KEMI-TORNION AMMATTIKORKEAKOULU

Pohjakarttasovellus

Inga Hiltunen

Tietojenkäsittelyn koulutusohjelman opinnäytetyö
Web-asiantuntijan suuntautumisvaihtoehto
Tradenomi
TORNIO 2011

TIIVISTELMÄ

Hiltunen, Inga. 2010. Pohjakarttasovellus. Opinnäytetyö. Kemi-Tornion ammattikorkeakoulu. Kaupan ja kulttuurin toimiala. 53 sivua. 1 liite.

Opinnäytetyön tilaajana on KTAMKin Kirjasto- ja tietopalvelut. Tavoitteena on luoda Kemin oppimiskeskuksen käyttöön interaktiivisen pohjakarttasovelluksen vaatimusmäärittely sekä alustava demoversio ilman varsinaista sisältöä, mutta varustettuna sisällön esimerkkeinä kuten valokuva sekä selostusteksti.

Tutkimuksessa on käytetty konstruktiivista tutkimusmenetelmää. Tämän menetelmän valintaan vaikuttaa tutkimuksen tavoitettu ja odotettu tulos, eli sovelluksen toimiva demoversio. Tutkimuksen konstruktiivisessa osuudessa käsitellään ratkaisun elinkaarta. Silloin ensin tutkitaan tapauksen ongelmat, sitten mallinnetaan konstruktio eli rakennetaan ratkaisumalli, joka teoreettisesti vastaa ongelmaan. Sitten kerätään aineisto ja sen pohjalla valmistetaan tarpeisiin vastaava sovelluksen demoversio. Suunnittelun vaiheessa käsittelen vaatimusmäärittelyn mukaista sovellusta kokonaisuudessa. Silloin käytän UML-mallinnuskieltä ja free 30-Day Microsoft Visio 2003 trial ohjelmaa. Sovelluksen toteutustyökaluina ovat free 30-Day Adobe Photoshop CS2 trial ohjelma, ActionScript 2.0-ohjelmointikieli, XML-merkintäkieli, lähdekoodieditori notepad++ ja free 30-Day Adobe Flash 8 trial ohjelma.

Tämän opinnäytetyön tuloksena syntyi KTAMKin Kirjasto- ja tietopalvelujen puolelta tilatulle Flash-sovellukselle vaatimusmäärittely sekä alustava demoversio. Demoversio on varustettu kahdella painikkeella, joiden toiminnan pohjalla on rakennettu sovelluksen interaktiivisuus. Koska toimeksiantona on interaktiivinen pohjakarttasovelluksen lopullinen versio kaikkineen mahdollisuuksineen, niin sovellusta tulee jatkokehittää vaatimusmäärittelyn mukaisilla ominaisuuksilla.

Asiasanat: ActionScript, XML, Flash, ohjelmointi

ABSTRACT

Hiltunen, Inga 2011. Floor map application. Bachelor's Thesis. Kemi-Tornio University of Applied Sciences. Business and Culture. Pages 53. Appendix 1.

The thesis was assigned by the Kemi-Tornio University of Applied Sciences (KTUAS) Library and Information Services. The goal of the thesis is to create the requirement specification and a preliminary version of the demo without any real content, but with content examples such as photo and description text for the interactive floor map of the learning center in Kemi.

The thesis is based on constructive research method. This method has been chosen because of the expected result of the thesis, namely the functional demo version of the application. In the constructive research section I discuss the solution's life cycle. Following the discussion of the life cycle, I research case study problems. This stage is followed by building a model construction, which corresponds to the theoretical problem. I collected theoretical material from books and on the basis of the literature I will build a demo version which meets the product needs. During the planning stage, I deal with the application requirements specification as a whole. I used UML modeling language and free 30-Day Microsoft Visio 2003 trial program. Application development tools used were as follows: free 30-Day Trial Adobe Photoshop CS2 program, ActionScript 2.0 programming language, XML markup language, open source notepad + + program and free 30-Day Trial Adobe Flash 8 trial program.

As a result of this thesis a preliminary version of the demo of the Flash application was created which was ordered by the KTUAS Library and Information Services. The demo version is equipped with two buttons and the interactivity of the application is based on the performances of the buttons. Since the goal of the original assignment by the KTUAS Library and Information Services is a completed version of interactive floor map with all options and functionalities, the application's demo version must be further developed in compliance with the requirement specification.

Keywords: ActionScript, XML, Flash, programming

SISÄLTÖ

1 JOHDANTO

1.1 Työn lähtötilanteen kuvaus

Tämän opinnäytetyön tilaajana on Kemi – Tornion ammattikorkeakoulun Kirjasto- ja tietopalvelut. Kirjasto koostuu neljästä yksiköstä, jotka sijaitsevat kahdessa eri kaupungissa sekä niiden kaupunkien eri osissa. Kirjastoyksikköjen moninaisuus tuottaa Kemi – Tornionlaakson koulutuskuntayhtymä Lappian uusille opiskelijoille opiskelun alussa vaikeuksia kirjastopalveluissa orientoitumiseen. Opinnäytetyön tilaaja kääntyi Kemi – Tornion ammattikorkeakoulussa toimivaan Liikeakatemiaan toimeksiannolla. Toimeksianto koskee sovellusta, joka esittelisi Kemin oppimiskeskuksen tilojen pohjakarttaa. Sovelluksen lopullisen version käyttäjinä ovat koulutuskuntayhtymän Lappia opiskelijat sekä muut kohderyhmät. Silloin sovelluksen avulla on mahdollista hahmottaa Kemin oppimiskeskuksen tiloja, palvelupisteiden sijaintia sekä kirjahyllyjen sisältöä virtuaalisesti interaktiivisen toiminnan avulla.

