PROFILOINTI JA DYNAMIIKKA MULTIMEDIATUOTTEESSA

OPINNÄYTETYÖ

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto

Kevät 2006

Jani Välkky

Lahden ammattikorkeakoulu
Mediatekniikan koulutusohjelma

VÄLKKY, JANI: Profilointi ja dynamiikka multimediatuotteessa

Teknisen visualisoinnin opinnäytetyö
71 sivua, 1 liitesivu

Kevät 2006

TIIVISTELMÄ

Galvatek Oy kaipasi yrityksen ja sen tuotteiden esittelemistä varten uuden ratkaisun, joka mahdollistaisi personoidut ja dynaamiset esitykset, olisi helppo ylläpitää ja toisi uutta ilmettä verrattuna aiemmin käytettyihin staattisiin PowerPoint-kalvoihin. Tämä opinnäytetyö käsittelee niitä perusperiaatteita ja tekniikoita, jotka liittyvät Galvatekille luodun yritysesittelysovelluksen suunnitteluun ja toteutukseen. Työn tarkoituksena oli tutkia, kuinka profiileja käyttämällä katsojalle voitaisiin välittää juuri häntä kiinnostavaa sisältöä ja mitkä tekniikat mahdollistaisivat sisällön muokkaamisen mahdollisimman helpolla tavalla.

Personoimalla sisältö voidaan katsojalle välittää oleellista, juuri häntä kiinnostavaa sisältöä. Profiilit voidaan luoda automaattisesti, seuraamalla käyttäjää tai ne voidaan laatia etukäteen hyödyntäen olemassa olevaa markkinatutkimusmateriaalia. Sisällön personointi voidaan hoitaa esimerkiksi automaattisesti profiilissa olevin avainsanojen perusteella. Yksinkertaisimmillaan personointi toteutuu, kun käyttäjä itse valitsee kohdennetun esityksen listalta.

Flashiin on mahdollista hakea tietoa suoraan teksti- tai XML-tiedostoista. Palvelimella sijaitsevia sivuja hyödyntäen voidaan sisältö hakea myös tietokannoista ja WWW-sovelluspalveluista. Tietoa siirretään Flashiin muuttuja-arvopareina tai XML-muodossa. XML:n käsittelyssä hyödynnetään yleensä ActionScriptin XML-luokkaa ja sen metodeja. Vaihtoehtoisesti voidaan käyttää valmiita komponentteja, joiden avulla työskentely tapahtuu täysin visuaalisesti.

XML on standardi, jonka todettiin sopivat hyvin tiedon tallennusmuodoksi. Se mahdollistaa oman helposti luettavan kuvailukielen luomisen ja on yhteensopiva monien järjestelmien kanssa. XSL-kielen avulla XML-dokumentteja voidaan helposti muuntaa ja tarjota kohdennettua sisältöä eri kanaviin. XML-tiedostojen muokkaaminen onnistuu tavallisella tekstieditorilla, mutta käytännöllisempää on hyödyntää XML-editoreita tai esimerkiksi Microsoftin Wordia, Exceliä tai Accessia. Flashin ja palvelinsivun yhdistelmä mahdollistaa lomaketyyppiset ratkaisut, joilla päivitykset voidaan toteuttaa entistä käyttäjäystävällisemmin.

Galvetekille tehty yritysesittelysovellus toteutettiin käyttäen Flash 8:aa ja sisältö tallennettiin XML-tiedostoihin. Tehty case oli yritykselle mieluinen ja osoitti, että Flashin avulla pystytään luomaan näyttäviä, helposti muokattavia, dynaamisia ja monipuolisia kohdennettuja esityksiä.

Asiasanat: Galvatek, Flash 8, XML, XSL, multimedia, personointi, kanavointi.

Lahti University of Applied Sciences
Faculty of Technology

VÄLKKY, JANI: Profiles and Dynamic Content in Multimedia Products

Bachelor's Thesis in Visualization Engineering
71 pages, 1 appendix

Spring 2006

ABSTRACT

Galvatek Oy needed a new tool for presenting their company and products in a more dynamic and personalized way. One main criterion was that the content would be more easily manageable and editable than it was with their current PowerPoint solution. This thesis concentrates on studying the main principles and techniques involved in creating the new company presentation software for Galvatek. The goal was to find out ways how to deliver personalized and interesting content to viewers, based on profiles. A technique that would make content editing as easy as possible also had to be found.

By utilizing profiles and personalizing the content one can deliver viewers information that actually interests them. Profiles can be generated automatically by studying the user's actions, or they can be created beforehand, based on market research material. The actual personalization can be done automatically by using keywords to find and display relative content. The simplest way of personalization is selecting interesting options from a list of choices.

Flash was selected as the main tool for creating the new company presentation software and the content was stored in XML files. Data can be imported to Flash directly from texts or XML files or one can use server-side pages to connect and load information from remote databases or web services. Data is transferred to Flash using attribute-value pairs or in XML format. XML data is usually processed in Flash, using the ActionScipt XML class and its methods. If one prefers to work visually, an alternative is to use the pre-made Flash components.

XML is a standard and was found to be an excellent format for saving data. It is compatible with many different kinds of software and still enables the possibility to create customizable tags and structures. XML documents can be transformed to suit different channels by using the XSL language. The actual editing of the XML files can be done using the simplest text editors, but it is much more practical to use official XML editors or programs such as Microsoft Word, Excel or Access. The combination of server-side pages and Flash can be used to create a form type page to edit and save XML content in a user-friendly way.

The company presentation software made for Galvatek Oy was considered a success and very useful by the target company. This showed that Flash is very versatile software if you need to create stylish, easily editable, dynamic and personalized multimedia presentations.

Keyword: Galvatek, Flash 8, XML, XSL, multimedia, profiles, channels.

SISÄLLYS

TERMEJÄ JA LYHENTEITÄ

DTD (Document Type Definition)

REST (Representational State Transfer)

RSS (Really Simple Syndication)

SOAP (Simple Object Access Protocol)

UDDI (Universal Description, Discovery, and Intregration)

XML (Extensible Markup Language)

XHTML (Extensible Hypertext Markup Language)

ActionScript on Flashin oma ohjelmointikieli, jonka nykyinen versio on 2.0.

Container element on elementti, joka voi sulkea sisäänsä asiasisältöä ja toisia elementtejä.

Dynaaminen sivu muodostuu kokonaan tai osittain web-palvelimella vastauksena käyttäjän lähettämään palvelupyyntöön. On staattisen sivun vastakohta.

Evästeet (cookies) ovat selaimen kovalevylle tallentamaa dataa käyttäjästä. Palvelin voi pyytää evästettä seuraavalla käyttökerralla ja hakea niistä käyttäjän aiemmin jo antamaa tietoa.

Jäsennin (parser) on tietokoneohjelma, jolla tarkistetaan onko jokin sille syötteenä annettu merkkijono rakenteeltaan tietyn kielen kelvollinen ilmaisu. Validoiva XML-jäsennin tarkistaa lisäksi, onko syöte XML-kielen yleisten sääntöjen mukainen ja onko se annetun dokumenttityyppimäärittelyn mukainen.

Luokka (class) tarkoittaa olio-ohjelmoinnissa olion mallia. Luokan ohjeiden mukaan luodaan ohjelmaa ajettaessa olio (object).

Metodi tarkoittaa olio-ohjelmoinnissa kyseisen olion tarjoamaa palvelua eli vastausta oliolle lähetettyyn sanomaan.

Neuroverkko on tiedon automaattista analysointia ja luokittelua varten kehitetty tekniikka, joka hyödyntää visuaalisia karttoja tulosten esittämiseen.

Olio on olio-ohjelmoinnissa tietorakenne, joka sisältää ominaisuuksia ja toimintoja.

Ontologia, viittaa tässä työssä W3C:n hyväksymään semaattisen webin standardiin, johon liittyy keskeisesti OWL-kieli. OWL on DAML+Oil-, RDF- ja DL-pohjainen XML-kieli, josta koneellisestikin voidaan päätellä ja johtaa uutta tietoa.

Relaatiokanta on tietokanta, jossa tiedot ja niihin liittyvät relaatiotiedot esitetään ja tallennetaan taulukkoina, joissa on rivejä ja sarakkeita. SQL-kielen vakiokomennoilla tietoa tietokannasta voidaan sekä hakea että yhdistellä.

Rich Media kuvaa WWW-sivulla olevaa interaktiivista multimedia-osaa, jonka etenemiseen myös sivuilla kävijällä on mahdollisuus vaikuttaa. Rich Media -elementit on yleensä tehty Flashilla, mutta myös virtaavat videoleikkeet lasketaan kuuluviksi Rich Mediaan.

Skeema on malli, jolla kuvataan XML-dokumenttien tiedon rakenne. Skeemat noudattavat itsekin XML-standardia.

Skripti on lyhyenpuoleinen usein toisen tietokoneohjelman sisällä sellaisenaan suoritettava tietokoneen ohjelmakoodi.

Staattinen sivu on sisältöpalvelimelle tallennettu valmis sisältötiedosto, joka lähetetään sellaisenaan vastauksena käyttäjältä tulleeseen palvelinpyyntöön. Tavalliset HTML-sivut, jotka on tallennettu web-palvelimen tiedostorakenteeseen, ovat staattisia sivuja.

Web-palvelin on ohjelmisto, joka ottaa vastaan selaimelta tulevia palvelupyyntöjä ja lähettää niihin vastauksia. Sen takana sijaitsee sivusto, sisällön hallinta tai sovelluspalvelin.

1 JOHDANTO

Yrityksillä on tänä päivänä haasteinaan kiristyvä globaali kilpailu, jonka luomiin ongelmiin markkinoinnin on vastattava. Asiakkaille pitäisi pystyä tarjoamaan ajantasaista ja juuri heitä kiinnostavaa tietoa ja esittää se mieleenpainuvalla tavalla. Hyvin suunnitellun kohdennetun multimediaesityksen pitäisi pystyä vastaamaan näihin haasteisiin sekä auttamaan uusien asiakkaiden tai tilausten saamisessa ja tuottaa kannattavaa liiketoimintaa.

Yksi multimediaan liittyvistä haasteista on, että tiedot muuttuvat jatkuvasti nopeaan tahtiin. Perinteiset staattiset PowerPointin kaltaiset esitykset eivät välttämättä kykene muuntumaan tarpeeksi nopeasti nykypäivän yrityksen tarpeiden mukaan. Tässä työssä pyritään tutkimaan ratkaisuja, joita hyödyntäen voitaisiin luoda visuaalisesti näyttäviä, personoituja ja dynaamisia esityksiä, joiden sisältöä voidaan nopeasti muuntaa. Yhtenä tavoitteena olisi tutkia, kuinka suuresta määrästä tietoa voitaisiin valmiiksi seuloa katsojaa kiinnostavaa tietoa ja tarjota siten personoidumpia elämyksiä. Personointiin liittyy keskeisesti profiilit, jotka kuvaavat käyttäjää ja ohjaavat sisällön tai ulkoasun personointia. Koska multimediaesityksiä voidaan nykyisin katsoa myös mobiililaitteilla, työssä tutustutaan myös siihen, kuinka samaa esityksen sisältöä voidaan hyödyntää useassa eri paikassa ja käyttötarkoituksessa.

Tässä työssä keskitytään tutkimaan Flash 8:n tarjoamia mahdollisuuksia tuottaa personoituja ja dynaamisia esityksiä. Dynaamisuudella tässä yhteydessä tarkoitan, että esityksissä käytetään muuttuvaa ulkoista sisältöä. Työssä tutustutaankin eri tekniikoihin, joiden avulla Flashiin voidaan tuoda tietoa ulkopuolisista lähteistä kuten uutisvirroista, tietokannoista, teksti- ja XML-tiedostoista. XML on keskeisessä asemassa tässä työssä ja monet esitellyt profiloinnin ja dynamiikan keinot pohjautuvat XML:ään. Työssä perehdytään siihen, mikä XML on ja miten sitä voitaisiin hyödyntää tiedon varastoimisessa ja välittämisessä järjestelmästä toiseen. Tutkitaan myös eri työkaluja, jotka mahdollistavat työskentelyn XML-muotoisen tiedon kanssa.

Tutkittuja teorioita hyödynnän lopun case-osiossa, jossa Galvatek OY:lle luotiin Flashilla uusi yritysesittelysovellus, joka korvasi aiemmin käytetyt PowerPoint-esitykset. Ne oli havaittu vanhahtavan näköisiksi, hankalasti hallinnoitaviksi eivätkä pystyneet tarjoamaan yksinkertaista ratkaisua personoidun sisällön esittämiseksi. Galvatek halusi uudistaa ja yhtenäistää esityksien ulkoasua, tarjota asiakkailleen personoidumpaa sisältöä ja luoda samalla vahva kuva Galvatekin osaamisesta suunnittelussa ja automaatiossa. Sovelluksen suunnittelussa tuli ottaa huomioon eri asiakkaat ja mahdollistaa personoitujen esitysten näyttäminen. Sisällön helppo päivitettävyys oli toinen keskeinen vaatimus, minkä takia oli tärkeää löytää ratkaisu, jonka avulla esitysten sisältöä voitaisiin muokata ilman Flashia. Annettu tehtävä oli haasteellinen mutta mielenkiintoinen.

2 MULTIMEDIAESITYKSET

2.1 Määritelmät

Dynamiikka ja profilointi multimediatuotteessa -otsikko sisältää jo heti kolme sanaa, jotka voivat tarkoittaa hyvinkin montaa eri asiaa. Siksi työssä lähestytään tätä laajaa aihetta määrittelemällä ja tarkentamalla noita termejä.

Multimedia on käsitteenä äärimmäisen laaja, ja Wikipedian tietojen mukaan IT-kuplan aikaan se oli yksi suosituimmista hypetyssanoista, joilla pyrittiin vain nostamaan projektin markkina-arvoa ilman todellista merkitystä. Multimediassa käytetään hyödyksi useita erilaisia ilmaisun muotoja, kuten tekstiä, kuvaa, liikettä ja ääntä, muodostamaan yhden kokonaisuuden. (Multimedia 2006.)

Vaikkei multimedia-termiä käytetäkään nykyään yhtä villisti kuin IT-kuplan aikoina, on sen käyttö edelleen melko väljää. Äärimmillään multimediaa voi olla kaikki, mikä ei ole pelkkää tekstiä, mutta yleensä se jää määritelmän ulkopuolelle. Multimedia tarkoittaa Webissä liikkuvan kuvan ja mahdollisesti äänen liittämistä sivuihin. Yleensä kyseessä on videoleike tai tietokoneella toteutettu animaatio tai Flash-esitys. (Korpela & Linjama 2003, 261.)

Multimediaesitys määritellään Tekniikan sanastokeskuksen (TSK) sivuilla olevan: ”multimedia-aineistoon perustuva esitys“ (2006). Tässä työssä multimediaesityksillä tarkoitetaan pääasiassa Macromedian Flashilla toteutettuja multimediaesityksiä, jotka voivat sisältää tekstiä, ääntä, kuvaa, videota tai animaatioita. Vaikka tässä työssä keskitytäänkin Flashilla toteutettuun multimediaan, voi monia esiteltäviä profiloinnin ja dynaamiikan tekniikoita hyödyntää työskenneltäessä myös muiden multimediantuotantoon soveltuvien ohjelmien parissa. Ideoita voi hyödyntää myös suunniteltaessa rikkaita Web-sivustoja, vaikka toteutukseen ei käyttäisikään Flashia.

2.2 Multimedian tuotanto-ohjelmat

2.2.1 Yleistä

Multimediatuotannossa tarvitaan ohjelma, joka pystyy tuottamaan valitun muotoisia esityksiä. Multimediatuotanto-ohjelmia on valtaisa määrä, eikä tämän työn tarkoitus ole lähteä tutkimaan niitä yksitellen. Työssä käsitellään joitain yleisimmin käytössä olevien ratkaisujen eroja ja perehdytään syvemmin Macromedian Flash-ohjelmiston tarjoamiin mahdollisuuksiin ja etuihin.

Multimediaa voi yksinkertaisuudessaan olla myös HTML-sivusto, jossa on linkkejä selaimen tunnistamiin kuva- ja äänitiedostoihin. Esimerkkinä voisi toimia sivusto, jossa on linkki MP3-muotoiseen äänitiedostoon. Käyttäjä pystyy lataamaan linkin

tiedoston ja kuuntelemaan sen. Tämä on varsin yksinkertainen tapa tuottaa multimediaa, mutta sopii hyvin tilanteisiin, joissa äänellä tai videolla ei ole suoraa yhteyttä tekstiin tai muuhun esitykseen. Linkkeihin pohjautuvia esityksiä voi luoda millä tahansa teksti- tai HTML-editorilla, eikä se vaadi HTML-osaamisen lisäksi muita taitoja. (Mischook 2006.)

2.2.2 PowerPoint

Yksinkertaisen multimediaesityksen saa luotua nopeasti käyttäen hyvin yleiseksi tullutta Microsoftin PowerPoint-ohjelmistoa. Mischookin mukaan PowerPoint on ehdottomasti helpoin tapa tehdä diaesityksiä, joissa on animoitujen siirtymien lisäksi ääni- ja videoelementtejä. PowerPoint esitykset ovat laajassa käytössä suuren yleisön hyväksymiä. Hän kuitenkin myöntää, että vaikka PowerPoint onkin nopea, sillä toteutettujen esitysten ulkoasu ei välttämättä häikäise katsojaa näyttävyydellään. (Mischook 2006.)

Jere Majava ilmaisee Piirtoheitin-verkkolehden artikkelissaan osuvasti keskeisen PowerPointia kuvaan ongelman: ”PowerPoint-esityksiä on helppo tehdä – mutta sitäkin tuskastuttavampi seurata”. PowerPoint-esityksiä näkee nykyisin joka paikassa, niihin törmää kokouksissa, konferensseissa, luennoilla ja jopa sähköpostien liitteinä. Majavan kirjoitus herätti myös laajempaa keskustelua sivulla, ja vastanneiden yleinen mielipide oli, että ihmisten koettiin tuottavan PowerPointilla samanlaisia, puuduttavia esityksiä, jotka voisi korvata myös muilla keinoin. Esimerkkeinä annettiin muun muassa PDF-dokumenttien ja webisivujen käyttöä vaihtoehtoisena esitystapana. (Majava 2004.)

Digitaalisen kuvantamisen ammattilainen Harry Waldman yhtyy mielipiteeseen, että PowerPointilla voidaan luoda ja esittää hyviä esityksiä mahdollisimman pienellä vaivannäöllä. Hän kuitenkin huomauttaa, että tässä on juuri se syy, miksi PowerPointille tulisi miettiä vaihtoehtoja. Kun esityksen tarkoitus on kouluttaa, opettaa, myydä tai motivoida, hyvä ei välttämättä ole riittävää. Toisilla ohjelmilla on kenties mahdollista esittää vielä havainnollisemmin jokin prosessi tai tekniikka. Yleistä luuloa vastaan Waldman huomauttaa myös, että joissain tapauksissa PowerPoint ei ehkä ole edes nopein tapa luoda visuaalista sisältöä esityksiin. (Waldman 2002.)

Toisaalta nykyaikana yritysmaailmassa on tärkeää pystyä tarjoamaan sisältöä nopeasti ja useassa eri muodossa, useisiin eri medioihin. PowerPoint ei pysty täysin vastaamaan näihin haasteisiin, ja Waldman esittääkin vaihtoehdoksi muun muassa Acrobatin tai Flashin käyttöä. Acrobatin edut ovat selkeästi mahdollisuuksissa käyttää hienostuneempaa grafiikkaa, fontteja ja muita design-konsepteja ja käyttää sisältöä hyödyksi useissa eri medioissa. Flash toisaalta mahdollistaa vieläkin näyttävämpien ja dynaamisempien esitysten muodostamisen, ja seuraavissa luvuissa tutkitaan multimediantuotantoa lähinnä Flashin näkökulmasta. (Waldman 2002.)

2.2.3 Flash

Yleisesittely

Flash on Macromedian (nykyisin Adoben omistama) kehittämä ja tällä hetkellä johtava, interaktiivisten web-sivujen, digitaalisten esitysten ja mobiilisisällön tuottamiseen tarkoitettu ohjelmisto. Flashia käyttää yli miljoona ammattilaista ympäri maailman, ja Flash Player on Macromedian tietojen mukaan asennettu tänä päivänä jopa 97,7 %:iin kaikista Internetiin kytketyistä koneista. Flash on tilastojen mukaan siis yleisempi kuin esimerkiksi Adobe Acrobat Reader. Tätä kirjoitettaessa Flashin uusimmat versiot olivat Flash 8 ja Flash 8 Professional. (Macromedia Flash Player Statistics, 2006.)

Flash multimedian tuotanto-ohjelmana

Flash tarjoaa mahdollisuuden luoda grafiikkaa, animaatiota ja interaktiivisuutta esityksiin täysin eri tasolla kuin esimerkiksi PowerPoint, jossa esitykset yleensä pohjautuvat lähinnä valmiisiin ja kaikille tutuksi tulleisiin pohjiin. Vaikka Flash tarjoaakin nykyisessä versiossaan jotain valmiita PowerPoint-tyylisiä pohjia, on perusperiaate se, että kaikki esityksen elementit luodaan itse. Kaikkiin elementteihin voidaan itse vaikuttaa ja asettaa niille haluttuja toimintoja, esimerkiksi mitä tapahtuu, kun kursori viedään jonkin kuvan päälle. (Waldman 2002.)

Esityksiä ei ole pakko toteuttaa Flashissa lineaarisesti, dia per dia -periaatteen mukaisesti, vaan Flash tarjoaa työkalut luoviin ja uusiin ratkaisuihin. Esityksen pystyy rakentamaan niin, että siellä voi linkkien avulla liikkua minne vain, milloin vain. Hyviin ominaisuuksiin lukeutuu vielä se, että näyttävästä sisällöstä ja animaatiosta huolimatta tiedostokoot pysyvät pienenä. (Waldman 2002.)

Waldmanin mukaan Flash voi olla monille hankala ohjelma oppia ja esitysten tekeminen voi viedä paljon aikaa. Hänen mukaansa Flash kuitenkin sopii ihmisille, jotka ovat valmiita käyttämään hieman enemmän aikaa ja vaivaa luodakseen visuaalisesti näyttävämpiä ja mieleenpainuvampia esityksiä.
(Waldman 2002.)

Flashin käytön osaaminen ja luova mieli mahdollistavat epäilemättä visuaalisesti näyttävien ja havainnollistan esitysten luomisen. Pelkkä ulkonäkö ja toimivat animaatiot eivät kuitenkaan tee esityksestä täydellistä, koska pohjimmiltaan sisältö on se, joka on tärkeintä (Samela 2002, 159). Katsojalle pitäisi pystyä näyttämään kohdennettua mielenkiintoista katsottavaa, jotta haluttu viesti saataisiin perille. Luvussa kolme tutustutaan periaatteisiin ja tekniikoihin, jotka auttavat tekemään tämän mahdolliseksi.

2.3 Multimediaesitysten tuottaminen

Samalla tavalla kuin web-sivujenkin toteuttamiseen, liittyy multimediatuotantoon myös suuri määrä suunnittelua ja asioita, joita tulee ottaa huomioon ennen varsinaisen tekemisen aloittamista. Korpelan mukaan tulisi ensinnäkin aina miettiä vastausta kysymykseen: ”Miksi www-sivuja tehdään?” Yksityisellä henkilöllä syitä voi tietenkin olla lukuisiakin, kenties halu kokeilla uutta opittua tekniikkaa tai halu tehdä vain ”jotain hienoa”. Yritysmaailmassa www-sivuja ja myös multimediaesityksiä tehdään, koska on tarve saada jokin haluttu viesti perille kuulijoille. Esimerkiksi myyntimiehillä on tarve välittää mahdollisimman vakuuttava kuva yrityksestään ja tuotteestaan asiakkaalle, jotta kaupat saataisiin tehtyä. (Korpela & Linjama 2003, 48.)

Toinen tärkeä mietittävä asia on se, kenelle sivuja tehdään. Web-sivuilla voi käydä kuka tahansa, mutta Korpelan mukaan kannattaa silti aina miettiä, kenelle sivusto on ensisijaisesti tarkoitettu, kieltä myöten. Sivustolla voi kuitenkin olla useitakin tärkeitä kohderyhmiä, joiden kaikkien tarpeita tulisi huomioida. (Korpela & Linjama 2003, 48.)

Samat säännöt koskevat myös Flashilla toteutettuja multimediaesityksiä. Suunnittelu alkaa aina miettimällä, kenelle esitys on kohdennettu ja mikä on viesti, joka halutaan välittää katsojalle. Lisäksi tulisi miettiä, millaisella kohdelaitteella esitys tullaan näyttämään, sillä kuten Macromedia mainostaa, on Flash myös mobiili. Flashilla toteutettua multimediaa on nykyisin mahdollista katsoa perinteisen tietokoneruudun lisäksi melkeinpä millä tahansa laitteella, jossa vain on visuaalinen näyttö. Flashia tukeviin laitteisiin Macromedia listaa useita eri kännyköitä, pda-laitteita ja kodin elektroniikan tuotteita. (Mobile & Devices, 2006.)

3 PROFILOINTI

3.1 Yleistä

Samelan mukaan ”käyttäjät haluavat oikeaa ja ajan tasalla olevaa tietoa juuri sillä hetkellä, kun tiedolle on käyttöä”. Samela kiteyttää tässä lauseessa hyvin tarkkaan syyn sille, miksi tiedon profilointi on niin tärkeää. Valtavasta määrästä tietoa ei ole mitään hyötyä, jos se esitetään väärällä tavalla, se on hankalasti saatavissa tai se on sekavasti organisoitua. Tulevaisuudessa informaation helppo saatavuus tulee olemaan yhä tärkeämmässä asemassa. (2002, 159.)

Termille profilointi ei löydy yhtä ainoaa tarkkaa määritelmää. Tässä työssä profilointiin liittyy joukko eri käsitteitä, kuten personointi, kanavointi sekä itse profiilit ja niiden muodostaminen. Kaikki nämä teemat nivoutuvat tietyllä tavalla yhteen ja muodostavat yhdessä toiminnallisen kokonaisuuden, johon tässä työssä viitataan laajemmin termillä profilointi.

Seuraavissa luvuissa tullaan perehtymään niihin perusperiaatteisiin ja tekniikoihin, jotka liittyvät profilointiin. Asiaa käsitellään yleisesti web-tuotannon kannalta, ja käytetyt esimerkit ovat myös pääsiassa web-ympäristöistä tuttuja. Samat yleiset perusperiaatteet ja tekniikat koskettavat kuitenkin paljolti myös Flashilla toteutettuja multimediaesityksiä.

