähivuosien tarpeet ja vaatimukset, sitä voidaan pitää hyvänä valintana. Tällöin tietysti on entistä tärkeämpää huolehtia siitä, että sisältö saadaan ulos sellaisessa muodossa, että se on siirrettävissä toiseen järjestelmään.

4.5.1. Kaupallisten ja vapaiden ohjelmistojen eroista

Valinta kaupallisten ja vapaiden ohjelmistojen välillä tehdään nykyään liian helposti ideologisista syistä. Pääasia kuitenkin on lopputuloksena saatavan sivuston laatu. Järjestelmän avoimuus, ”sulkeutuneisuus” tai maksullisuus eivät sinänsä ole kelvollisia kriteerejä sen valinnassa. Eri asia on, vaikuttavatko nämä ominaisuudet välillisesti esimerkiksi käytettävissä olevien taloudellisten tai henkilöstöresurssien kautta.

Kaupallisen ja suljetun järjestelmän valinnut asiakas on toimittajan armoilla. On kuitenkin esimerkkejä, jossa lopetetun tuotteen lähdekoodi on avattu ja lahjoitettu yhteisön kehitettäväksi. Esimerkiksi suosittu Firefox-selain on saanut alkunsa Netscapen avattua Communicator-selaimensa lähdekoodin [Netscape, 1998].

Mikäli avoimen lähdekoodin sovelluksella on riittävän iso yhteisö takanaan, se on yleensä turvallinen valinta. Toisaalta uusi ”hienompi ja parempi” järjestelmä voi lyhyessäkin ajassa haalia ison käyttäjäkunnan kilpailijoiltaan. Mikäli kehittäjäyhteisö on vielä nuori, se voi profiloitua tiettyihin avainhenkilöihin. Tällaisen siirtyminen muualle saattaa toimia esimerkkinä suurellekin joukolle ”seuraajia”. Tämä voi joskus kääntyä myös vahvuudeksi. Linux-käyttöjärjestelmille 3D-käyttöliittymää kehittänyt Compiz-projekti on esimerkki onnistuneesta ”irtaantumisesta”.

Compiz-projekti hajosi vuonna 2006 kahdeksi erillisprojektiksi (Compiz ja Beryl) näkemyserojen vuoksi. Projektit kehittyivät eri suuntiin kunnes huomattiin, että ne olivat keskinäisen kilpailun vuoksi kehittyneet huomattavan nopeasti. Projektit päätettiin yhdistää yhteisen hyvän nimissä [Compiz Fusion, 2007]. Lopputuloksena saatiin molempien yhteisöjen parhaat ideat ja aiempaa suurempi määrä kehittäjiä.

Virtuaalisissa yhteisöissä viestit vääristyvät helposti. Sähköposti ja pikaviestimet mahdollistavat nopean kommunikoinnin paikasta riippumatta. Niillä ei kuitenkaan pystytä kunnolla välittämään sosiaalista kontekstia, jolloin väärinkäsitykset voivat johtaa ongelmiin koko yhteisössä. Tätä epävarmuustekijää on yleisesti käytetty argumenttina avoimen lähdekoodin ohjelmistokehitystä vastaan. Siinä piileekin tietty riski, jota tosin vähentää yhteisön vahva identiteetti. Usko yhteiseen asiaan – tässä tapauksessa kehitettävään ohjelmistoon – vahvistaa yhteisöä. Konfliktit syttyvät nopeasti mutta nopeiden viestimien aikakaudella ne myös ovat ratkaistavissa nopeasti [Elliot ja Scacchi, 2003].

Monet avoimen lähdekoodin järjestelmät ovat vain muutaman tekijän aikaansaannoksia. Tällaiset projektit syntyvät johonkin henkilökohtaiseen tarpeeseen ja kun se on tyydytetty tai mennyt ohi, kehittäjä ei välttämättä tunne enää lojaaliutta projektia kohtaan. Avoin lähdekoodi on kuitenkin tullut jäädäkseen, ja eräiden arvioiden mukaan viiden vuoden kuluessa jopa 90% yrityksistä käyttää tavalla tai toisella avoimen lähdekoodin tuotteita [Gartner, 2008].

Myös kaupallisia ja suljettuja ohjelmistoja tekevät ihmiset. Siksi edellä avoimen lähdekoodin järjestelmistä sanottu pätee osin myös suljetulle puolelle. Motivaationa on tällöin työstä saatava palkka ja sen tuoma vastuu. Kun asiakas ostaa tuotteen, hän odottaa siltä tiettyä jatkuvuutta investointinsa vastineeksi.

5. Kokemukset esimerkkiprojektista

Verkkosivuston uudistamisprojekti voidaan viedä läpi monella eri tavoin. Tässä tutkimuksessa käsitellään Tampereen yliopiston Tietojenkäsittelytieteiden laitoksen vuorovaikutteisen teknologian tutkimusyksikön verkkosivuston uudistamista. Pääpaino on uudistusta varten tehdyn esimerkkisivuston toteutuksessa ja sen aikana tehdyissä huomioissa.

Esimerkkiprojektin toteutukseen valittiin käytännönläheinen lähestymistapa. Drupal välineenä oli melko vieras, tosin muistakaan sisällönhallintajärjestelmistä ei ollut syvällisempää kokemusta. Tämän vuoksi todettiin, että helpoin tapa saada käsitys järjestelmän kyvyistä olisi ryhtyä toteuttamaan esimerkkisivustoa jo varhaisessa vaiheessa.

5.1. Alkutilanne

Yksikön vanhat sivut oli toteutettu tietokantapohjaisella järjestelmällä, jonka rajat olivat tulleet vastaan niin ominaisuuksien kuin ylläpidollisten seikkojen suhteen. Vanhan sivuston ylläpito koettiin hankalaksi eikä sivuilla tämän vuoksi ollut sisältöä niin paljoa kuin olisi tarpeen. Uudistusta suunnitellessa uudelle sivustolle asetettiin mm. Taulukossa 1 näkyvät tavoitteet:

Tavoite
Sisällön on oltava ajan tasalla ja ajankohtaista.
Henkilö- ja yhteystietoja ylläpidetään nyt kahdessa erillisessä paikassa. Keskitetään ylläpito yhteen kohtaan sivustossa.
Sisällöstä on hyötyä ja sitä kohdistetaan eri käyttäjäkunnille, kuten tutkijoille, opiskelijoille ja rahoittajille.
Graafinen ulkoasu joka sisältää mm. valokuvia, kaavioita ja videoleikkeitä.
Johdonmukainen visuaalinen ilme kautta koko sivuston.
Sisällön päivittämisen pitää olla helppoa.
Mahdollisuus useisiin ylläpitäjiin joilla on eritasoisia oikeuksia sisältöön.

Taulukko 1. Uuden sivuston tavoitteita

Yksikön henkilökunta koostuu opetushenkilöstöstä ja tutkijoista. Tutkimustyön avuksi on toivottu ryhmätyötä tukevia työkaluja, kuten blogeja, keskustelufoorumeita ja ryhmäkohtaiset kalenterit. Myös opetustyössä olisi hyötyä siitä, että opettaja saisi helposti julkaistua kurssisivun tai esimerkiksi luentokalvot verkossa. Tarvittiin siis uusi sisällönhallintajärjestelmä, joka tarjoaisi helpommat työkalut sisällön ylläpitoon ja tuottamiseen.

Koska sisältöä tuotetaan monille käyttäjäryhmille vierailijoista opiskelijoihin ja oman tutkimusryhmän jäseniin, sisällönhallintaoikeuksien joustava ja helppo määrittely on tärkeää. Työntekijöille haluttiin myös mahdollisuus ylläpitää omia tietojaan. Näin vältyttäisiin ylimääräisiltä välikäsiltä, kun työntekijä voi itse päivittää tai korjata tietojaan. Sivuston tulevien käyttäjien kokemukset HTML-sivujen tuottamisesta ja ylläpidosta vaihtelevat suuresti. Siksi pidettiin tärkeänä sitä, että sisällön ylläpito olisi helppoa ja ylläpito-oikeuksia pystyttäisiin rajaamaan tarkastikin tietylle sivuston osalle. Näin kokematon käyttäjä ei saisi vahingossa ainakaan kovin suurta vahinkoa aikaiseksi.

Alkukartoituksen jälkeen sisällönhallintajärjestelmäksi valittiin Drupal [Drupal, 2008]. Kartoituksen tehnyt projektiryhmä myös hahmotteli tulevaa sisällön ryhmittelyä ja loi alustavat suunnitelmat sivuston ulkoasusta. Tehtäväkseni jäi siten luoda esimerkkitoteutus sivustosta ja dokumentoida toteutus niin, että lopullisen sivuston luominen sujuisi helpommin. Projektiryhmän toiveena oli, että toteutuksessa käytettäisiin mahdollisimman pitkälti valmiina saatavia laajennuksia. Ylläpidon yksinkertaistamiseksi itse kirjoitettua ohjelmakoodia pyrittäisiin välttämään. Näin haluttiin saavuttaa tila, jossa sivuston ylläpito koostuisi sisällön päivittämisen lisäksi vain valmiiden laajennusten päivittämisestä ilman ohjelmakoodin korjailua.

5.2. Drupal

Projekti päätettiin toteuttaa Drupalin versiolla 5. Valinta oli helppo, sillä version 5 ensijulkaisu oli tapahtunut noin puoli vuotta aiemmin ja valtaosa tärkeimmistä lisämoduuleista oli jo päivitetty yhteensopivaksi sen kanssa. Projektin alkaessa kuudes versio oli jo työn alla, mutta siitä ei ollut edes esiversiota saatavilla.

Välineenä Drupal 5 osoittautui melko haastavaksi joskin myös monipuolisesti laajennettavaksi ja joustavaksi. Haasteellisuutta järjestelmän käyttöönottoon tuo tekijöiden ideologia, jonka mukaan tärkeämpää on kehittää järjestelmää vauhdilla, jopa yhteensopivuuden kustannuksella [Buytaert, 2006]. Drupal-projektin kotisivulla olevia julkaisuja elinkaari-ilmoituksia vertailemalla voidaan päätellä, että uusi rajapinnaltaan erilainen versio julkaistaan suunnilleen kerran vuodessa. Viimeisimmän version lisäksi bugikorjauksin tuetaan yleensä kahta tai kolmea uusinta versiota. Pikaisesti laskettu 2-3 vuoden elinkaari ei Drupalin tapauksessa kuitenkaan kerro koko totuutta. Koska itse pääohjelma sisältää melko rajatun valikoiman työkaluja, täytyy sen päivitystä suunniteltaessa ottaa huomioon myös lisämoduulien tilanne. Päivitys voidaan tehdä vasta sitten, kun sivustossa käytetyt lisämoduulit on päivitetty uuden Drupal-version kanssa yhteensopiviksi.

5.2.1. Drupalin käsitteitä

Seuraavaksi käydään läpi muutamia Drupalin keskeisiä käsitteitä. Toteutustavasta riippuen eri sisällönhallintajärjestelmät saattavat käyttää lähes samoista asioista erilaisia

nimiä. Esimerkiksi Drupalissa yksittäinen sisältö on nimeltään solmu. Muissa järjestelmissä vastaava termi voisi olla esimerkiksi ”id” tai ”page”.

- **Solmu.** Drupalin perussisältöyksikkö on nimeltään *node* eli solmu. Kaikki järjestelmään tallennettu sisältö tallentuu solmuina: käyttäjät, staattiset sivut, kalenterimerkinnät jne. Solmuun viitataan suoraan sen tietokantaan tallennetulla id-numerolla, esimerkiksi http://testisivusto.org/node/4. Ilman lisämoduuleita asennetussa Drupalissa solmuihin ei talleteta kovinkaan paljoa tietoa, mutta solmun ominaisuuksia voidaan kasvattaa tarpeen mukaan lisämoduuleilla.
- **Moduuli.** Moduulit luovat tietokantaan oman taulunsa ja lisäävät solmuihin viittauksen tauluun. Jotkin moduulit voivat olla ylemmän tason moduuleita joihin taas alamoduulit voivat viitata. Näin saadaan aikaiseksi hyvin joustava järjestelmä, joka toisaalta on myös haavoittuvainen. Moduulit ovat yleensä harrastajien omaan käyttöönsä rakentamia lisäosia, joilla pyritään ratkomaan spesifejä ongelmia ja joiden ohjelmointirajapintojen säilyvyydestä ei ole takeita.
- **Näkymä.** Drupalin tapauksessa näkymällä tarkoitetaan sisältötyypin esitystapaa. Esimerkiksi blogimerkinnöillä voi olla useita näkymiä: yksittäinen blogimerkintä, luettelo yhden kirjoittajan blogimerkinnöistä ja luettelo kaikkien kirjoittajien uusimmista blogimerkinnöistä.
- **Sivupohja (template).** Yksi Drupalin vahvuuksista on sen joustava sivupohjiin pohjautuva ulkoasujärjestelmä. Sivuston yleiset elementit on määritelty omassa tiedostossaan. Mikäli näkymäkohtaisesti halutaan lisäksi poiketa sisällön oletusmuotoilusta, voidaan kullekin näkymälle luoda oma muokattu sivupohja. Tällöin sisällön määrittely tapahtuu muutamassa näkymän mukaan nimetyssä tiedostossa. Jos Drupalin sivupohjalle tuottamia tietoja halutaan muokata, sekin on mahdollista. Esimerkiksi artikkeliluettelon tarjoavalla sivulla voidaan typistää tekstin runko määrättyyn sanamäärään ja vaihtaa kirjoittajan nimen tilalle pelkät nimikirjaimet. Seuraavaksi näkymälle haetaan mahdollisia erikoismäärittelyjä sisältävä CSS-tyylitiedosto. Lopuksi määritellään PHP-kielellä ja HTML-muotoilukomentoja käyttäen, miten sisältö pitäisi muotoilla. Drupalin sivupohjajärjestelmän vahvuus on siinä, että kokonaisen sisältötyypin ulkoasu voidaan muokata hyvin pienellä vaivalla. Monesti vain muutaman rivin mittaisella sivupohjatiedostolla saadaan aikaan halutut muutokset.

