Poikonen Mikko Matias

Rikkaat Internet-sovellukset

Tietojärjestelmätieteen
kandidaatintutkielma
21.5.2007

Jyväskylän yliopisto
Tietojenkäsittelytieteiden laitos
Jyväskylä

TIIVISTELMÄ

Poikonen, Mikko Matias
Rikkaat Internet-sovellukset / Mikko Poikonen
Jyväskylä: Jyväskylän yliopisto, 2006, 33 s.

Kandidaatintutkielma

World Wide Webin käyttö alustana asiakas/palvelin-arkkitehtuurin mukaisten sovellusten toteutuksessa on ollut jatkuvassa kasvussa. Viimeisten vuosien aikana on noussut esille suuri määrä tekniikoita, joissa Web-sovelluksen asiakkaan puoleisen osan toteutus on saanut enemmän painoarvoa. Näillä tekniikoilla toteutuista Web-sovelluksista on alettu käyttää nimeä rikkaat Internet-sovellukset.

Tässä tutkielmassa pyritään selventämään, että mitä rikkailla Internet-sovelluksilla tarkoitetaan, mitkä ovat niiden vaatimukset ja mitä erilaisia lähestymistapoja niiden toteutukseen on olemassa. Tutkimus on tehty kirjallisuuskatsauksena tieteellisten artikkelien ja teosten pohjalta. Tutkimuksen tuloksena todetaan, että rikkaat Internet-sovellukset ovat Web-sovelluksia, joissa sovelluksen esityslogiikan lisäksi myös ainakin osa liiketoimintalogiikasta on toteutettu myös asiakkaan puolella. Rikkaissa Internet-sovelluksissa sovelluksen latauksen jälkeen asiakas ja palvelin kommunikoivat pääasiallisesti asynkronisesti. Käyttöliittymän toteutuksessa on käytetty korkeaa interaktiotasoa tarjoavia komponentteja. Asiakkaan puoleinen osa sovelluksesta voidaan toteuttaa joko Web-selaimen päällä toimivilla tekniikoilla tai Web-toimitusta hyödyntävillä tekniikoilla.

AVAINSANAT: Web-sovellukset, rikkaat Internet-sovellukset, Rich Internet Applications (RIA), ohut asiakas, rikas asiakas, Web-toimitus, Ajax

Tarkastaja: Vahvistamatta
Tietojenkäsittelytieteiden laitos
Jyväskylän Yliopisto

SISÄLLYSLUETTELO

1 JOHDANTO

Internetin asema tietojenkäsittelyssä on muuttunut yhä merkittävämmäksi viimeisten vuosien aikana. Yhtenä keskeisenä tekijänä Internetin merkityksen kasvussa on *World Wide Web*in suuri suosio ja kehitys. Nykyinen, lähes jokaiseen maailmankolkkaan ulottuva Internet omaa valtavan potentiaalin ihmisten saavuttamisessa ja halu sen yhä monipuolisempaan hyödyntämiseen teknisten innovaatioiden avustuksella on ilmeistä.

WWW (World Wide Web, Web) on *asiakas/palvelin-arkkitehtuuriin* (engl. *client/server architecture*) perustuva palvelu, jossa asiakas ja palvelin kommunikoivat keskenään käyttäen *HTTP-protokollaa* (HyperText Transfer Protocol). Tavallisten kotikäyttäjien ja pienyritysten mielenkiinto Internetiä kohtaan heräsi toden teolla 1990-luvun alussa WWW:n myötä. (Paananen, 2000). Alkuajoistaan lähtien WWW on ollut jatkuvan kehityksen alla ja on tultu pitkä matka sen alkuajoista, jolloin käyttö keskittyi staattisten dokumenttien toimittamiseen koneelta toiselle.

Web-sovelluksilla (engl. web applications) tarkoitetaan WWW:n päällä toimivia sovelluksia, joissa *Web-asiakas* (engl. *web client*) toimii *Web-palvelimen* (engl. *web server*) kanssa yhteistyössä suorittaen erilaisia tehtäviä. Tehtävien vaativuus voi vaihdella yksinkertaisten viestien näyttämisestä monimutkaisten transaktioiden hallintaan saakka. (Wahono & Cheng, 2002). Nykyisin Web-sovellukset ovat yhä enenevässä määrin tärkeä osa yritysten liiketoimintaa ja ne voivat olla tiivisti linkitettynä yritysten kriittisiin liiketoimintasovelluksiin (Antoniol, Di Penta & Zazzara, 2004). Esimerkkeinä liiketoiminta-alueista, joilla Web-sovelluksilla on tärkeä rooli, ovat sähköiset kauppapalvelut, pankki- ja pörssipalvelut sekä julkisen hallinnon palvelut.

Bozzonin, Comain, Fraternalin & Carughin (2006) mukaan Web-sovellusten kasvaneen monimutkaisuuden johdosta ne ovat alkaneet kärsiä ongelmista

käytettävyydessä ja käyttäjäinteraktiossa. Heidän mukaansa Web-sovellusten käyttökokemukset ovat huonompia työpöytäsovelluksiin verrattuna, koska ne reagoivat huonommin käyttäjän toimiin, aiheuttavat turhaa liikennöintiä asiakkaan ja palvelimen välillä sekä ne eivät tue toimintaa tietoliikenneyhteyksien katkeamisen aikana. Tällaisissa Web-sovelluksissa, jotka ovat vahvasti palvelinkeskeisiä, asiakasta kutsutaan nimellä *Ohut asiakas (*engl. *Thin client)*.

Rikkailla Internet-sovelluksilla (engl. *RIA*, *Rich Internet Applications*) viitataan Web-sovelluksiin, jotka pyrkivät puuttumaan Web-sovelluksissa ilmenneisiin ongelmiin. Ne tarjoavat kehittyneemmän käyttöliittymän monimutkaisten prosessien ja datan esittämiseen, minimoivat asiakkaan ja palvelimen välisen datan siirron sekä siirtävät käyttäjän interaktion ja tiedon esityksen hallinnan palvelimelta asiakkaalle. (Bozzon, Comai, Fraternali & Carughi, 2006).

Tässä tutkielmassa pyritään selventämään, että mitä rikkailla Internet-sovelluksilla tarkoitetaan, mitkä ovat niiden vaatimukset ja mitä erilaisia lähestymistapoja niiden toteutukseen on olemassa. Ohut asiakas Web-sovelluksia ja rikkaita Internet-sovelluksia tarkastellaan lähinnä korkean tason arkkitehtuurin ja vaatimusten näkökulmasta menemättä toteutuksen tarkempiin yksityiskohtiin. Lähtökohtana tutkimuksessa on aiemmassa Web-sovellusten tutkimuksessa esitetyt arkkitehtuurityylit. Rikkaita Internet-sovelluksia ja niiden vaatimuksia tarkastellaan näiden aiemmin esitettyjen tyylien pohjalta ja pyritään tunnistamaan, että eroavatko ne todella aiemmin esitetyistä Web-sovellusten tyyleistä.

Tutkielman toisessa luvussa tutustutaan WWW:n toimintaan ja tekniikoihin sekä ennenkaikkea ohut asiakas Web-sovelluksien arkkitehtuuriin ja toimintaan. Kolmannessa luvussa kuvaillaan, että kuinka rikkaat Internet-sovellukset eroavat ohut asiakas Web-sovelluksista, pyritään johtamaan rikkaille Internet-sovelluksille asetetut vaatimukset kirjallisuuden pohjalta sekä

tunnistetaan RIA-tekniikoiden eri kategoriat. Neljännessä luvussa, yhteenvedossa, esitellään tutkielman keskeisimmät tulokset ja kerrataan sisällön pääkohdat.