3.2 Personointi

3.2.1 Johdatus aiheeseen

Tiedon määrä kasvaa jatkuvasti ja samalla muuttuu yhä monimuotoisemmaksi. Internet on mahdollistanut tiedon levittämisen tehokkaasti eri kanavia pitkin, ja on varsin tyypillistä, että web-järjestelmät käyttävät samaa tietoa monessa eri paikassa. On tarpeen löytää keinoja, joilla tietoa voidaan hallinnoida entistä tehokkaammin. Hyvin organisoitu sisällön tuotanto- ja julkaisuprosessi mahdollistaa, että katsojalle ei tyrkytetä mitä tahansa aineistoa ja että sisällön laatu pysyy korkeatasoisena. Käyttäjää varten profiloitu korkeatasoinen sisältö lievittää siten osittain tietoähkyn aiheuttamaa tuskaa. (Samela 2002, 7 - 8.)

Tietotekniikan sovellus ilman sisältöä on hyödytön, tästä toimivat esimerkkinä hyvin monet epäonnistuneet Internetpalvelut. Internetin alkuaikoina käyttäjinä olivat lähinnä tekniikasta innostuneet ihmiset ja sovellukset suunnattiin tälle kohderyhmälle. Suuria massoja ei kuitenkaan yleensä kiinnosta mikään tekninen hienous, vaan sovelluksesta tai esityksestä saatava hyöty, joka on suoraan verrannollinen tarjottavaan sisältöön. Samelan mukaan viestinnällinen tarve on aina ensisijainen syy rakentaa web-järjestelmiä. Samaa ideaa tulisi mielestäni noudattaa myös Flash-multimediatuotannossa. (Samela 2002, 7 - 8.)

3.2.2 Personoinnin tavoitteet ja haasteet

Personoinnin tarkoituksena on parantaa käyttäjän saamaa palvelua ja kokemusta. Pyritään luomaan mahdollisimman henkilökohtaisia esityksiä ja kohdentamaan juuri haluttua sisältöä katsojalle. Web-ympäristöissä yhtenä tärkeänä tavoitteena voidaan pitää, että käyttäjä saataisiin mieltymään sivustoon ja palaamaan takaisin. Personoinnin motiiveina ovat monesti myös kaupalliset tarkoitukset, ja varsinkin monet mainospalvelut haluavat toteuttaa mahdollisimman personoituja ja sitä kautta tehokkaita mainoksia. (Kalliola 2004.)

Sisällön personointia käytetään hyväksi luotaessa henkilökohtaisempia ja osuvampia elämyksiä käyttäjille. Suuresta tietomäärästä yritetään seuloa katsojaa potentiaalisesti kiinnostavaa sisältöä ja jättää muu, oletetusti turha, tieto mahdollisesti kokonaan näyttämättä. Idea on yksinkertainen, mutta käytännön toteuttaminen sitäkin hankalampaa. (Samela 2002, 78.)

Personoinnin avulla voidaan luoda henkilökohtaisia palveluja, joissa käyttäjää palvellaan nimellä ja näytetään esimerkiksi kohdennettuja uutistietoja. Mikäli käyttäjän sijainti voidaan määritellä, voidaan paikkatietoa hyödyntää esimerkiksi paikallisuutisia näyttämällä. Käyttäjä pystyy mahdollisesti itse vaikuttamaan palvelun ulkonäköön ja toimintaan. Myös eri kieliversioita voidaan pitää personoinnin yhtenä muotona. (Kalliola 2004.)

Personoinnin keskeinen ongelma on se, että suunnittelijan on usein hankala tietää, mitä katsoja haluaisi nähdä tai mitä tietoa hän tarvitsee. Katsojan hakemat elämykset vaihtelevat muun muassa elämäntilanteen, tarpeiden ja mieltymysten mukaan. Samela esittää esimerkkinä ongelmatilanteen, jossa myyjä tekee jatkuvasti tarjouksia sinisistä villapaidoista, mutta asiakas haluaakin tällä kertaa punaisen. Asiakasta kiinnostavien asioiden ennakoiminen ei ole helppo tehtävä. (2002, 78.)

Internetin alkuaikoina monet nettisivut tarjosivat mahdollisuuden personoida sivujen ulkoasua, esimerkiksi värejä ja kirjasintyyppiä. Ulkoasuvalinnoista ei tullut kovinkaan suosittuja, sillä ne eivät tarjonneet merkittävän suurta lisäarvoa. Personointi koskeekin sen takia nykyisin lähinnä sivujen tieto- tai palvelusisältöä. Uutispalvelut on tyypillinen esimerkki personoidusta sisällöstä, palvelu pyrkii tarjoamaan asiakkaalle hänen toivomustensa mukaista uutissisältöä, joka voidaan lähettää hänelle esimerkiksi matkapuhelimeen. (Samela 2002, 79.)

3.2.4 Profiilit personoinnissa

Personointi perustuu aina tiettyihin, käyttäjään tai käyttäjäryhmään liitettyihin, parametreihin. Jotta käyttäjälle voitaisiin esittää jollain tavalla henkilökohtaisempaa tietoa, on käyttäjä ensin tunnistettava. Tunnistuksen jälkeen käyttäjä yhdistetään profiiliin, joka sisältää personointia ohjaavat asetukset. (Samela 2002, 79.)

Profiili itsessään on tietorakenne, joka kuvaa käyttäjää ja sisältää personointia ohjaavat parametrit. Profiilissa olevien tietojen avulla on mahdollista hakea tietomassasta käyttäjää kiinnostavaa tietoa ja ohjata näin dynaamista sivunmuodostusta. Asiasanat on yleisimmin käytetty keino toteuttaa sisällön personointi, asiasanana voi toimia esimerkiksi ”jääkiekko”. Käyttäjälle voitaisiin esimerkiksi hakea kaikista urheiluaiheisista uutisista vain jääkiekkoa koskevat artikkelit. Profiilissa voi olla lukuisia eri parametrejä, joille annetaan arvoiksi mieltymyksen mukaiset asiasanat. Asiasanoihin perustuva sisällön personointi on mahdollista toteuttaa tehokkaasti vain, jos alkuperäiseen sisältöön on liitetty kuvailutietoja käyttäen samoja asiasanoja. Avainsanoja voi yrittää toki hakea myös itse tekstimassasta, mutta se on kovin tehotonta ja osumatarkkuus on huono. (Samela 2002, 79.)

Personointitietoja ei kannata tallentaa suoraan sisällön yhteyteen, niin että esimerkiksi nettisivun metatiedoissa olisi lista profiileista, joille tieto näkyy. Profiilit kannattaa tallentaa erilleen itse sisällöstä, jolloin monesti hyödynnetään esimerkiksi tietokantoja. Toinen lähestymistapa Internet-ympäristöissä on tallentaa profiilitiedot selaimen evästeiksi. Muutaman asiasanan avulla pystytään yllättävän hyvin seulomaan sisältö haluttuun muotoon. Esimerkkejä ovat monet web-kaupat, joissa käyttäjän tekemien hakujen ja aiempien tilausten mukaan näytetään käyttäjälle mainoksia muista mahdollisesti kiinnostavista tuotteista. (Samela 2002, 79 - 81.)

Käyttäjän tunnistaminen tapahtuu Internetissä automatisoidusti esimerkiksi evästeiden tai istuntojen kautta. Evästeiden heikkoutena on se, että niiden avulla tunnistetaan työasema, ei varsinaista käyttäjää. Mikäli käyttäjä päättää vaihtaa konetta, samat profiilit eivät enää ole käytettävissä. Joillakin sivuilla käyttäjän tunnistaminen vaatii käyttäjän kirjautumisen sivuille, joka mahdollistaa tunnistamisen kaikkialta. (Samela 2002, 79 - 81.)

3.2.5 Personoinnin toteuttaminen

Sisällön personoimisen toteuttaminen on haastava prosessi, mikä johtuu muun muassa sisältötyyppien suuresta määrästä. Personoidut sivut ja muut esitykset pohjautuvat yleensä dynaamiseen sisältöön. Webissä käytetään tekniikkaa, jossa jokaiselle käyttäjälle tai käyttökerralle luodaan sivut aina uudestaan dynaamisesti. Sivu voidaan tuottaa kokonaan yhden dynaamisen sivun ohjelmakoodin perusteella, mutta tällainen ratkaisu on erittäin hankala ylläpitää. Sivu voidaan tuottaa myös yhdistelemällä pieniä palasia, mikä mahdollistaa helpommin muunneltavan sisällön. (Kalliola 2004.)

Sisällön personointiin ei aina välttämättä liity sisällön dynaamista muodostamista. Mikäli ennakkoon on jo tiedettävissä joukko profiileja, voidaan niitä varten toteuttaa omat sivunsa. Riittää, että tunnistetaan käyttäjä ja ohjataan tämä oikealle alisivulle. Periaate on varsin yksinkertainen, mutta mahdollistaa silti henkilökohtaisemman elämyksen katsojalle. Tekniikka sopii parhaiten tilanteisiin, joissa profiileja on vähän. (Samela 2002, 80 - 81.)

Yksinkertaisimmillaan personointi tapahtuu niin, että käyttäjä itse valitsee listasta häntä kiinnostavat aihealueet. Sisällön personointi onnistuu periaatteessa siis ilman käyttäjän tunnistamistakin. Monipuolisemmissa järjestelmissä pyritään kuitenkin päättelemään kiinnostuksen kohteet jo valmiiksi käyttäjän puolesta. (Kalliola, TKK Viestintätekniikan laboratorio, Personointi, 2004)

Esityksen personointi onnistuu huomattavasti helpommin kuin sisällön personointi, sillä käyttäjä yleensä hoitaa personoinnin itse, eikä prosessia tarvitse siten automatisoida (Kalliola 2004). Yksi esityksen personoinnin keino on tuttu intraneteistä, joissa personointi hoidetaan monesti rajaamalla käyttöoikeuksia. Tietyille käyttäjäryhmille tarjotaan pääsy vain rajattuun osaan sisällöstä. Pääkäyttäjät yleensä pystyvät näkemään kaiken sisällön, mutta tietyille alikäyttäjäryhmille näytetään vain heitä koskettavaa tietoa. Periaate on yksinkertainen, mutta tällaisen tietorakennelman ylläpito voi olla yllättävänkin hankalaa. (Samela 2002, 81 - 82.)

3.2.6 Profiilien muodostaminen

Yksittäisiä käyttäjiä koskevia profiileja muodostetaan web-ympäristöissä automaattisesti seuraamalla käyttäjän toimenpiteitä ja tallentamalla kerätty tieto selaimen evästeisiin (Samela 2002, 79). Joskus on tarvetta luoda profiileja jo etukäteen ja muodostaa sisältö niihin perustuen. Haasteina on pyrkiä ennalta arvaamaan, millaista sisältöä käyttäjä tai jokin käyttäjäryhmä mahdollisesti haluaa nähdä. (Sather, Ibanez & DeChant 1997, 36 - 38.)

Oltiinpa tuottamassa sitten tavallisia web-sivuja tai Flash-multimediaesityksiä, on aina tärkeää tietää mikä on kohdeyleisö. Tuote on parhaimmillaan silloin, kun katsoja voi tuntea, että näytettävä materiaali olisi tehty juuri hänelle. Katsojan täytyy kokea, että sisältö on kiinnostavaa ja koskettaa juuri häntä jollain tavalla. Tuotteen suunnittelussa on tärkeää ottaa huomioon kohdeyleisö ja eri asiakkaan ominaisuudet. Näiden tietojen pohjalta voidaan muodostaa käyttäjä- ja asiakasprofiileja. (Sather ym. 1997, 36 - 38.)

Asiakasprofiilit auttavat tunnistamaan asiakkaan henkilökohtaiset tarpeet. Profiilit auttavat yritystä kohdentamaan heidän markkinointiaan helpommin ja tehokkaammin. Oikein muodostetut profiilit auttavat säästämään aikaa ja lopulta myös tuottavat yritykselle enemmän rahaa, koska pystytään vähentämään hukkaan heitettyä aikaa vääränlaisen informaation esittämisessä asiakkaalle. (Kennaugh 2006.)

Monilla yrityksillä on jo valmiina olemassa oma brändi ja tarkasti laaditut asiakasprofiilit heidän tuotteilleen ja palveluilleen. Aina ei siis ole tarvetta aloittaa profiilien luomista tyhjästä. Valmista markkinatutkimusmateriaalia kannattaa muutenkin hyödyntää aktiivisesti, kun mietitään mahdollisia käyttäjiä luotavalle multimediatuotteelle. Pyrkimyksenä on tunnistaa kaikki mahdolliset käyttäjät ja luoda joukko profiileja havaittujen käyttäjäryhmien perusteella. (Sather ym. 1997, 36-38.)

Asiakasprofiilien tiedot jakautuvat yleensä kahden eri kategorian alle. On arvoja, jotka kuvaavat kuka asiakas on, esimerkkejä tällaisista tiedoista ovat ikä, sukupuoli, työpaikka, asuinpaikka, koulutus ja kansallisuus. Hieman käytännöllisempi tieto kuvastaa asiakkaan käyttäytymistä. Asiakkaat voivat jakautua esimerkiksi hintatietoisiin, trenditietoisiin tai kokeilunhaluisiin asiakkaisiin, joilla kullakin ryhmällä on omat kiinnostuksen kohteensa. (Kennaugh 2006.)

Profiilien muodostamisen vaadittavan tiedon kerääminen ei aina ole helppoa. Sen lisäksi, että voidaan hyödyntää aiempia profiileja ja markkinointitietoa, kannattaa hyödyntää myös yrityksen henkilökunnan omaa tuntemusta asiakkaista. Toisaalta myös kolmannen osapuolen tekemät tutkimukset asiakasryhmistä voivat osoittautua arvokkaaksi tiedonlähteeksi. Yksi parhaimpia informaation lähteitä ovat itse asiakkaat ja heidän kanssaan käydyt keskustelut. Mutta vaikka tietoa olisi kuinka paljon, joudutaan yleensä tekemään aina joitain omaan ammattitaitoon pohjautuvia oletuksia. Niitä tehtäessä kannattaa muistaa, ettei tee liian tiukkoja rajauksia ja että muotoilee arvot aina kuvaamaan jotain arvoväliä, esimerkiksi iäksi väli 25-35 eikä tasan 30. (Kennaugh 2006.)

Luotuja profiileja pitäisi muistaa aina myös soveltaa, jakaa ja päivittää usein. Ei saisi käydä niin, että profiili luodaan vain kerran, eikä huomioida ollenkaan asiakasryhmän mahdollisesti muuttuvia tarpeita. Profilointiin on syytä käyttää myös reilusti aikaa, kerättyä tietoa voidaan hyödyntää yrityksessä jatkossa lukuisissakin eri päätöksissä. Yrityksen valmiisiin asiakasprofiileihin on syytä myös suhtautua pienellä varauksella, sillä vaikka yritys luuleekin tuntevansa asiakkaansa, harva todellisuudessa on tutkinut asiaa perusteellisesti. (Kennaugh 2006.)

3.2.7 Huomioitavaa personoinnissa

Suunniteltaessa personointia hyödyntäviä järjestelmiä täytyy huomioida joitain tiettyjä perusasioita. Personointi ei saisi koskaan lisätä käyttäjän työmäärää, sillä kukaan ei todennäköisesti halua tehdä valtavasti töitä saadakseen hieman parempaa palvelua. Palvelun olisi syytä toimia myös ilman personointia, sillä kaikki käyttäjät eivät välttämättä hyödy personoinnista eivätkä edes halua personoitua sisältöä tai esitystapaa. (Kalliola 2004.)

Personointia varten kerättävä tieto on yleensä arkaluonteista tai henkilökohtaista. Nämä tiedot voivat olla yksityisyyden suojan alaisia, ja tulee olla hyvin tarkka, miten näitä tietoja käsittelee ja kerää. Tietojen täytyy liittyä tarjottavaan palveluun ja hyödyttää asiakasta. Yhdysvalloista löytyy paljon esimerkkejä, joissa käyttäjätietoja on kerätty luvattomasti ja päädytty sen takia jopa oikeussaliin asti. (Kalliola 2004.)

Nykyaikaiset tekniikat ovat vielä suhteellisen rajoittuneita kuvaamaan tietoa ja sen välisiä suhteita, eivätkä tekniikat ole vielä riittävän älykkäitä hoitamaan sisällön personointia täysin automatisoidusti (Kennaugh 2006). Parhaillaan kehitellään kuitenkin lukuisia eri tekniikoita, joilla voitaisiin paremmin hallita ja kuvata tietoa, ja helpottaa näin myös sisällön personointia. Uusiin tekniikoihin lukeutuvat muun muassa ontologiat ja uudenlaiset neuroverkot, joiden avulla pystytään luomaan jopa uutta tietoa yhdistelemällä olemassa olevaa tietoa uusilla tavoilla (Samela 2002, 166). Näiden tekniikoiden esittely vaatisi aivan oman työnsä, joten niihin tutustuminen jää lukijan itsensä hoidettavaksi.

3.3 Kanavointi

3.3.1 Aiheen esittely

Digitaalinen sisältö esiintyy pääasiassa kahdessa eri muodossa. Puhutaan tiedon tallennus- ja esitysmuodoista. Tallennusmuodolla tarkoitetaan vaihetta, jossa tieto on tallennettuna johonkin fyysiseen paikkaan, kuten tiedostoon. Selainohjelmat muodostavat sen tiedon pohjalta esitysmuodon, joka voi ohjelmasta riippuen olla aina hieman erilainen. Joskus on tarvetta muuntaa sisältö ensin tallennusmuodosta lähetysmuotoon, joka on tilapäinen vaihe välitettäessä tietoa selainohjelmalle. Muunnokset mahdollistavat monipuolisen tavan hallita ja kohdentaa oikeaa tietoa selainohjelman käytettäväksi. (Samela 2002, 24 - 25.)

Tyypillisin selainohjelma on Internetselain. Kuten moni on huomannut, erimerkkiset selaimet esittävät saman tiedon joskus hieman eri tavalla. Yleisesti puhutaan sisällön tulkattavuudesta, mikä on kaikkia kuvauskieliä ja digitaalista sisältöä koskeva ominaisuus. Esimerkiksi sama valokuva voidaan nähdä hieman erilaisena eri näytöillä, riippuen näytön erilaisista asetuksista. Tulkattavuus tuottaa yleensä ongelmia ja se tulisi ottaa huomioon jo suunnitteluvaiheessa. (Samela 2002, 24 - 25.)

Esitysmuotojen sijaan puhutaan monesti myös kanavista, joilla tarkoitetaan tapaa jakaa ja esittää digitaalista sisältöä. Yleinen esimerkki kanavasta on Internetin webpalvelu, jossa sisältöä tarkastellaan Internetselaimella. Muita kanavia voi olla esimerkiksi jokin mobiililaite, kuten matkapuhelin tai PDA-laite. Myös pienoismikroja ja digitaalista televisiota voidaan pitää kanavina. (Samela 2002, 26 - 27.)

Jokaisella kanavalla on tietyt tiedonsiirtoa ja esitettävyyttä koskevat rajoituksensa, ja ne tulee ottaa huomioon esitystä suunniteltaessa. Mobiililaitteita koskeva suurin rajoittava tekijä on käytettävissä olevan näytön koko, joka soveltuu vain pienien tietomäärien esittämiseen (Hiltunen, 2002, 165). Tiedon julkaiseminen eri kanaviin edellyttää, että tallennusmuodossa olevat tiedot muunnetaan tai täydennetään niin, että ne sen jälkeen ovat esitettävissä tarkoitetussa kanavassa. Alkuperäinen tallennusmuoto on sama, mutta muunnoksien kautta esitystapa voi olla hyvinkin erilainen. Periaatteessa olisi mahdollista laatia sisältöä, jota voidaan esittää täysin samanlaatuisena eri kanavissa, mutta käytännön rajoitteet rajoittavat teorian koskemaan vain lyhyitä tekstipätkiä. Samelan mukaan tulevaisuudessa kanavariippumaton sisältö tulee olemaan tekniikan kehittyessä yhä suuremmassa roolissa. (Samela 2002, 26 - 27.)

3.3.2 Kanavaan sovittaminen

Kanavaan sovittaminen on tarpeen tilanteissa, joissa alkuperäisen sisällön esittäminen halutulla laitteella ei ole järkevää tai teknisesti mahdollista. Tallennusmuotoa on muokattava tavalla tai toisella, jotta sisältö sopisi paremmin suunnitellulle kohdelaitteelle tai -ympäristölle. Sovittaminen voi olla tekninen prosessi, jossa esimerkiksi XML-tiedostoon liitetyn tyylitiedoston avulla sisällön ulkoasua muokataan sopivammaksi. Toisaalta sovittaminen voi olla toiminnallista, joka tarkoittaa esimerkiksi sitä, että lomake jaetaan useaan osaan lähetettäessä se mobiililaitteelle, jolla suurikokoisen lomakkeen katsominen kerralla olisi hyvin hankalaa. (Samela 2002, 84-85.)

Kanavointi ei aina koske sisällön sovittamista pelkästään tietylle laitteelle.
Internetissä on jo pitkään tutkittu ohjelmallisesti käyttäjän selainversio ja räätälöity sivua tukemaan paremmin selaimen ominaisuuksia. Samelan mukaan tämä on kanavoinnin vanhin muoto. (2002, 85.)

3.3.3 Kanavoinnin suunnittelu

Yleiskäyttöinen esitysmuoto

Mitään täysin universaalia tallennusmuotoa ei ole olemassa. Sisältöä voidaan kuitenkin pitää yleiskäyttöisenä, mikäli siitä on mahdollista helposti muodostaa eri kanaviin soveltuvat lähetysmuodot. Parhaimmillaan tilanne olisi, mikäli sisältö olisi tallennettu muotoon, joka voitaisiin suoraa lähettää eri kanaviin. Näin ei kuitenkaan yleensä ole, ja tietoon joudutaan tekemään muutoksia. (Samela 2002, 87.)

XML on tällä hetkellä sivujen ja tekstidokumenttien yleiskäyttöisin tallennusmuoto, ja sen pohjalta on suhteellisen helppo muodostaa personoitua tiettyyn kanaviin suunnattua sisältöä. XML-tiedostojen suora selailu on erittäin hankalaa, ja yleisesti tarvitaankin aina jokin muunnos- ja tyylitiedosto, joka kertoo miten XML-dokumentti näytetään. XML-tiedostojen muotoilu ja esittäminen voidaan hoitaa myös jonkin erillisen ohjelman, esimerkiksi Flashin avulla. XML:n hyödyntämisestä lisää luvussa 4.5 Flash ja XML. (Samela 2002, 87.)

Aineiston vakiointi

Vakioinnilla tarkoitetaan sisällön tallennusmuotojen rajoittamista sellaisiin, jotka täyttävät parhaiten tarkoituksensa. Tallennusmuodon tulisi olla mahdollisimman yhteensopiva muiden järjestelmien kanssa ja sen tulisi säilyä käyttökelpoisena ohjelmavaihdoksesta huolimatta. Vakiointi lisää aineiston monikäyttöisyyttä, mikä on tärkeää, koska yhä useammin sama aineisto halutaan näyttää eri kohderyhmille eri kanavien kautta. Samela ennakoi tiedon monikanavaisuuden olevan tärkeä kriteeri tulevaisuudessa, sillä tietoa halutaan selailla erilaisilla välineillä. Käyttäjät haluavat

hyödyntää tietokonepäätteiden lisäksi myös esimerkiksi omia matkapuhelimiaan tiedon hakemiseen ja selailuun.Vakiointi on mahdollista toteuttaa erottamalla sisällön rakenne ja ulkoasu eri kokonaisuuksiin. Oleellinen osa vakiointia on sisällön kuvailutietojen määritteleminen. Kuvailutieto on tietoa tiedosta, sitä käytetään kuvaamaan lyhyesti millaisesta tiedosta on kyse. XML-standardi on keskeisessä asemassa tiedon vakioinnissa ja kanavoinnin toteuttamisessa. (2002, 20 - 21.)

3.3.4 XML-tekniikan esittely

XML on metakieli

Webin yhteisiä ja yhteensopivia pelisääntöjä kehittävä World Wide Web Consortium (W3C) määrittelee, että XML on yksinkertainen ja erittäin joustava tekstiformaatti, joka on kehitetty suoraan SGML-pohjalta. Vaikka XML suunniteltiin alun perin vastaamaan laajamittaisen elektronisen julkaisemisen haasteisiin, se on osoittanut toimivansa äärimmäisen hyvin tiedon levittämisen muotona kaikkialla. (Extensible Markup Language, 2006)

XML:n kehitys alkoi vuonna 1996 ja se on ollut W3C:n virallinen suositus vuodesta 1998. Iän perusteella voisi luulla, että kyse ei ole vielä kovinkaan yleisestä tekniikasta, mutta XML:n juuret ovat peräisin jo 80-luvun alussa kehitetyssä SGML-kielessä. XML:n suunnittelijat omaksuivat aiemman tekniikan parhaat puolet ja toteuttivat sen pohjalta huomattavasti säännöllisemmän ja helppokäyttöisemmän kielen. XML tunnetaan nykyisin maailmanlaajuisena informaation välittämisen standardina. (Nykänen 2003.)

XML tulee sanoista Extensible Markup Language. Se on metakieli rakenteisen tiedon esittämiseen. XML on joukko sääntöjä, joita käytetään rakenteisten tekstiformaattien suunnitteluun. XML välttää monien standardien yleisimmät ongelmat, se on helposti laajennettavissa, järjestelmäriippumaton, tukee kansainvälistämistä ja on vieläpä täysin ilmainen. XML-kieltä ei pidä kuitenkaan sekoittaa ohjelmointikieliin, sillä yksinään XML toimii pelkästään tiedon varastona. XML-dokumenttien sisällön käsittelyyn ja katsomiseen tarvitaan aina jokin erillinen ohjelma. (Nykänen 2003.)

Koska XML on metakieli, sen avulla on mahdollista luoda omia kuvailukieliä. XML hyödyntää itse määriteltyjä elementtejä, joilla tietoa ja sen rakennetta kuvataan. Toisin kuin esimerkiksi HTML:ssä, elementtien nimille ei ole asetettu mitään rajaa, vaan niitä voidaan luoda täysin omien tarpeiden mukaan. Luotuja XML-dokumentteja voidaan käyttää tietokantojen tapaan tiedon varastona. Sen sijaan, että tietoa pilkottaisiin taulukoihin ja kenttiin, käytetään elementeistä rakentuvaa runkoa tiedon kuvaamiseen. (Nykänen 2003.)