5.2.2. Tekniikkaa

Drupal on suunniteltu toimivaksi erilaisilla Web-palvelimilla ja tietokannoilla. Käytännössä suositeltavin yhdistelmä on Apache-palvelinohjelmisto ja Mysql-tietokanta sekä uusin 5.2-versio PHP-ohjelmointikielestä [Drupal requirements, 2008]. Vaikka itse Drupal onkin testattu toimivaksi myös Microsoftin palvelinohjelmistolla ja Postgresql-tietokannalla, voi moduulien toiminnassa olla ongelmia näitä käytettäessä. Tämä todettiin myös käytännössä, kun esimerkkisivustoa alettiin koostaa käyttäen tietokantana Postgresql-tietokantaa. Tietokanta jouduttiin pian vaihtamaan Mysql:ään. Monet moduulit käyttivät ohjeiden vastaisesti Mysql-spesifejä käskyjä ja toimivat virheellisesti muilla tietokannoilla.

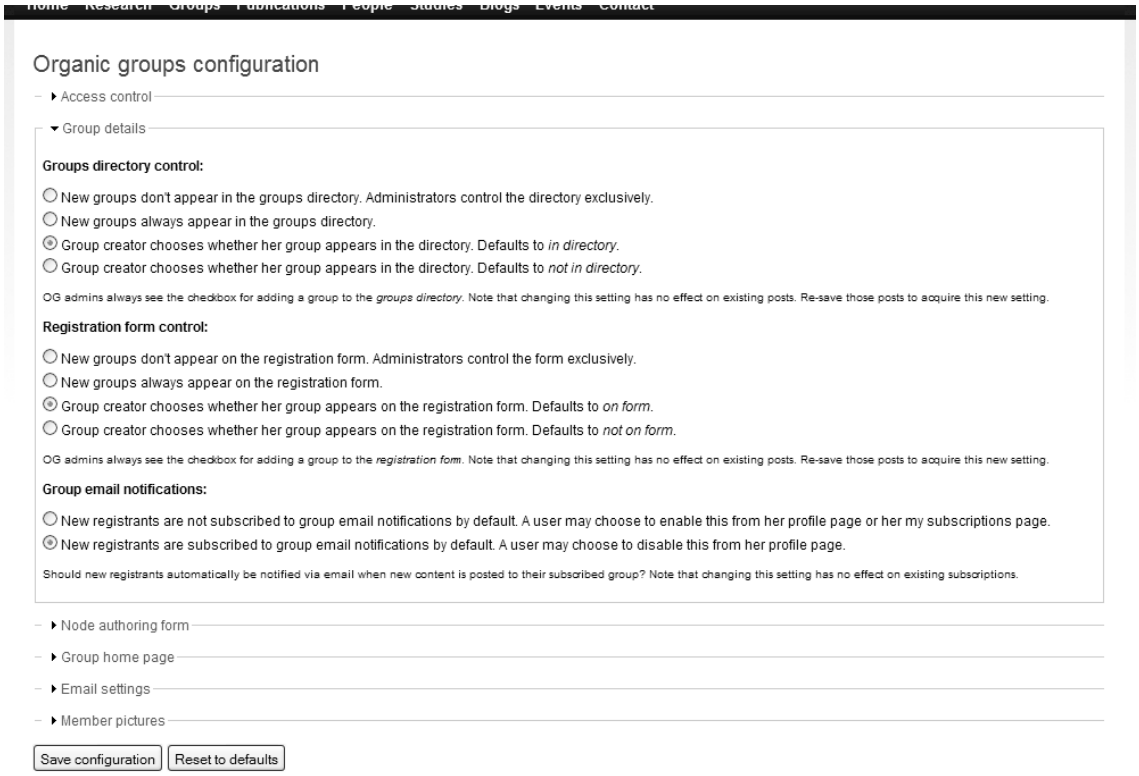
5.2.3. Ulkoasu ja saavutettavuus

Järjestelmän valinta osoittautui hyväksi myös saavutettavuuden suhteen: eräät muut sisällönhallintajärjestelmät käyttävät ulkoasun määrittelyyn edelleen sisäkkäisiä taulukoita [Weitzman et al., 2006]. Ennen CSS-tyylien yleistymistä tämä oli ainoa tapa muotoilla verkkosivujen ulkoasua. Varsinaisten asettelukäskyjen puuttuessa ratkaisuna oli sijoittaa sivun elementit (tekstikappaleet, kuvat, navigointivalikko) sivun kokoisen taulukon soluihin. Yleensä tarkkaan muotoilluilla sivuilla oli useita sisäkkäisiä taulukoita. Tällaiset sivut ovat epäkäytännöllisiä esimerkiksi näkövammaisten käyttämille ruudunlukuohjelmille. Ohjelmat lukevat sivulta kaikki merkit järjestyksessä [Näkövammaisten keskusliitto, 2008] ja taulukoiden käyttö sotkee yleensä tekstin loogisen rakenteen. Drupal käyttää sivun muotoiluun pelkästään CSS-tyylejä, jotka eivät haittaa ruudunlukuohjelman toimintaa.

Kun Drupalin muodostaman Web-sivun näyttää ilman CSS-tyylejä (esimerkiksi Firefox-selaimella valitaan *Näytä > Sivun tyylit > Ei tyylejä*), huomataan että joka sivun alku sisältää vasemman palstan navigointivalikon ja vasta tämän jälkeen varsinaisen sivukohtaisen sisällön. Esimerkkisivustossa tätä ongelmaa on kierretty siten, että kirjautumattomalla käyttäjällä navigointivalikko näkyy sivun oikeanpuoleisessa palstassa keskimmäisen sisältöpalstan jälkeen.

Kuva 4. Sivuston normaalinäkymä (1) ja ilman tyylitiedostoa (2).

Kuvasta 4 nähdään, että sisältö joka tyylejä käyttäen mahtuu jopa kerralla näytölle (näkymä nro 1), ”näkyy” ruudunlukijalle hyvin erilaisena (näkymä nro 2). Theofanos [2003] ehdottaa sivun alkuun sijoitettavaa linkkiä ”skip to main content” (siirry pääsisältöön). Tämän valitsemalla ruudunlukijaa käyttävä käyttäjä pääsee ohittamaan jokaisella sivulla toistuvat elementit ja siirtymään suoraan sisältöön.



Kuva 5. Drupalin ylläpitokäyttölittymä

Drupalin ylläpitokäyttöliittymän sivut näyttävät melko tavanomaisilta WWW-lomakkeilta, kuten kuvasta 5 nähdään. Käyttöliittymän muokkausmahdollisuudet rajoittuvat värien ja tekstityylien valintaan. Toteutus on selkeä, vaikka tekstiä onkin ajoittain runsaasti. Joidenkin muiden sisällönhallintajärjestelmien suosimia kuvakkeita tai alasvetovalikoita ei Drupalista löydy.

6. Esimerkkiprojektin toteutus

Tässä luvussa käsitellään sivuston uudistusprojektin sitä osaa, jossa luotiin esimerkkisivusto varsinaisen toteutuksen pohjaksi. Koska Drupal on modulaarinen järjestelmä, ohjelmakoodia ei oikeastaan tarvinnut kirjoittaa. Suurin osa työstä kului eri moduulien testaamiseen, valintaan ja yhteensovittamiseen. Mikäli valmiita moduuleita ei olisi löytynyt, olisi ne toteutettu itse. Muutamaa moduulia käytettiin hieman eri tavalla kuin moduulin tekijä on tarkoittanut, mutta se ei välttämättä tarkoita, että tapa olisi huono. Vaarana on, että ratkaisu ei toimi toivotulla tavalla enää Drupalin seuraavassa versiossa. Joka tapauksessa Drupalin versionvaihdoksessa on yleensä odotettavissa niin isoja muutoksia, että se vaatii runsaasti käsityötä ja tarkistuksia.

6.1. Toteutuksen vaiheet

Esimerkkisivuston toteutuksessa oli tavoitteena noudattaa yleisiä sivustoprojektin toteutusperiaatteita, joista tosin joustettiin melko paljon. Esimerkiksi käytettävyystestausta pitäisi suorittaa koko projektin ajan heti alusta lähtien [Krug, 2006, s. 135]. Tässä tapauksessa sivuston käytettävyyttä ja navigointia oli hahmoteltu projektiryhmän luonnoksissa. Toteutusvaiheessa edettiin luonnosten ohjaamana, mutta myös omaan vaistoon luottaen. Sivuston ulkoasu oli tässä vaiheessa vielä vailla lopullista muotoaan. Varsinaista visuaalista suunnittelua varten harkittiin ulkopuolisen graafisen suunnittelijan käyttämistä.

Projektin aikana oli yksi palaveri suunnitteluryhmän kesken sekä useampia keskusteluita ryhmän vetäjän kanssa. Vasta kun esimerkkisivusto alkoi olla lähes valmis, sitä esiteltiin yksikön koko henkilökunnalle. Tässä ainoassa kokouksessa käytiin sivuston toiminnallisuus läpi melko yksityiskohtaisesti. Kommenteissa ja keskustelussa ei tullut ilmi yllättäviä asioita jotka olisivat vaatineet suuria muutoksia.

Koska kenelläkään projektiin osallistuneista ei ollut syvällisempää Drupal-osaamista, päätettiin heti alkumäärittelyjen jälkeen ryhtyä toteuttamaan esimerkkisivustoa. Näin järjestelmän potentiaali ja rajoitukset saatiin selvitettyä käytännössä. Ratkaisu osoittautui suhteellisen hyväksi. Käyttäjäoikeuksia ja näkymiä olisi etukäteen voitu määritellä hieman tarkemminkin. Tämä olisi kuitenkin ollut melko vaikeaa, koska Drupalin kyvyistä ei ollut etukäteen tarkkaa tietoa.

6.2. Moduulien valinta

Seuraavassa käydään läpi esimerkkisivustoon valitut lisämoduulit. Samalla on tarkoitus tuoda esille Drupalin toimintaperiaatteita, toteutusprosessin vaiheita ja yleisesti toteutuksessa huomioitavia asioita. Taulukossa 1 on lueteltu asennetut moduulit ja niiden kuvaukset.

Moduuli	Kuvaus
Content Access	Tarjoaa työkalut sisällön näkyvyyden hallintaan.
CCK - Content Creation Kit	Lähes kaikkiin Drupal-sivustoihin asennettava moduuli. Mahdollistaa uusien sisältötyyppien luomisen ja muokkaamisen. Tuo myös uusia kenttätyyppejä solmuihin, esim. päivämäärä ja käyttäjätunnukseen viittaaminen.
Event, Event views	Mahdollistaa uudet sisältötyypit: tapahtumat ja kalenterin. Event views tarjoaa valmiita näkymiä, esimerkiksi kalenterisivun ja viimeisimpien tapahtumien listauksen.
Image	Työkalut kuvien siirtämiseen ja hallinnointiin. Lisää solmun muokkaussivulle haluttujen kenttien yhteyteen kuvakkeen, jonka kautta kuvia pystytään siirtämään palvelimelle ja liittämään tekstin sekaan.
Inline	Mahdollistaa tiedostojen (kuvien) liittämisen suoraan solmun tekstin joukkoon. Normaalisti Drupalissa on mahdollisuus ainoastaan solmuun liitettäviin ja siitä erikseen ladattaviin liitetiedostoihin.
Javascript tools	Muutamia javascript-työkaluja käytettävyyden parantamiseksi. Ominaisuuksia ovat esimerkiksi harvoin käytettävien elementtien romahduttaminen pieneksi ja popup-kalenteri päivämäärän valitsemiseksi.
Ldapauth	LDAP-autentikointi. Tarjoaa mahdollisuuden kirjautua Drupaliin LDAP-käyttäjätunnusta käyttäen. Moduuli on useiden muiden tavoin keskeneräinen. Se toimii ilmeisesti riittävän hyvin kehittäjänsä omassa ympäristössä, joten kehitystyö on lähes pysähtynyt.
Organic groups, Organic roles	Mahdollistaa käyttäjäryhmien luomisen. Ryhmillä on oma koostesivu jossa näkyy ryhmän viimeisimmät tapahtumat. Kaikelle sisällölle voidaan määritellä ryhmä, jolle se on tarkoitettu. Ryhmälle voidaan myös luoda esimerkiksi omat keskustelufoorumit.
Usernode	Tekee automaattisesti uuden solmun jokaista käyttäjätunnusta kohti. Tämä mahdollistaa sen, että käyttäjän tietoihin on helpompi viitata muista sisällöistä.
Views	Mahdollistaa mukautettujen näkymien rakentamisen helposti hiirellä valitsemalla. Ensin valitaan sisältötyyppi joka halutaan näkyväksi (esimerkiksi blogimerkintä), tämän jälkeen valitaan haettavat tiedot (esim. otsikko, teksti, kirjoittaja ja päivämäärä). Tämän jälkeen Views rakentaa sopivan näkymän, joka voidaan vielä muotoilla Views Theme Wizardin tuottaman PHP/html/CSS-koodin avulla.
Workflow, Workflow access	Moduuli on tarkoitettu solmun työstämisen ja julkaisun työvaiheiden rytmittämiseen. Tässä sitä ei kuitenkaan käytetä alkuperäiseen käyttötarkoitukseensa vaan osana solmujen

	näkyvyyden hallintaa. Workflow-moduulia käyttäen pystytään määrittelemään solmun näkyvyys niin anonyymien kuin autentikoitujen käyttäjäryhmien kesken helposti: editointisivulle saadaan radionapeilla toteutettu toiminto, josta valitaan kenelle solmu näkyy.