2 OHUT ASIAKAS WEB-SOVELLUKSET

WWW:n käyttäjämäärien ja tekniikoiden kehitys on ollut jatkuvaa palvelun lyötyä itsensä läpi Internetissä 1990-luvun alussa. WWW:n alkuaikoina palvelua käytettiin lähinnä palvelimelle tallennettujen dokumenttien suoraan toimittamiseen palvelimelta asiakkaalle. Nykyisin WWW toimii alustana mitä monimutkaisemmille sovelluksille, jotka käsittelevät käyttäjien erilaisia syötteitä ja ovat yhteydessä mm. tietokantoihin. Tässä luvussa esitellään ohut asiakas Web-sovellusten toimintaperiaatteita sekä niiden oleellisia osia. Web-sovelluksien toimintaa esitellään kokonaisuudessaan, mutta pääpaino on asiakkaan toimintojen esittelyssä.

2.1 Yleisiä Web-sovellusten tekniikoita

Seuraavassa esitellään joitakin tekniikoita, joita on käytetty laajasti Web-sovellusten toteutuksessa.

HTTP (HyperText Transfer Protocol) on yleiskäyttöinen, tilaton protokolla, jota voidaan käyttää hajautetuissa, yhteistoiminnallisissa (engl. collaborative) hypertekstijärjestelmissä. HTTP:aa on käytetty WWW:ssä vuodesta 1990 lähtien. HTTP on *pyyntö/vastaus -protokolla* (engl. *request/response protocol*), jossa asiakas lähettää pyynnön palvelimelle, jossa on sisällytettynä *pyynnön tyyppi* (engl. *request method*), pyydetyn resurssin sijainti eli *URI* (*Uniform Resource Locator*), protokollaversio sekä mahdollisia muita pyynnön muuttujia. Palvelimen vastaanotettua ja käsiteltyä pyynnön se lähettää vastauksen asiakkaalle, jossa on sisällytettynä vastauksen prokollaversio, onnistunutta tai epäonnistunutta pyyntöä ilmaisevan koodi, muita mahdollisia muuttujia sekä mahdollisesti pyyntöön liittyneeseen resurssiin liittyvä sisältö. (Hypertext Transfer Protocol -- HTTP/1.1, 1999).

HTML-kieli (HyperText Markup Language) on WWW-dokumenttien julkaisussa käytetty kieli. Se antaa mahdollisuuden, julkaista ja määritellä dokumenttien rakenteen erilaisine elementteineen, hakea tietoa hypertekstilinkkien määrittelemistä kohteista käyttäjän aloitteesta, suunnitella lomakkeita, joilla voidaan kerätä syötteitä käyttäjältä sekä sisällyttää mm. video- ja äänileikkeitä dokumentteihin. (HTML 4.01 specification, 1999).

Skriptauskieli on ohjelmointikieli, jota voidaan käyttää olemassa olevan järjestelmän manipulointiin, kustomointiin ja automatisointiin. Web-selain tarjoaa ympäristön (engl. host environment), jonka objekteja ja mahdollisuuksia (engl. facilities) voidaan muokata ja hyödyntää skriptauskielten avulla. (EcmaScript Language Specification, 1999)

DOM (Document Object Model) on ohjelmointirajapinta, joka on tarkoitettu validien HTML ja XML dokumenttien käsittelyyn. Se määrittelee dokumentin loogisen rakenteen sekä käytänteet, jolla dokumenttiin päästään käsiksi ja miten sitä voidaan muokata. Document Object Modelin avulla ohjelmoijat voivat rakentaa dokumentteja, navigoida niiden rakenteita ja lisätä, muokata tai poistaa elementtejä ja sisältöä. (Document Object Model (DOM) Level 3 Core Specification, 2004)

Liitännäiset (engl. Plug in) ovat sovellusohjelmia, jotka Web-selain voi käynnistää käsittelemään tiedostoja, joita se ei itse osaa avata. Liitännäiset integroidaan osaksi Web-selainta (Paananen, 2000).

ActiveX-kontrollit on Microsoftin kehittämä käyttöjärjestelmäsidonnainen tekniikka, jossa kontrolleja ladataan käyttäjän koneelle esimerkiksi WWW:n välityksellä, minkä jälkeen ne näkyvät käyttäjälle erilaisina pieninä käyttöjärjestelmäelementteinä, valintaikkunoina ja pieninä sovelluksina. (Paananen, 2000). Liitännäiset ja ActiveX-kontrollit voivat toimia vaihtoehtoisina tapoina toteuttaa samoja laajennoksia selaimeen. Selaimen

toimintaa laajentavat sovellukset voivat olla toteutettuna Microsoftin selaimissa ActiveX-kontrolleina ja muissa selaimissa liitännäisinä.

Java Applet-ohjelmat ovat pieniä, itsenäisiä sovelluksia, joita voidaan toimittaa käyttäjän koneelle esimerkiksi WWW:n välityksellä. Sovelluksia voidaan ajaa käyttäjän koneella joko työpöydällä tai Web-selaimen päällä. (Paananen, 2000)

2.2 Web-sovellukset

Web-sovellukset ovat yksi nopeimmin kasvavista ohjelmistotekniikan aloista. Web-sovellukset tarjoavat informaatiota ja palveluita suurille käyttäjämäärille. Palveluiden tason monipuolisuus ja monimutkaisuus ovat vaihtelevia. Käyttäjät ovat vuorovaikutuksessa sovelluksen kanssa tyypillisesti *Web-selaimen* välityksellä. Käyttäjän navigoidessa sovelluksessa tai lähettäessä dataa, uusia pyyntöjä lähetetään palvelimelle selaimen toimesta. (Elbaum, Chilakamarri, Fisher & RotherMel, 2006).

Web-sovelluksien ero normaaleihin Web-sivuihin verrattuna on se, että Web-sovellukset tarjoavat mahdollistavien tekniikoiden avulla käyttäjien syötteiden mukaan muuttuvaa sisältöä ja antavat käyttäjälle mahdollisuuden vaikuttaa palvelimella sijaitsevaan *liiketoimintalogiikkaan* (engl. *business logic*). Mikäli palvelimella ei ole liiketoimintalogiikkaa, ei Web-pohjaista järjestelmää tulisi kutsua Web-sovellukseksi. (Conallen, 2003)

Conallenin (2003) mukaan Web-sovelluksien tyypillisiä arkkitehtuurityylejä, jotka eroavat lähinnä asiakkaan toteutustavassa, on kolme; ohut asiakas, rikas asiakas ja Web-toimitus. Seuraavassa tyylejä on kuvattu lyhyesti asiakkaan näkökulmasta Castro, Kolp & Mylopoulous (2001) mukaan:

- Ohut asiakas-tyyli on kaikista yksinkertaisin toteutustyyli asiakkaan vaatimuksien näkökulmasta. Se vaatii ainoastaan standardin Web-selaimen asiakkaalta.

- Rikas asiakas laajentaa ohut asiakas tyyliä asiakkaan puolen skriptien ja objektien, kuten ActiveX-kontrollien ja applettien, avulla.

- Web-toimitus tarjoaa perinteisen asiakas/palvelin-arkkitehtuurin, jossa WWW:iä käytetään asiakkaan sovelluksen toimitukseen. Sovellusta ajetaan yleensä selaimen ulkopuolella.

Tässä luvussa käsitellään tarkemmin ohutta asiakasta käyttäviä Web-sovelluksia. Muihin kahteen tyyliin palataan tarkemmin kolmannessa luvussa, jossa käsitellään kehittyneempiä Web-asiakkaita.

2.3 Arkkitehtuuri

Sovelluksen tai tietojärjestelmän arkkitehtuuri on järjestelmän rakenne, joka käsittää sovelluksen elementit, elementtien ulkoisesti näkyvät ominaisuudet sekä niiden väliset yhteydet. (Bass, Clements, Katzman, 2003). Puhuttaessa sovelluksen arkkitehtuurista käytetään usein termejä *kerros* (engl. *layer)* ja *taso* (engl. *tier*). Tasot nähdään usein fyysisenä erona eri arkkitehtuurin elementtien välillä, kun taas kerrokset kuvaavat ennemmin loogisia eroja. Termien käyttö on vaihtelevaa ja niitä käytetään usein synonyymeinä. (Fowler, 2003). Tässä tutkielmassa termejä käytetään erillisinä edellä esitetyn jaottelun mukaisesti.