XML:n etuja

Helppous on yksi XML-dokumenttien tarjoama etu. Noudattamalla muutamia yksinkertaisia sääntöjä on varsin yksinkertaista luoda XML-dokumentteja käyttämällä esimerkiksi tekstieditoreita. Dokumenttien lukeminen on myös helppoa, elementtien nimistä saa yleensä selkeän käsityksen tiedosta ja sen rakenteesta. Tietokonejärjestelmien on myös helppo lukea sisältöä, kunhan niille on kerrottu ensin tarkasti elementtien merkitykset. (Jacobson 2006, 47.)

Joustavuutta pidetään XML:n oleellisempana vahvuutena. Tiettyjen sääntöjen puitteissa on mahdollista luoda vapaasti täysin omia elementtejä, attribuutteja ja tiedon rakenteita ja sovittaa ne omiin tarkoituksiin parhaiten sopiviksi. Tietoa voidaan jakaa eri ihmisten kesken käyttämällä XML-dokumentin rinnalla DTD-tiedostoja tai skeemoja, jotka määrittelevät dokumentin ”kieliopin” tai säännöt. (Jacobson 2006, 47.)

Koska XML erottaa sisällön sen ulkoasusta, XML-pohjaista tietoa voidaan hyödyntää useissa eri järjestelmistä. XML-dokumentin sisältö on esitettävissä helposti esimerkiksi web-selaimella. Toisia teknologioita tai ohjelmistoja hyödyntämällä XML-dokumenttiin voidaan tehdä muutoksia ja tuottaa sisältöä jaettavaksi useisiin eri kanaviin. Saman XML-dokumentin sisältöä voitaisiin tarkastella myös tulostettuna PDF-dokumenttina tai Flash-esityksenä. Sisältö voitaisiin julkaista myös kännykälle. (Jacobson 2006, 48-49.)

XML on täysin riippumaton järjestelmästä tai käytetystä laitteesta. Samaa tietoa voidaan yhtä hyvin hyödyntää PC, Mac tai mobiiliympäristöissä. XML on myös täysin ilmainen ratkaisu, eikä sitä omista mikään yritys tai kaupallinen organisaatio. Kuka tahansa saa käyttää XML-kieltä, eikä sen tuottamiseenkaan tarvita mitään maksullisia ohjelmia. (Jacobson 2006, 48-49.)

3.3.5 Kanavoinnin toteuttamisen periaatteet

Kanavointi on tapana toteuttaa erottamalla varsinainen sisältö, joka yritetään pitää mahdollisimman riippumattomana kanavista, sekä kanavittain vaihtelevat tyyli- ja muotoilumäärittelyt. Tällä tavalla turvataan, että sisältö on yhteensopiva eri ohjelmien ja laitteiden kanssa. (Samela 2002, 27.)

Kuten aiemmassa luvussa esiteltiin, XML soveltuu erittäin hyvin sisällön ja ulkoasun erottamiseen toisistaan. XML-tiedostoista on eri tekniikoita hyödyntäen helppo luoda personoitua tietoa haluttuihin kanaviin. XML-tiedostojen esitysmuotoa voidaan muokata muun muassa XSL-kielen avulla, hieman samaan tapaan kuin HTML-dokumentteja CSS:n avulla. Ulkoasun käsitteleminen voidaan toki hoitaa myös ohjelmallisesti käyttäen esimerkiksi Perliä, Javaa tai PHP:tä. Kuten luvussa

4.5 Flash ja XML tullaan huomaamaan, myös Flashia voidaan käyttää XML-sisällön muotoiluun ja esittämiseen. (Zaharia 2005a.)

3.3.6 XSL:n käyttö kanavoinnin toteuttamisessa

XSL-kieli XML-rakenteen muotoilussa ja muokkaamisessa

XML-tiedostojen katselu normaaleilla selainohjelmilla on hankalaa, koska ohjelmat eivät voi tietää, miten itse luotuja elementtejä tulisi käsitellä. XSL-kieli suunniteltiin, jotta XML-tiedostoja voitaisiin esittää halutussa muodossa. Kieli koostuu oikeastaan kahdesta eri kielestä: XSLT on XML-dokumenttien muotoilut mahdollistava kieli, XPath mahdollistaa liikkumisen XML-dokumentissa. (Zaharia 2005a.)

XSLT:n avulla XML-dokumentti voidaan muuntaa toiseksi dokumentiksi, esimerkiksi XHTML-muotoon tai ihan tavalliseksi tekstiksi. Mikäli XML-dokumentti halutaan esittää esimerkiksi tietokoneen web-selaimella, XSLT-kielen avulla voidaan muuntaa XML-elementit selaimen ymmärtämiksi XHTML-elementeiksi. XML-dokumenttipuulle on mahdollista tehdä rakenteellisempiakin muutoksia. XSL-mahdollistaa elementtien ja attribuuttien lisäämisen ja poistamisen, elementtien uudelleenjärjestelyn, ja sitä voidaan hyödyntää myös elementtien piilottamiseen, etsimiseen, valitsemiseen ja näyttämiseen. (Zaharia 2005d.)

XSL:n avulla voidaan hallita ja muokata XML-tiedostojen esitystapaa ja näytettävän tiedon osuutta. XSL soveltuu erinomaisesti tilanteisiin, joissa halutaan luoda XML-tiedostojen pohjalta personoitua sisältöä, ja levittää sitä lukuisille eri käyttäjille ja kohdelaitteille. (Zaharia 2005d.)

XSLT-kieli etsii tiedostoista yhtäläisyyksiä, se etsii tiettyjä hakuehtoja täyttäviä elementtejä ja tekee niille tiettyjä ohjeen mukaisia muunnoksia. Siltä kuitenkin puuttuu merkittävä osa useimpien ohjelmointikielien tarjoamista toiminnallisuuksista. XSL ei esimerkiksi osaa vaihtaa muuttujien arvoja ajon aikana. Monipuolisempaa toiminnallisuutta haettaessa onkin Zaharian mukaan viisaampaa turvautua oikeisiin web-ohjelmointikieliin. (Zaharia 2005d.)

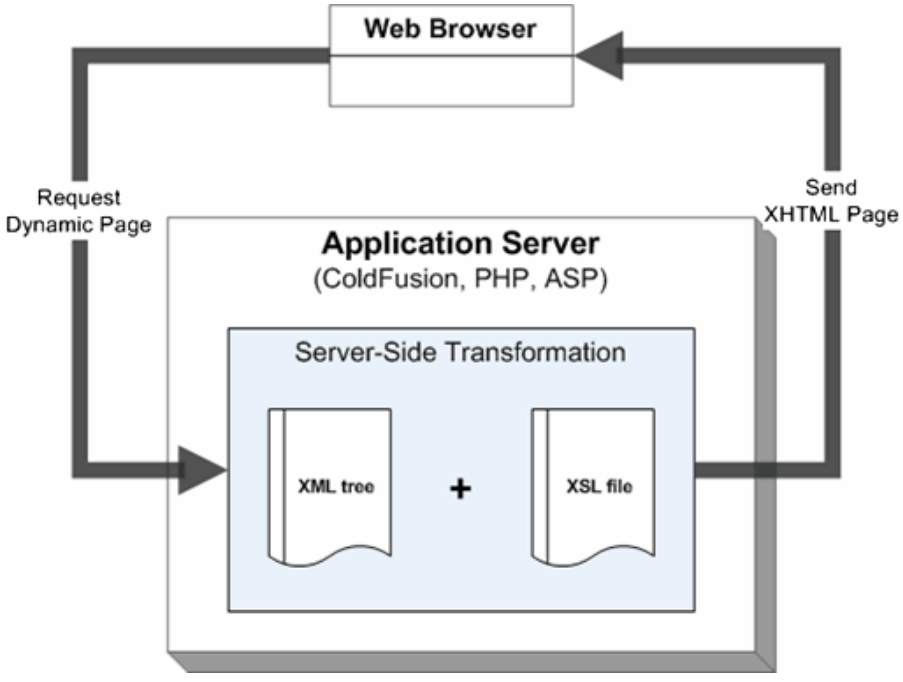
On syytä myös huomata, ettei XSL yksinään mahdollista alkuperäisten XML-dokumenttien muokkaamista. Sitä voidaan pitää tulkkina, joka osaa tehdä halutut käännökset ja välittää ne eteenpäin esimerkiksi selaimen näytettäväksi tai jonkin muun ohjelman käsiteltäväksi. (Zaharia 2005d.)

XSL:n hyödyntäminen kanavoinnissa

Tyypillinen esimerkki XSL:n käytöstä ja kanavoinnista, on tilanne, jossa XML-sivu halutaan esittää tietokoneen web-selaimella. Varsinainen muunnosprosessi voidaan hoitaa yleensäkin kahdella eri tavalla, joko suoraan selailuun käytettävällä ohjelmalla, tai erillisellä palvelimella sijaitsevaa ohjelmaa hyödyntäen. Toisaalta muunnoksien tekeminen voidaan toteuttaa myös paikallisesti käyttäen apuna esimerkiksi Microsoftin Wordiä tai Exceliä (katso luku 5.5). (Zaharia 2005a.)

Palvelin- ja asiakaspuolen tekniikoiden eroja

Sekä palvelin- että asiakaspuolen tekniikoissa on omat vahvuutensa ja heikkoutensa. Palvelinpuolen tekniikka mahdollistaa muunnosten tekemisen kokonaan palvelimella. Asiakkaan koneen ei tarvitse osata käsitellä muunnoksia, koska tieto on jo valmiiksi HTML-muodossa, ja siten luettavissa vanhemmillakin selaimilla. Etuina on myös, että itse XML-dokumentin ei tarvitse sijaita palvelimella, vaan se voidaan ladata mistä vain tai hyödyntää esimerkiksi XML-muotoista RSS-uutisvirtaa. Heikkoutena tekniikassa on, että palvelimelta täytyy löytyä tuki XML:lle ja XSL:lle. Läheskään kaikilta palveluntarjoajilta ominaisuutta ei löydy valmiina, mutta yleisimmät palvelinohjelmointikielet, kuten ColdFusion, ASP, .Net ja PHP, tukevat näitä tekniikoita. (Zaharia 2005b.)



KUVA 1. XSL-muunnosprosessi hyödyntäen palvelinpuolen tekniikkaa (Zaharia 2005b)

Kuvassa 1 on kuvattu yksinkertaistettuna prosessi, jossa web-selaimeen haetaan XML-tiedoston pohjalta luotu XHTML-sivu. Selain lähettää kyselyn dynaamiselle palvelinpuolen sivulle, joka lataa XML-dokumentin ja muodostaa siitä XML-dokumenttipuun. XSL-tiedoston määrityksiä hyväksi käyttäen muokataan dokumenttipuuta ja muodostetaan siitä selaimelle lähetettävä XHTML-sivu. (Zaharia 2005b.)

Asiakaspuolen muunnokset tehdään suoraan itse sisällön katseluun tarkoitetulla selaimella, eikä mitään erillisiä järjestelmiä tarvita. Heikkoutena on, että läheskään kaikki selainohjelmat eivät tunnista XML/XSL-muotoja. Uusimmat web-selaimet kuten Internet Explorer 6, ja Firefox 1.0.2 ja Opera 8 tukevat näitä kieliä, mutta läheskään kaikki vanhemmat versiot, esimerkiksi IE 5, eivät pysty käsittelemään XSL-muunnoksia. Asiakaspuolen muunnokset rajoittuvat koskemaan vain paikalliselle palvelimelle sijoitettuja XML-tiedostoja. (Zaharia 2005b.)

XPath osoittaa sijainnin XML-dokumenttipuussa

Macromedian mukaan XSL käyttää XPath-määreitä osoittamaan tiettyä sijaintia XML-dokumenttipuussa. Tekniikka käyttää samanlaista yksinkertaista polkurakennetta kuin käyttöjärjestelmät käyttävät tiedostojen sijainnin määrittelyssä (esimerkiksi C:\Program Files\Macromedia\.). Yksinkertainen XPath-määreitä käyttävä XSL-dokumentti voisi näyttää seuraavalta: (Zaharia 2005c.)

```
<xsl:template match="/">
  <table border="1">
    <tr>
     <th>Name</th>
     <th>Job</th>
     <th>Salary</th>
    </tr>
    <tr>
     <td><xsl:value-of se-
lect="company/department/employee/name"/></td>
     <td><xsl:value-of se-
lect="company/department/employee/job"/></td>
     <td><xsl:value-of se-
lect="company/department/employee/sal"/></td>
    </tr>
  </table>
</xsl:template>
```

Esimerkin XSL-tiedostossa käytetään tavallisia HTML-elementtejä muotoilemaan XML-dokumenttia tarkoituksena luoda HTML-dokumentti, joka tulostaa taulukkomuodossa työntekijän nimen, työn ja palkan. XPath-määre ”company/deparment/employee/name” viittaa XML-dokumentin ensimmäiseen name-elementtiin. Alla on nähtävissä alkuperäisen XML-tiedoston rakenne. Kuten esimerkistä nähdään, XPathin käyttö XSL-dokumenteissa on varsin yksinkertaista. (Zaharia 2005c.)

```
<department>
  <employee>
      <name>John Doe</name>
   <job>Software Analyst</job>
      <salary>2000</salary>
  </employee>
  <employee>
</department>
```

4 DYNAMIIKKA

4.1 Johdanto

Aiemmissa luvuissa kuvailtiin asioita, joita liittyy profiloitujen multimediaesitysten tekemiseen. Seuraavissa luvuissa käsitellään niitä tekniikoita, jotka mahdollistavat edellä esitettyjen ideoiden toteuttamista Flashissa. Tutkitaan, miten on mahdollista tuoda muuttuvaa tietoa mahdollisimman joustavasti ja yksinkertaisesti Flashin käytettäväksi.

4.2 Dynaaminen tieto

4.2.1 Määrittelyt

Tämän työn yhteydessä dynamiikalla viitataan tapaan rakentaa, jäsennellä ja luoda multimediaesityksiä niin, että niistä tulisi mahdollisimman joustavia ja helposti muokattavia. Toisaalta dynaamisuus liittyy myös esityksien ulkoasuun, Galvatek Oy:lle luotuihin esityksiin pyrittiin luomaan liikettä ja voimakkuutta, jotta katsoja jaksaisi kiinnostua seuraamaan esitystä.

Työssä puhutaan paljon dynaamisuudesta ja dynaamisesta tiedosta. Tietotekniikan liiton ATK-sanakirja määrittelee dynaamisen seuraavasti: ”ajasta tai tilanteesta riippuvainen tai jatkuvasti tai usein muuttuva” (ATK-sanakirja 2005). Toisaalta dynaamisuus viittaa sovelluskäytössä toimintoon, joka tapahtuu vasta hetkellä jona sitä tarvitaan, eikä ennalta määrätysti etukäteen. Dynaamisella tiedolla käsitetään esimerkiksi Internet-ympäristössä sellaista sisältöä, joka muuttuu jollain tavalla ulkopuolisten muuttujien tai käyttäjän syötteiden mukaan, ja haetaan selaimeen samalla kun käyttäjä katsoo sivuja. Dynaaminen tieto voi yksinkertaisimmillaan olla esimerkiksi Internet-sivuille lisättävä sen hetkinen päivämäärä. (What is dynamic? 2005.)

Dynaamisella sisällöllä (dynamic content) tarkoitetaan sisältöä, joka voi muuttua itsenäisesti ja on tallennettu eri paikkaan kuin itse esitys (Jacobson 2006, 12).

4.2.2 Dynaaminen tieto webissä

Vaikka webissä ollaankin siirtymässä yhä enemmän kohti dynaamisia www-sivuja, suurin osa sivuista on todennäköisesti yhä tavallisia staattisia sivuja. Staattisilla sivuilla tarkoitetaan sellaisia sivuja, joissa käyttäjä ei voi vaikuttaa sivujen sisältöön ja itse sisältö on suoraan HTML-tiedostoissa, eikä sitä haeta mistään ulkopuolisesta lähteestä. Staattisia sivuja käytetään yhä, koska ne ovat selvästi nopeampia kuin dynaamiset sivut (Samela 2002, 52). Tällaisten sivujen luominen on yksinkertaista, mutta voi ajan mittaan tuottaa suuria hankaluuksia, mikäli sivuja on paljon tai sivuja joudutaan päivittämään nopeasti (Shaffer 2005).

Staattisten sivujen vastapainona on olemassa älykkäämpiä ja monipuolisempia dynaamisia sivustoja. Niillä tarkoitetaan sivustoja, joissa sisältö haetaan ulkoisista tiedostoista tai tietokannoista ja kenties muokataan jollain tavalla käyttäjän toiveiden mukaan. Sivuilla näytettävä dynaaminen tieto voi yksinkertaisimmillaan olla esimerkiksi kävijälaskuri tai päivämäärä. Dynaamisia sivustoja luodaan käyttäen apuna esimerkiksi php- tai java-ohjelmointikieltä. (Shaffer 2005.)

Dynaamiset sivut mahdollistavat personoitujen sivujen toteuttamisen, jossa käyttäjä voi mahdollisesti itse ohjata tarkemmin millaista sisältöä haluaa nähdä. Monet sivut hyödyntävät dynaamista sisältöä käyttäjän huomaamattakin, esimerkiksi näytettävät mainokset saattavat vaihdella käyttäjän selailutottumusten tai aiemmin tehtyjen ostosten tai hakutulosten mukaan. (Shaffer 2005.)

4.3 Dynaaminen Flash ja FlashLite

Flash oli alkujaan yksinkertainen animaatiotyökalu, mutta on vuosien mittaan kehittynyt monipuoliseksi korkean tason multimediantuotanto-ohjelmaksi ja saavuttanut suosion ammattigraafikoiden ja web-kehittäjien keskuudessa. Flashia käytetään nykyään hyvin laajasti useissa eri käyttötarkoituksissa: yksinkertaisissa animaatioissa, kokonaisten web-sivujen toteuttamisessa, kännyköiden käyttöliittymissä ja e-opetuksessa. Yksinkertaiset Internetissä olevat animaatiot ovat ainakin monille varmasti tuttuja. (Jacobson 2006, 2.)

Ihmiset harvemmin kuitenkaan osaavat yhdistää Flashia Rich media -sovelluksiin. Flash mahdollistaa joustavien ja tyylikkäiden käyttöliittyminen toteuttamisen web-sivuille tai paikallisella koneella suoritettaville ohjelmille. Uusimmissa Flashin julkaisuissa on mukaan tullut erilaisia valmiita komponentteja, jotka ovat mahdollistaneet helpon ja nopean tavan luoda interaktiivisia esityksiä. (Jacobson 2006, 2.)

Flash tarjoaa mahdollisuuden tuoda dynaamista sisältöä esitykseen lukuisallakin eri tavalla; tietokannoista, tekstitiedostoista ja XML-dokumenteista. Flash-esitys toimii tavallaan ulkoasupohjana, jonka tyhjiin kohtiin ladataan tietoa ulkopuolisista lähteistä. Esityksen ja ulkoasun erottaminen toisistaan on hienoa sen takia, että esityksen sisältöä voi päivittää käyttämättä Flashia. Sisältöä pystyy muokkaamaan kaikki, joilla on pääsy tietolähteeseen, esimerkiksi tekstitiedostoon. (Jacobson 2006, 12.)

Mobiililaitteita varten Macromedia on kehittänyt oman Flash Lite Playerin, joka mahdollistaa Flashilla toteutettujen sovellusten esittämisen esimerkiksi kännykällä. Tätä kirjoittaessa uusin versio oli 2.0, jonka merkittävimmät uudistukset koskivat juuri ulkoisen tiedon käsittelyä. Flash Lite 2.0 mahdollistaa XML-muotoisen tiedon käsittelyn, kuvien ja äänen dynaamisen käytön ja tuen ActionScript 2.0. Uusimman version myötä myös mobiililaitteissa on mahdollista nähdä yhä dynaamisempi Flash-esityksiä. (Duran 2005.)

Dynaamisen sisällön käyttö tarjoaa sovelluskehittäjälle keinot toteuttaa profiloituja esityksiä, joissa saman ulkoasun yhteydessä näytetään hieman eri sisältöä kohdeprofiilin määrittelyjen mukaisesti. Flashin dynaamisia ominaisuuksia tarvitaan myös tilanteissa, joissa esityksiä halutaan jakaa useisiin eri kanaviin ja laiteprofiileihin. Dynaamisen sisällön käyttö helpottaa myös merkittävästi ylläpitoa ja sisällön päivittämistä.

4.4 Muuttuja-arvoparit

4.4.1 Perusteet

Yksi suoraviivaisimmista keinoista tuoda tietoa dynaamisesti Flashiin on sijoittaa tieto muuttuja-arvopareihin. Flash pystyy käsittelemään pitkääkin tekstimuotoista riviä, joka sisältää lukuisia eri muuttuja-arvopareja. Henkilötiedot voitaisiin esittää esimerkiksi kolmen muuttujan avulla: nimi, sukunimi ja ikä.

```
nimi=Jorge&sukunimi=Solis&ika=30
```

Kullekin muuttujalle on annettu oma arvo (Jorge, Solis, 30). Kokonaisuus on kirjoitettu yhteen, mutta parit on erotettu toisistaan aina &-merkillä. Muuttujien määrää ei ole rajattu, vaan kerralla voidaan välittää satojakin muuttuja-arvopareja. Esimerkin mukainen koodirivi voidaan kirjoittaa itse ja sijoittaa se esimerkiksi tekstitiedostoon, josta se haetan Flashiin. Monesti muuttuja-arvoparit tulostaa palvelimella oleva ohjelma, joka hakee tiedot tietokannasta ja välittää ne eteenpäin Flashiin. (Solis 2006.)

4.4.2 Säännöt ja rajoitteet

Flash asettaa tiettyjä rajoitteita syötettävän tekstin muotoon. On olemassa tiettyjä erikoismerkkejä, joiden käyttö voi aiheuttaa epätoivottuja tuloksia, ja joita ei tulisi käyttää sellaisenaan. Tähän ongelmaan tuo ratkaisun URL encoding -tekniikka, jossa erikoismerkit koodataan Flashin kannalta yksiselitteiseen muotoon. Seuraava muuttuja-arvopari luettaisiin esimerkiksi ”kulut=30”, plus-merkki jäisi kokonaan pois:

```
kulut=+30
```

Käyttämällä URL encoding -tekniikkaa Flash lukisi rivin oikein muodossa ”kulut=+30”:

```
kulut=%2b30
```

(URL Encoding: Reading special characters from a text file 2005.)

Character	Code	Character	Code
Space	%20	*	%2A
!	%21	+	%2B
"	%22	,	%2C
&	%26	-	%2D
$	%24	.	%2E
%	%25	/	%2F

KUVA 2. Lyhennetty lista yleisimmistä erikoismerkeistä ja niitä vastaavista merkintätavoista (URL Encoding: Reading special characters from a text file 2005)

4.4.3 FlashVars

On olemassa monia keinoja siihen, miten muuttuja-arvopareja voidaan lähettää Flashiin. Yksi helppo tapa tehdä se on hyödyntää FlashVars-parametriä. FlashVars on object- ja embed-elementtien erikoismääre HTML-kielessä. Näitä elementtejä käytetään linkittämään Flashin SWF-tiedosto HTML-dokumenttiin. (David 2005c.)

Käyttääksesi FlashVars-parametriä on ensin luotava HTML-dokumentti, johon liitetään Flash-sisältöä käyttäen object- ja embed-elementtejä (object-elementti toimii Internet Explorerin kanssa, embed-elementtiä tulee käyttää muiden selainten kanssa). FlashVars-parametrillä välitetään muuttuja-arvopareja Flash-tiedostolle seuraavanlaisesti:

```
<body bgcolor="#ffffff">
<object id="helloWorld " align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="helloWorld.swf" />
```

```
<PARAM NAME=FlashVars VALUE="var1=Hello%20World">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="helloWorld.swf" quality="high" bgcolor="#ffffff"
width="544" height="249" name=" helloWorld " flash-
vars="var1=Hello%20World" align="middle" allowScriptAc-
cess="sameDomain" type="application/x-shockwave-flash" plugin-
spage="http://www.macromedia.com/go/getflashplayer" />
</object></body>
```
(David 2005c.)

Esimerkissä Flashiin välitetään yksi muuttuja var1 arvolla ”Hello%20World”. Jotta arvo saadaan näkymään Flashissa tekstinä, on luotava Flash-tiedosto (helloWorld.swf), jonne lisätään dynaaminen tekstikenttä nimeltään line1_txt. Avuksi tarvitaan vielä seuraava lyhyt rivi ActionScriptiä, jolla lähetetty muuttuja sijoitetaan tekstikenttään:

```
line1_txt.text=var1;
```
(David 2005c.)

Oheisessa esimerkissä näytettiin käytännön tasolla, miten helppoa tiedon tuonti Flashiin voi olla. FlashVarsin käyttö rajoittuu melko yksinkertaisiin sovelluksiin, eikä sovi esimerkiksi hallinnoimaan isoja tekstimääriä. Tätä tekniikkaa hyödyntäen voitaisiin kuitenkin rakentaa esimerkiksi sovellus, joka tunnistaa käyttäjän kirjautumisen ja lähettää kirjautumistiedot dynaamisesti Flashiin. (David 2005c.)

4.4.4 LoadVars

Edellä käytiin läpi, miten muuttujia ja arvoja voidaan viedä Flashiin HTML-dokumentin kautta. Tieto voidaan varastoida myös ulkopuoliseen tekstitiedostoon ja hakea tieto sitä kautta Flashiin. Tekstitiedostossa oleva tieto pitää kuitenkin olla samalla tavalla tallennettu muuttuja-arvopareiksi, jotta Flash osaa käsitellä sitä oikein. (Solis 2006.)

Version kuusi myötä Flashiin esiteltiin uusi LoadVars-objekti, joka oli aiempaan LoadVariables- ja LoadVariablesNum-toimintoja kehittyneempi ja joustavampi tapa käsitellä ja välittää muuttujia. Seuraavaksi käydään läpi ainoastaan uudempaa LoadVars-tekniikkaa, sillä se on Moran mukaan yleensä järkevämpi vaihtoehto. (Mora 2005.)

LoadVars-objektia hyödynnetään ladattaessa tietoa ulkopuolisesta tekstitiedostosta. Mikäli tekstitiedosto olisi nimeltään tiedot.txt ja siellä olisi rivi ”nimi=Jorge”, se saataisiin ladattua Flashiin seuraavanlaisella koodilla:

```
myData = new LoadVars()
    myData.load("tiedot.txt")
    myData.onLoad = function(succes){
```

```
    if(succes){
        etunimi.text = this.nimi
    } else trace ("Virhe ladattaessa tiedostoa!")
}
```

Esimerkkikoodissa luodaan aluksi LoadVars-olio, jonka jälkeen käytetään load-metodia lataamaan tiedoston sisältö ja, lopuksi tutkitaan onLoadin avulla, milloin sisältö on ladattu, ja sijoitetaan se haluttuun tekstikenttään. Mikäli lataus ei onnistunut, siitä tulostetaan virheilmoitus. Tekstikenttään nimeltä etunimi latautuu teksti Jorge. (Solis 2006.)