Taulukko 2. Sivustoon asennetut lisämoduulit.

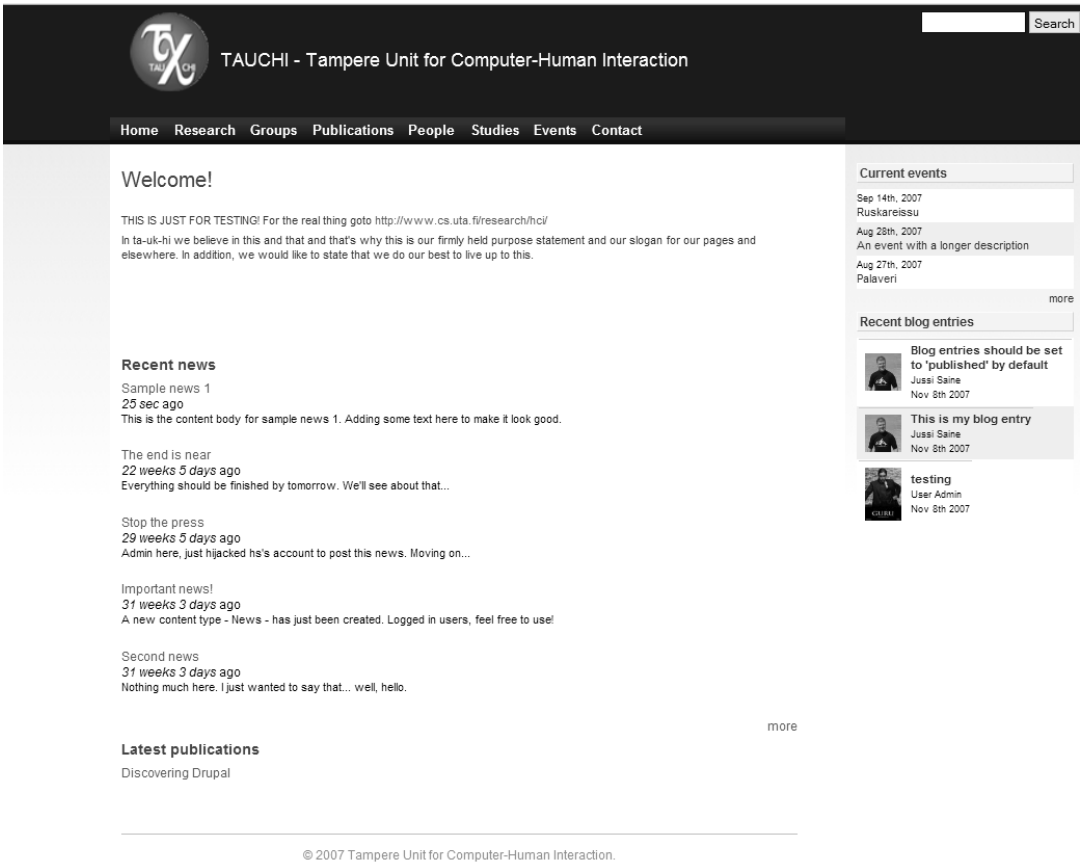
Kuten taulukosta 2 näkyy, jopa melko tavanomaisella sivustolla tarvitaan suhteellisen suuri määrä lisämoduuleita. Drupalissa on havaittavissa jopa Web-sovelluskehysmäisiä piirteitä. Sovelluskehyksen ideanahan on tarjota sivuston toteutusta nopeuttavia ”rakennuspalikoita”, jotta kaikkea toiminnallisuutta ei tarvitsisi toteuttaa alusta lähtien. Drupal vie toteutuksen vielä askeleen sovelluskehyksiä pidemmälle, koska normaalitilanteessa Drupal-pohjaisen sivuston toteutuksessa ei oikeastaan tarvitse tehdä ohjelmointityötä. Toinen tällaista sovelluskehysajattelua tukeva huomio taulukosta on se, että Drupaliin on asennettava lisäosana sellaista toiminnallisuutta, joka useimmissa muissa järjestelmissä kuuluu ilman muuta perusvarustukseen. Esimerkkinä tästä mainittakoon ryhmäominaisuudet (Organic Groups) ja sisällön oikeuksien määrittely (Content Access). Drupalin vahvuus on siinä, että näihinkin toimintoihin on valittavissa erilaisia moduuleita. Tällöin saadaan varmasti valittua kuhunkin tarkoitukseen sopivin työkalu.

6.3. Räätälöityjä osioita

Edellisessä kappaleessa mainittiin, että ohjelmointityöltä voidaan Drupal-pohjaisen verkkosivuston käyttöönotossa välttyä. Esimerkkiprojektissa näin ei kuitenkaan ollut, vaan muutamaan otteeseen jouduttiin järjestelmän rajoituksia kiertämään omalla ohjelmakoodilla. Lisäksi useita osioita jouduttiin räätälöimään Drupalin omilla työkaluilla. Seuraavassa esitellään muutamia räätälöityjä osioita.

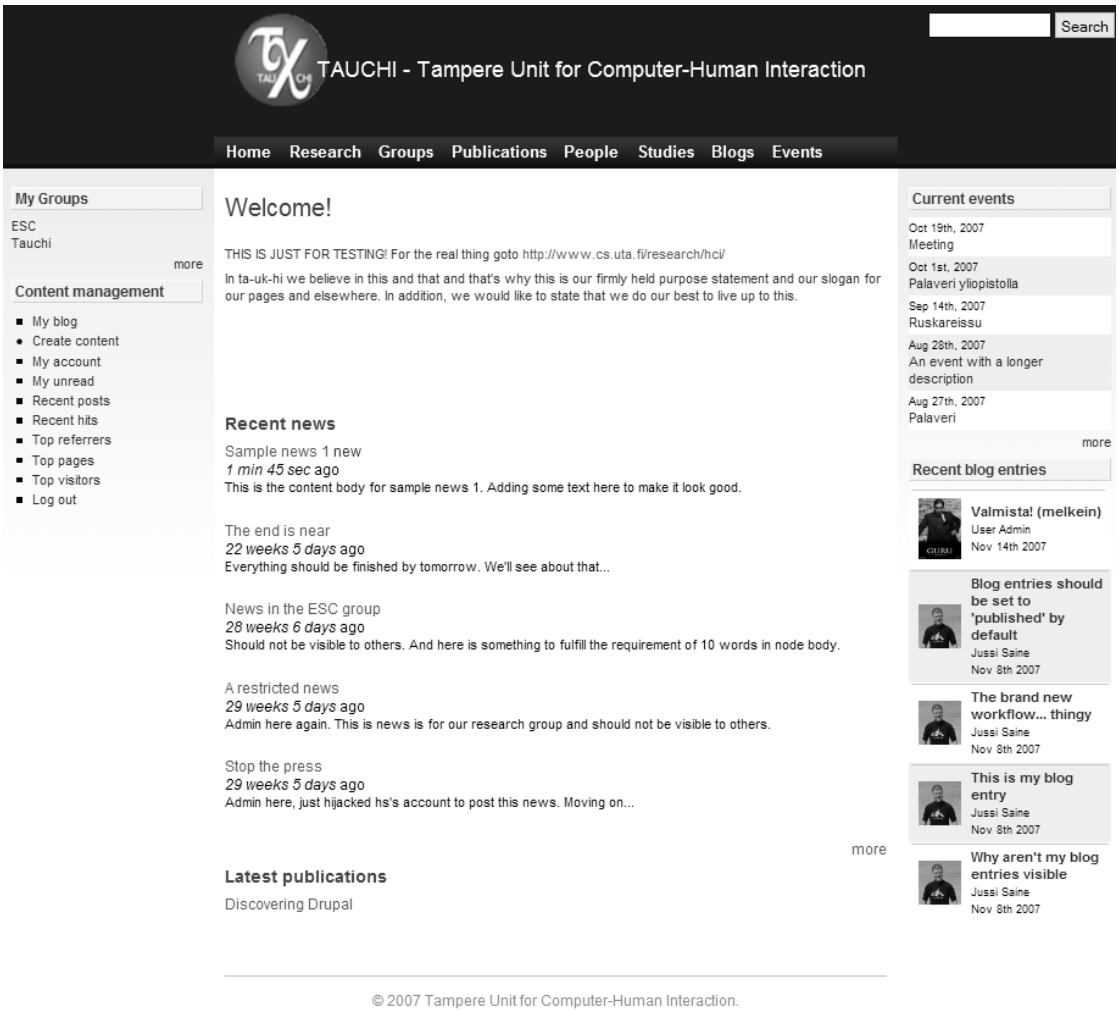
6.3.1. Etusivu

Etusivu on sisältötyyppinä hyvin hankala. Kun linkki jollekin sisältösivulle nostetaan etusivulle, sen suosio kasvaa selvästi verrattuna muihin sivuihin [Krug, 2006, s. 97]. Tällöin tietysti kaikki sisällöntuottajat haluavat linkin juuri omalle sivulleen. Jos etusivu tupataan liian täyteen asiaa, vaarana on, että se muuttuu liian sekavaksi ja informaatioähky karkottaa vierailijat sivustolta. Esimerkkiprojektin tapauksessa etusivulle haluttiin nostaa ajankohtaisia tietoja. Näin vierailija näkisi nopeasti esimerkkejä siitä, mitä sivustolla on tapahtunut edellisen käynnin jälkeen.



Kuva 6. Anonyymin käyttäjän etusivu

Kuvassa 6 näkyy esimerkkiprojektissa toteutetun verkkosivuston etusivu yksinkertaisimmillaan. Yläreunan navigointivalikon ja hakupalkin lisäksi näkyvissä on tervetulotoivotus ja neljä erilaista listaa. Oma lista on niin uutisille, blogimerkinnöille, tapahtumille kuin viimeisimmille julkaisuillekin. Jokainen lista näyttää uusimmat aihepiirinsä sisällöt. Näin vierailija näkee heti, mitä merkittävää sivustolla on viime aikoina tapahtunut. Kaikille sisältötyypeille ei ole omaa listaa etusivulla. Mikäli tällaisia sisältöjä halutaan erityisesti mainostaa, niistä voidaan kirjoittaa uutinen, joka näkyy muiden uutisten joukossa etusivun keskeisimmällä paikalla.