Web-sovellusten toimintalogiikka rakentuu usein *kolmikerroksisesta arkkitehtuurista*. Arkkitehtuurin alin kerros tarjoaa mekanismit tiedon tallentamiseen. Tyypillinen tallennuspaikka on relaatiotietokanta. Arkkitehtuurin ylin kerros tarjoaa yleensä HTML rajapinnan, jonka Web-selain voi muotoilla käyttäjälle ymmärrettävään muotoon. Keskimmäinen kerros sitoo kerrokset yhteen ja määrittelee sovelluksen toimintalogiikan. (Haustein & Pleumann, 2005). Eri lähteissä on käytetty hieman erilaisia nimityksiä näistä arkkitehtuurin kerroksista. Tässä tutkielmassa kerroksista käytetään ylhäältä alaspäin lueteltuna seuraavia nimityksiä: *Esityslogiikka (engl. Presentation logic)*,

Liiketoimintalogiikka (engl. *Business logic*) ja *Datalähdelogiikka (engl. Data source logic)*. Seuraavassa kuvataan kerroksia tarkemmin Fowlerin (2003) mukaan.

Esityslogikka määrittelee, että kuinka käyttäjän ja sovelluksen välinen interaktio hallitaan. Esityslogiikka voidaan toteuttaa esimerkiksi yksinkertaisesti komentorivi- tai tekstipohjaisen järjestelmän avulla, mutta nykyisin esityslogiikka toteutetaan yleensä graafisen tai HTML-pohjaisen käyttöliittymän avulla. Esityslogiikka tason ensisijaiset vastuut ovat informaation näyttäminen käyttäjälle ja käyttäjän komentojen välittäminen muille kerroksille.

Liiketoimintalogiikka määrittelee sovelluksen *toiminta-alueeseen* (engl. *domain)* liittyvän logiikan ja toimenpiteet. Se käsittää käyttäjän syötteisiin ja tallennettuun dataan liittyvää laskennan, esityslogiikkatasolta tulevan datan validoinnin ja kommunikoinnin datalähteiden kanssa.

Datalähdelogiikka määrittelee sovelluksen kommunikoinnin muiden järjestelmien kanssa, jotka suorittavat tehtävien sen puolesta. Muut järjestelmät voivat olla esimerkiksi viestintäjärjestelmiä, transaktioiden seurantajärjestelmiä tai muita sovelluksia. Suurimmassa osassa liiketoimintaan liittyvistä järjestelmistä suurin osa datalähdelogiikka ovat tietokannat, jotka huolehtivat tiedon pysyvästä tallentamisesta.

Sovelluksen esitys-, liiketoiminta- ja datalähdelogiikka on ohut asiakas Web-sovelluksien arkkitehtuurissa jaettu eri tasojen välille. Seuraavassa on kuvattu sovelluksen tasoja Conallenin (2003) mukaan:

Web-selain toimii yleisenä asiakkaan käyttöliittymänä sovellukseen. Sovellus käyttää selainta Web-sivujen pyytämiseen palvelimelta. Palvelimen palauttama dokumentti pitää sisällään määritellyn HTML käyttöliittymän, joka pitää sisällään kontrolleja (tekstinsyöttökenttiä, painikkeita ym.). Kaikki käyttäjäinteraktio tapahtuu selaimen välityksellä.

Web-palvelin toimii liityntäpisteenä (engl. access point) kaikille Web-selaimille. Web-selain päättelee pyynnön tyypistä riippuen, että voiko se vastata pyyntöön itse (kts. *staattinen sivu*) vai liittyykö pyyntöön *dynaaminen sivu,* jolloin pyyntö ohjataan *sovelluspalvelimelle*.

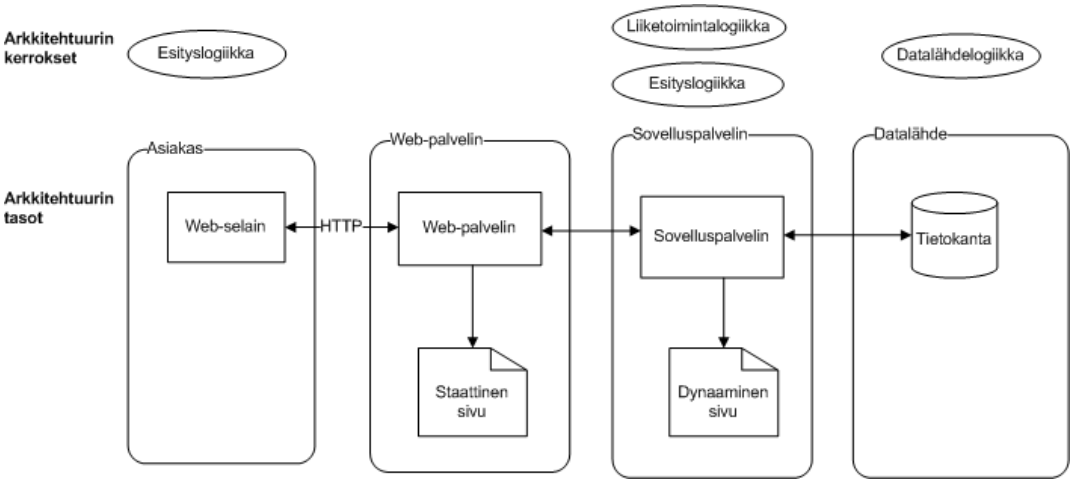
Staattinen sivu on Web-sivu, joka ei tarvitse käsittelyä palvelimen puolella. Web-palvelin palauttaa staattisen sivun asiakkaalle sellaisenaan kuin se on palvelimen tiedostorakenteessa.

Dynaaminen sivu on Web-sivu, joka käy läpi käsittelyä palvelimen puolella. Tyypillisesti dynaamiset sivut on toteutettu palvelimella sivuina, jotka pitävät sisällään skriptauskieliä. Usein käytettyjä tekniikoita ovat esimerkiksi *Active Server Pages, Java Server Pages ja PHP*- tekniikat.

Sovelluspalvelin on pääasiallinen palvelinpuoleisen liiketoimintalogiikan toteuttaja. Sovelluspalvelin on vastuussa dynaamisten sivujen suoritettava koodin ajamisesta. Sovellus- ja Web-palvelin saattavat sijaita myös samalla koneella ja jakaa toimintaympäristönsä. Loogisesti nämä kuitenkin ovat kaksi eri kokonaisuutta.

Tietokantapalvelin on vastuussa sovelluksen pysyväisen tilan säilyttämisestä. Tilaa pidetään yllä tietokannassa. Palvelin on yleensä *relationaalisen tietokannan hallintajärjestelmä* (engl. *relational database management system*)