Samalla periaatteella voitaisiin tehdä Soliksen esimerkkiä noudatteleva hieman monipuolisempi sivu, jossa näkyy henkilön nimi, kuvaus ja valokuva. Silloin Flashiin tehtäisiin kaksi tyhjää dynaamista tekstikenttää, jotka nimettäisiin etunimi ja txt_kuvaus, sekä tyhjä MovieClip nimeltään kuva_paikka. Tekstitiedosto tiedot.txt voisi näyttää seuraavalta:

```
nimi=Jorge&kuvaus=Mukava ihminen&valokuva=omakuva.jpg
```

Flash koodi olisi muuten sama kuin aiemmin, mutta lisänä rivit:

```
txt kuvaus = this.kuvaus
kuvan_paikka.loadMovie(this.valokuva)
```

(Solis 2006.)

Tekstitiedostosta pystytään siis lataamaan muuttujia ja arvoja ja niiden avulla muodostamaan sisältö Flash-esitykseen. Pelkän tekstisisällön lisäksi on mahdollista ladata ulkopuolisesta lähteestä myös kuvamateriaalia. Flash hakisi tyhjän MovieClipin sisään valokuvan nimeltä omakuva.jpg ja näyttäisi sen ajettaessa esitystä. (Solis 2006)

4.5 Flash ja XML

4.5.1 Pohjatietoa

Flash 5 oli ensimmäinen versio, joka salli ulkoisen tiedon tuonnin Flashiin XML-muodossa. Tämä ominaisuus on säilynyt tärkeänä piirteenä myös myöhemmissä Flashin julkaisuissa. Sisäänrakennetun XML-parserin myötä Flash pystyy lukemaan sisään ja lähettämään ulospäin XML-muotoista tietoa. MX 2004 -versio toi mukanaan ensimmäistä kertaa datakomponentit, jotka automatisoivat ja helpottavat työskentelyä ulkoisen XML-tiedon kanssa. Lisää komponenteista ja komponenttien välisistä datasidoksista luvussa 4.6. (Jacobson 2006, 3.)

Erinomaiset multimedian ja käyttöliittymien tuottamiseen tarkoitetut työkalut tekevät Flashista oivan valinnan, kun halutaan toteuttaa käyttäjä-näkymä ohjelmalle, joka käyttää XML-sisältöä. Sisältöä voidaan hakea lukuisista eri kohteista ja on mahdollista käyttää jopa Microsoftin Office 2003 -paketin ohjelmia hallinnoimaan Flash-esityksen sisältöä. (Jacobson 2006, 3.)

Tietoa voidaan tuoda Flashiin toki aiemmin esitellyilläkin menetelmillä esimerkiksi tekstitiedostosta. Muuttuja-arvoparien muodostamaa tiedostoa on kuitenkin paljon hankalampi käsitellä ja muokata, varsinkin jos tiedoston koko kasvaa suureksi. Tällaisissa tapauksissa XML:n tuoma rakenteellinen tiedon jäsentely tarjoaa huomattavasti paremman tavan jäsennellä tietoa. Tapa kuvata tietoa rakenteellisesti on jotain, mitä kaikkien on helppo ymmärtää. Kun XML:n säännöistä pidetään kiinni, pystyvät tietoa ihmisen lisäksi lukemaan helposti myös lukuisat eri järjestelmät ja ohjelmat, mukaan lukien Flash. (David 2005d.)

4.5.2 XML:n käytöstä Flashissa

Kuten luvussa 3.3.4 esiteltiin, mahdollistaa XML:n omien tägien luomisen halutun tiedon kuvaamiseen. Voitaisiin luoda esimerkiksi XML-dokumentti, joka sisältää säätiedot, kuten lämpötilan, tuulen nopeuden ja pilvisyyden.

```
<saatiedot>
   <lampotila>76</lampotila>
   <tuuli suunta="Kaakko" nopeus="12"/>
   <pilvisyys>Aurinkoinen</pilvisyys>
</saatiedot>
```

XML näyttää hyvin samalta kuin HTML, mutta XML:n toimintatarkoitus on aivan eri. HTML-tiedostot kertovat selaimelle aina myös tavan, jolla teksti tai kuvat näytetään. XML keskittyy pelkästään kuvaamaan itse tietoa, attribuuteilla ja elementeillä voi olla mikä tahansa sovelluskehittäjän antama tarkoitus. (Statler 2002b.)

XML-muotoisen tiedon tulkinta ja näyttäminen jää sen ohjelman varaan, joka käsittelee sitä. Esimerkin mukaiset säätiedot voitaisiin näyttää yhdessä Flash-esityksessä pelkkänä tekstinä, mutta toisessa esityksessä animoituna tuulimittarina ja pilvisyyskarttana. Sisällön ja ulkoasun erottaminen onkin keskeinen teema XML-dokumentteja käsiteltäessä, samaa sisältöä voidaan hyödyntää useissa eri sovelluksissa. (Statler 2002b.)

4.5.3 XML:n sääntöjä ja kielioppia

HTML:ään verrattuna XML on huomattavasti herkempi koodissa oleville virheille. Koska HMTL-dokumentit perustuvat ennalta määriteltyihin tägeihin ja kielioppiin, ne voivat tehdä tiettyjä oletuksia ja korjata mahdollisia virheitä. Internet-selainta ei esimerkiksi haittaa, vaikka se ei löytäisi lopetustägiä kappaleen aloittavalle tägille . Selain osaa myös automaattisesti paikata atribuuteista puuttuvat heittomerkkit. XML-dokumentissä -tägi voisi kuitenkin tarkoittaa käytännössä mitä vain, eikä dokumenttia lukeva ohjelma voi mitenkään ennakoida sen merkitystä ja tulostaisi virheen puuttuvan lopetustägin takia. (Statler 2002b.)

XML-dokumentin on noudatettava tarkasti sille asetettuja kielioppisääntöjä. Sillä täytyy olla yksi ja ainoastaan yksi juurielementti. Ei-tyhjillä elementeillä on aina sekä aloitus- että lopetuselementti. Tyhjät elementit voidaan merkitä erikseen, esimerkki tyhjästä elementistä on HTML:stä tuttu rivinvaihtoelementti
, joka merkittäisiin
. On huomattava, että elementtien nimissä isot ja pienet kirjaimet tulkitaan eri kirjaimiksi. Attribuuttien arvot on muistettava sijoittaa aina lainausmerkkien sisään. (XML 2006.)

Elementtien rakenteessa on huomioitava, että vaikka elementit saavatkin olla sisäkkäisiä, ne eivät saa mennä ristiin toisten elementtien kanssa. Elementit on aina lopetettava samalla tasolla kuin ne aloitetaan. Seuraava esimerkki havainnollistaa asiaa:

```
<b><i>Tämä teksti on sekä lihavoitu, että kursivoitu</b></i>
//elementit ristissä
<b><i>Tämä teksti on sekä lihavoitu, että kursivoitu</i></b>
//elementit oikein sisennettyinä
```

Ylempi olisi täysin hyväksyttävä merkkaustapa HTML-kielessä, mutta ei toimisi XML-dokumentissa. (Refsnes 2006.)

HTML-dokumentit sisältävät yleensä rivivälejä ja sisennyksiä, jotka helpottavat koodin lukemista. Selain osaa automaattisesti jättää huomioimatta nuo tyhjät tilat, mutta esimerkiksi Flash olettaa, että XML-dokumentissa olevilla tyhjillä väleillä on aina merkitys. Mikäli XML-koodia ei halua kirjoittaa ilman selventäviä välejä ja rivivaihtoja, asian voi kiertää käyttämällä esimerkiksi XML-olion ignoreWhite ominaisuutta. (Statler 2002b.)

4.5.4 XML-luokka

XML-luokka on kaikista yleisin tapa työskennellä XML-pohjaisen ulkoisen sisällön kanssa. Luokat ovat perusta kaikille olio-ohjelmointikielille. Luokka määrittelee tavan, jolla olio toimii, siinä määritellään myös kaikki olion ominaisuudet ja metodit. ActionScriptin XML-luokka sisältää kaiken tarvittavan tiedon, joka mahdollistaa työskentelyn XML-olion ja sitä kautta XML-pohjaisen tiedon kanssa. (Jacobson 2006, 125.)

ActionScriptin XML-oliota käytetään lataamaan, jäsentämään, lähettämään, luomaan ja muokkaamaan XML-dokumenttipuuta. Dokumenttipuu kuvaa XML:n rakennetta hierarkisena rakenteena. Ensimmäinen XML-olion käyttöön liittyvä vaihe on luoda siitä uusi instanssi. (Statler 2002b.)

```
myXML = new XML();
```

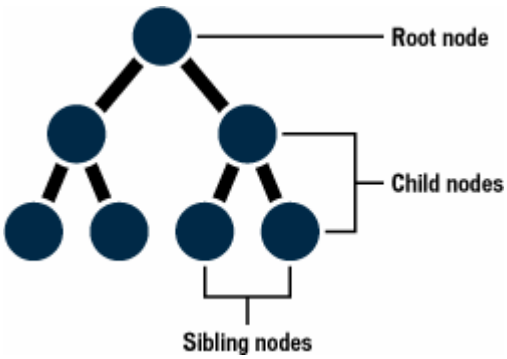
Sen jälkeen on mahdollista ladata Flashiin haluttu XML-dokumentti käyttämällä XML.load-metodia.

```
myXML.load("path_to_xml/data.xml");
```

Kun dokumentti ladataan XML-olioon Flash-esityksen aikana, käynnistyy Flashin Playerin parseri, joka muodostaa ladattavan tiedoston pohjalta XML-dokumenttipuun. Kuva 3 esittää edellä kuvattua prosessia. Kuvan 3 XML-dokumenttipuu muodostuu toisiinsa yhdistetyistä ympyröistä, joita kutsutaan nodeiksi. Nodit muodostavat sukurakenteen, jossa on kantaisiä, jälkeläisiä ja sisaruksia. (Statler 2002a.)

Kuva 3. XML dokumentin lataaminen Flashiin (Statler 2002a)

Rakenteen ylimpänä oleva juurinodi (root node) vastaa XML-dokumentin ensimmäistä juurielementtiä. Polveutuvat lapsinodit (Child nodes) muodostuvat XML-dokumentin juurielementin alla olevista sisäkkäisistä elementeistä. Sisarnodeiksi (sibling node) sanotaan dokumenttipuussa samalla tasolla olevia rinnakkaisia nodeja. (Statler 2002a.)



KUVA 4. XML dokumenttipuun rakenne (Macromedia, (Statler 2002a)

Aikaisemmin esitellyssä säätilaa kuvaavassa XML-dokumentissa <saatiedot> toimisi juurinodina dokumenttipuussa. Muut elementit olisivat lapsinodeja, koska ne on asetettu juurielementin alle. Ne sijaitsevat myös samalla tasolla, joten ne olisivat toisilleen myös sisaria. (Statler 2002a.)

ActionScriptin XML-olio tunnistaa kahdenlaisia XML-nodeja, teksti- ja elementtinodeja. Ensimmäinen pystyy viittaamaan mihin tahansa teksti- tai numeromuotoiseen dataan, joka sijaitsee XML-dokumentissa. Elementtinodeilla viitataan aina tietyn nimiseen XML-dokumentin elementtiin. Nodien avulla voidaan osoittaa tiettyä kohtaa dokumenttipuussa, ja hakea sieltä haluttu tieto käyttäen hyväksi erilaisia XML-luokan tarjoamia metodeja. (Statler 2002a.)

XML-luokan käyttöön liittyy paljon ActionScript-koodia, eikä sitä sen takia käydä tarkemmin läpi tämän teoriaosuuden puitteissa. XML-luokka oli kuitenkin keskeisessä roolissa case-osiota toteutettaessa, jota käydään tarkemmin läpi luvussa kuusi.

4.5.5 XML ja palvelinpuolen sivut

Flash-esityksissä voidaan hyödyntää myös dynaamisesti generoitua XML-sisältöä. Sisällön tuottaminen tapahtuu käyttämällä palvelinpuolen sivua, joka tulostaa XML-muotoista dataa. Dynaaminen data saadaan ladattua Flashiin kertomalla sille ladattavan XML-tiedoston sijaan palvelinpuolen sivun osoite, joka tulostaa XML-muotoista dataa. (Jacobson 2006, 329.)

Tekniikkaa voidaan hyödyntää esimerkiksi tekemällä kysely tietokantaan ja palauttamaan tulos XML-muodossa. Toinen yleinen tekniikka on käyttää palvelinpuolen sivua tutkimaan jokin kansiorakenne ja tulostamaan se ja löydetyt tiedostot XML-muodossa. (Jacobson 2006, 64.)

Palvelinpuolen sivu voidaan toteuttaa esimerkiksi ColdFusion, .NET, JAVA tai PHP-kielillä. Edellä mainitut tekniikat ovat viime vuosina kehittyneet tukemaan yhä laajemmin ja paremmin XML-muotoista dataa. Esimerkiksi uusimmassa PHP:n 5 versiossa olleet muutokset keskittyivät juuri XML-ominaisuuksien parantamiseen. Tukemalla XML-kieltä Flash mahdollistaa työskentelyn useiden eri järjestelmien kautta. (David 2005e.)

Dynaaminen XML-data on jatkuvasti muuttuvaa ja tarjoaa tiettyjä etuja verrattuna staattisiin XML-tiedostoihin, joiden sisältö muuttuu vasta kun käyttäjä itse päivittää sitä jonkin editorin kautta. Osa datan käsittelystä voidaan jättää palvelimen työstettäväksi, joka voi tietyissä tilanteissa parantaa järjestelmän nopeutta. Palvelin pystyy myös suodattamaan valmiiksi XML-tietoa ja karsimaan siitä ylimääräisiä kohtia pois. Tällä on vaikutusta myös tietoturvallisuuteen, koska piilotettavaa tietoa ei tarvitse lähettää ollenkaan asiakaskoneen käsiteltäväksi. (Shiell, James, Addey, Auld, Rafter, Spencer, Kent, Surguy & Gudmundsson 2002.)

4.5.6 XMLSocket-yhteydet

XMLSocket-luokan ominaisuuksien avulla Flash pystyy ylläpitämään jatkuvaa yhteyttä palvelimeen ja välittämään reaaliaikaista dataa. Tämä on erittäin käytännöllistä sovelluksissa, joissa vaaditaan nopeutta ja erittäin lyhyitä vasteaikoja palvelimelta. Tyypillisiä esimerkkejä ovat live-verkkokeskustelusovellukset, jotka tunnetaan paremmin chattisovelluksina. Tavallinen HTTP:hen perustuva chattisovellus lähettää jatkuvasti kyselyitä palvelimelle ja vastaanottaa uusia viestejä käyttäen HTTP-kyselyitä. XMLSocket-yhteyksiin perustuva ratkaisu ylläpitää jatkuvaa yhteyttä palvelimelle, joka mahdollistaa uuden tiedon lähettämisen heti, eikä vasta kyselyn jälkeen. (XMLSocket, 2006.)

XMLSocket-luokan käyttö vaatii, että palvelimelle on asennettu ohjelmisto, joka ymmärtää käytettävää protokollaa. Nämä palvelinohjelmistot voidaan toteuttaa esimerkiksi Java- tai Python-ohjelmointikielillä. Valmiita kaupallisia XMLSocket-yhteydet mahdollistavia sovelluksia on esimerkiksi UNITY 2.0, Shovemedia ja Swocket. (David 2006.)

Macromedian mukaan voi olla haastavaa saada palvelin keskustelemaan Flashin XMLSocket-luokan kanssa. Mikäli ohjelmalta ei välttämättä vaadita reaaliaikaista interaktiivisuutta, kannattaa ulkoinen sisältö hakea Flashiin käyttäen jotain toista, helpompaa tekniikkaa. (XMLSocket, 2006.)

4.5.7 XML ja WWW-sovelluspalvelut (Web services)

Sovelluspalvelut helpottavat tiedon jakamista

Tekniikan sanastokeskus (TSK) määrittelee WWW-sovelluspalvelun olevan ”Verkkopalvelimessa toimiva ohjelma, joka tarjoaa standardisoitujen Internet-yhteyskäytäntöjen avulla palveluja sovellusten käytettäväksi.” Sivu täsmentää vielä määrittelyä: ”WWW-sovelluspalvelun ja asiakassovelluksen välinen yhteys voidaan toteuttaa SOAP-yhteyskäytännön avulla. SOAP hyödyntää XML-kieltä, ja SOAPin ja XML:n avulla toteutettua sovellusta voidaan käyttää millä tahansa alustalla.”

WWW-sovelluspalveluiden perusidea on siinä, että tietoa voitaisiin jakaa verkon yli mahdollisimman helposti eri järjestelmien kesken käyttäen tiettyjä standardeja. Yritykset voivat WWW-sovelluspalveluita käyttäen esimerkiksi tarjota yleisölle rajoitettua ja seulottua tietoa. Yleisölle voidaan välittää julkista dataa, mutta samalla rajoittaa pääsy osaan materiaalista. Käyttäjä tekee pyynnön sovelluspalvelulle, joka palauttaa pyynnön XML-muodossa, joka tosin on enemmänkin tietovirta kuin kiinteä dokumentti. (Jacobson 2006, 367.)

Haettua tietovirtaa voidaan hyödyntää lukuisilla eri tavoilla, sitä voidaan näyttää selaimessa tai se voidaan viedä Flashiin dynaamisena sisältönä. Flashin MX2004-version myötä Flashista tuli yksi harvoista asiakaspuolen ohjelmista (ohjelma asennettu paikalliselle tietokoneelle), joka tuki WWW-sovelluspalveluita käyttäen hyödyksi SOAP-protokollaa. (David 2005e.)

Yrityksen WWW-sovelluspalvelun kuvaamiseen käytetään UDDI-rekisteriä, joka sisältää kuvauksen käytettävissä olevista palveluista ja tietoa, kuinka niihin pääsee käsiksi. WSDL-kieli kuvaa WWW-sovelluspalvelua käyttäen tavallista XML-muotoa. Sillä kuvataan, mitä operaatioita voidaan tehdä, mitä parametrejä täytyy antaa ja mitä tietoa saadaan palautteena sovelluspalvelulta. (Jacobson 2006, 368.)

WWW-sovelluspalvelua voidaan hyödyntää käyttämällä tiedon hakemiseen SOAP-, REST- tai RSS-standardeja. SOAP käyttää standardoituja XML-viestejä tehdessään pyyntöjä ja hakiessaan vastauksia sovelluspalvelulta. REST ei noudata mitään tiettyä standardimuotoa viesteissä. Kyselyt tehdään URL-osoitteen kautta lisäämällä muuttujia osoitteen perään. RSS on tunnettu uutisvirtojen muoto, joka tarjoaa tietoa XML-muotoisen uutisvirran avulla käyttäen hyödyksi aivan tavallisia XML-elementtejä. RSS-elementtien nimet on standardisoitu huomattavasti tarkemmin kuin REST:ssä, mutta siitäkin on olemassa eri versioita. (Jacobson 2006, 367 - 368.)

SOAP

SOAP-viestejä voidaan välittää vaikka sähköpostilla, mutta Flash tukee ainoastaan HTTP-pohjaisia viestejä. Flash pystyy ottamaan yhteyden WSDL-tiedostoon ja automaattisesti selvittämään, mitä toiminnallisuuksia SOAP-pohjainen WWW-sovelluspalvelu tarjoaa. Yhteyden ottamiseen voidaan käyttää Flash Professional -versioiden mukana tulevaa WebServiceConnectoria. Sama voidaan toteuttaa myös ActionScriptillä, käyttäen WebServiceConnector- tai Web Service -luokkia. Haettua tietoa voidaan käyttää täyttämään Flash-esityksen dynaamisia tekstikenttiä tai UI-komponentteja. (Jacobson 2006, 380.)

4.5.8 REST ja RSS

REST WWW-sovelluspalveluita voidaan hyödyntää myös Flashissa. Käytön edellytyksen on, että tuntee palvelun URL-osoitteen, juurikaan muuta lisätietoa ei tarvitse. Uusimpien Flash Playereiden tietoturva-asetusten takia apuna joudutaan yleensä käyttämään palvelinpuolen sivua, joka hakee XML-muotoisen tiedon ja välittää sen Flashiin. Tietyissä tilanteissa apuna voidaan käyttää myös luvussa 4.7 esiteltyä Flash Remoting -tekniikkaa. (Jacobson 2006, 369 - 371.)

RRS-uutisvirrat on yksi REST WWW -sovelluspalveluiden muoto. Niiden sisältöä voidaan hyödyntää Flashissa monesti suoraan samaan tapaan kuin kiinteitä XML-dokumentteja. Uusimmat Flash Playerit eivät tosin oletuksena anna ladata tietoa muulta kuin siltä palvelimelta, jolla itse Flash-esityskin on. Siksi palvelinpuolen sivuja joudutaan käyttämään tässäkin tapauksessa, mikäli RRS-uutisvirtaan ei ole määritelty oletusasetuksia korvaavia uusia tietoturva-asetuksia. (Jacobson 2006, 375.)

Suomessa esimerkiksi Taloussanomat tarjoaa tuoreimpien uutisiaan RRS-muodossa. Tekniikka mahdollistaa lehden mukaan uuden tavan seurata uutisia etsimättä niitä ympäri verkkoa. Käyttäjä pystyy tilaamaan omalle koneelleen RSS-feedin, jonka pohjalta lukuohjelma kokoaa listan kaikista tilauksista ja näyttää sen mukaiset uutiset. (Taloussanomien uutisotsikot sivuillesi 2006.)

XML-tiedon käsittely Flashissa tapahtuu kummassakin tilanteessa mieltymyksen mukaan käyttämällä joko XMLConnector-komponenttia tai XMLConnector- ja XML-luokan metodeja (Jacobson 2006, 368).

4.6 Datakomponenttien käyttö XML:n kanssa

4.6.1 Komponenttien esittely

Macromedia mainostaa, että Flash 8 on ohjelma, jolla voidaan tuottaa erittäin vaikuttavia elämyksiä web-ympäristöissä. Komponentit ovat rakennuspalikoita, joiden avulla rakennetaan monipuolisia web-sovelluksia, jotka mahdollistavat nämä elämykset. Komponentti on movie clip, jolla on tiettyjä paikallisia ominaisuuksia ja ActionScript-metodeja, ominaisuuksia ja eventtejä, jotka kaikki yhdessä mahdollistavat tapoja muokata komponenttia Flash-esityksen ajon aikana. Komponenttien idea on siinä, että kehittäjät voisivat helposti jakaa ja käyttää uudelleen luomaansa koodia. Komponentteihin voidaan kätkeä monipuolista toiminnallisuutta, eikä suunnittelijan ei tarvitse osata ActionScriptiä ja hän voi silti luoda monipuolisia ja dynaamisia sovelluksia eri komponenttien avulla. (Using Components 2006.)

Komponentteja voi rakentaa itse tai niitä voi ladata valmiina esimerkiksi Macromedian Flash Exchange -palvelun kautta. Flash 8 Professional -version mukana tulee valmiinakin joukko varsin hyödyllisiä komponentteja. Flashin oma ohje jakaa ne kuuteen eri ryhmään: Data-, FLVPlayback-, Media-, User interface-, Managers- ja Screens-komponentteihin. Data-komponentit mahdollistavat ulkopuolisen tiedon lataamisen ja muokkaamisen. FLVPlayback-komponentti on valmis videosoitin, jossa on toiminnot videon lataamiseen ja soittamiseen. Media-komponentit mahdollistavat streaming-sisällön soittamisen ja ohjaamisen. UI-komponentit mahdollistavat interaktion ohjelman kanssa, esimerkiksi tekstin syöttäminen. Managereilla ei ole visuaalista ulkoasua, vaan niitä käytetään ohjamaan tiettyjä ominaisuuksia, kuten movie clipin sijaintia näyttämön z-akselilla. (Using Components 2006.)

4.6.2 Data- ja UI-komponenttien ymmärtäminen

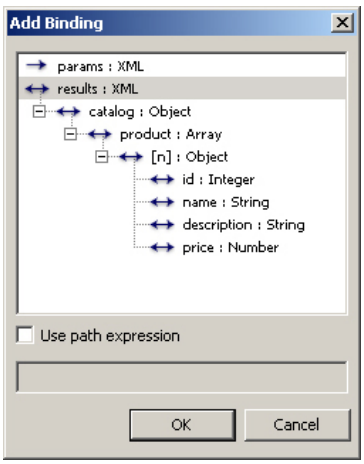
Dynaamisen multimediaesityksen luonnin kannalta kiinnostavimpia komponentteja ovat Data- ja UI-komponentit. Seuraavissa luvuissa pyritään havainnollistamaan, miten näiden Flashin mukana toimitettujen komponenttien avulla pystytään luomaan dynaaminen XML-pohjainen esitys ja silti työskentelemään haluttaessa täysin visuaalisessa tilassa eli niin, ettei riviäkään ActionScriptiä tarvitse kirjoittaa käsin. (Jacobson 2006, 325 - 326.)

Komponentti	Käyttötarkoitus
DataHolder	DataHolder toimii tiedon varastona ja sitä voidaan käyttää välittämään tietoa eri komponenttien välillä. Toisin kuin DataSet, se ei seuraa UI-komponenteissa tapahtuvia muutoksia.
DataSet	Toimii muuten kuin DataHolder, mutta mahdollistaa UI-komponenteissa tehtyjen muutosten seurannan. Flashissa tehdyistä muutoksista voidaan lähettää tietoa ulkopuolisille ohjelmille ja päivittää muuttuneet tiedot alkuperäiseen tietolähteeseen, esimerkiksi XML-tiedostoon.
RDBMSResolver	Toimii tietokantojen kanssa, käytetään lähettämään tieto Flashissa tehdyistä muutoksista alkuperäiseen tietoon.
WebServiceConnector	Mahdollistaa SOAP-pohjaisen tiedon käsittelyn.
XMLConnector	Ottaa yhteyden ulkopuoliseen XML-dokumenttiin. Komponenttia voidaan käyttää XML-tiedon lukemiseen tai kirjoittamiseen. Käytetään yhdessä DataSet ja XUpdateResolver -komponenttien kanssa, kun tarvitaan muuttaa ulkopuolista tietoa käyttäen apuna palvelinpuolen sivuja.
XUpdateResolver	Luo XUpdate-määrittelyn mukaisia lausuntoja, jotka kuvaavat XML-tietoon tehtyjä muutoksia Flashissa. Käytetään yhdessä DataSet komponentin kanssa.