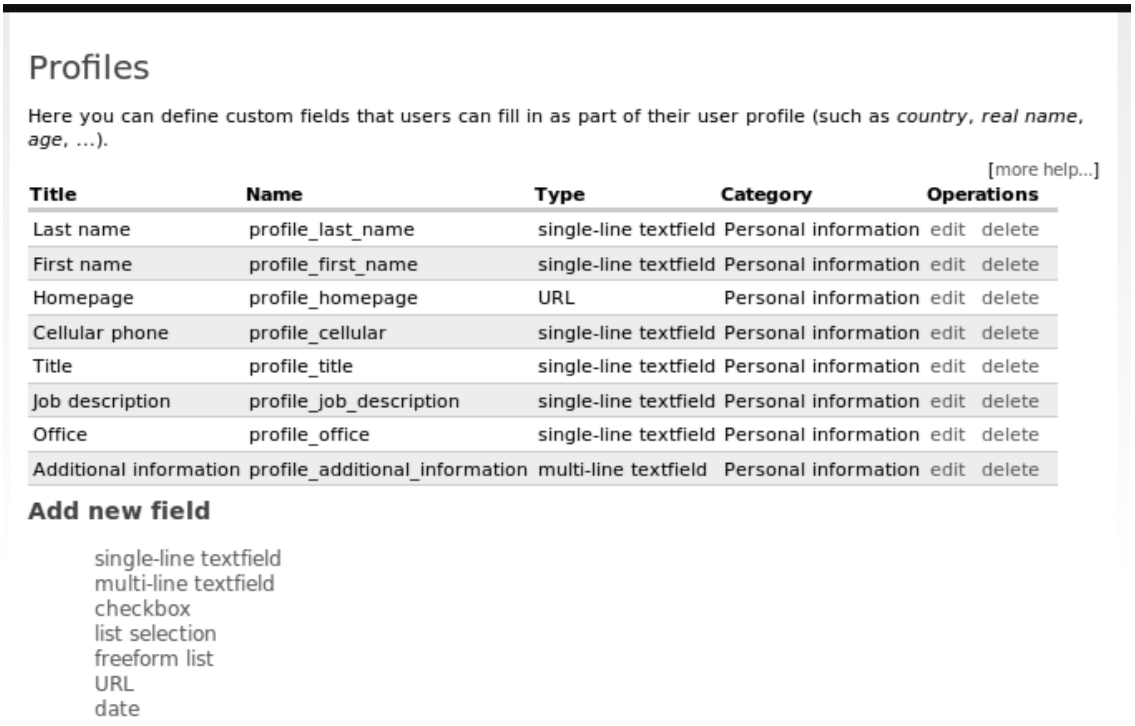


Kuva 7. Kirjautuneen käyttäjän etusivu

Kuvassa 7 on etusivu sellaisena kuin se näkyy kirjautuneelle käyttäjälle. Keskimmäinen ja oikeanpuoleinen palsta pysyvät samoina, mutta lisäksi näytölle ilmestyy kolmas palsta vasempaan reunaan. Siitä löytyvät linkit käyttäjän omien ryhmien etusivuille ja työkalut sisällöntuottamiseen ja käyttäjätilin hallintaan. Lisäksi saadaan tarkempia tietoja viimeaikaisista tapahtumista, esimerkiksi viimeisimmät foorumiviestit ja viime käynnin jälkeen muuttuneet sisällöt.

6.3.2. Henkilötiedot

Normaalisti Drupalissa jokaisesta käyttäjästä tallennetaan käyttäjätunnuksen, salasanan ja käyttöoikeuksien lisäksi vain sähköpostiosoite ja kuva. Lisämoduuleilla käyttäjille on mahdollista luoda yksityiskohtaisempi profiili ja sinne halutut kentät. Näitä voidaan vielä lajitella eri kategorioihin.



Profiles

Here you can define custom fields that users can fill in as part of their user profile (such as *country*, *real name*, *age*, ...).

[more help...]

Title	Name	Type	Category	Operations
Last name	profile_last_name	single-line textfield	Personal information	edit delete
First name	profile_first_name	single-line textfield	Personal information	edit delete
Homepage	profile_homepage	URL	Personal information	edit delete
Cellular phone	profile_cellular	single-line textfield	Personal information	edit delete
Title	profile_title	single-line textfield	Personal information	edit delete
Job description	profile_job_description	single-line textfield	Personal information	edit delete
Office	profile_office	single-line textfield	Personal information	edit delete
Additional information	profile_additional_information	multi-line textfield	Personal information	edit delete

Add new field

single-line textfield
multi-line textfield
checkbox
list selection
freeform list
URL
date

Kuva 8. Käyttäjäprofiilin kenttien muokkausnäkymä.

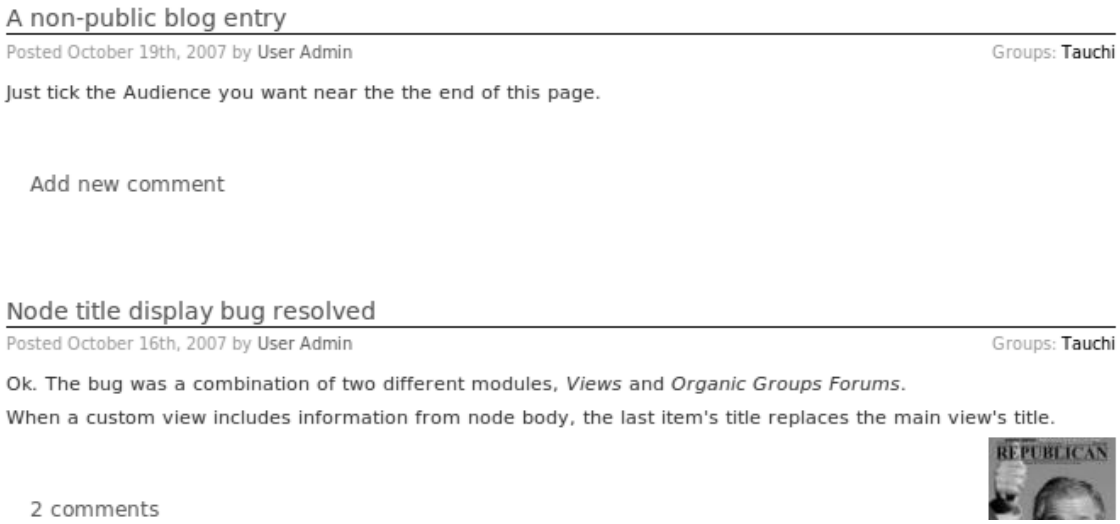
Kuvassa 8 näkyy joitakin luotuja kenttiä ja listauksen alla mahdolliset uudet kenttätyypit. Näihin kenttiin on mahdollista viitata ympäri sivustoa. Näin esimerkiksi blogimerkinnän tekijän nimen yhteyteen voidaan laittaa vaikkapa sähköpostiosoite tai virkanimike.

Jotta järjestelmään syötetyistä henkilötiedoista olisi mitään hyötyä, ne täytyy saada jotenkin näkyville. Tätä varten vakiomuotoista henkilötietosivua muokattiin siten, että sivulla näkyy nimen lisäksi valokuva, yhteystiedot ja linkki julkaisuihin joissa kyseinen henkilö on ollut osallisena.

Drupal näyttää oletuksena esimerkiksi blogimerkinnän kohdalla merkinnän tekijän käyttäjätunnuksen. Blogimerkinnän näyttävän sivupohjan lähdekoodia muokattiin siten, että käyttäjätunnus piilotetaan ja sen sijaan näytetään yhdistelmä ”etunimi sukunimi”. Tämä on paitsi käyttäjäystävällistä, myös tietoturvan kannalta parempaa, kun kirjautumiseen käytettäviä käyttäjätunnuksia ei ole nähtävillä.

6.3.3. Blogit

Blogien ominaisuuksien muokkauksessa ei puututtu varsinaisiin blogimerkintöihin. Sen sijaan luotiin listasivu, jossa merkinnöistä näkyy oleelliset tiedot:



Kuva 9. Muokattu blogimerkintöjen listanäkymä.

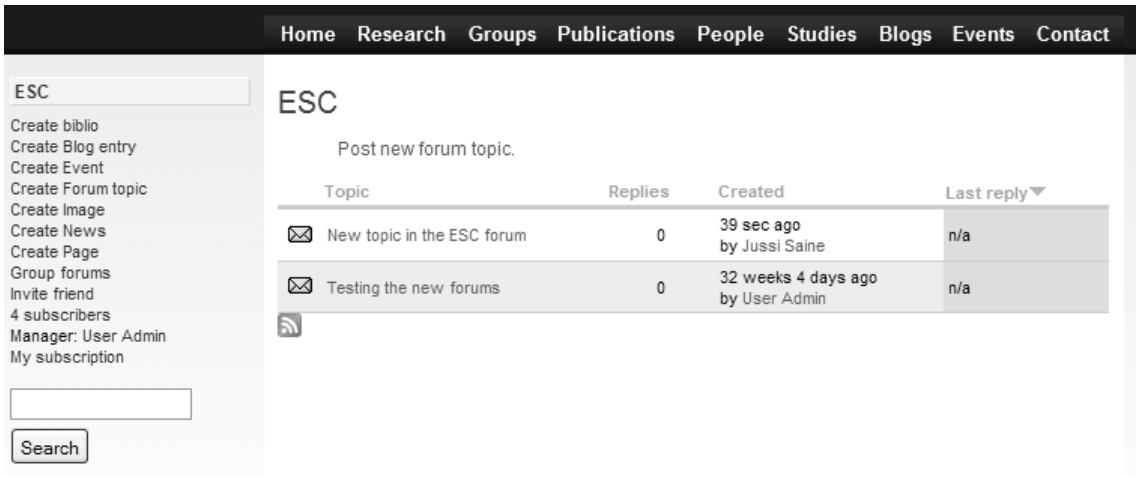
Kuvassa 9 näkyy ote sivusta, jossa listataan tietyn käyttäjän kirjoittamat blogimerkinnät. Muutoksia Drupalin oletusnäkymään on useita:

- käyttäjätunnuksen sijasta näkyy käyttäjän nimi (”User Admin”),
- oikeassa reunassa näkyy merkinnän näkyvyys. Tässä tapauksessa merkinnän näkevät kaikki ”Tauchi”-nimiseen ryhmään kuuluvat käyttäjät ja
- tekstistä näytetään vähän alkua pelkän otsikon sijasta.

Erityisen tärkeäksi koettiin se, että solmun yhteydestä käy aina ilmi, mille käyttäjäryhmälle se näkyy. Sisällön näkyvyys on tärkeä käyttäjäkokemukseen vaikuttava tekijä, mutta myös tietoturvatekijä. Kun lukija näkee heti sivuston kohderyhmän, mahdolliset väärät näkyvyysasetukset havaitaan nopeasti.

6.3.4. Ryhmäsivujen toteutus

Ryhmät toteutettiin Drupalin Organic Groups- ja Organic Groups Forums -lisämoduulien avulla. Kun käyttäjä avaa ryhmän etusivun, näytölle avautuu listaus uusimmista sellaisista sisällöistä, joissa kyseinen ryhmä on määritelty vastaanottajaksi.



Kuva 10. Ryhmän toimintovalikko ja keskustelufoorumi

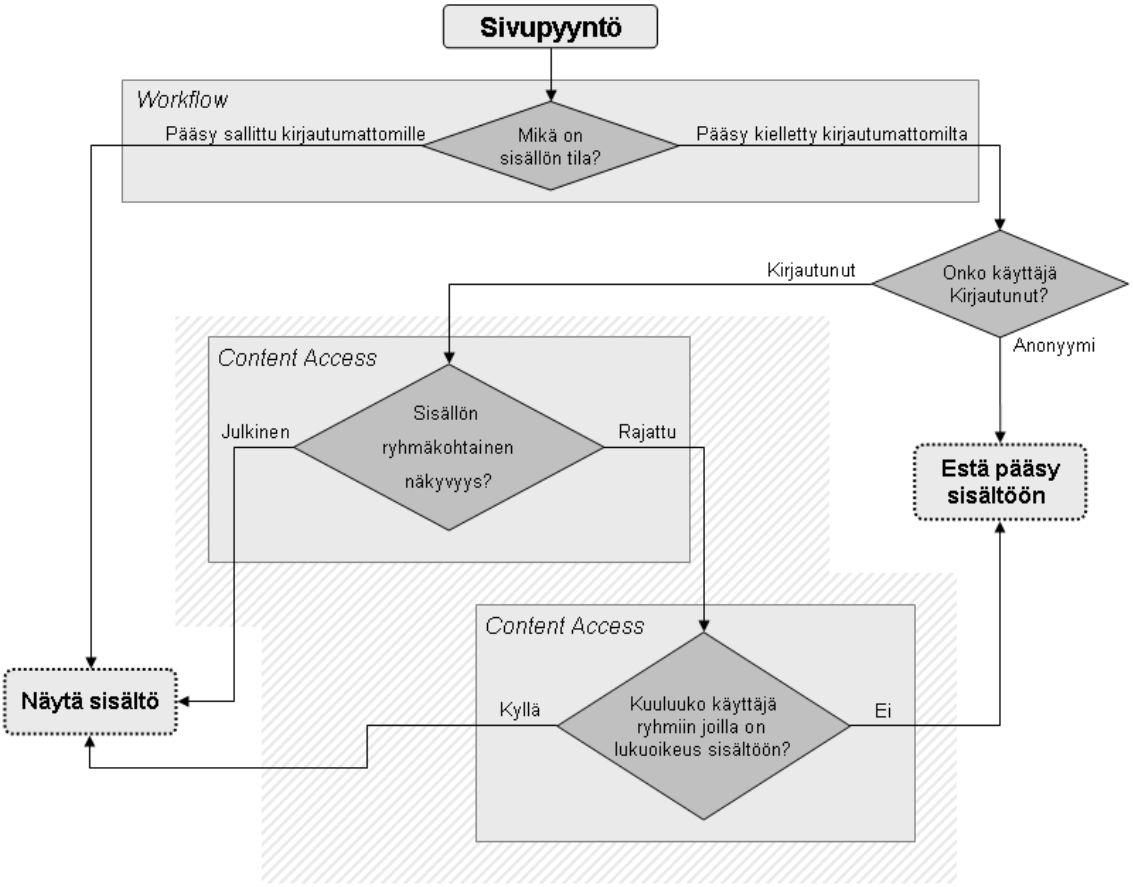
Kuvan 10 vasemmassa reunassa näkyy valikko niistä toiminnoista, joita ryhmän sisällä voi tehdä. Kun sisältöä luodaan sen kautta, näkyvyys määräytyy automaattisesti oikealle ryhmälle. Kuvassa nähdään myös ryhmän keskustelufoorumin etusivu.