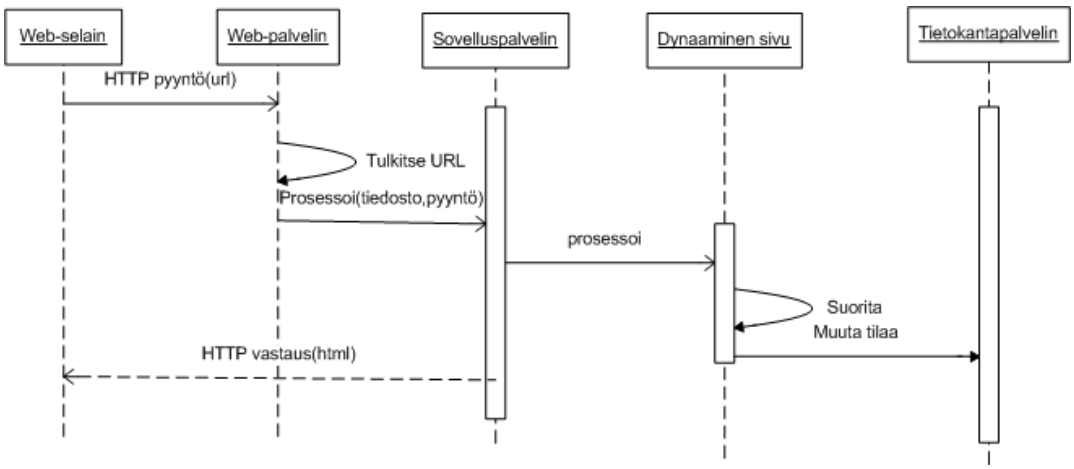
Ohut asiakas Web-sovellukset ovat vahvasti palvelinkeskeisiä. Tällaisessa Web-sovelluksessa Web-selain toimii yksinkertaisena informaatiota näyttävänä terminaalina, jolla ei ole tietoa sovelluksen suorituksen tilasta. Asiakkaan vastuulle jää siis vain palvelimelta saadun HTML-dokumentin näyttäminen käyttäjälle ymmärrettävässä muodossa. (Pascarello, 2006). Web-sovelluksen liiketoimintalogiikka on kokonaisuudessaan Web-/sovelluspalvelimen vastuulla, jonka vastuulla on myös kommunikointi tiedon pysyvästä tallentamisesta vastaavaan tasoon (Kuva 1).



Kuva 1: Ohut asiakas Web-sovellusten arkkitehtuuri

2.4 Toiminta

Kuvassa 2 on esitetty dynaamisen sivun haku sekvenssikaavion avulla. Suoritus alkaa asiakkaan Web-selaimelta, joka pyytää Web-palvelimelta dynaamista sivuresurssia lähettämällä HTTP-pyynnön palvelimelle. Web-palvelin tulkitsee, että pyydetyn resurssin suoritus vaatii sovelluspalvelimen kutsumista. Sovellupalvelin prosessoi dynaamisen sivun sisällön ja muuttaa tarvittaessa tietokantapalvelimelle pysyvästi tallennetun liiketoimintalogiikkaa heijastavan datan tilaa. (Conallen, 2003)



Kuva 2:Ohut asiakas Web-sovelluksen toiminta

2.5 Asiakkaan vaatimukset

Conallenin (2003) mukaan ohut asiakas Web-sovelluksen tärkein vaatimus on mahdollisuus toimittaa sovellus mahdollisimman laajasti käyttäjille. On tärkeämpää saavuttaa suuri määrä käyttäjiä käyttämällä rajoitetuin toiminnoin varustettua asiakasta kuin toimittaa kehittyneempi asiakas rajoitetuille käyttäjämäärille. Tästä syystä ohut asiakas Web-sovellukset vaativat vain minimaalista kokoonpanoa asiakkaan puolella. Asiakkaan ainoa vaatimus on standardi Web-selain, joka osaa pyytää ja rendereöidä HTML-sivuja sekä ottaa vastaan ja lähettää yksinkertaisia tietotyyppejä.

Ohut asiakas Web-sovelluksen muiden komponttien, kuten Web-palvelinten ja sovelluspalvelinten toiminnan ja vaatimusten tarkempi käsittely on rajattu tämän tutkielman ulkopuolelle.

2.6 Hyviä ja huonoja puolia

Web-sovellusten ohut Web-asiakas malli soveltuu parhaiten Web-sovelluksiin, joissa asiakkalla on vain minimaalinen määrä suoritustehoa tai asiakkaan konfiguraatiota ei voida hallita. Asiakkaalta vaaditaan ainoastaan mahdollisimman yksinkertainen konfiguraatio, kuten standardeja lomakkeita näyttävä Web-selain. (Castro, Kolp & Mylopoulos, 2001)

Suuri etu kaiken liiketoimintalogiikan toteuttamisessa palvelimella on helppo päivitettävyys ja ongelmien korjaaminen, koska tärkeät liiketoimintalogiikkaa käsittelevät prosessit suoritetaan yhdessä paikassa. Näin kehittäjien ja ylläpitäjien ei tarvitse huolehtia uusien sovellusversioiden toimituksesta ja sovelluksen komponenttien välillä ei näin ollen ole esimerkiksi eri versioihin liittyviä eroja. Myöskään sovelluksen yhteensopivuudesta muiden asiakkaan suoritusympäristön sovellusten kanssa ei tarvitse huolehtia yleisen Web-selainpohjaisen suoritusympäristön johdosta. Merkittävänä rajoitteena

arkkitehtuurissa on kuitenkin toistuva edestakainen kommunikointi palvelimen kanssa jokaiseen päätöksentekoon liittyvän tapahtuman yhteydessä. Tämä heikentää sovelluksen vastaanottavaisuutta (engl. responsiveness) pidentyneiden vasteaikojen takia. Vasteaikoja ja turhia sivulatauksia voidaan vähentää hieman käyttämällä esimerkiksi JavaScript-koodia. (Fowler, 2003)

Bozzon, Comai, Fraternali & Carughi (2006) mukaan Web-sovelluksien kasvaneen monimutkaisuuden vuoksi ne ovat alkaneet kärsimään ongelmista käytettävyydessä ja käyttäjäinteraktiossa. Heidän mukaansa käyttäjäkokemus ei ole verrattavissa työpötäsovelluksiin, koska reagointi käyttäjän toimiin on hitaampaa verkon viiveiden sekä ylimääräisen tiedonsiirron takia ja sovellukset eivät tue käyttöä palvelinyhteyden katketessa.

Ohutta asiakasta hyödyntävillä Web-sovelluksissa tarkoitetaan siis sovelluksia, jossa asiakkaan puolella ei suoriteta minkäänlaista liiketoimintalogiikkaan liittyvää prosessointia. Käyttäjän syötteitä ei esimerkiksi tarkisteta skriptauskielten avulla, vaan kaikki tällainen edes löyhästi liiketoimintalogiikkaa sivuuttava suoritus jätetään palvelimen harteille. Tätä tutkielmaa kirjoitettaessa JavaScriptin käyttö Web-sovelluksissa ja Web-selaimien tuki skriptien ajamiseen on hyvin laajaa. Puhtaan ohuen asiakkaan hyödyntäminen on kuitenkin edelleen perusteltua monissa tapauksissa ja tämä mahdollisimman yksinkertainen lähtökohta asiakkaan puolella toimii hyvänä referenssinä muiden Web-sovellusarkkitehtuurien tarkasteluun.

3 RIKKAAT INTERNET-SOVELLUKSET

Yksinkertainen ohut Web-asiakas sopii parhaiten tilanteesen, jossa asiakkaalla on käytettävissään vain rajattu määrä suoritustehoa tai asiakkaan konfiguraatiolta ei voida edellyttää edistyksellisempia ominaisuuksia. Web-sovellusten tapauksessa minimivaatimuksissa asiakkaalta vaaditaan vain Web-selain, joka osaa näyttää HTML-dokumentteja ja kommunikoida palvelimen kanssa HTTP:n avulla. Nykyiset Web-selaimet pystyvät kuitenkin paljon muuhunkin kuin pelkkään HTML-dokumenttien näyttämiseen asiakkaalle ja WWW:n kautta voidaan toimittaa vaivattomasti myös selaimen ulkopuolella ajettavia sovelluksia. Ohut asiakas Web-sovellusten puutteiden, Web-sovellusten monimutkaistumisen ja tekniikoiden kehittymisen myötä on esitetty, että esille olisi noussut uusi Web-sovellusten sukupolvi, jossa asiakkaan vastuu Web-sovelluksen suorituksessa on huomattavasti suurempi. Näistä sovelluksista käytetään nimitystä rikas Internet-sovellus.