KUVA 5. Data-komponenttien yleiset käyttötarkoitukset (Jacobson 2006, 326)

Flashin mukana tulee valmiina myös joukko visuaalisia UI-komponentteja, jotka pystyvät keskustelemaan data-komponenttien kanssa. Sellaisia ovat Combobox, List ja DataGrid komponentit. Nämä UI-komponentit pystyvät vastaanottamaan tietoa kukin hieman eri muodoissa, vaikka käyttävätkin kaikki tiedonlähteinään olioista muodostuvia taulukoita. DataGrid-komponentti pystyy vastaanottamaan ja esittämään tietoa monimutkaisistakin taulukoista. Kaksi ensimmäistä voi sisältää vain tietoa ja sitä kuvaavan nimen listalla. (Phillips 2006.)

Kuten luvussa 4.5 Flash ja XML todettiin, XML:n avulla voidaan kuvata tietoa ja sen rakennetta helposti lähestyttävän puumaisen rakenteen avulla. Vaikka itse XML-tiedoston tutkiminen editorilla paljastaakin helposti tiedon rakenteen, Flash ei XML-luokan kanssa työskenneltäessä tarjoa työkaluja rakenteen havainnollistamiseksi. Käytettäessä XMLConnector-komponenttia XML-dokumentin rakenne on nähtävissä bindings- tai skeema-välilehden kautta. Phillipsin mielestä XML-rakenteen visualisoiminen kuvan 6 mukaisella tavalla yksinkertaistaa merkittävästi työskentelyä XML-parissa. (Phillips 2006.)

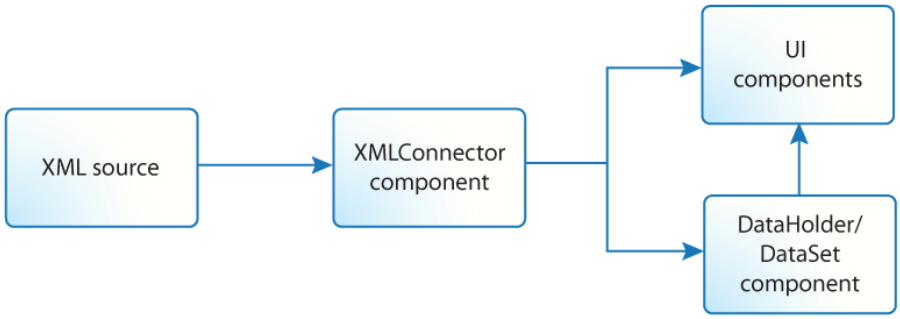


KUVA 6. XML-rakenne Flashin bindings-välilehden näyttämänä

4.6.3 XMLConnector-komponentin ymmärtäminen

XMLConnector-komponentti on vaihtoehtoinen tapa käsitellä XML-tietoa verrattuna XML-luokkaan (ks. luku 4.5.4). Sen sijaan, että käytetään ActionScipt-koodia XML-tiedostojen käsittelyyn, voidaan samat toiminnot toteuttaa komponenttien avulla ja työskennellä täysin visuaalisesti. XMLConnector-komponentti mahdollistaa XML-pohjaisen tiedon lähettämisen, vastaanottamisen sekä tiedon kaksisuuntaisen vastaanottamisen ja lähettämisen. (Jacobson 2006, 327.)

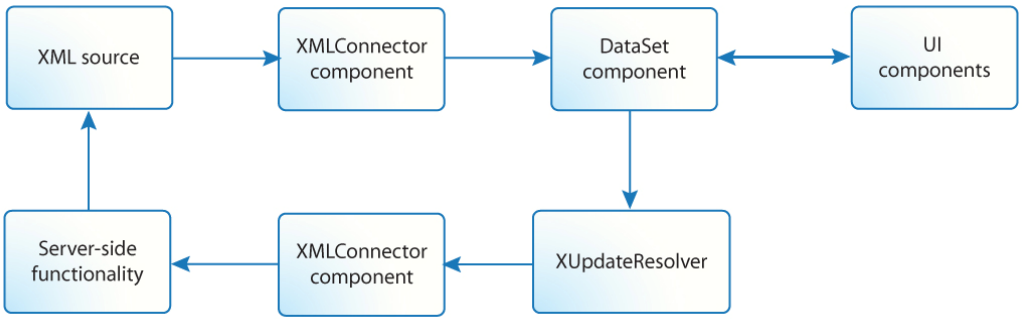
Pelkkä ulkopuolisen XML-tiedon lataaminen on yksinkertaisimmillaan varsin suoraviivainen prosessi. Ladataan tieto käyttäen XMLConnector-komponenttia. Sidotaan XML-sisältö suoraan yhteen tai useampaan visuaaliseen UI-komponenttiin. Joissain tapauksissa voi olla järkevää sitoa tieto ensin DataHolder- tai DataSet-komponenttiin ja vasta sitä kautta UI-komponentteihin, mutta Jacobsonin mukaan se ei ole kovin yleistä. (Jacobson 2006, 327.)



KUVA 7. XML-tiedon tuonti Flashiin komponenttien avulla

Rakennettaessa sovellus, jossa käyttäjän pitää päästä muokkaamaan ja tallentamaan XML-tietoa, muuttuu kokonaisprosessi hieman monimutkaisemmaksi. Flash ei itse pysty esityksen aikana tallentamaan mitään tietoa, vaan apuna joudutaan käyttämään palvelinpuolen sivuja, jotka sitten hoitavat muutosten tekemisen ja tallentamisen.

Työ aloitetaan kuitenkin aina samalla tavalla lataamalla XML-tieto käyttäen XMLConnectoria, mutta tässä tapauksessa tietoa ei voida sitoa suoraan UI-komponentteihin, vaan väliin joudutaan liittämään DataSet-komponentti, joka tarkkailee UI-komponenteissa tapahtuneita muutoksia XML-tietoon. DataSet pystyy generoimaan listan kaikista tehdyistä muutoksista ja lähettämään sen XUpdateResolverin käsiteltäväksi. Siellä lista muutoksista käännetään muotoon, jota palvelinpuolen sivu pystyy käsittelemään ja jonka pohjalta se pystyy tallentamaan ja tekemään muutokset alkuperäiseen XML-dokumenttiin.



KUVA 8. Editoitavan XML-tiedon käyttö Flashissa

4.6.4 Komponenttien väliset sidokset (Data Binding)

Edellisessä kappaleessa oli lyhyt maininta komponenttien välisistä sidoksista, mutta perehdytään niihin nyt hieman tarkemmin. Datasidosten avulla tietoa pystytään välittämään komponentilta toiselle. Määritellään siis tiettyjä ominaisuuksia komponenteille, jotka sen pitää välittää tietylle kohteelle. Yksinkertaistettuna datasidos tekee sen, että kun komponentin A ominaisuus X muuttuu, uusi ominaisuus kopioidaan komponentin B ominaisuudeksi Y. (Data Binding, 2006.)

Yksi hyödyllisimmistä käyttökohteista datasidoksille on määritellä niiden avulla datavirran kulku UI-, data management- ja connector-komponenttien välillä, jotka ovat yhteydessä ulkopuoliseen dataan kuten WWW-sovelluspalveluihin, XML-dokumentteihin tai relaatiotietokantoihin. (Data Binding, 2006.)

Sidosten tekeminen hoidetaan Flashissa täysin visuaalisesti, Component Inspector -paneelin kautta. Sidoksella määritellään aina sidottavan arvon lisäksi kohde ja suunta. Valittavana olevat sidottavat arvot vaihtelevat komponenttikohtaisesti: listakomponentilla on huomattavasti lukuisia sidottaviksi kelpaavia arvoja, kun taas tekstikenttäkomponentilla on vain yksi arvo ”text”. Sidoksilla on aina suunta ja kohde, tieto lähtee ulos jostain komponentista ja päätyy jonkin toisen komponentin sisään. Sidoksista on muistettava, että ne on tehtävä aina Flash-dokumentin ensimmäisessä kehyksessä. Sidokset eivät päde komponentteihin, jotka on lisätty aikajanalle jossain myöhemmässä vaiheessa. Sidokset eivät myöskään toimi käytettäessä useita eri scenejä samassa Flash-dokumentissa. (Jacobson 2006, 338.)

Vaikka sidosten ymmärtäminen aluksi voikin vaikuttaa hankalalta, on hyvä huomata, että niiden avulla pystytään luomaan monipuolisia sovelluksia kirjoittamatta itse riviäkään ActionScript-koodia. Jacobsonkin vakuuttaa, että komponenttien käyttö ja mahdollisuus työskennellä visuaalisesti sidosten kanssa helpottaa työskentelyä ulkoisen XML-tiedon kanssa. (Jacobson 2006, 338.)

4.6.5 Komponenttien kanssa työskentely käyttäen ActionScriptiä

Kaikissa tilanteissa ei ole mahdollista työskennellä visuaalisesti, ei edes komponentteja käytettäessä. Mikäli komponentteja lisätään näyttämölle dynaamisesti actionscriptin avulla, niitä ei ole olemassa ennen kuin valmis esitys koostetaan. Siksi tietojen sitominenkaan ei ole mahdollista ”component inspector” -paneelin kautta. Onneksi komponenttien kanssa voi työskennellä perinteisesti ActionScriptin avulla. (Jacobson 2006, 354 - 357.)

XML Connector -luokka on skriptiversio XMLConnector -komponentista. Luokan avulla pystytään luomaan uusia instansseja komponentista ja muokkaamaan komponentin ominaisuuksia ActionScriptillä. Luokan metodeja ja ominaisuuksia voidaan käyttää hyödyksi haetun XML-sisällön kanssa työskenneltäessä (Jacobson 2006, 354 - 357.)

Tietojen sitominen voidaan toteuttaa myös ActionScriptin avulla. Käytetään hyväksi DataBindingsClasses-scriptikomponenttia, joka sisältää joukon tarpeellisia luokkia datasidosten tekemiseen. Luokkien tarjoamien metodien avulla on mahdollista määritellä sidoksien alku ja loppupisteet, ja ne tiedot joita sidoksien kautta välitetään. (Jacobson 2006, 358.)

Datasidosten toteuttaminen käsin ohjelmoimalla on haastava prosessi ja vaatii, että käyttäjä on perehtynyt hyvin ActionScriptin käyttöön. En sen vuoksi käy läpi sitä koodia, jolla XML-tietoa voitaisiin hakea ja esittää Flashissa, käyttäen hyödyksi komponentteja ja ActionScriptiä. Jacobson myös on sitä mieltä, että datakomponenttien skriptaaminen käsin on melko haastavaa, joten kannattaa käyttää visuaalisia työkaluja hyödyksi, kun se vain on mahdollista (Jacobson 2006, 363).

4.6.6 UI-komponenttien ulkoasu

Yleistä ulkoasuista

Flash 8 tarjoaa kehittäjän käyttöön valmiiksi rakennettuja komponentteja, joilla on oletuksena tietty ulkonäkö ja toiminnot. Niitä voi käyttää esityksissä suoraan sellaisenaan, mutta aina se ei riitä, koska monesti on tarvetta muokata niitä paremmin omaan esityksen ulkonäköön soveltuviksi. Siihen on olemassa pääpiirteittään kolme eri tapaa: tyylit, skinit ja teemat. (Customizing Components 2006.)

Tyylimäärittelyt

Ehkä yksi helpoiten lähestyttävä ja suoraviivaisin tapa muokata komponenttien ulkoasua on tehdä se tyylien avulla. Flashin mukana tulleilla UI-komponentilla on vaihteleva määrä eri ominaisuuksia, joita voidaan muokata tyylien avulla. Näitä ominaisuuksia voidaan muokata UIObject.setStyle() metodin avulla. SetStyle on ohittaa hierarkiassa kaikki muut komponentin muotoilut ja sen avulla asetettu ulkoasu ohittaa mahdolliset tyylimäärittelyt ja teemat. (Using styles to customizise component color and text 2006.)

Tyylejä voidaan määritellä usealla eri tavalla ja eri paikkaan:
1. Määritellään tyyli yksittäiselle komponentille. Tämä on tehokas ja nopea tapa joissain tapauksissa, mutta hankalaa ja raskasta, kun komponentteja on lukuisia.
2. Muokataan globaalia tyylimäärittelyä, joka asettaa tyylin kaikkien dokumentin komponenttien kohdalla.
3. Luodaan oma tyylimäärittely ja asetetaan se tietylle joukolle komponentteja.
4. Luodaan oletus tyylimäärittely tiettyä luokkaa varten, jolloin kaikki luokan pohjalta tehdyt komponentit saavat saman ulkoasun.
5. Määritellään tyylit tiettyä dokumentin osaa varten hyödyntäen tyylien periytymistä. Komponentti perii ominaisuuksia säilytinelementiltä, jonka sisällä se on, esimerkiksi toinen komponentti tai movieclip.
(Setting styles on a component instance 2006.)

Yksittäisen komponentin tekstin värin vaihtaminen onnistuu helposti, setStyle() metodin avulla:

```
komponentinNimi.setStyle("ominaisuudenNimi", arvo);
tekstiLaatikko.setStyle("color","blue");
```

Mitä useampaa komponentin ominaisuutta muokataan, sitä enemmän prosessointiaikaa kuluu komponentin näyttämiseen Flash-esityksen aikana. Kun joudutaan muokkaamaan useampaa eri tyyliominaisuutta, olisi Flashin ohjeen mukaan järkevää ja ohjelman toiminnan kannalta nopeampaa luoda komponentti dynaamisesti ja määritellä ominaisuudet jo luontivaiheessa. (Setting styles on a component instance 2006.)

Sen sijaan, että määritellään jokaiselle komponentilla aina omat tyyliasetukset, voidaan muuttaa kaikkien dokumentissa olevien komponenttien oletustyyliä muokkaamalla arvoja globaalisesti (_global.style). Jokainen dokumentin komponentti käyttää globaalia tyylimäärittelyä, kunnes jokin toinen määrittely liitetään siihen. On huomattava, että joitain arvoja ei voi muokata globaalitasolla. Jos vaihdetaan esimerkiksi taustaväriä (backgroundColor), ei sillä olisi mitään vaikutusta TextArea- ja TextInput-komponentteihin, koska taustan väri on määritelty näiden komponenttien luokan arvoksi, ja luokkamäärittelyt ohittavat hierarkiassa globaalit arvot.

```
_global.style.setStyle("color", 0xCC6699)
```
(Setting global styles 2006.)

Kolmas vaihtoehto on, että luodaan oman globaalin tyylimäärittelyn ja asetetaan se käytettäväksi joukolle komponentteja. Määritellään tälle omalle tyylille halutut arvot ja asetetaan halutut komponentit käyttämään kyseistä tyylimäärettä.

```
_global.styles.omaTyyli.setStyle(nimi, arvo);
komponentinNimi.setStyle("styleName", "omaTyyli");
```
(Seting custom styles for groups of components 2006.)

Kaikille valmiille UI-komponenteille on mahdollista luoda myös luokkakohtainen tyylimäärittely, joka määrittelee, miltä kaikki sen luokan komponentit näyttävät. Joillekin komponenteille, kuten TextArealle ja TextInputille, on oletuksena luotu jo luokkakohtainen tyylimäärittely, joka ohittaa esimerkiksi globaalit tyylimäärittelyt. Suunnittelun kannalta on ehdotonta tietää, missä järjestyksessä Flash käsittelee eri tyylimäärittelyjä. Flash etsii arvoja järjestelmällisesti seuraavanlaisesti:
1. Flash etsii komponentin paikallista tyylimäärittelyä.
2. Flash tutkii, onko komponenttiin liitetty styleName-ominaisuuden kautta jokin uusi tyylimäärittely.
3. Flash etsii tyylejä luokan oletus tyylimäärittelyistä.
4. Mikäli arvo on periytyvien arvojen listalla, Flash etsii arvoa isäntäkomponentilta hierarkiarakenteen mukaisesti.
5. Tutkitaan globaalit tyylimäärittelyt
6. Jos arvoa ei vieläkään löydy, sen arvoksi tulee undefined eli määrittelemätön
(Using global, custom, and class styles in the same document 2006.)

Skinit

Skinit ovat movie clip -symboleja, joita komponentti käyttää esittämään niiden ulkoasua. Osa skineistä koostuu graafisista elementeistä, osa sisältää vain ActionScript-koodia, joka piirtää komponentin Flash-esityksen aikana. Komponenttien skinejä ei pääse muokkaamaan suoraan siitä dokumentista, jossa ne sijaitsevat. Komponenttien skinit sijaitsevat erillisissä .fla-tiedostoissa, joita kutsutaan teemoiksi. Jokaisella teemalla on oma ulkoasu ja toiminnot, mutta skinien nimet ja linkkiinformaatio ovat yhteneviä. (About skinning components, 2006.)

Jokainen komponentti koostuu lukuista eri skineistä. Pelkästään vierityspalkkikomponentissa (ScrollBar) oleva alaspäin osoittava nuoli (down arrow) koostuu neljästä eri skinistä; ScrollDownArrowDisabled, ScrollDownArrowDown, ScrollDownArrowOver ja ScrollDownArrowUp, jotka kuvastavat nuolen ulkoasua eri tiloissa. Tarvitaan yhteensä 13 eri skiniä kuvaamaan koko komponentin ulkoasua. Osa skineistä on kuitenkin yhteisiä muiden komponenttien kanssa, esimerkiksi vierityspalkin skinejä käyttää myös monivalintaruutu- ja listakomponentit. (About skinning components 2006.)

Macromedian ohjeen mukaan helpoin tapa vaikuttaa komponentin ulkoasuun skinejä käyttämällä on kopioida ja muokata jo käytössä olevan teeman skinielementtejä. Koska uudet muokatut skinit ovat samannimisiä kuin entiset, Flash osaa automaattisesti päivittää komponenttien ulkoasun. Muutokset tulevat voimaan koskien kaikkia komponentteja, jotka käyttävät kyseistä skiniä. Tämä on täysin graafinen toimenpide, eikä vaadi ActionScriptin käsittelyä. (About skinning components 2006.)

Teemat

Teemat ovat kokoelmia tyylejä ja skinejä, jotka yhdessä määrittelevät miltä komponentti näyttää. Varsinaisen graafisen ulkoasun määräävien skinien lisäksi teemaan sisällytetään tyylejä, joilla voidaan vaikuttaa esimerkiksi oletusarvoisen tekstin kokoon tai väriin. (About themes, 2006.)

Flashin perusteema on nimeltään Halo (HaloTheme.fla). Mukana tulee myös yksinkertaisemman näköinen Sample-teema, joka tarjoaa kuitenkin enemmän tyyliominaisuuksia käyttöön. Tuleekin aina tarkistaa kumpi teema tukee haluttua tyyliominaisuutta, sillä kaikki eivät toimi molempien teemojen kanssa. Uusia omia teemoja voi lähteä luomaan helposti ottamalla kopio esimerkiksi Halo-teemasta ja tekemällä siihen haluttuja muutoksia. (About themes, 2006.)

4.6.7 Komponentit ja Flash Lite 2.0

Flash Lite 2.0 mahdollistaa monipuolisen Flash-sisällön esittämisen useissa eri mobiilikanavissa. Kuten edellä havaittiin, on komponenttien avulla suhteellisen helppo luoda elämyksellisiä Flash-sovelluksia. Harmittavasti helppoudella on hintansa ja komponenttien asettamat muisti- ja prosessointitehovaatimukset ovat liian vaativia monille mobiililaitteille. Macromedia suosittelee, että Flashin mukana toimitettuja peruskomponentteja ei käytettäisi ollenkaan Flash Lite -sovelluksissa. (About components in Flash Lite 2.0, 2006.)

4.7 Flash Remoting

Flash Remoting on Macromedian luoma teknologia, jonka tarkoitus on helpottaa ja tehostaa palvelimen ja Flash-esityksen välisiä yhteyksiä, mahdollistaen monipuolisten ja nopeiden WWW-sovellusten luonnin.

Flash Remoting on tekniikka, jonka avulla Flash-esitys saa käyttöönsä kaikki palvelimen toiminnot mahdollistaen tietokantayhteydet, sähköpostin lähettämisen sekä tiedostojen lukemisen ja kirjoittamisen. Tietoa ei lähetetä palvelimelta tavalliseen tapaan muuttuja-arvopareina, vaan se kulkee AMF-muodossa (ActionScript Message Format). AMF on oma formaatti, jonka avulla voidaan välittää isompia tietokokonaisuuksia käyttäen ActionScipt-muotoisia taulukoita. Flash pystyy lukemaan tiedon suoraan näistä taulukoista, eikä sen tarvitse käsitellä kaikkia muuttujia ja arvoja yksitellen. AMF:n avulla voidaan välittää myös XML-muotoista dataa. (Solis 2006.)

Flash Remoting rakentuu ActionScript-tiedostoista, joten Flashin käyttäjille ohjelmointikieli on tuttu. Flash Remoting tarjoaa tietoturvallisemman ja nopeamman tavan Flashille keskustella palvelinpuolen sovellusten ja tietokantojen kanssa. Flash Remoting on täysin yhteensopiva sovellusten kanssa, jotka on rakennettu tekniikoilla: Macromedia ColdFusion, .NET, JAVA ja SOAP-pohjaiset WWW-sovelluspalvelut. Amfphp mahdollistaa myös täysin lisenssivapaan tavan toteuttaa Flash Remoting -pohjaisia sovelluksia. (David 2005a.)

4.8 Yhteydet tietokantoihin

Luotaessa laajoja Internetsivuja tai multimediasovelluksia, joissa on paljon sisältöä ja joita joudutaan muokkaamaan tiheään, ovat tietokannat monesti tehokkain tapa varastoida ja jäsennellä tietokokonaisuutta. Flashiin on ollut mahdollista tuoda dataa tietokannoista sitten Flash 3:n julkaisun. Yhteys tietokantoihin on suoritettu teknologialla nimeltään server scripting. Tyypillisimpiä server scripting -kieliä on Microsoftin ASP, Java Server Pages, PHP ja Macromedian ColdFusion. Palvelinpohjainen sivu ottaa yhteyden tietokantaan ja hakee sieltä halutun sisällön ja välittää sen Flashille. WWW-ympäristössä prosessi etenee usein viidessä vaiheessa:

Vaihe 1. Selaimeen ladataan WWW-sivu, johon on integroitu Flash-esitys.
Vaihe 2. Flash-esitys lähettää pyynnön palvelinpohjaiselle WWW-sivulle.
Vaihe 3. Sivu ottaa tietokantayhteyden ja vastaanottaa pyynnön mukaisen sisällön.
Vaihe 4. Tietokannasta haetusta sisällöstä muodostetaan dynaaminen WWW-sivu, joka lähetetään Flash-esitykselle.
Vaihe 5. Flash vastaanottaa dynaamisen sivun ja ennalta määrätyn ActionScriptin mukaan sijoittaa sisällön esitykseen niin, että käyttäjä voi nähdä sen.
(David 2005b.)

Tietokanta voi olla luotu käyttäen esimerkiksi Microsoftin Accessia tai MySQL:ää. Jälkimmäistä suositaan yleensä, koska se on lisenssivapaa ja toimii laajoillakin tietokannoilla, joissa on paljon liikennettä. Tieto voidaan siirtää Flashiin käyttäen aiemmissa luvuissa esiteltyjä tekniikoita: FlashVars, LoadVars, XML, WWW-sovelluspalvelut ja XMLSocket yhteydet. Tekniikka kannattaa valita aina tilannekohtaisesti ja oman osaamisen mukaan Flash tarjoaa useita vaihtoehtoja. (David 2005b.)

4.9 Oikean tekniikan valinta

4.9.1 XML:n heikkouksia

Tämä opinnäytetyö keskittyy paljolti XML-pohjaisiin tekniikoihin ja niiden hyödyntämiseen Flashilla tuotetussa dynaamisessa multimediaesityksessä. Mutta kuten Jacobsonkin tuo kirjassaan esille, on huomattavaa, ettei XML ole aina välttämättä paras ratkaisu. XML vaatii tekstisisällön lisäksi melko paljon koodia kuvaamaan tiedon rakennetta, minkä seurauksena dokumenteista voi tulla hyvinkin isoja ja Flash-sovelluksesta liian hidas. (2006, 427.)

4.9.2 Vasteajat

Flash-ohjelmalta vaadittavat vasteajat vaikuttavat paljolti valittavaan tekniikkaan. XML-käsittely on suhteellisen hidas prosessi dokumentit voivat olla hyvinkin kookkaita ja Flash joutuu aina käymään läpi koko rakenteen ja etsimään sieltä haluttu informaatio. Kaikki tietenkin haluavat, että ohjelma olisi mahdollisimman nopea, mutta yleisesti XML-käsittelystä aiheutuva, muutaman sekunnin viive on varsin hyväksyttävä. Vasteajoista tuleekin yleensä ongelman vasta, kun suunnitellaan todella laajoja, organisaatiotason ohjelmia. (Jacobson 2006, 428.)

Vaadittaessa nopeita vasteaikoja kannattaa tekniikaksi valita Flash Remoting, joka tarjoaa Flash-esitykselle huomattavasti nopeamman tavan käsitellä ulkopuolista tietoa. Flash Remoting -tekniikka pakkaa siirrettävän tiedon pienempään kokoon ja mahdollistaa Flash-esityksen käyttää palvelinpuolen koodia suoraan, aivan kuten paikallistakin ActionScript-koodia. Ongelmana on, että ihmisen on erittäin hankala lukea käytettävää AMF-viestimuotoa ja toisekseen Flash Remoting on maksullinen tuote ja aiheuttaa lisäkuluja. (Jacobson 2006, 428 - 429.)

4.9.3 Tietokannat ja tietokantojen käsittely

Flash ei pysty suoraan hakemaan tietoa tietokannoista, vaan se haetaan sieltä palvelinpuolen sivua käyttämällä ja muuntamalla haettu tieto Flashin ymmärtämään muotoon. Monissa tapauksissa on nopeinta lähettää tieto Flashiin käyttäen muuttuja-arvopareja ja LoadVars-oliota. Toisaalta kyselyn tuloksena voidaan muodostaa myös XML-tiedosto ja lähettää se Flashin käsiteltäväksi. Kumpikaan tekniikka ei ole ylivertainen toisiinsa verrattuna. Jotkin tietokannat osaavat palauttaa kyselyn tuloksen suoraan XML-muodossa. Tällöin palvelinpuolen sivun ei tarvitse prosessoida dataa, ja XML-muotoinen tieto voidaan välittää suoraan Flashin käytettäväksi. Tällaista ominaisuutta kannattaa yleensä hyödyntää, sillä se on todennäköisesti käytössä nopeampi ja helpommin toteutettavissa. (Jacobson 2006, 429 - 430.)