6.3.5. Uutiset ja tapahtumat

Uutisia varten luotiin uusi sisältötyyppi. Tapahtumille saatiin valmis sisältötyyppi asentamalla Event-moduuli. Näiden ero on siinä, että tapahtumalla voi olla alku- ja loppuaika, kun taas uutinen ei sellaisia tarvitse. Uutiset voidaan järjestää aikajärjestykseen sen mukaan, milloin niitä on viimeksi muokattu. Tapahtumien ja uutisten erottamista pohdittiin muutamaan otteeseen. Lopulta ne siis päädyttiin luomaan erillisinä sisältötyyppeinä. Vaikka yksikin sisältötyyppi olisi voinut riittää, tämän ratkaisun avulla kummastakin voidaan luoda erilliset listat. Tarvittaessa niistä voidaan luoda yhdistetty lista, jonka luominen on helpompaa kuin erityyppisten sisältöjen erottaminen yhdestä sisältötyypistä.

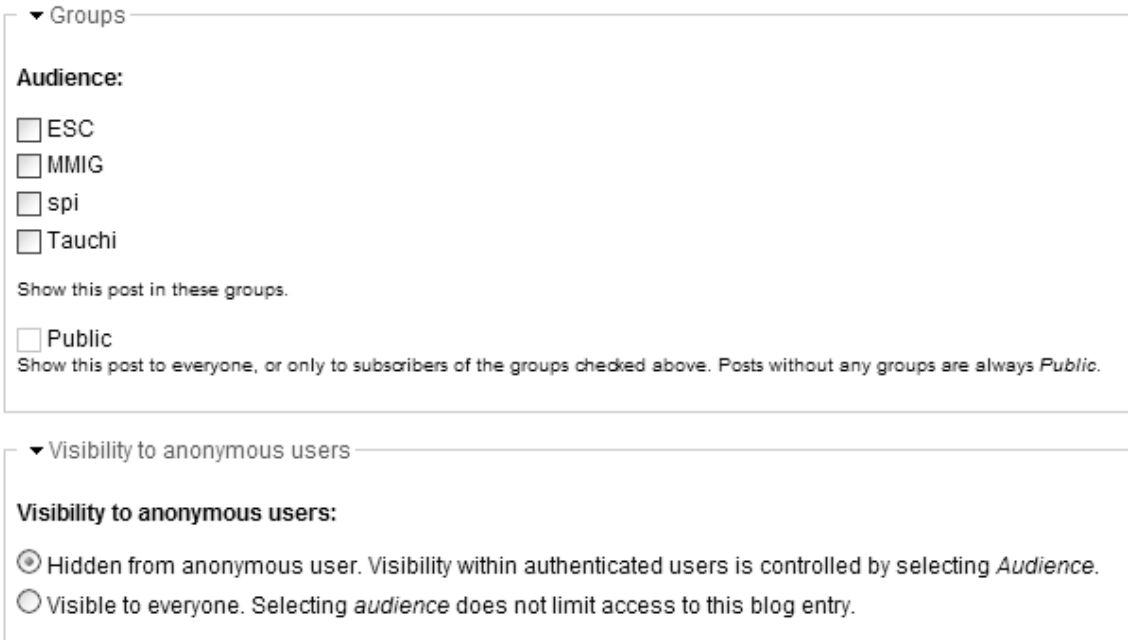
6.3.6. Lukuoikeuksien määrittely

Sisällön näkyvyyttä ei haluttu kontrolloida vain sisältötyypin mukaan, koska se olisi monessa tilanteessa ollut liian karkea jaottelu. Sen sijaan päädyttiin toteuttamaan oikeudet yksittäisten solmujen tasolla kuitenkin siten, että oletusarvo näkyvyydelle olisi määriteltävissä sisältötyypeittäin. Ratkaisu toteutettiin yhdistelemällä Workflow- ja Content Access -moduulien toiminnallisuuksia. Workflow-moduulissa sisällöntuotannolle pystytään määrittämään eri tiloja, normaalisti nämä olisivat esimerkiksi *luonnos*, *tarkistettava* ja *julkaistu*. Tässä tapauksessa tiloja on kaksi, *Hidden from anonymous users* (piilotettu anonyymeiltä käyttäjiltä) ja *Visible to everyone* (näkyvissä kaikille).



Kuva 11. Sisällön lukuoikeuksien tarkistuskaavio.

Toteutuksessa on hieman päällekkäistä logiikkaa, kuten kuvasta 11 käy ilmi. Osa tästä päällekkäisyydestä näkyy valitettavasti loppukäyttäjän käyttöliittymässä asti. Tämä nähdään kuvasta 12:



Kuva 12. Sisällön näkyvyyden määrittely

Kuvasta 12 nähdään, että sisällön näkyvyys määritellään kahdessa eri kohdassa. Periaate on, että otsikon ”Visibility to anonymous users” alla oleva asetus on voimakkaampi kuin kohdan ”Audience” asetukset. Jos alemmassa kohdassa on määritetty sisältö kaikille näkyväksi, sitä ei enää pysty rajaamaan tietyille ryhmille. Tämä toteutus tuntuu aluksi oudolta, mutta sillä saavutetaan yksi tärkeä etu jota ei muuten ole mahdollista toteuttaa: uutta sisältöä luodessa se on oletusarvoisesti piilotettu kirjautumattomilta käyttäjiltä. Sisällön näkyminen liian laajalle yleisölle ei siis onnistu vahingossa. Ilman tätä toteutusta ei uuden sisällön näkyvyyttä rajata mitenkään. Kuvan 10 valinnoilla sisällön näkyvyys voidaan siis määritellä täysin vapaasti turvallisin oletusarvoin.

7. Toteutuksen ongelmia ja ratkaisuehdotuksia

Esimerkkiprojektissa esille tulleet Drupaliin liittyvät ongelmat voidaan jakaa muutamaan kategoriaan. Näistä ensimmäinen ja tärkein on järjestelmän modulaarisesta luonteesta johtuva hajanaisuus ja moduulien laadulliset ongelmat. Monesti moduuleihin on voinut eksyä suoranaisia ohjelmointivirheitä, joiden kiertäminen vaatii selvitystyötä.

Toinen ongelmakategoria liittyy käyttöoikeuksiin ja sisällön näkyvyyteen. Drupal suhtautuu avoimen lähdekoodin periaatteiden mukaisesti myös sivuston sisältöön. Oletuksena kaikki sisältö on kaikille avointa. Lisämoduulien käyttöön on turvauduttava heti, jos on tarvetta yksityiskohtaisemmille näkyvyysmäärittelyille.

Kolmas ongelma on hieman yllättäen kehittäjien kova into hyödyntää uusinta tekniikkaa. Tästä johtuen vanhoihin versioihin perustuvien sivustojen päivittäminen uuden version kanssa toimivaksi on työläs urakka ja vaatii aina runsaasti käsityötä. Muita ongelmia ovat mm. dokumentaation pinnallisuus ja ainakin vielä versiota 5 vaivaava raskaus ja hitaus. Seuraavassa käydään yksityiskohtaisemmin läpi edellä mainittuja ongelmia.

7.1. Moduulit

Modulaarisuus on Drupalin suurin vahvuus ja tietyssä mielessä myös suurin heikkous. On toki käytännöllistä, että järjestelmä on helposti laajennettavissa, ja että laajennuksia löytyy lähes joka tarpeeseen. Huono puoli tässä on se, että myös kehittäjiä on tuhansia ja kaikilla on eri motivaatiot moduulien kehitykseen. Useimmissa tapauksissa moduuli on saanut alkunsa, kun järjestelmään on haluttu uusi ominaisuus. Kun tämä toiminnallisuus on saatu toimimaan kehittäjän omassa järjestelmässä, motivaatio moduulin jatkokehitykseen voi loppua. Tällöin moduulin toiminta muissa ympäristöissä voi olla vajaa tai jopa täysin virheellinen.

Moduulien kehittäjät voivat perustaa oman tukisivuston Drupalin kotisivujen alle. Sieltä saattaa löytyä jonkun muun käyttäjän keksimä keino moduulin ohjelmointivirheiden kiertämiseen. Tällaisten korjausten käyttämisestä seuraa kuitenkin ylläpidollisia ongelmia. Kun moduulien ohjelmakoodia ryhdytään korjaamaan itse, miten pystytään varmasti huolehtimaan korjausten toimivuudesta tulevaisuudessa? Kun ohjelmointivirhe ilmoitetaan korjatuksi, täytyy korjauksen toteutus testata. Jos se on tehty eri tavalla kuin oma korjaus, on tarkistettava, ettei siitä aiheudu ongelmia.

Esimerkkisivustoa tehtäessä muutamat moduulit olivat keskeneräisiä jopa siinä määrin, että osasta toiminnallisuutta täytyi luopua virhetilanteiden välttämiseksi. Esimerkiksi LDAP-moduulin tarkoitus on päivittää käyttäjän tiedot hakemistopalvelusta Drupaliin, mutta ohjelmointivirheen vuoksi kaikki Drupalin käyttäjätietokentät tyh-

jenevät mikäli hakemistopalvelun käyttäjätietoihin ei ole tallennettu tämän sähköpostiosoitetta.

Eräs virhe koski Views- ja Organic Groups -moduulien yhteistyötä. Kun Views-moduulilla tehtiin listauksia joihin haettiin solmun pääsisältö, kaikki otsikot menivät sekaisin sivuilla, joilla tällainen listaus oli näkyvissä. Ongelma on tyyppiesimerkki siitä, mitä tapahtuu kun moduuleja kirjoitetaan joko vanhan ohjelmointirajapinnan ohjeiden mukaan tai vaihtoehtoisesti ohjeita hieman soveltaen. Riittää kun jokin muuttuja nimetään väärällä tavalla, jolloin käsissä on hankalasti ratkottava ongelma, jonka selvittämiseen voi kulua runsaasti aikaa. Ratkaisuna tähän kyseiseen ongelmaan oli tehdä Organic Groups -moduulin lähdekoodiin pieni muutos. Korjauksen toimivuus on muistettava tarkistaa aina ongelmaan osallisia moduuleita päivitettäessä.

7.2. Oikeuksien toteutus

Drupalin idea sisällön näkyvyyden suhteen ainakin versioon 5 asti on se, että käyttäjäoikeuksia määritellään sisältötyypin perusteella, ei niinkään alueittain tai yksittäisen sisällön tasolla. Käyttäjä voi esimerkiksi nähdä kaikki tai ei yhtään kalenterimerkintää. Vastaava karkea jaottelu toistuu kaikilla sisältötyypeillä. Ongelman kiertämiseksi lähdettiin kehittämään toteutusta, jolla oikeuksia pystyttäisiin hienosäätämään tarkemmin mutta silti riittävän helposti.

Käyttäjäoikeuksia hallinnoivia moduuleita löytyy monia, mutta mikään niistä ei yksinään riittänyt halutun lopputuloksen saavuttamiseen. Lopulta päädyttiin käyttämään yhteensä kuutta eri moduulia lisäosineen: Content access, Organic groups (OG), OG roles, Menu per role, Workflow ja Workflow access. Lopputulos ei ole kovin elegantti, mutta toimii luotettavasti ja käyttäjän näkökulmasta melko loogisesti. Jokaisesta moduulista käytettiin niitä ominaisuuksia, joista oli hyötyä kokonaisuuden suhteen. Tästä aiheutui tiettyä päällekkäisyyttä teknisessä mielessä, joten esimerkiksi anonyymien käyttäjien lukuoikeus sisältöön on mahdollista määritellä kahdella tavalla. Käytännössä ratkaisusta tuli melko siisti ja näkyvyys määritetään kaksiportaisesti: ensin määritellään pääsy anonyymeille ja kirjautuneille käyttäjille, tämän jälkeen vielä haluttaessa kirjautuneille käyttäjille käyttäjäryhmän mukaan.

Yllä kuvattu ratkaisu kuvaa hyvin Drupalin joustavuutta. Mikäli jotain toiminnallisuutta ei löydy, se on lähes aina toteutettavissa tavalla tai toisella. Joustavuuden kääntöpuoli on liika yksilöllisyys. Tietty toiminnallisuus voi eri Drupal-sivustoissa olla toteutettu hyvin erinäköisin ratkaisuin. Tällöin myös ongelmat ovat erilaisia ja avun saanti vaikeampaa yhtenäisten toteutustapojen puuttuessa.