Kohu uusien ja uudestaan lanseerattujen vanhempien tekniikoiden ympärillä on ollut kiihtyvää. Ala on kuitenkin sen verran nuori, että tieteellinen tutkimus on puutteellista. Perusteellinen selvitys siitä, että mitä rikkailla Internet sovelluksilla tarkoitetaan, on kiven alla. Tässä luvussa pyritään selventämään, että mitä rikkaat Internet-sovellukset ovat, mitä vaatimuksia RIA-tekniikoille asetetaan, mitä erilaisia lähestymistapoja niiden toteuttamiseen on olemassa ja kuinka lähestymistapojen mukaiset sovellukset toimivat pääpiirteissään.

3.1 Arkkitehtuuri

Rikkaat Internet-sovellukset tarjoavat edistyneitä käyttöliittymiä monimutkaisten prosessien ja datan esittämiseen minimioiden asiakkaan ja palvelimen välisen tiedonsiirron siirtämällä suuren osan esityslogiikasta ja osan liiketoimintalogiikasta palvelimelta asiakkaalle. Yleensä RIA ladataan asiakkaan puolelle sisältäen jonkin verran alustavaa dataa. Tämän jälkeen

asiakas hallitsee datan näyttämistä sekä tapahtumien hallintaa ja kommunikoi palvelimen kanssa vain, kun se tarvitsee lisää dataa tai haluaa siirtää tietoa palvelimelle. (Bozzon, Comai, Fraternali & Carughi, 2006)

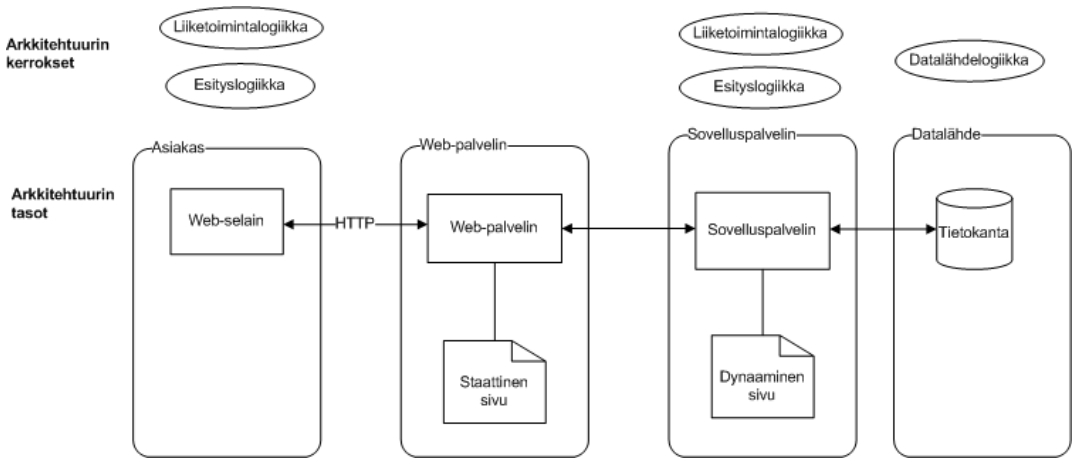
Merkittävä ero Rikkaiden Internet-sovellusten ja ohut asiakas Web-sovellusten välillä piilee juuri osan siirtymisestä palvelimelta asiakkaalle. Asiakkaan vastuulla uudessa arkkitehuurissa on pelkän datan näyttämisen sijasta myös sovelluksen käyttöliittymän muokkaus, käyttäjän toimiin reagointi ja ainakin osittaisen liiketoimintalogiikkaa heijastavan tietomallin ylläpito ja muokkaus.

Bozzon, Comai, Fraternali & Carughi (2006) ovat esittäneet, että RIA-tekniikat voidaan jakaa neljään eri kategoriaan: *skriptaukseen perustuviin* (engl. *Scripting-based*), *liitännäisiin perustuviin* (engl. *Plugin-based*), *selaimeen perustuviin* (engl. *Browser-based*) ja *Web-pohjaisiin työpöytäsovelluksiin perustuviin* (engl. *Web-based Desktop*) tekniikoihin.

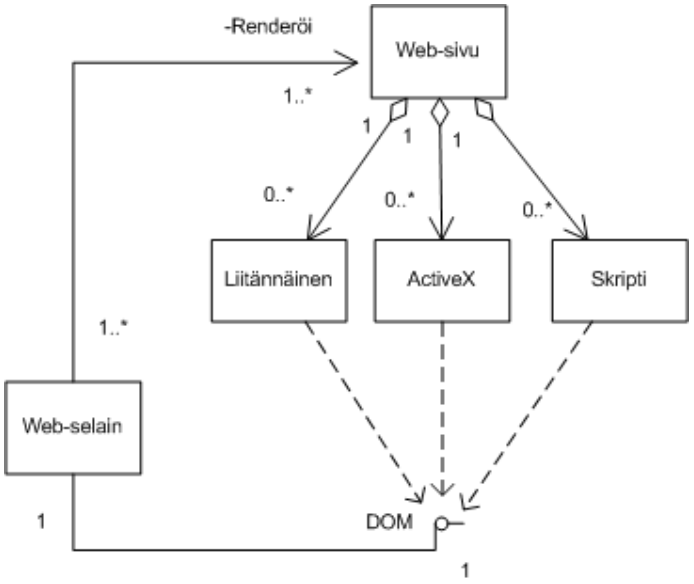
Näistä kolmen ensimmäisen kategorian tekniikat toimivat selaimen päällä. Viimeisellä kategorialla tarkoitetaan samaa kuin Web-toimituksella.

3.1.1 Selainpohjaiset Web-sovellukset

Selainpohjaisissa sovelluksissa sivuresursseja pyydetään palvelimelta aivan kuten ohut asiakas arkkitehtuurissakin (Kuva 3). Sivuresurssit voivat olla joko dynaamisia tai staattisia sivuja. Kun sivu on ladattu ja renderöity selaimen toimesta, dokumenttiin upotetut skriptit ajetaan. Sivuun upotettuja appletteja pyydetään palvelimelta erikseen. Appletit suoritetaan latauksen valmistuttua. Appletit ja upotetut skriptit voivat kommunikoida keskenään DOM-rajapinnan kautta (Kuva 4). (Conallen, 2003). Applettien suoritusta koskevat säännöt ja mahdollisuudet pätevät myös muihin ActiveX-kontrollien ja liitännäisten käsittelemiin tiedostoihin.



Kuva 3: Selainpohjaisten RIA-tekniikoiden arkkitehtuuri



Kuva 4: Asiakkaan elementit

Seuraavaksi esitellään eri lähestymistapojen mukaisia toteutustapoja yleisellä tasolla. Lähestymistapojen alle kuuluvien tekniikoiden toiminnan tarkempi käsittely rajataan tämän tutkielman ulkopuolelle tekniikoiden suuren määrän vuoksi.

3.1.2 Skriptaukseen perustuvat Web-asiakkaat

Skriptaukseen perustuvissa rikkaissa Web-asiakkaissa asiakkaan puoleinen logiikka saavutetaan skriptauskieliä, kuten JavaScript, hyödyntämällä. Käyttäjärajapinnat perustuvat HTML:n ja *CSS*:n (*Cascading Style Sheets*) yhdistelmään. (Bozzon, Comai, Fraternali & Carughi, 2006). Suurin osa skriptaukseen perustuvien rikkaiden Web-asiakkaiden toteutukseen käytetyistä tekniikoista ovat olleet käytössä jo jonkin aikaa, mutta nykyään tekniikoihin viitataan yleensä nimityksellä *Ajax (Asynchronous Javascript + XML)*. Puderin (2006) mukaan termi Ajax nousi esiin ensimmäistä kertaa esiin vuonna 2005 Jesse Garrettin toimesta.