Palvelinpuolen sivuja on käytettävä yhdessä Flashin kanssa, mikäli halutaan keskustella tietokantojen kanssa, käyttää nopeaa Flash Remotin tekniikkaa, tallentaa jokin XML-tietovirta kiinteäksi XML-dokumentiksi tai kun Flashia halutaan käyttää tiedon muokkaamiseen ja tallentamiseen. Näitä tekniikoita käyttäen Flash-esityksen käyttö rajoittuu tilanteisiin, joissa on yhteys palvelimelle, eikä esitystä voi ottaa mukaan esimerkiksi CD:lle. (Jacobson 2006, 429 - 430.)

4.9.4 Valmiit XML-tiedostot

Flash-esityksissä hyödynnetään yleensä jo valmista tietoa, joka on varastoitu tiettyyn paikkaan. Mikäli tieto on jo valmiiksi tallennettu XML-muotoon, ei ole mitään syytä lähteä viemään sitä tietokantaan tai muuntamaan tekstitiedostoksi, koska siitä aiheutuisi vain yliääristä työtä. XML rakennetta voidaan XSL muunnoksien myötä usein muokata vielä paremmin Flashille sopivaan muotoon. (Jacobson 2006, 429.)

XML-tiedostot sopivat hyvin myös tilanteisiin, joissa Flash-esityksen tulee toimia ilman yhteyttä palvelimelle, esimerkiksi kun esitys halutaan tallentaa CD:lle tai DVD:lle. Jopa XML-tietovirta on mahdollista tallentaa palvelinpuolen sivua hyödyntämällä staattiseksi XML-tiedostoksi, jolloin tietovirran sisältöä voidaan käyttää ilman yhteyttä palvelimelle. Monet ohjelmistot mahdollistavat nykyisin myös tiedon tallentamisen XML-muodossa, jolloin niitäkin voidaan käyttää Flash-esitysten sisällön tuottamiseen. Esimerkiksi Microsoft Office 2003:n mukana tulevat Word, Excel ja Access mahdollistavat monille tutun käyttöliittymän kautta helposti lähestyttävän tavan käsitellä XML-tietoa. (Jacobson 2006, 429 - 430.)

4.9.5 XML-tiedon tuominen Flashiin

Kuten luvussa 4.5 esiteltiin, mahdollistaa Flash lukuisia eri tapoja työskennellä XML-datan kanssa. Käytettäviä tekniikoita ovat XML-luokka, valmiit datakomponentit, XMLConnector-, WebServiceConnector- tai Web Service -luokat ja XMLSocket-luokka. (Jacobson 2006, 432.)

Mikäli Flash-sovelluksen täytyy pystyä toimimaan reaaliaikaisen XML-datan kanssa ja vaaditaan, että ohjelmaa voi käyttää samanaikaisesti useakin käyttäjä, ainoa valittava tekniikka on käyttää XMLSocket-luokkaa ja omaa socket-palvelinta. Tämä on ainoa tekniikka, joka mahdollistaa täysin reaaliaikaisen XML-datan syöttämisen Flashiin. (Jacobson 2006, 433.)

Yksi tekniikan valintaan vaikuttava tekijä on kehitykseen käytettävän Flashin versio. Flash 5:stä aikaisemmilla versiolla ei ole mahdollista käyttää XML-dataa. Flash MX 2004:n ja Flash 8:n Professional-versiot mahdollistavat kaikkien tässä työssäkin esiteltyjen tekniikoiden toteuttamisen. Standard-versioiden mukana ei tule valmiita datakomponentteja tai Connector-luokkia. (Jacobson 2006, 433 - 434.)

Jacobsonin mukaan datakomponenttien käyttö sopii ihmisille, jotka työskentelevät mieluiten visuaalisesti käyttäen Flashin työkalupaneeleja. Toiset toteuttavat samat toiminnallisuudet mieluummin kirjoittamalla ActionScriptiä ja käyttävät hyödyksi joko XML-, WebService- tai Connector luokkia. ActionScriptin käyttö tarjoaa hieman enemmän joustavuutta, mutta kummallakin päätyylillä päästään kuitenkin samoihin lopputuloksiin. (Jacobson 2006, 433 - 434.)

5 XML-datan luominen ja muokkaaminen

5.1 Yleistä

XML-dokumenttien kanssa voi työskennellä lukuisilla eri tavoilla. Niiden luominen ja muokkaaminen on suhteellisen helppoa, sillä siihen ei välttämättä tarvita mitään erityistä ja kallista ohjelmistoa. Työn pystyy tekemään yksinkertaisimmillaan käsin käyttäen apuna mitä tahansa tekstieditoria, esimerkiksi Windowsin mukana tulevaa Muistiota. Vaihtoehtoisesti dokumenttien käsittelyn voi hoitaa ohjelmallisesti, koskematta itse varsinaiseen XML-tiedostoon ollenkaan. Voidaan hyödyntää esimerkiksi Microsoftin Office-paketin tuotteita tai jotain omaa, esimerkiksi PHP:llä tai JSP:llä tehtyä, palvelinpuolen sivua, joka tarjoaa lomaketyyppisen ratkaisun tiedon päivittämiseen. (Jacobson 2006, 39.)

5.2 Tekstieditorit

Varmasti eräs tutuimmista tavoista käsitellä XML-tietoa on käyttää siihen Muistion tai SimpleText-ohjelman kaltaisia pelkistettyjä tekstieditoreita. Kenen tahansa on helppo käyttää niitä, mutta ne eivät tarjoa mitään erityisiä työkaluja helpottamaan työskentelyä XML:n kanssa. Käyttäjä joutuu itse kirjoittamaan jokaisen rivin koodia ja muistamaan lisätä jokaisen tagin itse. Työaikaa voi kulua hukkaan paljonkin, ainakin työskenneltäessä isojen dokumenttien kanssa. (Jacobson 2006, 58.)

Tavalliset tekstieditorit eivät myöskään osaa värikoodata kirjotettua koodia, mikä osittain hankaloittaa työskentelyä. Ne eivät myöskään tarjoa minkäänlaista mahdollisuutta tarkistaa, onko XML-dokumentti oikein laadittua ja sisältääkö se virheitä. Mahdolliset virheet huomataan vasta kun ensimmäisen kerran yritetään avata XML-tiedostoa, mistä aiheutuu lisää hukkaan heitettyjä työtunteja. Tiedostojen käsittelyssä voidaan hyödyntää myös tuttuja HTML-editoreita, esimerkiksi HomeSite tai BBEdit. Etuina on, että HTML-editorit osaavat yleensä värikoodauksen ja pystyvät automatisoimaan työprosessia osittain. Ne voivat esimerkiksi lisätä oikean dokumenttimääritelmän dokumentin alkuun ja automaattisesti täydentää tagien loppuosia kirjoitettaessa koodia. (Jacobson 2006, 58 - 59.)

5.3 XML-editorit

XML-editori on ohjelma, joka on suunniteltu nimenomaan XML-dokumenttien käsittelemiseen. Useimmat niistä sisältävät työkaluja, jotka helpottavat merkittävästi työskentelyä, kuten tägien automaattinen täydentäminen ja dokumentin oikeellisuuden ja rakenteen tarkistaminen. Tavallisia tekstieditoreita käytettäessä mahdolliset virheet XML-dokumentissa tulevat esiin vasta tiedostoa avattaessa. Sen lisäksi, että testaaminen on tällä tavalla hankalaa, ongelmana on monesti se, että virheilmoitukset eivät ole kovin kuvaavia ja virheen paikantaminen voi olla hankalaa. Monet XML-editorit, kuten XMLSpy, antavat hyvinkin tarkkaa tietoa virheen laadusta ja osoittavat selkeästi, missä kohden koodia virhe syntyy. (Jacobson 2006, 59 - 61.)

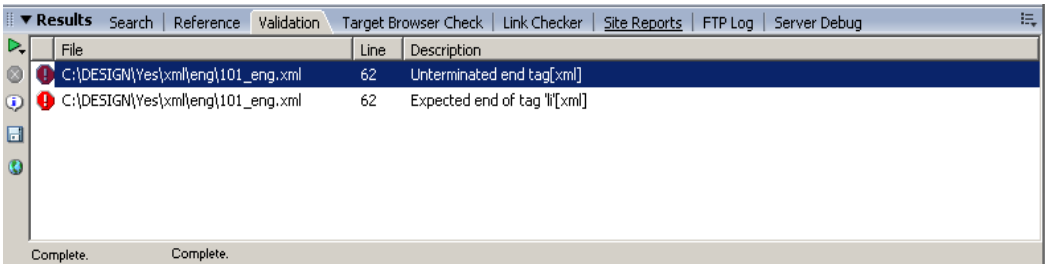
XML-editorit tutkivat yleensä kahden tyyppisiä virheitä. Tutkitaan, noudattaako dokumentti oikeaoppista XML-rakennetta, eli tarkistetaan muun muassa, ovatko tägien nimet oikein ja löytyvätkö aina vastaavat aloittavat ja lopettavat tägit. On monesti myös mahdollista tarkistuttaa, noudattaako dokumentti tiettyä DTD:tä tai skeemaa. Silloin virhe voi syntyä esimerkiksi tapauksessa, jossa elementtien järjestys ei noudata skeeman mukaista määrittelyä. (Jacobson 2006, 61 - 63.)

XML-editoreita on monenlaisia. On olemassa useita maksullisia ohjelmia, kuten suositut Altovan XMLSpy ja SyncRO Softin <oXygen/>, mutta myös hyviä ilmaisia freeware-ohjelmia, esimerkiksi XMLEditPro ja XMLFox. Vaikka XML-editorien käyttö ei missään nimessä ole pakollista, on se Jacobsonin mukaan erittäin kannattavaa ja nopeuttaa työskentelyä. (Jacobson 2006, 61 - 63.)

5.4 Macromedia Dreamweaver

Dreamweaver on yleisesti tunnettu ammattilaistason työkalu web-suunnittelijoiden keskuudessa, jotka käyttävät sitä työskennellessään JavaScriptin, CSS:n tai HTML:n kanssa. Vaikkei Dreamweaver olekaan mikään täysimittainen XML-editori, sisältää se reilun joukon työkaluja XML-dokumenttien käsittelemistä varten. (Ruse 2004.)

Monien aitojen XML-editorien tapaan osaa Dreamweaver värikoodata XML-koodia, tarkistaa XML-dokumenttien oikeellisuuden ja ilmoittaa mahdollisista virheistä. Tarkistuksen voi halutessaan hoitaa nopeasti myös esikatselutilassa. Silloin Dreamweaver avaa dokumentin oletusselaimella esimerkiksi Internet Explorerilla, jolloin virheilmoitusten laatu riippuu täysin selaimesta. Dreamweaverin oma validaattori osaa myös verrata XML-dokumenttia sitä vastaavaan DTD:hen tai skeemaan. (Ruse 2004.)



KUVA 9. Dreamweaverin Results paneeli näyttämä virheilmoituksen validoinnin jälkeen

Rusen mukaan tilanteissa, joissa työskennellään XML-pohjaisten tietokantayhteyksien kanssa tai hyödynnetään Web Service -palveluita, ei Dreamweaver suinkaan ole paras mahdollinen työkalu. Se tarjoaa kuitenkin yhdessä Flashin ja ColdFusionin kanssa monipuoliset mahdollisuudet työskennellä XML tiedon kanssa. (Ruse 2004.)

5.5 Microsoft Office 2003

5.5.1 Esittely

Office-tuotteet ovat tuttuja lähes kaikille tietokoneita käyttäville, mutta harva tietää, että niitä voidaan hyödyntää tehokkaasti työskenneltäessä XML-dokumenttien kanssa. Microsoftin Office 2003 -paketti sisältää sisäänrakennetun tuen XML:lle. Word-, Excel- ja Access-ohjelmilla voi avata, muokata ja muuntaa XML-dokumentteja. (Kelly 2003.)

Normaalissa käytössä Word 2003 tallentaa XML-dokumentin käyttäen WordprocessingML-kieltä, kun taas Excel käyttää SpreadsheetML-kieltä. Kummatkin kielet noudattavat XML-standardia. Niitä käytettäessä XML-dokumenteista tulee kuitenkin hyvin raskaita ja hankalasti luettavia, sillä elementeillä on määritelty kaikki Office-dokumenttiin liittyvä, kuten dokumentin asetukset, muotoilut ja itse tieto. (Jacobson 2006, 66 - 67.)

Mikäli käytössä on Office 2003 Professional tai Enterprise edition, tiedostoille voidaan kuitenkin määritellä skeema tai XSL-tyylisivu, jolloin voidaan itse päättää täysin, millaista XML-koodia ohjelma kirjoittaa. Tämä onkin monesti oletusarvoista tallennustapaa järkevämpi vaihtoehto, ja käsittelen siihen liittyviä tekniikoita tarkemmin seuraavissa luvuissa. (Jacobson 2006, 66 - 67.)

5.5.2 Office-tuotteiden etuja

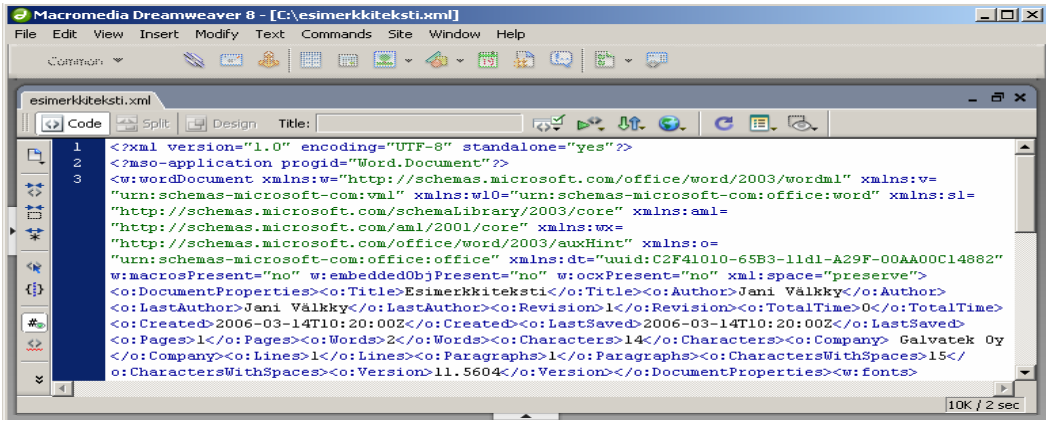
Microsoftin Office on epäilemättä yksi suosituimmista yrityksien käyttämistä ohjelmistoista. Työkalut löytyvät yleensä valmiina, ja ainakin Exceliä ja Wordiä osataan keskimäärin käyttää melko hyvin. Jacobson on havainnut työssään, että ihmisten on suhteellisen helppo oppia työskentelemään myös XML-tiedostojen kanssa, jos käytössä on Wordin tai Excelin kaltainen tuttu ohjelma. (Jacobson 2006, 218.)

Flash-sovelluksessa voidaan hyödyntää tehokkaasti dynaamista sisältöä ja erottaa ulkoasu ja sisältö toisistaan. Käyttämällä hyväksi Office-paketin tuotteita on sisällön päivittäminen mahdollista toteuttaa varsin käyttäjäystävällisellä tavalla. Sisältöä päivittäville ihmiselle ei tarvitse lähteä opettamaan kokonaan uutta ohjelmistoa vaan riittää, että oppii muutaman uuden tekniikan jo ennestään tutusta ohjelmasta. Näin Flash-tuottajat voivat tarjota asiakkailleen helpon tavan muuttaa ja päivittää sisältöä koskematta itse Flash-tiedostoihin. (Jacobson 2006, 218.)

XML- tai teksti-editorien käyttäminen voi olla monelle tägejä ymmärtämättömälle haastava tilanne ja mahdollisuus tehdä virheitä on suuri. Officessa voidaan kuitenkin piilottaa kaikki tägit ja ohjata käyttäjää antamaan oikeita syötteitä ja muodostamaan kelvollisia XML-dokumentteja, joita voidaan hyödyntää esimerkiksi Flashissa. (Jacobson 2006, 218.)

5.5.3 Word 2003

Helpoimmallaan Wordilla voidaan luoda XML-dokumentteja käyttäen ’Tallenna nimellä’ -toimintoa ja valitsemalla tiedostomuodoksi ’.xml’. Tällöin Word tallentaa dokumentin WordprocessingML muotoon. Jatkokäsittelyn kannalta ongelmana on, että Word tuottaa valtavan määrän koodia kuvatessaan jo yhtä ainoata sanaa. Syynä on se, että Word tallentaa XML-dokumenttiin itse tiedon lisäksi kaikki Word-asiakirjan asetukset ja muotoilut. (Jacobson 2006, 66.)



KUVA 10: Esimerkki Wordin tuottamasta WordprocessingML koodista, katsottuna Macromedian Dreamweaverilla

Mikäli käytettävissä on Wordin Professional-, Enterprise- tai Standalone-versio, voidaan käyttää hyväksi skeemoja. Niiden avulla voidaan varmistaa, että tallennettava XML-dokumentti noudattaa haluamaa muotoa ja rakennetta. Merkittävää on, että voidaan huomattavasti pienentää syntyvän XML-tiedoston kokoa ja käyttää haluttuja tägejä tiedon kuvaamiseen. (Jacobson 2006, 66 - 68.)

Jacobsonin ohjeen mukaan skeemaa käyttävän XML-dokumentin luonti on kolmivaiheinen. Ensiksi luodaan uusi skeema-tiedosto, jonka pääte on yleensä ’.xsd’. Siinä luodaan säännöt XML-dokumentille. Seuraavassa esimerkkikoodissa on näytetty, miltä tyypillinen skeema-tiedosto näyttää. Siinä määritellään, että uutiset-elementillä on aina oltava pvm-, otsikko- ja sisalto-elementit, jotka koostuvat string-muotoisesta datasta. Elementtien tulee aina myös esiintyä tässä tietyssä järjestyksessä. (Jacobson 2006, 68 - 70.)

```
<?xml version="1.0">
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="uutiset">
   <xsd:complexType>
    <xsd:sequence>
        <xsd:element name="pvm" type="xsd:string"/>
        <xsd:element name="otsikko" type="xsd:string"/>
        <xsd:element name="sisalto" type="xsd:string"/>
       </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Toiseksi on luotava uusi Word 2003 -pohja ja asetettava se käyttämään luotua skeemaa. Wordissa luodaan dokumentin rakenne ja jätetään siihen muokattavia kohtia, joihin varsinainen tieto tallennetaan. Word-pohja voi koostua, esimerkiksi tässä tapauksessa kolmesta lomakentästä, jotka asetetaan käyttämään skeeman mukaisia elementtejä ”pvm”, ”otsikko” ja ”sisalto”. (Jacobson 2006, 68 - 70.)

Luotua pohjaa hyödynnetään uusien XML-dokumenttien luomisessa. Tuloksena saadaan oikein muodostettu XML-dokumentti, joka on merkittävästi lyhyempi ja selkeämpi kuin vastaava WordprocessinML-versio. Vanhoja jo olemassa olevia XML-dokumentteja on myös helppo muokata Wordilla, mikäli niiden skeemat on määritelty oikein. (Jacobson 2006, 70 - 71.)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<uutiset xmls="omaSchema">
      <pvm>Päivämäärä</pvm>
      <otsikko>Uutisotsikko</otsikko>
      <sisalto>Uutisen varsinainen sisalto</sisalto>
<uutiset>
```

Word on parhaimmillaan käsiteltäessä sellaisia XML-dokumentteja, joissa tieto ei ole jatkuvasti toistuvaa. Word toimiikin parhaiten tilanteissa, jossa XML-dokumentti luodaan lomaketyyppisen ratkaisun avulla täydentämällä tyhjiä kohtia valmiille pohjille. (Jacobson 2006, 70 - 72.)

5.5.4 Excel

Excel on hyvä käsiteltäessä taulukkorakenteista tietoa, mutta se ei sovellu isojen tekstimassojen hallintaan yhtä hyvin kuin Word. Kaikilla Excelin versiolla pystyy kuitenkin Wordin tyyliin avaamaan ja tallentamaan XML-tietoa, jota voi käyttää esimerkiksi Flashissa. Excel tallentaa oletusarvoisesti XML-dokumentit käyttäen SpreadsheetML-muotoa. Se muistuttaa WordprocessinML:n kaltaista ihmisten kannalta hankalasti luettavaa ja raskasta koodia. Tiedon käyttäminen esimerkiksi Flashissa olisi työläs prosessi. (Jacobson 2006, 73 - 75.)

Skeemoja voidaan hyödyntää Excelissä samalla tavalla kuin Wordissa ja niiden avulla voidaan huomattavasti yksinkertaistaa koodia. Toisin kuin Word, Excel luo jokaiselle avattavalle XML-dokumentille oman skeeman, mikäli sellaista ei ole jo määritelty. Samalla Excel luo aina XML-dokumenttikartan, sitä käytetään luomaan yhteyksiä Excelin solujen ja skeemassa määriteltyjen elementtien välille. Dokumenttikartta kertoo mikä solun kenttä kuuluu millekin XML-elementille. Sitä hyödynnetään aina XML-tiedostoja tuotaessa Exceliin tai vietäessä Excelistä. (Using XML in Excel 2003.)

Oheinen kuva 11 havainnollistuu selkeästi, miten eri tiedostomuodot toimivat keskenään työskenneltäessä XML-dokumenttien kanssa Excelissä. Vaiheessa 1 Exceldokumentille kerrotaan mitä skeema-tiedostoa (.XSD) käytetään määrittelemään XML-rakennetta. Vaiheessa 2 luodaan dokumenttikartta ja määritellään, mihin soluun tai XML-listaan mikäkin skeema elementti viittaa. Vaiheessa 3 haetaan tuotava XML-dokumentti ja sijoitellaan sisältö soluihin aiemmin tehdyn kartoituksen mukaan. Vaiheessa 4 voidaan tietoa muokata halutulla tavalla, lisäämällä, poistamalla tai siirtämällä. Lopuksi tieto haetaan kartoitetuista soluista, tarkistetaan oikeellisuus vertaamalla sitä XSD-tiedostoon ja tallennetaan XML-tiedostoon. (Using XML in Excel 2003.)

KUVA 11: XML datan käyttö Excelissä (Using XML in Excel 2003.)

Excel-dokumenttien rakenne on suoraviivainen, tietoa käsitellään riveinä ja sarakkeina. On muistettava, että XML-tiedon tulee olla kuvattavissa taulukkomaisesti, eikä esimerkiksi sisäkkäisiä elementtejä ole mahdollista käyttää. Seuraavan XML-koodin voisi kuvata Wordissa, muttei Excelissä:

```
<otsikko>
  Kirjan on kirjoittanut
  <kirjailija>Sas Jacobson</kirjailija>
</otsikko>
```

Excelissä työskenneltäessä pienin mahdollinen yksikkö on yksi solu, joka ei voi sisältää toisia soluja. Ohjelma osaisi näyttää esimerkin XML-datasta vain otsikkoelementin alla olevan kirjailija-elementin ja sisällön ”Sas Jacobson”, teksti ”Kirjan on kirjoittanut” jäisi kokonaan huomioimatta. (Jacobson 2006, 75 - 80.)

5.5.5 Access

Access tarjoaa Excelin ja Wordin tapaan mahdollisuuden kerätä valtavan määrän XML-pohjaista dataa, tuoda sitä muokattavaksi ohjelmaan ja tallentaa eteenpäin mahdollista jatkokäsittelyä varten. Accessiin pystyy tuomaan XML-muotoista dataa helposti eri tietokannoista tai tekstitiedostoista, riittää että XML noudattaa oikeaoppista rakennetta. (Using XML and Access, 2003.)

Access käsittelee ja tallentaa XML-tiedostoja aivan eri tavalla kuin Word ja Excel. Ohjelma ei käytä mitään valmista WordprocessinML:n kaltaista monimutkaista rakennetta, vaan XML-dokumentti rakennetaan suoraan käyttäen taulukoiden ja kenttien nimiä. (Jacobson 2006, 80 - 86.)

Access luo ja nimeää taulukot sellaisten elementtien mukaan, joilla on lapsielementtejä. Kentät ja niiden nimet määräytyvät lapsielementtien mukaan, varsinainen data haetaan näiden elementtien sisällöstä. Juurielementtiä ei huomioida taulukoita luotaessa, ja sama koskee oletusarvoisesti myös elementtien mahdollisia attribuutteja. Tiedon tallennuksessa noudatetaan myös samoja periaatteita. (Using XML and Access, 2003.)

Mikäli ei haluta käyttää taulukoiden kenttien oletusnimiä, voidaan ongelma kiertää tekemällä ensiksi kysely ja tallentamalla se XML-muotoon. Accessin tuottama XML-koodi on huomattavasti selkeämpää kuin Wordin tai Excelin oletuksen tuottama koodi. Luotuja XML-tiedostoja voidaan helposti käyttää hyödyksi omissa Flash-sovelluksissa. (Jacobson 2006, 80 - 86.)

Jacobsonin mukaan Accessin hyviä puolia on se, että voidaan työskennellä relaatiollisen tiedon kanssa ja pystytään hakujen avulla rajaamaan tulostettavan XML-tiedon määrää. Monimutkaisten relaatioiden siirtäminen XML-muotoon on kuitenkin Jacobsonin mukaan haastavaa ja esimerkiksi Accessista tutuille ensisijaisille avaimille ei ole suoraa vastinetta XML:n siirrettäessä. (Jacobson 2006, 291.)

5.6 Server-side files (Palvelinpuolen sivut)

Yksi yleisimmästä webissä käytetyistä tavoista kerätä tietoa on käyttää siihen HTML-lomakkeita. Yhdistämällä lomakkeisiin palvelinpuolen sivujen toiminnallisuus on mahdollista toteuttaa sivu, jonka kautta voidaan luoda ja muokata XML-dokumentteja. Lomakkeiden kenttien tieto lähetetään palvelinpuolen sivulle, joka tallentaa sen suunnitellulla tavalla XML-muotoon. XML-tiedostoja voidaan käyttää hyödyksi myös itse lomakkeiden dynaamisessa muodostamisessa. Lomakepohja rakennetaan XML-dokumentin tietojen perusteella, jossa voi olla ohjeita siihen minkä nimisiä lomakekenttiä luodaan ja minkälaista tietoa niihin saa syöttää. (Mooney 2004.)