7.3. Suunnittelufilosofia

Drupalin kehitystä ohjaa into parantaa ja kehittää jatkuvasti uusia ominaisuuksia ja ottaa uusimmat tekniikat käyttöön mahdollisimman pian. Ohjelmisto voitti "2007 Overall Open Source Content Management System Award" -palkinnon [Packt, 2007] ja sitä on kehitetty jo vuodesta 2000 lähtien. Drupal ei siis ole mikään hetkessä ohimenevä ilmiö. Siksi voisikin kuvitella, että järjestelmän rajapinnat olisivat hioutuneet riittävän pitkälle, jotta niitä ei tarvitsisi muuttaa joka versioon. Näin ei kuitenkaan ole.

Tässä toimintatavassa on hyvätkin puolensa, mikäli halutaan kulkea kehityksen kärjessä. Luotettavasti toimiva päivitystoiminto voisi olla toimiva ratkaisu, jos kyseessä olisi tavanomainen yhden toimittajan tekemä järjestelmä. Koska Drupal on rakenteeltaan pirstaleinen, kaikki yhdistelmät huomioonottavan päivitystoiminnon tekeminen on kuitenkin mahdotonta. Samasta syystä myös varmuuskopiointitoiminnon toteuttaminen on hankalaa. Drupal-sivustojen varmuuskopiointiin suositellaankin tapaa, jossa sisällöstä otetaan tietokantavedos ja sivuston ohjelmatiedostot kopioidaan talteen [Drupal 5 getting started, s.10].

7.4. Ongelmanratkontaa ja vaihtoehtoisia toteutustapoja

Tässä luvussa käsitellään muutamia vaihtoehtoisia järjestelmiä, joilla esimerkkiprojektia olisi voitu lähteä toteuttamaan. Tarkoituksena ei ole tarjota kattavaa vertailua eri vaihtoehdoista. Ennemminkin pyritään arvioimaan, miten muissa järjestelmissä oltaisiin pystytty välttämään Drupal-toteutuksessa kohdatut ongelmat. Samalla luodaan pikainen katsaus Drupal 6:n uudistuksiin.

7.4.1. Django

Django [2008] on Python-pohjainen Web-sovelluskehys, joka soveltuu kaikentyyppisten WWW-sivustojen alustaksi. Sen vahvuuksia ovat Pythonin selkeys ja oliopohjaisuus. Django-sovellukset jakaantuvat rakenteellisesti malleihin, näkymiin ja sivupohjiin. Lisäksi siinä on kehittynyt URL-arkkitehtuuri ja automaattinen Admin-käyttöliittymä sivuston ylläpitoon. Django-sovellus on mahdollista jopa kirjoittaa siten, että sisällön (esim. WWW-sivu) muokkaaminen tapahtuu Adminin tietokantanäkymässä. Tällöin ohjelmoijan tarvitsee vain kirjoittaa URL-skeema ja sivupohja jolla sisältö näytetään käyttäjille.

Tietokantojen luominen tapahtuu suoraan niiden määrittelytiedostojen perusteella. Esimerkiksi verkkolehden artikkelille voidaan kirjoittaa seuraava malli josta Django rakentaa tietokantaan oikeat taulut:

```
class Artikkeli(models.Model):
    julkaisupvm = models.DateTimeField()
    otsikko = models.CharField(max_length=200)
    teksti = models.TextField()
    toimittaja = models.ForeignKey(Toimittaja)
    class Admin: pass
```

Kun Django-sovellus vastaanottaa sivupyynnön, se tarkistaa ensin URL-skeemasta, mitä pyynnölle pitää tehdä. Se voi esimerkiksi ohjata pyynnön oikealle näkymälle, joka tekee pyydetyt asiat, esimerkiksi hakee tietokannasta viimeisimmät 5 uutista. Tämän jälkeen se lähettää uutiset sivupohjalle joka koostaa sivun valmiiksi.

Django-sivuston eri vaiheita voidaan kehittää samanaikaisesti. Ensin määritellään mitä tietoja sivu tietokannasta haluaa ja määritellään muuttujien nimet. Tämän jälkeen ohjelmoijan tehtävä on toteuttaa näkymä (*view*) jolla muuttujat voidaan tarjota sivulle ja graafikko voi suunnitella tiedot vastaanottavan HTML-sivupohjan. Koska Djangon template-kieli on erittäin selkeää, HTML-sivun toteutus onnistuu myös ohjelmointia osaamattomalta graafikolta kunhan vain yksinkertaiset if-lauseet ja muuttujat ovat tuttuja käsitteitä. Esimerkiksi seuraava koodinpätkä tulostaa artikkelien lyhennelmät sivulle:

```
<h1>Viimeisimmät uutiset</h1>
<ul>
  {% for artikkeli in artikkelit %}
  <li>
    <h2>{{ artikkeli.otsikko }}</h2>
    <h5>{{ artikkeli.julkaisupvm }}</h5>
    <p>{{ artikkeli.teksti|truncatewords:30 }}</p>
  </li>
  {% endfor %}
</ul>
```

7.4.2. Soveltuvuus esimerkkiprojektiin

Djangon lähtökohdat esimerkkiprojektin toteuttamiseen ovat hyvin erilaiset Drupaliin verrattuna. Drupal-toteutuksessa suurin osa työstä menee moduulien valintaan ja niiden yhteensovittamiseen, ohjelmakoodia ei ole välttämättä pakko kirjoittaa lainkaan. Django taas tarjoaa työkalut joilla sovellusten kirjoittaminen on nopeaa, mutta ne täytyy joka tapauksessa ohjelmoida itse. Djangon vahvuus on kuitenkin siinä, että sillä kirjoitettujen sovellusten koodi on helposti luettavaa ja vikatilanteet pystytään eristämään tarkasti pieneen ohjelman osaan.

Drupal-toteutuksessa aiheutui jonkin verran ongelmia moduulien välisen yhteistyön toimimattomuudesta. Yksi rivi eräässä moduulissa aiheutti sen, että oikeassa tilanteessa kaikkien sivun otsikoiden teksti korvautuikin tietokannasta sieltä täältä haetuilla tiedoilla. Koska asennettuja moduuleita oli tässä vaiheessa jo toistakymmentä, syyllisen etsin-

tä osoittautui hyvin työlääksi prosessiksi. Käytetty ratkaisu osoittautui hyödylliseksi myös muiden Drupalin ongelmien ratkomisessa.

Koska Drupalissa moduulien yhteensopivuus on monesti syynä ongelmiin, kannattaa niitä ratkoa testiympäristössä asentamalla moduuleita kunnes virhetilanne saadaan toistettua. Pelkkä moduulien poistaminen ja lisääminen yksitellen ei toimi, koska ne ovat jo tehneet vian aiheuttavia muutoksia tietokantaan. Lopulta otsikkotekstejä sotkevan ongelman aiheuttaja pystyttiin rajaamaan muutaman moduulin yhdistelmään. Näitä tietoja yhdistelemällä erään moduulin tukisivuilta löytyi vikaraportti tilanteesta ja onneksi myös tilapäiskorjaus ongelman korjaamiseksi.

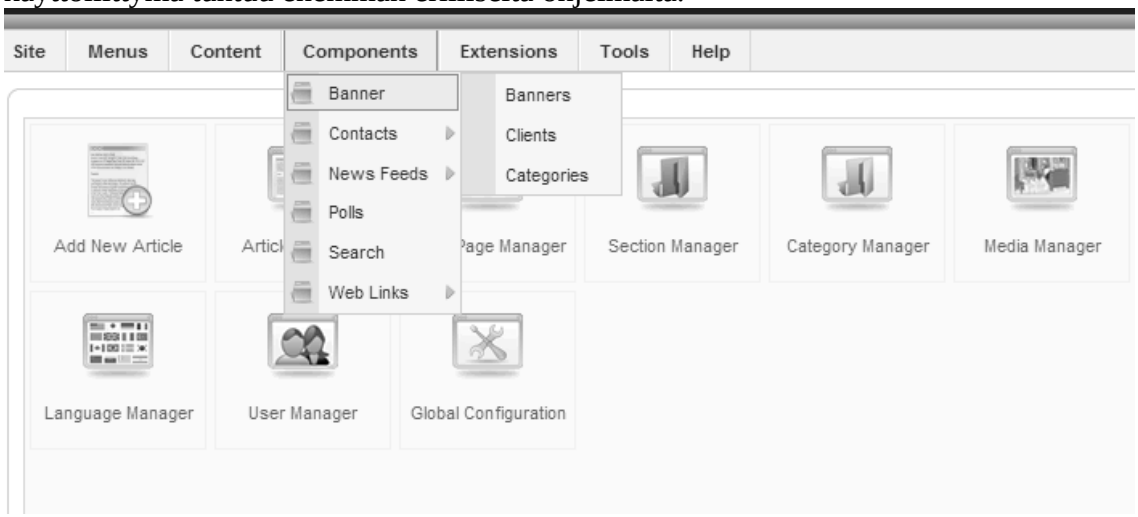
Djangolla ongelmaa ratkottaessa se pystytään heti rajaamaan juuri siihen funktioon (ohjelman osa, joka vastaanottaa tietoa ja palauttaa sen käsiteltynä), jota haettu URL-osoite kutsuu. Tämän jälkeen vian aiheuttaja paljastuu nopeasti. Näin esimerkiksi Drupal-toteutuksessa havaittu LDAP-integrointiongelma olisi ollut korjattavissa hetkessä. On tietysti huomioitava, että itsekirjoitetun ohjelmakoodin virheiden selvittäminen on aina helpompaa kuin toisen ohjelmoijan ajatusten selvittäminen. Drupalin tapauksessa tämä on vielä hieman hankalampaa, koska joka moduulilla on eri ohjelmoijat jotka toimivat toisistaan riippumatta. Sen ohjelmakoodin toiminnan selvittämistä haittaa myös se, että PHP-koodia kirjoittamisessa on monenlaisia käytäntöjä ja ohjelmakoodi on näin tekijänsä näköistä [Phpro, 2007]. Djangon käyttämä Python taas edellyttää esimerkiksi sisennysten käyttöä, jolloin päästään eroon monista häiriötekijöistä kuten silmukoiden suluttamisesta [Lutz and Ascher, 2003, s.149-151] .

Yhteenvetona voidaan todeta, että Django voisi soveltua hyvin myös sisällönhallintajärjestelmän toteuttamiseen. Se ei kuitenkaan esimerkkiprojektissa olisi ollut varteenotettava vaihtoehto toteutuksen välineeksi. Samankaltaiseen tulokseen päädyttiin myös IBM:n Internet technology teamin tutkimuksessa, jossa käsiteltiin niinikään sisällöntuotantojärjestelmän valintaprosessia [Weitzman, 2006]. Tuolloin yhtenä toteutusvaihtoehtona olisi ollut Ruby on Rails, joka on pitkälti Djangon tyyppinen Web-sovelluskehys. Vaikka valinta olisi ollut mielenkiintoinen ja olisi tarjonnut vapaat kädet kaikenlaisiin toteutusvaihtoehtoihin, se ei aikarajan vuoksi ollut todellinen vaihtoehto. Esimerkkiprojektin tapauksessa uusi järjestelmä olisi pohjimmiltaan ollut juuri sitä mitä vanhakin: itse toteutettu ylläpitojärjestelmä, jossa jokainen uusi ominaisuus tai muutos olisi vaatinut ohjelmointia. Toki järjestelmästä saataisiin helposti yhtä modulaarinen kuin Drupalista, jolloin uusien ominaisuuksien lisääminen olisi nopeaa. Django ei ole ainakaan nopein väline tilanteessa, jossa valmiitakin ratkaisuja on olemassa.

7.4.3. Joomla

Joomla on yksi suosituimmista ilmaisista sisällönhallintajärjestelmistä. Sen ylläpitokäyttöliittymä poikkeaa suuresti esimerkiksi Drupalin vastaavasta. Siinä missä Drupalin

ylläpito hoidetaan jopa vanhanaikaisen näköisten WWW-lomakkeiden avulla, Joomlan käyttöliittymä tuntuu enemmän erilliseltä ohjelmalta.



Kuva 13. Joomlan hallintapaneeli

Kuvasta 13 nähdään, että Joomlan ylläpitokäyttöliittymän etusivu näyttää hieman Windows-ohjauspaneelilta ja näytön yläreunassa on valikot eri ylläpitotoiminnoille. Peruskäyttäjälle näkymä on siis tutumpi kuin Drupalissa tai monessa muussa sisällönhallintajärjestelmässä.

Joomla tarjoaa oletusasennuksessa hyvän tekstieditorin. Tekstin joukkoon on erittäin helppoa liittää kuvia. Kuvien lisääminen on huomattavan helppoa verrattuna esimerkiksi Drupalin lisävarusteina saataviin moduuleihin, joista yksikään ei tunnu toimivan yhtä loogisesti.