Ajaxissa käytetään XHTML- ja CSS-kieliä standardinmukaisten näyttöjen luontiin, Document Object Modelia dynaamiseen näyttämiseen ja interaktioon, *XMLHttpRequest-oliota* asykroniseen datan siirtoon ja JavaScript-kieltä sitomaan kaikki palaset toisiinsa. Näiden uudentyyppisten Web-sovellusten ytimessä on yhteen Web-sivuun perustuva käyttöliittymä, joka mahdollistaa rikkaan interaktion sovelluksen kanssa. Tässä mallissa muutoksia tehdään yksittäisiin Web-sivun komponentteihin koko sivun uudelleen lataamisen sijasta. (Mesbah & Deursen, 2006)

Ajaxiin liittyvillä tekniikoilla on mahdollista toteuttaa rikkaita Web-asiakkaita puhtaalta pöydältä, mutta markkinoilla on myös suuri määrä eri valmistajien tarjoamia Ajax-implementaatioita, jotka pyrkivät helpottamaan Ajax-sovelluskehitystä. Jo pelkkiä avoimeen lähdekoodiin perustuvia Ajax-implementaatioita on saatavilla joitakin kymmeniä (Open source applications foundation, 2007).

Skriptaukseen perustuvien rikkaiden Web-asiakkaiden vaatimat suoritusympäristöt löytyvät moderneista Web-selaimista ja siten ne eivät

yleensä vaadi ylimääräisten sovellusten tai suoritusympäristöjen asentamista käyttäjän koneelle.

3.1.3 Liitännäisiin perustuvat Web-asiakkaat

Liitännäisiin perustuvat Web-asiakkaat hyödyntävät selaimeen asennettuja liitännäisiä, jotka tarjovavat edistyksellisempiä tiedon näyttämiseen ja käyttäjän toimien hallintaan liittyviä ominaisuuksia. (Bozzon, Comai, Fraternali & Carughi, 2006).

Liitännäisiin perustuvien Web-asiakkaiden toteutusmahdollisuudet ovat laajat. Selaimen liitännäisiksi rekisteröidyille sovellusohjelmille voidaan ohjata esimerkiksi XML-tiedostoja, skriptejä, valmiiksi käännettyjä sovelluksia tai mediatiedostoja. Liitännäisiin perustuvia rikkaiden Web-sovellusten toteutukseen käytettyjä tekniikoita ovat esimerkiksi *Adoben Flash Player*-liitännäisen päällä toimivat Adoben *Flex*- ja *Flash-sovellukset* sekä *Laszlo Systemsin OpenLaszlo-sovellukset*, Java Applet-sovellukset ja *Microsoftin Silverlight-sovellukset.*

Liitännäisiin perustuvan Web-asiakkaan toiminta vaatii yleensä asianomaisen liitännäisen asentamiseta Web-selaimeen.

3.1.4 Selaimeen perustuvat Web-asiakkaat

Selaimeen perustuvissa Web-asiakkaissa käytettävä Web-selain tukee deklaratiivisesti määriteltyjen rikkaiden käyttöliittymien tulkkaamista ja renderöintiä ajon aikana. (Bozzon, Comai, Fraternali & Carughi, 2006). Selaimeen perustuvissa rikkaissa Web-asiakkaissa Web-selain sisältää ajoympäristön, joka mahdollistaa rikkaiden Web-sovellusten toteuttamisen. *Mozillan XUL-sovellukset* ovat esimerkki tällaisista sovelluksista.

Selaimeen perustuvien rikkaiden Web-asiakkaiden vaatima suoritusympäristö löytyy yleensä tietyistä selaimista jo valmiina, mutta muihin selaimiin tukea ei ole saatavana välttämättä ollenkaan.

3.1.5 Web-toimitukseen perustuvat Web-asiakkaat

Web-toimitukseen perustuvat Web-asiakkaat ladataan WWW:n välityksellä, mutta niitä suoritetaan selaimen ulkopuolella (Bozzon, Comai, Fraternali & Carughi, 2006). Web-toimitukseen perustuvia rikas Web-asiakas tekniikoita ovat esimerkiksi *Java Web Start-sovellukset, Adoben Apollo-sovellukset ja Microsoftin Smart Client-sovellukset.*

Java Web Start-sovelluksien avulla käyttäjille annetaan mahdollisuus käynnistää Java-sovelluksia klikkaamalla linkkiä Web-selaimessa. Java Web Start-sovelluksia ajetaan Web-selaimen ulkopuolella käyttäjän koneella ja siksi ne pääsevät vapaammin käsiksi paikallisiin resursseihin. Kun sovellus on kerran ladattu käyttäjän koneelle, ei sitä tarvitse ladata enää uudestaan uusien suorituskertojen yhteydessä ellei sovelluksesta ole ilmestynyt uudempaa versiota. (Fleury, Basney & Welch, 2006)

Web-toimitukseen perustuvat asiakkaat muistuttavat hyvin paljon normaaleja työpöytäsovelluksia. Web-toimittavien sovelluksien asennustoimet ovat kuitenkin hyvin yksinkertaiset työpöytäsovelluksiin verrattuna. Lisäksi, Web-toimitukseen perustuvat asiakkaat eroavat työpöytäsovelluksista siten, että niiden suoritusympäristöstä on rajattu pääsy paikallisiin resursseihin. Web-toimitukseen perustuvat asiakkaat huolehtivat itse kokonaisuudessaan esityslogiikasta, koska ne eivät perustu Web-sivuihin.

3.2 Asiakkaan vaatimukset

Puhuttaessa rikkaista Internet sovelluksista ja rikkaista Web-asiakkaista, termillä ”rikas” viitataan usein muihinkin ominaisuuksiin kuin asiakkaan lisääntyneeseen vastuuseen. Preciadon, Linajen. Sánchezin ja Comain (2006) mukaan RIA yhdistää työpöytäsovellusten interaktiivisen, multimediaa hyödyntävän käyttöliittymän Web-sovellusten kanssa. Rikkailla Internet sovelluksilla viitataan myös saumattomasti toteutettuun interaktioon käyttäjän kanssa. Yu, Benatallahi, Saint-Paul & Casati (2006) mukaan ideaalitilanteessa Web-sovellusten tulisi tarjota samanalainen tuottavuuden taso kuin työpöytäsovellusten. Tuottavuudella tarkoitetaan sitä, että kuinka tehokkaasti käyttäjät voivat suorittaa työtehtävänsä sovelluksella. He tunnistavat neljä vaatimusta, jotka interaktiivisen Web-sovelluksen tulisi täyttää:

- Korkea käytettävyys (engl. high usability): Käyttäjille tulisi tarjota työpöytäsovelluksissa käytettyjen interaktiivisten käyttöliittymäkomponenttejen tapaisia komponentteja.

- Turvallinen suoritusympäristö (engl. secure client environment): Asiakaspuolen suoritusympäristön tulisi suorittaa vain turvallista koodia, joka ei voi vaarantaa käyttäjän koneen tietoturvallisuutta.

- Lyhyet viiveet (engl. low latency): Käyttäjät odottavat lyhyitä vasteaikoja sovelluksilta. Asynkronista kommunikontia tulisi käyttää tiedon hakemiseen kokonaisten sivulatausten välttämiseksi.

- Yhenäinen ulkoasu (engl. consistent look and feel): Web-sovellusten tulisi käyttää samoja käytänteitä kuin työpöytäsovellusten, jotta käyttäjät oppisivat käyttämään nopeammin Web-sovelluksia.

Preciado, Linaje. Sánchez & Comai (2006) esittävät seuraavia vaatimuksia Rikkaille Internet sovelluksille ja niiden toteutukseen liityville tekniikoille:

- Interaktio: Mahdollisuus määritellä, että miten sovellus reagoi käyttäjän toimiin.

- Multimedia: Mahdollisuus käyttää kuvaa, ääntä, videota, reaaliaikaista ja streaming multimediaa.

- Visuaalinen jatkuvuus (engl. visual continuity): Mahdollisuus välttää häiritseviä ruudunpäivityksiä.

- Synkronisaatio: Tuki sekä ennalta määriteltyjen, että käyttäjän toimiin liittyvien tapahtumien määrittelyyn.

- CASE-välineet (engl. Computer-Aided Software Engineering): Sovelluskehitystä helpottavien CASE-välineiden saatavuus.