5.7 FLASH

5.7.1 Uuden XML-datan luominen

Flashin avulla on mahdollista luoda XML-sisältöä ilman, että sitä ensin tuodaan jostain ulkopuolisesta tiedostosta. Tämä voi olla hyödyllistä, mikäli Flash-esityksestä halutaan lähettää XML-muotoista tietoa ulospäin toisen järjestelmän käsiteltäväksi ja tallennettavaksi. Yksinkertaisin tapa muodostaa XML-muotoista tietoa on tehdä se kirjoittamalla suoraan XML-tägejä ja lisäämällä niitä luotuun XML-olioon. Tämä on varsin yksinkertainen tapa luoda XML-muotoista dataa. (Jacobson 2006, 175 - 177.)

```
var myXML:XML = new XML("<login><name>Sas</name></login>");
```

Toinen vaihtoehto on hyödyntää enemmän ActionScriptiä ja käyttää XML-luokan metodeja: createElement, createTextNode, appendChild, insertBefore, cloneNode ja removeNode. Näitä metodeja hyödyntäen on mahdollista yksi osa kerrallaan rakentaa täydellinen XML-rakenne ja lisätä sinne halutut tekstielementit. Jälkimmäistä tapaa käytettäessä syntyy huomattavasti enemmän koodia, mutta tarjoaa myös mahdollisuuden muokata jo olemassa olevaa XML-rakennetta. Luokan metodeja käyttämällä voidaan tietylle elementille lisätä myöhemmin esimerkiksi uusia attribuutteja ja antaa niille arvoja. (Jacobson 2006, 177 - 182.)

5.7.2 XML-sisällön muokkaaminen Flashilla

Sen lisäksi, että Flashilla voidaan luoda kokonaan uusia XML-dokumenttipuita, on tärkeää, että pystytään muokkaamaan jo olemassa olevaa sisältöä. Monesti on tarvetta tarjota käyttäjälle mahdollisuus muokata tekstikenttien sisältöä ja tallentamaa se. Onkin varsin tyypillistä, että Flashiin avataan XML-tiedosto, jonka sisältö ladataan tekstikenttiin. Käyttäjä pystyy lomakemaiseen tapaan muokkaamaan niitä ja lähettämään muutokset tallennettavaksi. (Jacobson 2006, 187.)

Flashin avulla voidaan muokata avattua dokumenttipuun rakennetta käyttämällä hyödyksi muun muassa XML-luokan cloneNode ja removeNode metodeja. Tyypillisiä muokkaustoimenpiteitä on olemassa olevien arvojen muokkaaminen, elementtien kopioiminen ja poistaminen sekä elementtirakenteen uudelleen järjestäminen. Vaikka Flash mahdollistaakin dokumenttipuun muokkaamisen, on syytä huomata, ettei se voi kuitenkaan tallentaa muutoksia alkuperäiseen XML-tiedostoon. Mikäli muutoksia ei hyödynnetä pelkästään Flash-ohjelman aikana, ne täytyy lähettää eteenpäin palvelinpuolen sivun käsiteltäväksi. Ainoastaan niillä on oikeudet tehdä muutoksia varsinaisiin XML-tiedostoihin. (Jacobson 2006, 188.)

XML-muotoista tietoa voidaan lähettää ulospäin Flashista käyttämällä XML-luokan send- tai sendAndLoad-metodeja. Kummatkin lähettävät tietoa samalla tavalla, mutta jälkimmäinen myös vastaanottaa vastauksen palvelinpuolen sivulta. Vastaanottava sivu voidaan toteuttaa vapaasti halutulla ohjelmointikielellä. (Jacobson 2006, 197 -199.)

Flashin avulla on suhteellisen helppo muokata XML-sisältöä. Huonona puolena on se, ettei Flash pysty vertaamaan luotavaa sisältöä valmiiseen skeemaan tai DTD:hen. Ei ole olemassa suoraan mitään ratkaisua, jolla voitaisiin tarkistaa, että lähetettävä tieto on oikein muodostettua. Jacobsonin mukaan ongelman voi kuitenkin kiertää tekemällä tarkistus palvelinpuolen tiedostossa, joka käsittelee mahdolliset päivitykset. (Jacobson 2006, 208 - 209.)

5.7.3 Data-komponenttien hyödyntäminen

XML-luokan tarjoamien menetelmien lisäksi XML-muotoisen datan prosessointi, luominen ja muokkaaminen onnistuvat myös valmiiden datakomponenttien avulla. Periaatteet ovat tismalleen samat, mutta kuten luvussa 4.6 näytettiin, datakomponenttien avulla voidaan toteuttaa samoja toiminnallisuuksia tarvitsematta osata kirjoittaa riviäkään ActionScriptiä. Jacobsonin mukaan se voi hyvinkin nopeuttaa kehitysprosessia monissa tapauksissa. (Jacobson 2006, 325 - 328.)

XMLConnector-komponentti hallitsee tiedon siirtämisen kumpaankin suuntaan ja sitä voidaan käyttää XML-luokan send-metodin tapaan, lähettämään XML-dokumenttipuun sisältö palvelinsivun työstettäväksi ja tallennettavaksi. Mikäli komponenteilla halutaan rakentaa sovellus, jolla voi päivittää olemassa olevaa XML-dataa, joudutaan käyttämään aiemmin esiteltyjen komponenttien lisäksi DataSet- ja XUpdateResolver-komponentteja. (Jacobson 2006, 325 - 328.)

DataSet-komponenttia tarvitaan seuraamaan UI-komponenteissa tapahtuvia muutoksia. Komponentti pystyy sen jälkeen luomaan listan kaikista muutoksista ja lähettämään sen XUpdateResolver-komponentille. Tämä muuntaa listan muutoksista

lausunnoiksi, joita palvelinpuolen-sivu pystyy ymmärtämään ja käsittelemään. XMLConnector-komponentti lähettää tiedot palvelimelle xupdatePacket-muodossa ja palauttaa mahdolliset tulokset xupdateResults-muodossa. (Jacobson 2006, 344 - 348.)

Palvelinpuolen sivun tulisi aina päivitykset käsiteltyään lähettää Flash-sovellukselle takaisin paketti, joka sisältää mahdolliset virheilmoitukset tai päivitysoperaation myötä syntyneet uudet muutokset. Paketti tulisi lähettää aina, vaikka päivitysten käsittely ei onnistuisi tai ei olisi mitään uusia tuloksia mitä palauttaa. Tämä täytyy ottaa huomioon toteutettaessa palvelinpuolen sivua. (Server-side requirements for resolving XML data, 2006.)

XUpdateResolverista on syytä huomata, että se käyttää XUpdate-lausuntoja kuvaamaan tehtyjä muutoksia. XUpdate on standardi, jota käytetään yleisesti kuvamaan XML-dokumenttiin tehtäviä muutoksia ja sille löytyy tuki muun muassa lukuisista XML-perusteisista tietokannoista. Lausunnoissa kuvataan DataSet-komponentin huomaamat tiedon lisäykset, muokkaukset ja poistot. Alla esimerkki XUpdateResolverin tuottamista XUpdate-lausunnoista. (Resolving XML data with the XUpdateResolver component, 2006.)

```
<?xml version="1.0"?>
<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate">
   <xupdate:insert-after select="/addresses/address[1]" >
      <xupdate:element name="address">
         <xupdate:attribute name="id">2</xupdate:attribute>
         <fullname>Lars Martin</fullname>
         <born day='2' month='12' year='1974'/>
         <town>Leizig</town>
         <country>Germany</country>
      </xupdate:element>
   </xupdate:insert-after>
</xupdate:modifications>
```

6 CASE: Yritysesittelysovellus Galvatek Oy:lle

6.1 Kohdeyrityksen esittely

Case toteutettiin Lahtelaiselle Galvatek Oy:lle, joka on ammattilainen pinta- ja vedenkäsittelylaitoksien suunnittelussa ja toimituksessa. Galvatek toimittaa myös laitoksiin liittyviä komponentteja ja huoltopalveluja. Yrityksellä on takanaan yli 30 vuoden kokemus alan kansainvälisiltä markkinoilta, ja se on toimittanut yli 500 pinta- tai vedenkäsittelylaitosta 25:een eri maahan. Yritys on toimittanut ratkaisuja metalli-, konepaja- ja ilmailuteollisuuden tarpeisiin ja sekä jätevedenkäsittelyyn. Galvatekin asiakkaita ovat olleet esimerkiksi sellaiset suuret yhtiöt kuten Nokia, Oras, Finnair, Volvo, Saab ja SEASL. (Galvatek Oy 2006.)

6.2 Casen tarkoitus ja tavoitteet

Ennen casen aloittamista Galvatek käytti yrityksen ja sen tuotteiden esittelyyn tavallisia PowerPoint-esityksiä. Niiden sisällön hallinnoiminen ja yhtenäisen ulkonäön saavuttaminen oli todettu hankalaksi. Yritys kaipasi tähän ongelmaan ratkaisuksi uutta, helpommin käytettävää sovellusta, jonka toivottiin mahdollistavan myös aiempaan verrattuna näyttävämmät esitykset. Alettiin kehitellä ratkaisua, joka pystyisi ajan myötä korvaamaan PowerPointin käytön Galvatekin esittelytilaisuuksissa.

Yritysesittelysovellusta käytettäisiin uuden sähköisen yritysesittelymateriaalin luomisessa ja näyttämisessä asiakkaille. Esityksissä olisi tekstin lisäksi myös valokuvia, laitosten 3D-visualisointeja ja -animaatioita, joilla yhdessä pyrittäisiin antamaan vahva kuva yrityksen osaamisesta suunnittelussa ja automaatiossa.

Esityksestä toivottiin löytyvän polkuja, jotta voitaisiin esitystilanteen mukaan näyttää kohdennetumpaa sisältöä sen hetkisille katsojille. Käytännössä esimerkiksi vesipuolen asiakkaita varten pitäisi löytyä oma esitys, joka keskittyisi esittelemään vedenkäsittelylaitoksia ja käytettyjä prosesseja. Myös sisällön eri kieliversiot tulisi huomioida suunnittelussa.

Tieto on jatkuvasti muuttuvaa, ja suunnittelussa piti ottaa huomioon, että sisällön päivittäminen onnistuisi vaivattomasti. Toteutustekniikka tulisi valita myös niin, että sovellus toimisi myös hieman vanhemmilla tietokoneilla. Esitykset pidettäisiin tietokoneen näytöltä tai videotykillä, ja niiden tulisi olla siirrettävissä helposti koneesta toiseen esimerkiksi muistitikun tai CD:n avulla. Kehitettävälle yritysesittelysovellukselle annettiin nimeksi lyhyesti YeS.

6.3 Työkalut ja valitut tekniikat

Tuotettava yritysesittelymateriaali päätettiin toteuttaa käyttäen hyödyksi Macromedian Flash 8 Professionalia. Flash oli sopiva valinta, sillä se mahdollistaa ensinnäkin visuaalisesti näyttävän ulkoasun toteuttamisen ja toisekseen tukee lukuisia eri vaihtoehtoja hyödyntää ulkoista sisältöä. Koska esityksen tuli toimia suoraan CD:ltä, ei sisältöä voitu sijoittaa tietokantoihin tai käyttää esimerkiksi WWW-sovelluspalveluita tiedon lähteenä. Jäljelle jäivät vaihtoehdoiksi käyttää tavallisia tekstitiedostoja tai XML-tiedostoja. Jälkimmäinen havaittiin huomattavasti järkevämmäksi ratkaisuksi, koska sillä voitiin kuvata myös tiedon rakennetta ja sen käsittely Flashissa oli helpompaa kuin suurien muuttuja-arvopari joukkojen.

XML-tiedostojen kanssa työskenneltiin käyttäen Macromedian Dreamweaver 8 -ohjelmaa, joka osoittautui varsin kelvolliseksi XML-editoriksi. Esittelymateriaalin graafisen ulkoasun suunnittelussa ja sisällön editoimissa hyödynnettiin Adoben Photoshop- ja Illustrator CS2 -ohjelmia. Havaittiin, että grafiikan siirtäminen Illustratorista Flashiin oli varsin helppoa leikkaa ja liimaa -tekniikalla. Illustrator-tiedostot voitiin tallentaa myös suoraan Flashin swf-muotoon ja näin luotuja erillisiä kaavioita voitiin käyttää kuvatiedostojen tapaan yritysesittelysovelluksen sisältönä.

6.4 Personoidut esitykset

Yksi casen keskeisiä tavoitteita oli luoda järjestelmä, joka mahdollistaisi kohdennetun sisällön esittämisen asiakkaille. Tehtävä oli sikäli helppo, että järjestelmän ei ollut tarpeellista automaattisesti tunnistaa käyttäjää tai kohdeyleisöä. Tuotetta käyttäisi pääasiassa Galvatekin henkilökunta ja he pystyisivät tilanteen mukaan itse valitsemaan halutun kohdennetun esityksen.

Galvatek Oy:llä oli selkeä kuva heidän asiakaskunnastaan ja tiedoista, joita he kullekin ryhmälle haluaisivat esitellä. Esitysmateriaalista tuli löytyä yleinen firman taloudellisia tietoja esittelevä esitys, joka olisi suunnattu tyypillisesti asiakkaan hallituksen jäsenille. Lisäksi tuli löytyä erityisesti vedenkäsittelyn, pintakäsittelyn, huollon ja automaation ratkaisuihin painottuvia esityksiä.

Valmista asiakastuntemusta voitiin hyödyntää suoraan sisällön personoinnissa, eikä uusia asiakasprofiileja tarvinnut muodostaa. Esitellyn kuuden pääryhmän lisäksi haluttiin, että pystyttäisiin luomaan ja esittämään vielä tarkemmin yksittäistä asiakasta koskevaa sisältöä. Haasteena oli siis toteuttaa järjestelmä, jolla pystyttäisiin helposti hallinnoimaan ja esittämään kaikkia näitä eri personoituja esityksiä.

6.5 Sisällön hallinta

6.5.1 Tiedostettuja haasteita

Kaikissa esityksissä tulisi olemaan osittain samoja tekstejä ja kuvia, joten täytyi miettiä, kuinka samaa sisältöä voitaisiin hyödyntää useissa eri paikassa. Varsinkaan valokuva- ja videomateriaalia ei ollut järkevää lähteä kopioimaan jokaiseen esitykseen, koska se olisi vienyt turhaa levytilaa.

Teksti- ja valokuvamateriaalin lisäksi esityksissä tultaisiin käyttämään myös erilaisia kaavioita, animaatiota ja videota. Eri sisältömuodot täytyi säilyttää muokattavuuden takia omina tiedostoinaan ja miettiä rakenne, jolla tiedostot tallennettaisiin järjestelmällisesti niin, että niiden etsiminen, muokkaaminen ja käyttäminen olisi mahdollisimman helppoa.

6.5.2 XML-tiedostot

Esitysten sisällön tallennusmuodoksi valittiin XML-tiedostot. Tekniikka mahdollisti helpon tavan eriyttää sisältö esityksen ulkoasusta ja tehdä sisällön päivittämisestä siten yksinkertaista. Jokaisesta omasta esityksestä luotiin oma XML-tiedosto, joka sisälsi kaiken esityksessä tarvittavan tiedon. Esityksen tekstit tallennettiin suoraan näihin XML-tiedostoihin, kuvista ja muista graafisista elementeistä tallennettiin vain tiedoston linkkitieto.

Galvatekin toiveiden mukaan esityksen tuli koostua PowerPoint-tyylisesti yksittäisistä dioista. Ne saattoivat koostua erityyppisestä sisällöstä, yhdistäen esimerkiksi tekstiä, kuvaa ja animaatiota. XML-rakenne suunniteltiin tämän periaatteen mukaisesti. Seuraava on esimerkkipätkä muodostetusta XML-tiedostosta:

```
<slideShow name="General Presentation" showHeader="SURFACE AND WA-
TER TREATMENT TECHNOLOGY" showHeader2="" effects="off">
      <slide template="t3">
             <header>AVIATION INDUSTRY</header>
             <text>
              <p class="headline">Etching plants</p><br />
              <ul>
               <li>Engine manufacture</li>
               <li>Engine overhaul</li>
               <li>Airline engine service</li>
              </ul>
             </text>
             <images>
               <link>images/kuva4.jpg</link>
               <link>images/kuva17.jpg</link>
             </images>
             <video>
             </video>
      </slide>
</slideShow>
```

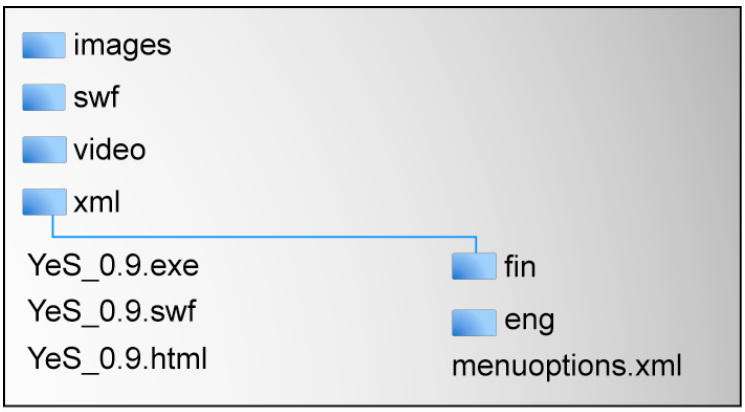
Yhtä esitystä koskevat tiedot ovat koottuna juurielementin <slideShow> attribuutteihin, joista tärkeimmät ovat esityksen nimi ja sen pääotsikko. Yhtä diaa vastaava sisältö löytyy <slide>-elementin sisältä. Tekstisisältö on sijoitettuna <text>-elementin alle XHTML-muodossa. XHTML-siksi, että sen avulla oli helppo ohjata sisällön muotoilua Flashissa. Video- ja kuvasisällön linkit on kuvattu sijoitettu <link>-elementteihin.

Esityksiä kuvaavia XML-tiedostoja tuli pakostikin olemaan lukuisia ja jonnekin oli tallennettava tieto kaikista esityksistä. Koska sovelluksen tuli toimia suoraan myös CD:ltä, ei ratkaisuna voitu käyttää suoraan esimerkiksi palvelinpuolen sivuja, joka olisi tulostanut kaikkien tietyssä kansiossa olevien XML-tiedostojen nimet ja välittänyt tiedon Flashiin. Tämän takia jouduttiin luomaan yksi kaikkia esityksiä kuvaava XML-tiedosto (menuOptions.xml), johon tallennettiin kaikkien esitysten nimet ja linkki vastaavan XML-tiedostoon. Yritysesittelysovelluksen alkuun luotiin kuvan 12 mukainen valikko, josta käyttäjä pystyy valitsemaan halutun kielen ja valmiiksi personoidun esityksen.

KUVA 12. YeS alkuvalikko, josta valitaan haluttu esitys

6.5.3 Tiedostorakenne

Valitun tekniikan johdosta tiedostoja tulisi olemaan paljon, eikä kaikkia olisi ollut järkevää tallentaa samaan hakemistoon. Ratkaisuna oli luoda kuvan 13 mukainen hakemistorakenne, jossa kunkin tyyppiset eri tiedostot tallennetaan omiin hakemistoihinsa. Jokaista eri kieltä varten on oma kansio XML-kansion alla ja niitä luodaan lisää sitä mukaan, kun tarvitaan erikielisiä esityksiä. Tiedosto YeS_0.9.exe on julkaistu muoto Flash-esityksestä ja sitä käytetään käynnistämään varsinainen yritysesittelysovellus YeS.

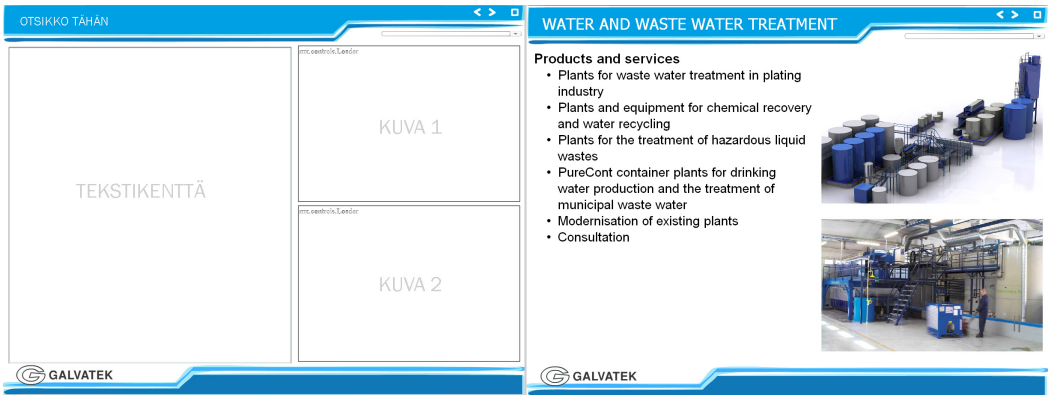


KUVA 13. Tiedostoja varten luotu hakemistorakanne

6.6 Flash-toteutus

6.6.1 Esittely

Koska esitysten täytyi olla helposti muokattavissa, ei tekstiä tai kuvia ollut järkevää sisällyttää Flash-esitykseen. Päivitykset olisivat vaatineet aina Flashin käyttöä ja esityksen uudelleenjulkaisua. Siksi päädyttiin rakentamaan yksi Flash-esitys, joka sisältää joukon erilaisia tyhjiä pohjia, joiden älykkäisiin kenttiin pystytään esityksen aikana lataamaan tekstiä, kuvaa, videota, animaatiota tai yhdistelmää edellisistä.



KUVA 14. Vasemmalla tyhjä Flash pohja, oikealla valmis esitys samalla pohjalla

6.6.2 Dynaamisen tiedon tuonti Flashiin

Luvussa 4 Dynamiikka esiteltiin lukuisia eri keinoja tuoda tietoa dynaamisesti Flashiin. Yritysesittelysovellusta luotaessa päädyttiin käyttämään XML-luokan tarjoamia keinoja, koska se havaittiin kaikkein selkeimmäksi ja joustavimmaksi tekniikaksi. Samat toiminnallisuudet olisi voitu toteuttaa myös käyttämällä valmiita datakomponentteja, mutta työ haluttiin toteuttaa tekniikalla, joka olisi mahdollista toteuttaa myös Flashin Standard-versioilla.

XML-tiedostojen sisältö ladattiin Flashiin XML-luokan load-metodia hyödyntäen. Muodostetusta XML-dokumenttipuusta poimittiin halutut tiedot ja sijoitettiin ne Flashin paremmin ymmärtämään taulukkomuotoon. Taulukoista tietoa oli helppo käyttää Flashissa normaalien muuttujien tapaan, ja sijoittaa arvoja UI-komponentteihin. Kaikki muu, tekstiä lukuun ottamatta, ladattiin Flashiin käyttäen valmista Loader-komponenttia, jolle tarvitsi vain antaa oikean ladattavan tiedoston nimi taulukon tietojen perusteella. Loader-komponentin avulla kuvien skaalaaminen oikeaan kokoon oli helppoa, eikä itse lataamisprosessia tarvinnut koodata käsin ActionScriptillä.

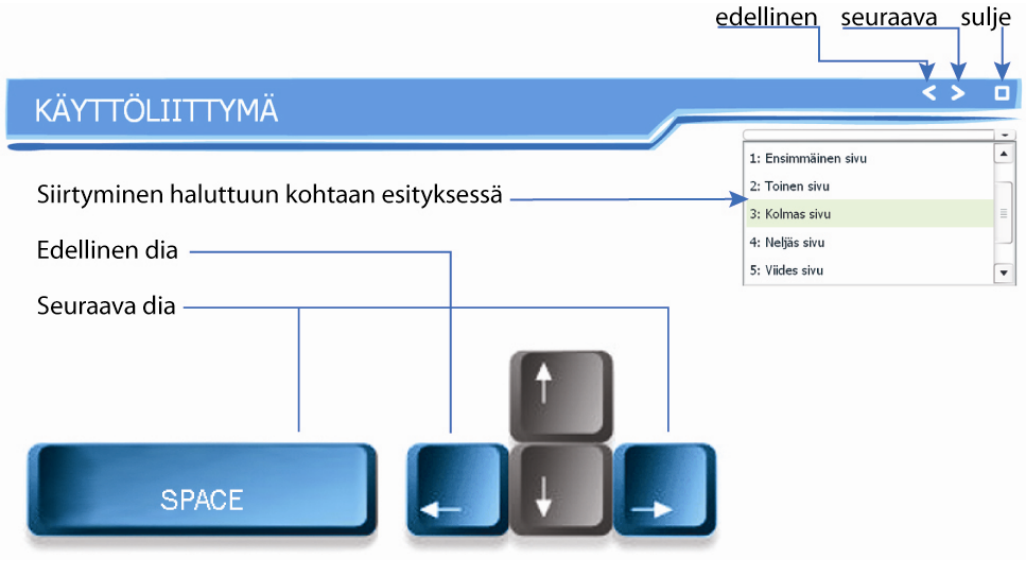
6.6.3 Muotoilut

Tekstien esittämiseen käytettyjen UI-komponenttien oletusfontti on ”_sans” ja koko 10. Oletusmuotoiluja käyttämällä esityksistä olisi tullut kovin mitättömän näköisiä, ja tekstien ulkoasua haluttiin muokata. Muotoilu hoidettiin tyylimäärittelyiden avulla käyttämällä yksinkertaista setStyle()-metodia. Osa muotoiluista, kuten fontin määrittely, tehtiin ”_global”-tasolla, minkä ansiosta muutos kosketti kaikkia komponentteja.

Varsinaisen päätekstikentän, eli TextArea-komponentin muotoilu hoidettiin HTML:n ja erillisen CSS-tyylisivun avulla. HTML:n avulla oli helppo lisätä tekstin joukkoon muun muassa listoja ja muita webistä tuttuja muotoiluja. Erillisen CSS-tiedoston ansiosta tekstikenttien sisältö oli helppo hallinnoida Flashin ulkopuolelta. Ratkaisun ansiosta voitaisiin esimerkiksi vaihtaa kaikkien esitysten tekstin väri muokkaamalla ainoastaan yhtä riviä CSS-tiedostossa. Muunnoksia voidaan hallinnoida siis täysin ilman Flashia.

6.6.4 Käyttöliittymä

Galvatekin yhtenä vaatimuksena yritysesittelysovelluksen suhteen oli, että se olisi helposti käytettävää. Yritys oli aikaisemmin käyttänyt paljon Microsoftin PowerPoint-ohjelmaa, joten navigaatio päätettiin toteuttaa hyvin samantyylisesti hyödyntäen nuolinäppäimiä ja välilyöntiä siirryttäessä eri diojen välillä. Nuolinäppäinten lisäksi navigaatio onnistuu hiiren ja esityksessä olevin painikkeiden avulla. Esitykseen toteutettiin myös alavetovalikko, jonka avulla voidaan hypätä mihin tahansa kohtaan esityksessä.



KUVA 15. Yritysesittelysovelluksen käyttöliittymä ja näppäinkartta

6.6.5 Flash-tehosteet

Flash mahdollistaa aivan erilaisen tavan tuottaa interaktiivisuutta ja näyttävyyttä esityksiin kuin esimerkiksi PowerPoint. Flashin avulla esityksiin voidaan lisätä esimerkiksi animoituja virtauskaavioita Galvatekin linjan toiminnasta. Flashin tarjoamia mahdollisuuksia havainnollistettiin lisäämällä kuvien yhteyteen mahdollisuus suurentaa ne koko näytön kokoiseksi. Työ painottui pääasiassa tutkimaan personoidun ulkoisen tiedon käyttöä Flashissa, eikä graafisia tehosteita hyödynnetty sovelluksen ensimmäisessä versiossa sen enempää.