Käyttäjäoikeuksien hallinta on melko rajoittunutta. Karkeasti ottaen käyttäjät jaetaan järjestelmän ylläpitäjiin ja sisällön ylläpitäjiin. Molemmissa ylläpitäjäryhmissä on lisäksi muutamia eritasoisia käyttäjäryhmiä. Mikäli oikeuksia halutaan määritellä tarkemmin, joudutaan turvautumaan lisäosiin. Sopivien moduulien valintaa vaikeuttaa se, että osa Joomlan lisäosista on maksullisia.

Tavoitettavuus voi muodostua ongelmaksi, koska Joomla käyttää taulukoita sisällön asetteluun. On tietysti mahdollista luoda sellainen sivupohja, jossa taulukoita ei käytetä, mutta vakioasetuksin tämä ei siis ole mahdollista. Kokonaisuutena Joomla on nopea ottaa käyttöön, mutta erityisesti käyttäjäoikeuksien toteutus on jopa vielä hankalampaa kuin Drupalissa.

7.4.4. Drupal 6

Drupal 6 julkistettiin helmikuussa 2008. Versioon 5 verrattuna se lupaa runsaasti uudistuksia mm. käytettävyyteen, tietoturvaan ja ylläpitoon. Julkaisu-uutisen jatkokeskustelussa osa käyttäjäkunnasta ilmoitti tyytymättömyytensä Drupalin liian nopeaan kehitystahtiin. Lisämoduulien kehittäjät eivät pysy vauhdissa mukana, mikäli uusi ja radikaa-

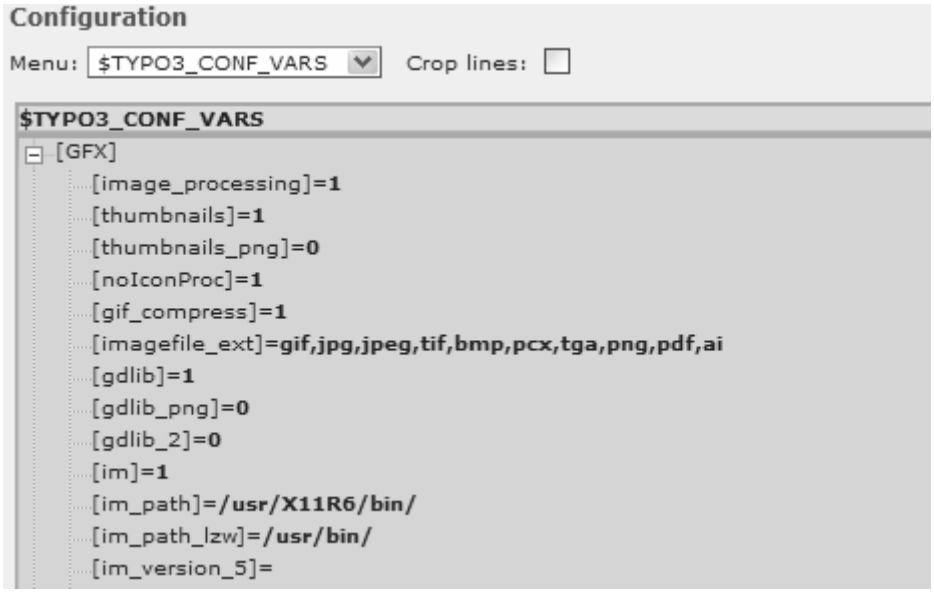
listi erilainen versio pääjärjestelmästä julkaistaan aina kerran vuodessa [Drupal 6 rel. notes, 2008]. Tämä näkyy myös käytännössä, sillä vielä muutama kuukausi Drupal 6:n julkistamisen jälkeen sille ei ole julkaistu edes tärkeimpiä esimerkkiprojektissa käytettyjä moduuleita.

Yksi luvatuista uudistuksista on parantunut kielituki. Käytännössä tämä tarkoittaa lähinnä sitä, että solmun kieliasetus on entistä helpompaa valita. Mukana on toki muitakin uudistuksia, kuten esimerkiksi hyvin toimiva tuki oikealta vasemmalle kirjoitettaville kielille. Järjestelmä ei edelleenkään itse hoida kieliasetusta esimerkiksi siten, että yhdelle sisällölle voisi kirjoittaa erikieliset versiot ja Drupal ymmärtäisi ne yhtenä kokonaisuutena tarjoten aina käyttäjän asetusten perusteella sopivinta kieliversiota [Knaddison, 2008]. Käyttäjän huoleksi jää määritellä esimerkiksi yhteystietosivun erikieliset versiot siten, että Drupal osaa valita oikean kieliversion.

Käyttöikeuksien toteutus on edelleen idealtaan sama kuin Drupalin aiemmissa versioissa. Käyttäjille ja käyttäjäryhmille annetaan oikeuksia sisältötyypeittäin. Yksittäisen sisällön näkyvyyttä ei edelleenkään voi määritellä ilman lisämoduulien asentamista. Muut tietoturvan uudistukset ovat lähinnä kosmeettisia, esimerkiksi liian heikosta salasanasta varoitetaan nyt jo salasanaa vaihdettaessa.

7.4.5. Muita järjestelmiä

Lopuksi luodaan vielä silmäys muutamaan laajuudeltaan eritasoisiin sisällönhallintajärjestelmiin. Typo3 [2008] on hyvin monipuolinen ja samalla myös monimutkainen järjestelmä. Osittain tästäkin syystä sen kehitystahti ei ole yhtä nopea kuin muilla järjestelmillä. Typo3:n käytön aloittaminen voi vaatia jopa viikkokausien perehtymisen [Weitzman et al., 2006]. Monimutkaisuudesta johtuen myös sen toiminnallisuus poikkeaa nykystandardeista.



Kuva 14. Typo3:n asetusvalikko

Kuvassa 14 näkyy ote Typo3:n asetusvalikosta. Siinä missä useimmat sisällönhallintajärjestelmät tarjoavat graafiset työkalut ja lomakkeet asetusten muokkaamiseen, Typo3 avaa käyttäjälle asetustiedoston sisällön. Osaavalle käyttäjälle asetusten mukauttaminen on nopeaa, mutta muiden täytyy arvuutella mitä lukuisat, jopa kryptiset lyhenteet tarkoittavat. Käyttäjäkommenttien perusteella Typo3:n oppimiskäyrä on poikkeuksellisen jyrkkä, mutta kärsivällisyys palkitaan järeällä, vakaalla ja monipuolisella järjestelmällä [Open source CMS - Typo3, 2008].

Wordpress [2008] on blogisivustojen ylläpitoon tarkoitettu järjestelmä. Se on myös hyvä vaihtoehto, kun tavoitteena on rakentaa suhteellisen yksinkertainen verkkosivusto pienellä vaivalla. Toteutus onnistuu myös ilman laajennuksia, mutta lisämoduuleitakin on saatavilla. Esimerkiksi etusivun hallintaan, sivuhierarkian luomiseen ja käyttäjäroolien määrittelyyn on olemassa lisäosia, jotka yhdistämällä Wordpressistä saadaan tehtyä ”oikea” sisällönhallintajärjestelmä [Peatling, 2007]. Tällöin järjestelmä sisältää mahdollisuuden blogien ylläpitoon ja sisältösivujen kirjoittamiseen, mutta esimerkiksi keskustelufoorumit puuttuvat edelleen.

7.4.6. Yhteenveto

Sisällönhallintajärjestelmien teknisissä toteutuksissa on suuria eroja. Jotkin järjestelmät ovat omaksuttavissa hyvin nopeasti, kun taas sovelluskehys vaatii omaksumisen lisäksi myös ohjelmointitaitoja. Vaikka yksinkertaisen järjestelmän valinta tuntuisi houkuttelevalta, sivuston vaatimukset on pidettävä erityisen tarkkaan mielessä. Erityisesti käyttäjä- ja lukuoikeuksien toteutus on asia, johon kannattaa kiinnittää huomiota. Liian yksinkertainen oikeuksien toteutus kostautuu moninkertaisena työmääränä silloin, kun tarvitaan oletusasetuksia tarkempia oikeusmäärittelyitä. Tällöin on syytä harkita vakavissaan, olisiko monimutkaisen järjestelmän opettelu sittenkin kannattavaa. Siihen kulutettu aika tulee ansaittua takaisin nopeasti, kun vältytään suppeamman järjestelmän rajoitusten aiheuttamalta lisätyöltä.

8. Yhteenveto

Tässä tutkimuksessa käsiteltiin verkkosivuston uudistusprojektia ensin kokonaisuutena yleisellä tasolla. Tämän jälkeen esimerkkiprojektia apuna käyttäen käsiteltiin lisäksi sisällönhallintajärjestelmän valintaa ja esimerkkisivuston toteutusvaihetta. Tutkimuksen painopiste oli erityisesti valmiissa sisällönhallintajärjestelmissä, joilla pystytään toteuttamaan monipuolisia verkkosivustoja suhteellisen pienellä vaivalla. Sisällönhallintajärjestelmän valinta -luvussa tehtiin katsaus sisällönhallintajärjestelmien yleisiin ominaisuuksiin ja mahdollisuuksiin. Tämän jälkeen esiteltiin erästä verkkosivuston uudistusprojektia ja sen osana rakennettua esimerkkisivustoa. Sivusto toteutettiin Drupal-sisällönhallintajärjestelmällä, jonka toiminta esiteltiin myös lyhyesti. Myöhemmin sivuston toteutuksesta poimittiin joitakin kohdattuja ongelmia ja mielenkiintoisia huomioita. Lisäksi arvioitiin sitä, miten toteutus olisi onnistunut muutamilla muilla toteutustavoilla.

Verkkosivuston uudistusprojekti saattaa kuulostaa helpolta tehtävältä. Näin voikin olla, mikäli pohjatyöt tehdään riittävän huolellisesti. Monesti projektiin saatetaan lähteä jopa tietämättä varsinaisesti miksi sivusto pitäisi uudistaa tai mitä toiminnallisuutta uudelta sivustolta halutaan. Vanhan sivuston analysointi, lähtökartoitus ja käyttäjähaastattelut tehtäväanalyysin apuna vievät aikaa, mutta ovat nähdyn vaivan arvoisia. Kun sivusto on ”valmis”, täytyy sen kunnosta silti huolehtia jatkossakin esimerkiksi säännöllisen auditoinnin kautta.

Sisällönhallintajärjestelmän valinta helpottuu huomattavasti, kun uuden sivuston ominaisuudet ovat selvillä. Käytettävä järjestelmä ei lopulta ole tärkein tekijä lopputuloksen kannalta. Valintaa tehdessä pitää vain huomioida, että kaikki toivotut asiat saadaan valittavalla sisällönhallintajärjestelmällä toteutettua. On myös selvitettävä, että valittu järjestelmä ei ole liian monimutkainen opittava projektin aikarajan puitteissa. Jos järjestelmää on tarkoitus käyttää osana jokapäiväistä työntekoa (tiedostojen jako, keskustelut, blogit, työryhmätilat), sen on oltava riittävän nopea, luotettava ja helppokäyttöinen. Muuten käyttö voi jäädä vähälle käyttäjien löytäessä korvaavia menetelmiä tehtävistä suoriutumiseen. Toisaalta liian suppea järjestelmä voi rajoittaa käyttöä liikaa, jolloin lopputulos on sama käyttäjien hakeutuessa muiden välineiden käyttäjiksi.

Esimerkkiprojektissa verkkosivuston testiversiota lähdettiin toteuttamaan melko pian määrittelyjen jälkeen. Toinen mahdollisuus olisi ollut opetella käytettävän sisällönhallintajärjestelmän ominaisuuksia ennen esimerkkisivuston perustamista. Tästä ei välttämättä olisi ollut mainittavaa hyötyä lopputuloksen kannalta. Testisivustoa tehtäessä opitaan välineen toiminta käytännössä, eivätkä tehdyt virheet kuitenkaan välity lopulliseen sivustoon.

Jatkokysymyksiä voitaisiin esittää esimerkiksi siitä, mitkä tekijät ohjaavat sisällönhallintajärjestelmien kehittymistä. Useat järjestelmät tarjoavat työkaluja hyvin monen-

laiseen sisällöntuotantoon. Millainen verkkosivusto olisi mahdollista toteuttaa yhdistelemällä useita tiettyyn sisältötyyppiin keskittyviä järjestelmiä? Voitaisiin esimerkiksi tutkia käytännön tapoja yhdistää erikoistunut keskustelufoorumi, hyvät ryhmätyöominaisuudet tarjoava järjestelmä ja esimerkiksi dokumenttien julkaisujärjestelmä. Onko olemassa tai kehitteillä standardeja, joilla sisällönhallintajärjestelmät saataisiin toimimaan saumattomassa yhteistyössä keskenään? Kaikilla järjestelmillä on omat puutteensa ja vahvuutensa. Sisällön määrän jatkuvasti kasvaessa yhteistoiminnan kehittäminen ja järjestelmän vaihtamisen helpottaminen voi tulevaisuudessa olla entistä tärkeämpää.