- Kerroksittainen sovelluskehitys (engl. n-tier development): Mahdollisuus erotella sovellus erillisiin kerroksiin (esim. datan ja näkymien erottaminen toisistaan).

- Dynaaminen datan haku: Mahdollisuus hakea dataa palvelimelta ja viedä dataa palvelimelle ajon aikana.

- Rinnakkaiset (engl. parallel) pyynnöt erillisiin tietolähteisiin: Mahdollisuus hakea dataa yhdestä tai useammasta lähteestä yhtäaikaisesti synkronisesti ja asynkronisesti.

- Personalisaatio: Tuki lokalisoitoijen, saavutettavien (engl. accessibility), useita laitetyyppejä tukevien sovellusten toteutukseen.

- Interaktiivinen yhteistyö: Mahdollisuus reaaliaikaiseen yhteistyöhön eri käyttäjien kanssa.

Bozzon, Comai, Fraternali & Carughi (2006) vertailevat asiakas/palvelin-arkkitehtuurin mukaisia työpöytäsovelluksia, ohut asiakas Web-sovelluksia ja Rikkaita Internet sovelluksia yhdeksän ominaisuuden perusteella (Taulukko 1).

Ominaisuus	Työpöytäsovellus	Ohut asiakas	RIA
Universaali asiakasympäristö	Ei	Kyllä	Kyllä
Asiakkaan asennus	Monimutkainen	Yksinkertainen	Yksinkertainen
Interaktio ominaisuudet	Kyllä	Rajoitetut	Rikkaat
Palvelimen liiketoimintalogiikka	Kyllä	Kyllä	Kyllä
Asiakkaan liiketoimintalogiikka	Kyllä	Rajoitettu	Kyllä
Kokonaiset sivulataukset	Ei	Kyllä	Ei
Toistuva kommunikointi palvelimen kanssa	Ei	Kyllä	Ei
Asiakas-palvelin kommunikaatio	Kyllä	Ei	Kyllä
Toiminta yhteyden katketessa	Kyllä	Ei	Kyllä

Taulukko 1: Aiempien Web-sovellusten ja rikkaiden Internet-sovellusten vertailu

Edellä esitetyissä RIA–sovelluksia koskevissa listauksissa ei ole sinällään ristiriitaisuuksia toistensa kanssa. Ominaisuuksien ryhmittelyssä ja termistössä

on käytetty hieman erilaisia lähestymistapoja ja tästä syystä RIAn vaatimusten vertailu on hieman sekavaa. Lisäksi, vaatimuksia ei ole selitetty kovinkaan kattavasti missään näistä lähteistä. Ominaisuuksien tunnistamisessa on käytetty lähinnä Adoben omia ja puolueettomilla tutkimuslaitoksilla teettämiä teknisiä raportteja (pääasiassa Duhl (2003) ja Allaire (2002)) sekä Garner tutkimuslaitoksen Driver, Valdes & Phifer (2005) tekemää selvitystä rikkaista Internet sovelluksista. Kattavaa tieteellistä tutkimusta rikkaiden Internet-sovellusten vaatimuksista ei ole vielä tehty.

Vaatimukset ovat kuitenkin riittävän selkeästi listattuna, jotta niiden yhteneväisyyksiä voidaan tunnistaa. Seuraavassa vaatimuksia on jaoteltu uudestaan kuuden eri kategorian alle. Kategoriat ovat suoritusympäristö, käytettävyys, datan haku, multimedia, CASE-välineet ja kerroksittainen kehitys:

Suoritusympäristö:

- Yleinen, helposti saatavilla oleva suoritusympäristö, kuten Web-selain
- Sovelluksen asentamisen suoritusympäristöönsä tulisi olla mahdollisimman helppoa
- Suoritusympäristössä tulisi voida suorittaa vain turvallista koodia, joka ei vaaranna asiakkaan tietoturvallisuutta

Käytettävyys:

- Rikas interaktio. Mahdollisuus määritellä, että miten sovellus reagoi käyttäjän toimiin. Sekä käyttäjän toimiin liittyvien tapahtumien että ennalta määrättyjen tapahtumien määrittelemisen tulisi olla mahdollista
- Sovellusten tulisi käyttää samanlaisia käytänteitä kuin työpöytäsovellusten käytön opettelemisen helpottamiseksi. Sovellusten

tulisi tarjota työpöytäsovellusten tapaisia interaktiivisa käyttöliittymäkomponetteja

- Häiritseviä ruudunpäivityksiä tulisi pystyä välttämään
- Tuki lokalisoitoijen, saavutettavien (engl. accessibility), useita laitetyyppejä tukevien sovellusten toteutukseen.

Datan haku:

- Datan hakuun liittyvien vasteaikojen tulisi olla mahdollisimman lyhyet. Asynkronista kommunkointia tulisi käyttää kokonaisten sivulatausten välttämiseksi
- Dataa tulee pystyä viemään palvelimelle ja hakemaan palvelimelta ajonaikana asynkronisesti.
- Tuki useiden rinnakkaisten pyyntöjen suorittamiseen eri tietolähteisiin samanaikaisesti
- Sovelluksen pitää pystyä tarjoamaan mahdollisuus sovelluksen rajattuun käyttöön palvelinyhteyden katketessa

Multimedia:

- Mahdollisuus käyttää kuvaa, ääntä, videota, reaaliaikaista ja streaming multimediaa

Kerroksittainen kehitys:

- Mahdollisuus erotella sovellus erillisiin kerroksiin (esim. datan ja näkymien erottaminen toisistaan)
- Myös asiakkaan puolelle tulee voida toteuttaa liiketoimintalogiikkaa

CASE- välineiden tuki:

- Asiakaspuolen kasvaneen monimutkaisuuden takia on tärkeää, että myös sovelluksen asiakaspuolen kehitystä voitaisiin tukea CASE-välineiden avulla

3.3 Vertailu aiempiin Web-sovellusten vaatimuksiin

Edellä esitetyt rikkaiden Internet-sovellusten vaatimukset määrittelevät Web-sovelluksen asiakkaan vaatimukset huomattavasti laajemmin, kuin esimerkiksi Conallenin (2003) esittämät Web-sovellusten arkkitehtuurityylit. Conallenin (2003) mukaan rikasta Web-asiakasta hyödyntäviin Web-sovelluksiin luetaan kaikki Web-sovellukset, joissa asiakkaan puolella on käytetty liitännäisiä tai esimerkiksi skriptausta toteuttamaan osa liiketoimintalogiikasta. Edellä esitetyt vaatimukset asettavat kuitenkin paljon muitakin vaatimuksia asiakkaalle, kuten asynkronista kommunikointia palvelimen kanssa.

Rikkaita Internet-sovelluksia on esitetty uudenlaiseksi Web-sovellusten kategoriaksi. Niille esitettyjen vaatimusten ollessa huomattavasti kattavampia asiakkaan osalta, näin näyttäisi myös olevan.

4 YHTEENVETO

Web-sovellusten suuren suosion ja sovelluksiin kohdistuvien kehitysvaatimuksien myötä on noussut esille uusi sukupolvi Web-sovellustekniikoita. Web-sovellustekniikoiden fokuksessa on ollut perinteisissä ohut asiakas Web-sovelluksissa ilmenneiden ongelmien ratkaiseminen ja työpöytäsovellusmaisempien käyttöliittymien toteutus. Yleisin näistä sovelluksista käytetty nimitys on rikas Internet sovellus. Tekniikoita on ilmestynyt markkinoille suuri määrä eri valmistajien toimesta, mutta alan tieteellinen tutkimus on vielä keskenräistä.