6.6.6 Julkaiseminen

Flashilla toteutettu .fla-tiedosto julkaistiin lopuksi .html-, .swf- ja .exe-muodoissa. Jälkimmäisen havaittiin olevan kaikista helpoin käyttää, koska se toimi kaikilla koneilla jopa ilman Flash Playerin asennusta. Html-sivuun ladattava swf-tiedosto vaatii toimiakseen, että koneelta löytyy Flash Player. Kopioimalla koko kansiorakenne muistitikulle tai polttamalle se CD:lle, saatiin yritysesittelysovellus siirrettyä ja esitettyä sisältö onnistuneesti useilla eri kannettavilla ja pöytäkoneilla.

6.7 Valmis yritysesittelysovellus

Flash-sovellus lataa tekstisisällön esityksen aikana XML-tiedostoista ja hyödyntää samalla tavalla ulkoisia JPG-kuva- ja SWF-videotiedostoja. Kuvia ja tekstiä pystyy helposti vaihtamaan vain korvaamalla aiempia tiedostoja, eikä muokkaaminen vaadi Flashin käyttöä ollenkaan. Kaikkien esitysten ulkoasua voidaan hallinnoida yhden Flash-tiedoston kautta, ja siihen tehdyt muutokset vaikuttavat kaikkiin sovelluksen hallinnoimiin esityksiin. Ominaisuus takaa, että kaikki yrityksen esitykset noudattavat samaa yhtenäistä graafista ulkoasua.

Sisällön profilointi on toteutettu XML-tiedostojen avulla, käytännössä jokaista profiilia vastaa oma esitys ja oma XML-tiedosto. Toteutustapa on varsin yksinkertainen, mutta se oli nopea toteuttaa ja se mahdollistaa kohdennettujen esitysten luomisen ja näyttämisen. Liitteessä 1 on esimerkkikuvia sovelluksella toteutetusta Galvatekin yleisesittelystä.

Täysin Flashilla toteutettu yritysesittelysovellus on tiedostokooltaan varsin pieni: swf-tiedosto vie vain reilun 100Kt ja exe-muotoinenkin, Playerin sisältävä tiedosto, vie vain vajaat 2000Kt. Itse työstettävä fla-tiedosto vie UI-komponenttien takia reilusti yli 6000Kt, mutta kuten huomataan, koko pienenee merkittävästi julkaisuvaiheessa. Varsinaista tallennuskapasiteettiä vievät sisällön kuva- ja videotiedostot, joiden määrä lopulta ratkaisee lopulliseen tilantarpeen. XML-tiedostoissa sijaitseva tekstisisältö menee myös erittäin pieneen tilaan, keskivertokoko yhtä esitystä kuvaavalle tiedostolle on 5-20Kt.

6.8 Jatkokehitysajatuksia

Nykyisellään sisällön muokkaaminen tapahtuu XML-tiedostoja editoimalla, mikä voi asiaan perehtymättömälle olla liian monimutkainen prosessi. Seuraava tärkeä askel olisikin luoda erillinen, sisällön päivittämistä varten tarkoitettu sovellus. Sen avulla päivitykset voitaisiin hoitaa helposti ja kootusti yhdessä paikassa, eikä päivittäjän tarvitsisi ymmärtää mitään XML-tiedostoita. Saman sovelluksen kautta voitaisiin hoitaa myös valokuvien ja muun materiaalin lataaminen oikeisiin hakemistoihin.

Päivitysohjelma voitaisiin toteuttaa esimerkiksi Flashin ja PHP:n yhdistelmänä. Flash ei itse pysty tekemään muutoksia fyysisiin tiedostoihin, minkä takia apuna käytettäisiin palvelimella olevaa PHP-sivua, joka hoitaisi XML-tiedostojen käsittelyn Flashilta tulevien ohjeiden mukaisesti. Tiedon päivittämisen lisäksi sovellukseen voitaisiin toteuttaa myös mahdollisuus luoda kokonaan uusia esityksiä ja hyödyntää niissä materiaalia aiemmista esityksistä. Flash käyttöliittymä olisi varsin yksinkertainen toteuttaa valmiita komponentteja hyödyntäen, jolloin tiedostojen päivittämisessä käytettäisiin kuvan 8 mukaista tekniikkaa.

Esityssovelluksen visuaalisuutta voitaisiin helposti kehittää ja lisätä esityksiin lisää toiminnallisuuksia ja animaatiota, jotka hyvin toteutettuina olisivat kiinnostavia eivätkä silti veisi huomiota itse sisällöltä. Esimerkiksi PowerPointista tutut siirtymät voitaisiin toteuttaa paljon persoonallisemmin Flashin avulla. Mahdollisuudet ovat lähes rajoittamattomat. Tässä case:ssa ei ollut tarvetta julkaista sisältöä mobiilikanaviin tai Internettiin, mutta luotua sovellusta voitaisiin kehittää toimimaan myös kyseisissä ympäristöissä. Flash-esityksiä on mahdollista katsoa Internetselaimella ja esimerkiksi Nokian Series 60 puhelimilla. Siirrettäessä sisältöä verkon yli täytyi-

si kiinnittää lisää huomiota kuvien ja videoiden pakkaamisen, jotta nopeus saataisiin pysymään kohtalaisena. Mobiilikanavissa haasteina olisi kohdelaitteen näytön pieni koko sekä grafiikan ja kuvien sovittaminen sille sopivaksi.

6.9 Casen arviointia

Yritysesittelysovelluksen ensimmäinen versio saatiin ajallaan valmiiksi. Sitä pystyy käyttämään kohdeyrityksen esittelemiseen, ja se tarjoaa pohjan luoda kohdennettuja esityksiä eri asiakkaille. Case on mielestäni onnistunut ja täyttää tärkeimmät sille asetuista tavoitteista. Yritysesittelysovelluksen ensimmäinen versio mahdollistaa valmiiden dynaamisten esitysten pitämisen Galvatekin asiakkaille.

Galvatekin mielestä yksi case-sovelluksen parhaimpia puolia on se, että se mahdollistaa ulkonäöltään yhtenäisten esityksien pitämisen kaikilta yrityksen henkilöiltä. Aiemmin ongelman oli se, että kukin myyntimies esitteli oman näköisiään PowerPointilla luotuja esityksiä. YeS:n avulla kaikilla esityksillä on sama yhtenäinen ammattimaisen näköinen ulkoasu. Mahdollisuus käyttää valmiita kohdennettuja esityksiä koettiin tärkeäksi tavoitteessa viestittää oikeaa tietoa asiakkaalle. Flashilla toteutettu ulkoasu havaittiin kiinnostavammiksi kuin staattiset PowerPoint-kalvot ja esimerkiksi kuvien suurentamisen esityksen aikana koettiin olevan erittäin käytännöllinen lisäetu, koska monesti on tarvetta esitellä kuvista myös yksityiskohtia. Esityksen ulkoasun ja sisällön erottaminen päivitettävyyden helpottamiseksi koettiin toimivaksi ratkaisuksi. XML-tiedostojen muokkaamiseen toivottiin kehitettävän lomaketyyppinen ratkaisu, jossa käyttäjä näkisi vain pelkän sisällön, ei XML- ja XHTML-tageja.

Casen tekeminen oli suhteellisen haastavaa, koska minulla oli hyvin rajoitetusti aikaisempaa kokemusta Flashilla toteutetuista sovelluksista. ActionScriptin opettelu vaati paljon aikaa, mutta tuotti myös tulosta. Casea tehdessä opin valtavasti uusia asioita XML:n käytöstä, Flash 8:sta, komponenteista, ActionScript-ohjelmoinnista ja -animoinnista. Case tarjosi mahdollisuuden hyödyntää monialaista osaamistani ja pääsin ohjelmoinnin taitojen lisäksi käyttämään ja kehittämään myös graafisen suunnittelun ja kuvankäsittelyn taitojani.

7 JOHTOPÄÄTÖKSIÄ

Multimediatuotannossa kannattaa aina miettiä tarkkaan, mikä on kohdeyleisö ja laite, jolla multimediaa esitetään. Esitysten sisältö on hyödyllistä pyrkiä kohdentamaan mahdollisimman tarkkaan kyseistä katsojaa varten ja välttää turhan tiedon esittämistä. Personoinnissa on hyvä hyödyntää mahdollisesti jo olevaa markkinointimateriaalia ja muodostaa valmiita profiileja kuvaamaan eri käyttäjiä tai asiakkaita.

Mikäli haluaa säilyttää sisällön muokattavuuden mahdollisimman hyvänä, kannattaa sisältö pitää aina erillään ulkoasusta. Näin esimerkiksi Flash-esitysten sisällön muokkaaminen onnistuu täysin ilman Flashia. Päivittämisessä voidaan hyödyntää muita helpommin käytettäviä ja kenties täysin ilmaisia ohjelmia. Samaa sisältöä voidaan hyödyntää myös eri kanavissa ja esittää eri laitteilla, eikä alkuperäiseen tietoon tarvitse tehdä mitään muutoksia. Tiedon hallinnointi muuttuu helpommaksi, kun tietoa ei tarvitse muokata moneen eri paikaan. Päivitystehtäviä on myös helpompi jakaa, ja graafikko voi keskittyä esimerkiksi täysin ulkoasun muokkaamiseen, eikä hänen tarvitse huolehtia tekstisisällön ajantasaisuudesta.

Tiedon tallennusmuotona XML on hyvä ratkaisu, sillä se toimii useiden eri järjestelmien kanssa eikä sen käsittelyyn tarvita mitään erikoisia ohjelmia. Vaikka XML on standardi, mikään järjestelmä ei pysty tulkitsemaan sitä ilman kaavaa, joka määrittelee, mitä itse luodut tagit tarkoittavat. Kaava voi olla DTD- tai skeema-tiedosto, mutta se voidaan määritellä myös itse ohjelmaan, joka lukee XML-tiedostoa. Case-osiossa kaavan sijaan Flashille kerrottiin suoraan, kuinka XML-elementit pitäisi tulkita. Tarvittaessa XML-muotoista tietoa voidaan myös muuntaa eri muotoon käyttäen esimerkiksi XSL-kieltä. Muunnoksia tarvitaan, mikäli samaa sisältöä halutaan hyödyntää eri kanavissa ja muodostaa saman tiedon pohjalta eri esityksiä esimerkiksi matkapuhelimia tai varten.

Flash havaittiin varsin monipuoliseksi sovelluskehitystyökaluksi, jolla pystytään luomaan varsin näyttäviä ja dynaamisia sovelluksia ja hyödyntämään niissä ulkoista tietoa. Haettava tieto voi olla esimerkiksi tekstitiedostossa, tietokannassa, XML-tiedostossa, uutisvirrassa tai kokonaan toisessa järjestelmässä, josta se lähetetään Flashiin palvelinpuolen sivun kautta käyttäen muuttuja-arvopareja tai XML-muotoa. XML-tiedon kanssa voidaan työskennellä visuaalisesti käyttäen valmiita UI- ja datakomponentteja. Toinen vaihtoehto on työskennellä ActionScriptin kanssa ja hyödyntää XML-luokkaa ja sen metodeja. Komponenttien avulla saa nopeasti luotua monipuolisia Flash-sovelluksia, mutta ne asettavat tiettyjä rajoitteita, minkä takia ohjelmointiin tottuneet käyttävät yleensä mieluummin jälkimmäistä tekniikkaa.

XML-tiedostojen luominen ja muokkaaminen on helppoa, eikä siihen vaadita tekstieditoria kummempaa työkalua. Markkinoilta ja Internetistä löytyy lukuisia kaupallisia ja ilmaisia ohjelmia helpottamaan työtä. Enemmän XML-tiedostojen kanssa

työskentelevän kannattaa ainakin hankkia kunnollinen XML-editori, joka mahdollistaa tiedostojen rakenteen ja kieliopin tarkastamisen. Microsoftin Wordia, Exceliä ja Accesia voidaan joissain tapauksissa hyödyntää XML-tiedostojen käsittelyssä. Jo tutuksi tulleita valmisohjelmia voidaan käyttää siten sisällön muokkaamiseen ja jopa muuntamiseen toiseen muotoon.

Tämä työ kumoaa selkeästi yleisen harhaluulon, että Flash soveltuisi lähinnä web-animaatioiden ja -grafiikan tekemiseen eikä pystyisi keskustelemaan muiden järjestelmien kanssa. Flashin avulla voidaan nimittäin toteuttaa hyvinkin monipuolisia sovelluksia ja tarjota katsojalle uudenlaisia rikkaita elämyksiä sekä personoitua sisältöä. Arkkitehtuurin hyvän skaalautuvuuden takia sisältöä voidaan julkaista myös useisiin eri kanaviin. Esimerkiksi uusi Flash Lite 2.0 takaa, että Flash-sovelluksia tullaan tulevaisuudessa näkemään yhä useammassa mobiililaitteessa, kuten matkapuhelimissa. Yksi Flashin vahvuuksia on se, että sen avulla voidaan toteuttaa visuaalisesti näyttäviä esityksiä ilman, että päivitettävyys tai käytettävyys kärsii. Pikemminkin ne paranevat. Case osoitti, että Flashilla toteutettu esitys päihittää esimerkiksi vanhahtavat staattiset PowerPoint-esitykset niin näyttävyydessään, käytettävyydessään kuin sisällön päivittämisen helppoudessa. Joten mikäli etsitään keinoa tarjota kohdeyleisölle ajantasaista ja personoitua sisältöä visuaalisesti näyttävällä tavalla, on Flash ehdottomasti harkitsemisen arvoinen valinta käytettäväksi työkaluksi.

LÄHTEET

About components in Flash Lite 2.0 [verkkodokumentti]. Macromedia, 2006 [viitattu 7.1.2006] Saatavissa:
http://livedocs.macromedia.com/flashlite/2/main/00000033.html

About skinning components [verkkodokumentti]. Macromedia, 2006 [viitattu 7.1.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00003003.html

About themes [verkkodokumentti]. Macromedia, 2006 [viitattu 7.1.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00003010.html

ATK-sanakirja [verkkodokumentti]. Tietotekniikan Liitto Ry 2005 [viitattu 3.2.2006] Saatavissa: http://www.tt-tori.fi/atk-sanakirja/suomi/su005.htm#586

Customizing Components [verkkodokumentti]. Macromedia, 2006 [viitattu 4.1.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00002991.html

Data Binding [verkkodokumentti]. Macromedia, 2006 [viitattu 2.1.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00000760.html

David, M. Flash Remoting [verkkodokumentti]. Informit.com, 2005a [8.12.2005]. Saatavissa: http://www.informit.com/guides/content.asp?g=flash&seqNum=218

David, M. Getting Dynamic Data into Macromedia Flash [verkkodokumentti]. Informit.com, 2005b [viitattu 9.12.2005].
Saatavissa: http://www.informit.com/articles/article.asp?p=28510&seqNum=1

David, M. Using Flash Variables [verkkodokumentti]. Informit.com, 2005c [viitattu 20.1.2006]. Saatavissa:
http://www.informit.com/guides/content.asp?g=flash&seqNum=170

David, M. Why Structuring Your Data is Always a Good Thing [verkkodokumentti] Informit.com, 2005d [viitattu 22.1.2006]. Saatavissa:
http://www.informit.com/guides/content.asp?g=flash&seqNum=180

David, M. Working with XML - Web Services [verkkodokumentti]. Informit.com, 2005e [viitattu 19.1.2006]. Saatavissa:
http://informit.staging.informit.mttech.com/guides/content.asp?g=flash&seqNum=199

David, M. XMLSocket Server [verkkodokumentti]. Informit.com, 2006 [viitattu 23.2.2006]. Saatavissa: http://informit.staging.informit.mttech.com/guides/content.asp?g=flash&seqNum=213

Duran, J. Exploring the New Features in the Flash Lite 2.0 Preview [verkkodokumentti] Macromedia, 2005 [viitattu 12.3.2006]. Saatavissa: http://www.macromedia.com/devnet/devices/articles/flashlite_v2_preview.html

Extensible Markup Language [verkkodokumentti]. W3C, 2006 [viitattu 5.1.2006). Saatavissa: http://www.w3.org/XML/

Galvatek Oy, 2006 [verkkodokumentti].[viitattu 26.2.2006] Saatavissa: http://www.galvatek.fi

Hiltunen, M. 2002. Mobile user experience, Edita, IT Press, Helsinki.

Jacobson, S. 2006. XML for Flash, Foundation, Friends of ED, Berkeley, CA, Yhdysvallat).

Kalliola, J. Personointi [verkkodokumentti]. TKK Viestintätekniikan laboratorio 2004 [viitattu 29.2.2006]. Saatavissa: http://www.media.hut.fi/~as75108/2004/4-personointi.pdf

Kelly, P. Microsoft Office System and XML: Bringing XML to the Desktop [verkkodokumentti]. Microsoft, 2003 [viitattu 14.3.2006]. Saatavissa: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dno2k3ta/html/odc_xmlinoffice2003_summarydoc.asp

Kennaugh, C. Discover the importance of target audience profiling [verkkodokumentti] Microsoft, 2006 [viitattu 20.2.2006]. Saatavissa: http://office.microsoft.com/en-us/FX011454641033.aspx

Korpela, J. & Linjama, T. 2003, Web-suunnittelu. Docendo Finland Oy, Jyväskylä.

Macromedia Flash Player Statistics[verkkodokumentti]. Macromedia, 2006 [viitattu 28.2.2006] Saatavissa: http://www.macromedia.com/software/player_census/flashplayer/

Majava J. Paha, paha PowerPoint [verkkodokumentti]. Piirtoheitin, 2004 [27.2.2006]. Saatavissa: http://www.valt.helsinki.fi/piirtoheitin/ppt1.htm)

Mischook, Creating Multimedia Content [verkkodokumentti], Killersites, 2006 [10.1.2006]. Saatavissa:

http://www.killersites.com/articles/articles_CreatingMultimediaContent%20.htm

Mobile & Devices [verkkodokumentti]. Macromedia, 2006 [viitattu 10.3.2006]. Saatavissa: http://www.macromedia.com/mobile/

Mooney, A. Sample Application: XML Form Generator [verkkodokumentti]. Aspalliance.com, 2004 [viitattu 15.3.2006]. Saatavissa: http://aspalliance.com/487

Mora, A. The Power of LoadVars Object [verkkodokumentti] Devarticles, 2005 [viitattu 20.1.2006]. Saatavissa: http://www.devarticles.com/c/a/Flash/The-Power-of-LoadVars-Object/

Multimedia [verkkodokumentti]. Wikipedia, 2006 [viitattu 10.1.2006]. Saatavissa: http://fi.wikipedia.org/wiki/multimedia

Nykänen, O. XML 10 kohdan tiivistelmänä [verkkodokumentti]. W3C Suomen toimisto, 2003 [viitattu 6.1.2006] Saatavissa: http://www.w3c.tut.fi/translations/xml/xmlin10pts/

Phillips, A. Another Way to Run Your Flash Application [verkkodokumentti]. MX Devoloper's Journal, 2006 [viitattu 2.1.2006]. Saatavissa: http://mxdj.sys-con.com/read/47124.htm

Refsnes, J. XML Syntax [verkkodokumentti]. Xmlfiles.com, 2006 [viitattu 16.3.2006]. Saatavissa: http://www.xmlfiles.com/xml/xml_syntax.asp

Resolving XML data with the XUpdateResolver component [verkkodokumentti]. Macromedia, 2006 [viitattu 10.3.2006] Saatavissa: http://livedocs.macromedia.com/flash/8/main/00000780.html 15.03.2006

Ruse, K. , XML and Dreamweaver [verkkodokumentti]. Xml.com, 2004 [viitattu 14.3.2006]. Saatavissa: http://www.xml.com/pub/a/2004/06/09/dreamweaver.html

Samela, J. 2002. Verkkosisällön hallinta. Edita Prima OY, Helsinki

Sather, A., Ibanez, A.& DeChant, B. 1997. Creating Killer Interactive Web Sites: The Art of Integrating Interactivity and Design. Hayden Books, Indianapolis, Indiana, Yhdysvallat.)

Server-side requirements for resolving XML data [verkkodokumentti]. Macromedia, 2006 [viitattu 10.3.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00000795.html 15.03.2006

Setting global styles [verkkodokumentti] Macromedia, 2006 [viitattu 4.1.2006]

Saatavissa: http://livedocs.macromedia.com/flash/8/main/00002995.html

Setting styles on a component instance [verkkodokumentti] Macromedia, 2006 [viitattu 4.1.2006] Saatavissa:
http://livedocs.macromedia.com/flash/8/main/00002994.html

Shaffer, G. Dynamic Versus static Web Content [verkkodokumentti] GeodSoft, 2005 [viitattu 9.12.2005]. Saatavissa: http://geodsoft.com/book/basics/dynstat.htm

Shiell, A., James, J.,Addey, D. Auld, C., Rafter, J., Spencer P., Kent, A. Surguy, I. & Gudmundsson, O. Practical XML for the Web [verkkodokumentti]. Web Developers, 2002 [viitattu 14.3.2006]. Saatavissa:
http://wdvl.internet.com/Authoring/Languages/XML/PracticalXML/

Solis, J. Loading dynamic data into Flash [verkkodokumentti]. The Flash-DB, 2006 [viitattu 2.3.2006]. Saatavissa: http://www.flash-db.com/Tutorials/loading/loadingData.php?page=1

Statler, T. About the XML object [verkkodokumentti]. Macromedia, 2002a [viitattu 15.3.2006]. Saatavissa:
http://www.macromedia.com/support/flash/applications/jpeg_slideshow_xml/jpeg_slideshow_xml05.html)

Statler, T. Creating a JPEG slide show with XML [verkkodokumentti] Macromedia, 2002b [viitattu 15.3.2006]. Saatavissa:
http://www.macromedia.com/support/flash/applications/jpeg_slideshow_xml/index.html

Taloussanomien uutisotsikot sivuillesi [verkkodokumentti]. Taloussanomat, 2006 [viitattu 8.3.2006]. Saatavissa: http://www.taloussanomat.fi/uutisotsikot.asp

Tekniikan sanastokeskus (TSK) [verkkodokumentti], 2006 [viitattu 10.1.2006]. Saatavissa: http://www.tsk.fi/termitalkoot/

URL Encoding: Reading special characters from a text file [verkkodokumentti]. Macromedia, 2005 [viitattu 4.3.2006]. Saatavissa:
http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_14143

Using Components [verkkodokumentti]. Macromedia, 2005 [viitattu 27.12.2005]. Saatavissa:
http://livedocs.macromedia.com/flash/8/main/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part5_Using_Components.html

Using global, custom, and class styles in the same document [verkkodokumentti] Macromedia, 2006 [viitattu 4.1.2006]
Saatavissa:http://livedocs.macromedia.com/flash/8/main/00002999.html

Using styles to customizise component color and text [verkkodokumentti], Macromedia, 2006 [viitattu 4.1.2006] Saatavissa:
http://livedocs.macromedia.com/flash/8/main/00002992.html

Using XML and Access [verkkodokumentti]. Microsoft, 2003 [viitattu 14.3.2006]. Saatavissa: http://office.microsoft.com/en-au/assistance/HA010345601033.aspx

Using XML in Excel [verkkodokumentti]. Microsoft, 2003 [viitattu 14.3.2006]. Saatavissa: http://office.microsoft.com/en-gb/assistance/HA011895301033.aspx

Waldman, H. Three good reasons to stop using PowerPoint [verkkodokumentti]. Presentations Magazine, 2002 [viitattu 28.2.2002]. Saatavissa:
http://www.presentations.com/presentations/technology/article_display.jsp?vnu_content_id=1731990)

What is dynamic? [verkkodokumentti]. Webopedia 2005 [viitattu 9.12.2005]. Saatavissa: http://www.webopedia.com/TERM/d/dynamic.html

XML [verkkodokumentti]. Wikipedia, 2006 [viitattu 16.3.2006]. Saatavissa:
http://fi.wikipedia.org/wiki/XML

XMLSocket [verkkodokumentti]. Macromedia, 2006 [viitattu 28.2.2006]. Saatavissa: http://livedocs.macromedia.com/flash/8/main/00002906.html

Zaharia, M. XSL Overview [verkkodokumentti]. Macromedia, 2005a [viitattu 05.03.2006] Saatavissa:
www.macromedia.com/devnet/dreamweaver/articles/xsl_overview.html

Zaharia, M. XSL Overview, How Do XML and XSL Work Together? [verkkodokumentti]. Macromedia, 2005b [viitattu 17.2.2006]. Saatavissa:
http://www.macromedia.com/devnet/dreamweaver/articles/xsl_overview_03.html

Zaharia, M. XSL Overview, What is XPath? [verkkodokumentti] Macromedia, 2005c [viitattu 17.2.2006] Saatavissa:
http://www.macromedia.com/devnet/dreamweaver/articles/xml_overview_02.html

Zaharia, M. XSL Overview, What is XSL and Where Should You Use It [verkkodokumentti] Macromedia, 2005d [viitattu 15.2.2006]. Saatavissa:
http://www.macromedia.com/devnet/dreamweaver/articles/xsl_overview.html

KUVALÄHTEET:

KUVA 1. Zaharia, M. XSL Overview, How Do XML and XSL Work Together? [verkkodokumentti]. Macromedia, 2005 [viitattu 17.2.2006]. Saatavissa: http://www.macromedia.com/devnet/dreamweaver/articles/xsl_overview_03.html

KUVA 2. URL Encoding: Reading special characters from a text file [verkkodokumentti]. Macromedia, 2005 [viitattu 4.3.2006]. Saatavissa laajempana: http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_14143

KUVA 3. Statler, T. About the XML object [verkkodokumentti]. Macromedia, 2002 [viitattu 4.3.2006]. Saatavissa: http://www.macromedia.com/support/flash/applications/jpeg_slideshow_xml/jpeg_slideshow_xml05.html

KUVA 4. Statler, T. About the XML object [verkkodokumentti]. Macromedia, 2002 [viitattu 4.3.2006]. Saatavissa: http://www.macromedia.com/support/flash/applications/jpeg_slideshow_xml/jpeg_slideshow_xml05.html

KUVA 5. Jacobson, S. 2006. XML for Flash, Foundation, Friends of ED, Berkeley, CA, Yhdysvallat.

KUVA 11: Using XML in Excel [verkkodokumentti]. Microsoft, 2003 [viitattu 9.3.2006). Saatavissa: http://office.microsoft.com/en-gb/assistance/HA011895301033.aspx

LIITTEET

Liite1: Kuvakaappauksia luodusta yritysesittelysovelluksesta (YeS)