Viiteluettelo

[Adobe, 2008] Adobe systems, inc., *Flash player version penetration,* http://www.adobe.com/products/player_census/flashplayer/version_penetration.html (tarkastettu 22.4.2008)

[Adobe Dreamweaver,2008] Adobe systems, inc., *The official home of Adobe Dreamweaver CS3,* http://www.adobe.com/products/dreamweaver/ (tarkastettu 9.5.2008)

[Alexandrou, 2004] Marios Alexandrou, *Five Types of Web Sites,* http://www.mariosalexandrou.com/blog/?p=42 (tarkastettu 20.4.2008)

[Blogger, 2008] Google, *Blogger: Luo blogi nyt – maksutta,* https://www.blogger.com/start (tarkastettu 9.5.2008)

[Boucher and Smith, 2000] Jan Boucher and Marion Smith, *Web site redesign follies,* Proceedings of the 28th Annual ACM SIGUCCS Conference on User Services (SIGUCCS '00), (2000), 14-18.

[Brain, 2005] Marshall Brain, *The different types of web sites,* http://webkew.blogspot.com/2005/04/lesson-2-different-types-of-web-sites.html (tarkastettu 20.4.2008)

[Brown et al., 2005] James Brown, Mariela Hristova, Thomas Nelson and Matthew Russel, *Towards a dynamic community identity: Transitioning to a CMS-driven CWRL web site,* http://www.cwrl.utexas.edu/node/160 (tarkastettu 20.4.2008).

[Buckinghamshire, 2008] Buckinghamshire County Council, *Homepage,* http://www.buckscc.gov.uk/bcc/ (tarkastettu 20.4.2008)

[Buytaert, 2006] Dries Buytaert, *Drupal Backward compatibility,* http://buytaert.net/backward-compatibility (tarkastettu 20.4.2008)

[Carlin, 2006] Sascha A. Carlin, *PHP bytecode cacher review october 2006,* http://itst.net/wp-content/uploads/2006/10/PHP%20Bytecode%20Cacher%20Review.pdf (tarkastettu 27.4.2008)

[Chou, 2002] Elaine Chou, *Redesigning a large and complex website: How to begin, and a method for success.* Proceedings of the 30th Annual ACM SIGUCCS Conference on User Services (SIGUCCS '02), 2002, 22-28.

[Compiz Fusion, 2007] Compiz fusion team, *Compiz fusion release 0.5.2 is out!* http://lists.compiz-fusion.org/pipermail/community/2007-August/000091.html (tarkastettu 20.4.2008)

[Deshpande et al., 2002] Yogesh Deshpande, Anupama Chandrarathna and Athula Ginige, *Web site auditing - First step towards re-engineering*. Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering (SEKE '02), 2002, 731-737.

[Django, 2008] Lawrence Journal-World, *Django, The web framework for perfetionists with deadlines,* http://www.djangoproject.com (tarkastettu 20.4.2008).

[Drupal, 2008] Drupal.org, *drupal.org | Community plumbing,* http://drupal.org/ (tarkastettu 20.4.2008).

[Drupal 5 getting started, 2008] Drupal.org, *Drupal 5 getting started,* http://drupal.org/files/getting-started_2.pdf (tarkastettu 20.4.2008).

[Drupal 5 theme guide, 2008] Drupal.org, *Drupal 5 theming guide,* http://drupal.org/theme-guide/5 (tarkastettu 20.4.2008).

[Drupal requirements, 2008] Drupal.org, *System requirements,* http://drupal.org/requirements (tarkastettu 20.4.2008).

[Drupal 6 rel. notes, 2008] Drupal.org, *Drupal 6.0 released,* http://drupal.org/drupal-6.0 (tarkastettu 22.4.2008).

[Episerver, 2007] Episerver AB, *Product Brochure, Episerver CMS 5 (Finnish),* http://www.episerver.com/downloads/Documents/Product_Brochures_EPiServer_CMS5/Product_Brochure_EPiServer_CMS5%20(Finnish).pdf (tarkastettu 20.4.2008).

[Episerver käsikirja, 2007] Episerver AB, *Episerver CMS 5 toimittajan käsikirja,* http://www.episerver.com/downloads/Documents/AuthorManuals/Suomi/Editors%20Manual%20EPiServer%20CMS%251%20(Finnish)%20RevA.pdf (tarkastettu 27.4.2008)

[Fogg et al., 2003] B.J. Fogg, Cathy Soohoo, David R. Danielson, Leslie Marable, Julianne Stanford and Ellen R. Tauber. *How Do Users Evaluate the Credibility of Web Sites?: A Study with Over 2,500 Participants.* Proceedings of the 2003 Conference on Designing For User Experiences (DUX '03), 2003.

[Gartner, 2008] Gartner, Inc. *The state of open source, 2008,* http://www.gartner.com/DisplayDocument?id=638643 (tarkastettu 20.4.2008).

[Goto ja Cotler, 2003] Kelly Goto ja Emily Cotler, *Verkkopalveluprojekti.* Edita Prima, Helsinki, 2003.

[Griffiths et al., 2002] G. Griffiths, B.G. Hebbron, M.A. Lockyer and B.J. Oater. *A simple method & tool for web engineering.* Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering (SEKE '02), 2002, 755-762.

[Halvey and Keane, 2007] Martin Halvey and Mark Keane, *An assessment of tag presentation techniques.* Proceedings of the 16th international Conference on World Wide Web (WWW '07), 2007, 1313-1314.

[Inplace CMS, 2008] Inplace CMS, *Inplace CMS for static pages,* http://inplace.sourceforge.net/index.html (tarkastettu 27.4.2008)

[Knaddison, 2008] Greg Knaddison, *New i18n and l10n features in Drupal 6,* http://www.masteringdrupal.com/screencast/new-i18n-l10n-features-drupal-6 (tarkastettu 22.4.2008)

[Krug, 2006] Steve Krug, *Don’t make me think: A common sense approach to web usability, second edition.* New riders publishing, Berkeley, USA (2006).

[Kuznetsov, 2006] Stacey Kuznetsov, *Motivations of contributors to wikipedia.* SIGCAS Comput. Soc. 36 (2006).

[Lipman, 2003] Doug Lipman, *Types of Web Sites*, http://www.storydynamics.com/Articles/Professional_Development/web_types.html (tarkastettu 20.4.2008).

[Lutz and Ascher, 2003] Mark Lutz and David Ascher, *Learning Python, second edition.* O'Reilly media, inc., Sebastopol, USA, 2004.

[Morrison et al., 2002] Mike Morrison, Joline Morrison and Anthony Keys, *Integrating web sites and databases*. Commun. ACM 45, 2002, 81-86.

[Myspace, 2008] Myspace.com, *Myspace*, http://www.myspace.com/ (tarkastettu 22.4.2008)

[Netscape, 1998] Netscape, *Netscape accelerates communicator evolution with first release of next-generation communicator source code to developer community via mozilla.org*, http://wp.netscape.com/newsref/pr/newsrelease591.html (tarkastettu 20.4.2008).

[Niederacher, 1999] Klaus Niederacher and Alexander Wahler, *Concept for content administration of database powered multimedia websites.* Proceedings of the Seventh ACM international Conference on Multimedia (MULTIMEDIA '99), 1999.

[Näkövammaisten keskusliitto, 2008] Näkövammaisten keskusliitto, *Näkövammaisten tietotekniset apuvälineet*, http://www.nkl.fi/kuntoutus/atk/apuvaline.htm (tarkastettu 20.4.2008).

[NCSA Mosaic, 2008] University of Illinois, *About NCSA Mosaic*, http://www.ncsa.uiuc.edu/Projects/mosaic.html (tarkastettu 20.4.2008).

[Nielsen, 2000] Jakob Nielsen, *Alertbox, January 9, 2000: Is navigation useful*, http://www.useit.com/alertbox/20000109.html (tarkastettu 20.4.2008).

[Open source CMS] The open source collective, inc., *OpenSourceCMS*, http://www.opensourcecms.com/ (tarkastettu 20.4.2008).

[Open source CMS - Typo3, 2008] Open source CMS, *Typo3 - OpenSourceCMS*, http://www.opensourcecms.com/index.php?option=com_content&task=view&id=181 (tarkastettu 27.4.2008)

[Packt, 2007] Packt Publishing Ltd, *2007 Open Source CMS Award, 2008*, http://www.packtpub.com/award (tarkastettu 20.4.2008).

[Parr, 2006] Terence Parr, *Web application internationalization and localization in action*. Proceedings of the 6th international Conference on Web Engineering (ICWE '06), (2006), 64-70.

[Peatling, 2007] Andy Peatling, Blaze new media, *Five Wordpress "CMS enabling" plugins*, http://blazenewmedia.com/articles/five-wordpress-cms-enabling-plugins/ (tarkastettu 27.4.2008)

[Phpro, 2007] Professional PHP Development, *PHP Tutorials examples PHP coding style*, http://www.phpro.org/tutorials/PHP-Coding-Style.html (tarkastettu 20.4.2008).

[Ritter et al., 2005] Frank Ritter, Adrew Freed and Onida Haskett, *Discovering user information needs: The case of university department web sites*. Interactions 12, 5 (Sep. 2005), 19-27.

[Rivadeneira et al., 2007] A. Rivadeneira, Daniel Gruen, Michael Muller and David Millen, *Getting our head in the clouds: toward evaluation studies of tagclouds.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07), 2007, 995-998.

[Roberts et al., 2003] Linda Roberts, Leigh Ranklin, Edward Silver, Darryl Moore, Stephanie Plunkett, David Washburn and Brenda Wilch-Ringen. *Looks good to me*. CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI'03), 2003, 818-819.

[Robertson, 2005] James Robertson, *Will your chosen CMS vendor go bust?* Step two designs, http://www.steptwo.com.au/papers/cmb_vendorbust/pdf/CMb_VendorBust.pdf (tarkastettu 22.4.2008)

[Robertson, 2006a] James Robertson, *Separate design and the CMS*, Step two designs, http://www.steptwo.com.au/papers/cmb_designcms/pdf/CMb_DesignCMS.pdf (tarkastettu 22.4.2008)

[Robertson, 2006b] James Robertson, *Top 10 mistakes when selecting a CMS*, Step two designs, http://www.steptwo.com.au/papers/kmc_selectionmistakes/pdf/KMC_SelectionMistakes.pdf (tarkastettu 22.4.2008)

[Second Life, 2008] Linden Research, Inc. *Second Life: Official site of the 3D online virtual world*, http://secondlife.com (tarkastettu 20.4.2008).

[Swarma, 2008] Swarma limited, *Hosting for an agile web*, http://www.webfaction.com/ (tarkastettu 22.4.2008)

[Tagadelic, 2008] Bèr Kessels, *Tagadelic*, http://drupal.org/project/tagadelic (tarkastettu 27.4.2008)

[Theofanos, 2003] Mary Theofanos ja Janice Redish. *Bridging the gap: between accessibility and usability*. ACM interactions 10, 2003, 36-51.

[Typo3, 2008] Typo3 association, *Typo3 content management system – developer resource*. http://typo3.org/ (tarkastettu 27.4.2008)

[Tähtinen, 2007] Mika Tähtinen, *Virtuaaliset kauppakassit*, Tietokone 7-8, Sanoma Magazines Oy, 2007, 60-66.

[vBulletin, 2008] vBulletin.org, *vBulletin.org Forum - The Official vBulletin Resource!*. http://www.vbulletin.org/forum/portal.php (tarkastettu 9.5.2008)

[W3C, 2008] World Wide Web Consortium, http://WWW.w3.org (tarkastettu 20.4.2008).

[W3C CSS, 2008] World Wide Web Consortium, *Cascading Style Sheets Specifications*, http://WWW.w3.org/Style/CSS/#specs (tarkastettu 20.4.2008).

[W3C HTML, 2008] World Wide Web Consortium, HTML 4.01 Specification, http://www.w3.org/TR/html401/ (tarkastettu 23.4.2008)

[Weitzman et al., 2006] Louis Weitzman, Alister Lewis-Bowen and Stephen Evanchick, *Using open source software to design, develop, and deploy a collaborative web site, Part 1: Introduction and Overview.* http://www.ibm.com/developerworks/ibm/library/i-osource1/ (tarkastettu 20.4.2008).

[Welho, 2008] SW Television, *100 megan laajakaista tulee Welhon kaapeliin,* http://www.welho.fi/Yksityisille/Laajakaista/LisatietoaJaOhjeita/100M/tabid/72393/Default.aspx (tarkastettu 23.4.2008)

[Wordpress, 2008] Wordpress.org, *Wordpress > blog tool and weblog platform*. http://wordpress.org/ (tarkastettu 27.4.2008)

[Xavier University Library, 2008] Xavier University Library, *xu.tutor: Evaluating Websites: Types of Websites*, http://WWW.xavier.edu/library/xututor/evaluating/types_of_websites.cfm (tarkastettu 20.4.2008).

[Kingsley-Hughes, 2008] Adrian Kingsley-Hughes (ZDNet) *Firefox 3.0 beta 5 - fastest browser yet!* http://blogs.zdnet.com/hardware/?p=1648 (tarkastettu 20.4.2008).