Tämän tutkielman tarkoituksena oli selvittää, että mitä rikkailla Internet sovelluksilla tarkoitetaan, minkälaisia vaatimuksia sovelluksille on asetettu aiemman tutkimuksen pohjalta sekä minkälaisia erilaisia lähestymistapoja sovellusten toteuttamiseen on olemassa. Tutkielmassa lähdettiin liikkeelle ohut asiakas Web-sovellusten tutkimisesta. Aimpien tekniikoiden käsittelyn jälkeen edettiin rikkaiden Internet-sovellusten arkkitehtuurin ja esitettyjen lähestymistapojen tutkimiseen. Tämän jälkeen edettiin aiemmassa tutkimuksessa esitettyjen sovellusten vaatimuksen kartoittamiseen. Eri tutkijaryhmien esittämistä vaatimuksista johdettiin uudelleen organisoidut vaatimukset rikkaille Internet sovelluksille ja vaatimuksia vertailtiin aiemmin esitettyjen Web-sovellusten tyylien vaatimuksiin.

Rikkaat Internet-sovellukset kaventavat kuilua Web-sovellusten ja työpöytäsovellusten välillä. Rikkaat Internet-sovellukset yhdistävät ohut asiakas Web-sovellusten ja työpöytäsovellusten hyviä puolia. Rikkaat Internet-sovellukset siirtävät suuren osan esityslogiikasta ja osan liiketoimintalogiikasta asiakkaan puolelle. Ne ovat laajalti saavutettavissa olevia Webin kautta toimitettavia sovelluksia, jotka eivät vaadi monimutkaisia asennustoimia asiakkaan puolella. Sovelluksia ajetaan joko Web-selaimessa tai Web-selaimen ulkopuolella suojatussa suoritusympäristössä. RIAt tarjovat mahdollisuuden

rikkaan interaktion määrittelyyn asiakkaan kanssa minimioiden asiakkaan ja palvelimen välisen tiedonsiirron käyttämällä asynkronista kommunikointia palvelimen kanssa. Asynkronisen kommunkoinnin mahdollistamana sovelluksen asiakkaan puoleinen osa voi toimia myös palvelinyhteyden katketessa.

Rikkaiden Internet-sovelluksen vaatimusten todettiin olevan huomattavasti laajempia kuin aiemmin esitettyjen Web-sovellusten arkkitehtuurien asiakasta koskevien vaatimusten. Tästä syystä, todettiin, että rikkaat Internet-sovellukset eroavat aiemmista Web-sovellusten tyyleistä.

Rikkaiden Internet-sovellusten asiakkaan puoleiseen toteutukseen on esitetty useita erilaisia lähestymistapoja, jotka voidaan jakaa karkeasti Web-selaimen päällä toimiviin tekniikoihin ja selaimen ulkopuolisissa suoritusympäristöissä toimiviin tekniikoihin. Web-selaimen päällä toimivat tekniikat voidaan jakaa edelleen skriptaukseen perustuviin tekniikoihin (Ajax), liitännäisiin perustuviin tekniikoihin sekä selaimeen perustuviin tekniikoihin. Aiemmassa tutkimuksessa on ristiriitaisuutta siitä, että pitäisikö Web-selaimen ulkopuolella suoritettavia sovelluksia kutsua rikkaiksi Internet sovelluksiksi.

RIA-tekniikoiden hyödyntäminen Web-sovelluksen toteutuksessa tarjoaa joitakin selviä etuja lähestymistavasta riippumatta. Tällaisia hyviä puolia ovat esimerkiksi minimoitu tiedonsiirto asiakkaan ja palvelimen välillä sekä jouhevampi sovelluksen käyttäminen.

Alan tutkimus on edelleen melko puutteellista ja tutkimussarkaa on paljon. Erilaiset rikkaiden Internet-sovellusten toteutukseen käytettävät lähestymistavat eroavat melko paljon toisistaan ja niiden soveltuvuus erilaisiin sovelluskehitystarpeisiin on yksi mielenkiintoinen tutkimusaihe. Lisäksi saman lähestymistavan alle kuuluvien tekniikoiden keskinäinen vertailu ja se, että mitkä tekniikoista mahdollistavat RIAlle asetettujen vaatimusten täyttäminen ovat muita tulevaisuuden tutkimuksen aiheita.

LÄHDELUETTELO

Allaire, J., 2002, Macromedie Flash MX – A Next generation rich client, Tekninen raportti, Adobe, [viitattu 20.4.2007], saatavilla WWW-osoitteesta < http://www.adobe.com/resources/business/rich_internet_apps/whitepapers.html>

Antoniol, G. Di Penta, M., Zazzara, M. 2004, Understanding Web applications through dynamic analysis, Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop, (120-129)

Bass, L., Clements, P., Katzman, R., 2003, Software Architecture in Practice 2nd ed., Pearson Education inc.

Bozzon, A., Comai, S., Fraternali, P., Carughi, G.T., 2006, Conceptual modeling and code generation for rich internet applications, Proceedings of the 6th international conference on Web engineering, ACM Press, (353-360)

Conallen, J., 2003, Building Web Applications with UML 2nd ed., Pearson Education inc.

Document Object Model (DOM) Level 3 Core Specification, 2004, W3C, saatavilla WWW-osoitteesta <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/>

Elbaum, S., Chilakamarri,K-R., Fisher, M., Rothermel, G., 2006, Web application characterization through directed requests, Proceedings of the 2006 international workshop on Dynamic systems analysis, ACM Press, (49-56)

Fleury, T., Basney,T., Welch, T., 2006, Single sign-on for java web start applications using myproxy, Proceedings of the 3rd ACM workshop on Secure web services, Proceedings of the 3rd ACM workshop on Secure web services, ACM Press, (95-102)

Fowler, M., 2003, Patterns of enterprise application architecture, Pearson Education inc.

Castro, J., Kolp, M., Mylopoulos, J., 2001, A Requirements-Driven Development Methodology, Proceedings of Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Springer Berlin / Heidelberg, (108-123)

Driver, M., Valdes, R., Phifer, G., 2005, Rich Internet Applications are the next Evolution of the Web, Tekninen raportti, Gartner

Duhl, J., Rich Internet Applications, 2003, Tekninen raportti, IDC, [viitattu 20.4.2007], Saatavilla WWW-osoitteesta < http://www.adobe.com/resources/business/rich_internet_apps/whitepapers.html>

ECMAScript Language Specification, 1999, Ecma International, [viitattu 7.4.2007], saatavilla WWW-osoitteesta < http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

Haustein, S., Pleumann, J., 2005, A model-driven runtime environment for Web applications, Teoksessa Software and Systems modeling, Springer Berlin / Heidelberg, (443-458)

HTML 4.01 Specification, 1999, W3C, [viitattu 5.4.2007], saatavilla WWW-osoitteesta <http://www.w3.org/TR/html401/>

Hypertext Transfer Protocol – HTTP/1.1, 1999, The Internet Society, [viitattu 5.4.2007] saatavilla WWW-osoitteesta <http://www.ietf.org/rfc/rfc2616.txt>

Mesbah, A, van Deursen, A., 2007, An Architectural Style for Ajax, Proceedings of Software Architecture, 2007. WICSA '07. The Working IEEE/IFIP Conference, (9-9)

Open source applications foundation, 2007, Survey of AJAX/JavaScript Libraries, [viitattu 21.4.2007], saatavilla WWW-osoitteesta <http://wiki.osafoundation.org/Projects/AjaxLibraries>

Pascarello, E., Crane, D., Ajax in action, 2005, Manning Publications Co.

Preciado, J.C., Linaje, M., Sánchez, F., Comai, S., 2005, Necessity of methodologies to model rich Internet applications, Proceedings of Web Site Evolution, 2005. (WSE 2005). Seventh IEEE International Symposium on, (7-13)

Puder, A., A Code Migration Framework for AJAX Applications, 2006, Teoksessa Distributed Applications and Interoperable Systems, Springer Berlin / Heidelberg, (138-151)

Yu, J., Benatallah, B., Casati F., Saint-Paul, R., 2006, OpenXUP: an alternative approach to developing highly interactive web applications, Proceedings of the 6th international conference on Web engineering, ACM Press, (289-296)