1.2 Työn tavoitteet, tutkimusongelma ja rajaus

Opinnäytetyön tavoitteena on laatia Kemi – Tornion ammattikorkeakoulun Kirjasto- ja tietopalvelujen tarpeisiin pohjakarttasovelluksen vaatimusmäärittely sekä alustava demoversio. Demoversio toimii pohjana tulevalle vaatimusmäärittelyn mukaiselle pohjakarttasovelluksen lopulliselle versiolle kaikkineen ominaisuuksineen.

Tutkimusongelmana on mallintaa ja rakentaa selkeä ja looginen pohjakarttasovelluksen alustava demoversio. Tässä demoversiossa toteutetaan yksinkertaista vuorovaikutusprosessia: sovelluksen käyttöliittymän kautta XML-dokumentista haetaan tietty tieto ja tulostetaan takaisin käyttöliittymän tulostusalueelle. Haasteena on Flash-sovellukseen ulkopuolisen tiedon lataus XML-dokumentista, joka sijaitsee paikallisella koneella. **Teoriaosuudessa** tarkastelen interaktiivisen Web-pohjaisen sovelluksen suunnittelu- sekä käytettävyyden periaatteita. **Toteutusosiossa** valituilla työkaluilla luon sovelluksen alustavaa demoversiota. Työn tuloksena on sovelluksen alkio, jota jatkokehitettään vaatimusmäärittelyn mukaisesti.

Vaatimusmäärittelyä laatiessa minulle selvisi, että toimeksiantajan toivomuksen mukainen sovelluksen kokonaisuus koostuu enemmän kuin kahdesta osasta. Jokainen

osa tarvitsee toteutusta varten omaa toteutustekniikkaa sekä ohjelmointikieltä. Samoin, piti määritellä osien yhteiset rajapinnat. Rajoitan työtäni siten, että teen vaatimusmäärittelyn pohjalta sovelluksen toimivan demoversion ilman ylläpito-osiota sekä tietokantaa. Samoin, tämän opinnäytetyön piireissä en valmista sovelluksen sisältöä. Sovelluksen demoversion toiminnallisuuden esimerkkeinä toimivat yksi valokuva sekä yksi selostustekstitiedosto.

1.3 Tutkimusmenetelmät ja työvälineet

Tutkimusmenetelmänä valitsin konstruktiivisen tutkimuksen, koska opinnäytetyön tavoitteena on tiettyyn tapaukseen vastaavan ongelmaratkaisun löytäminen ja sitten sen valmistaminen. Silloin luodaan uusi menettely tai väline, jolla ratkaistaan jokin ongelma. Tuloksena on sovelluksen alustava demoversio pohjana jatkokehitykselle. Konstruktiivinen tutkimus voidaan nähdä eräänä soveltavan tutkimuksen muotona, jolle on ominaista sellaisen uuden tiedon tuottaminen, joka tähtää johonkin sovellutukseen tai tavoitteeseen (Kasanen, Lukka & Siitonen 1991). Minun työni, konstruktiivisena tutkimuksena voidaan jakaa prosessiin:

Käytännön ongelman havaitseminen: Kirjasto- ja tietopalvelun henkilökunta huomasi, että kirjasto tarvitsee asiakaskäyttöön sovelluksen, jota käyttämällä kirjaston asiakkaat voivat tutustua ennakkoon Kemin oppimiskeskuksen toimitiloihin sekä palvelupisteiden sijaintiin ennen varsinaista käyntiä.

Alustavan ratkaisumallin eli konstruktion hahmottaminen: Ensimmäisellä tapaamisella minä ja toimeksiantaja keskustelimme niistä vaatimuksista, joihin valmiin sovelluksen on vastattava. Keskustelun tuloksena minä sain alustavan ymmärryksen ongelman ratkaisuun. Sen jälkeen minä etsin ja tutkin aiheeseen sopivaa tutkimusaineistoa.

Mallin rakentaminen: Minä rakensin alustavan sovelluksen demo käyttämällä UML – mallinnuskieltä teoriatietojen pohjalla. Dokumentoin vaatimusmäärittelyyn.

Varsinainen toteutusvaihe: Tein pohjakarttasovelluksen alustavan demoversion sopivilla työkaluilla ja testasin sovelluksen toimintaa työasemallani. Tarvittaessa, jos minulla oli jotain kysyttävää, niin pidin yhteyttä toimeksiantajaan.

Konstruktiivisen tutkimuksen tiedonkeruumenetelmänä pidin tietojenkäsittelyalan kirjallisuuslähteisiin tutustumista, sekä opiskeluaikana tehtyjen omien muistiinpanojen hyödyntämistä. Kirjallisuus käsittelee ohjelmistotuotantoa ja käytettävyyttä.

Huomasin, että tutkimusmaailmassa harvoin käytetään jotain yhtä olemassa olevista menetelmistä, koska usein tutkijaa kiinnostava tutkimuksen ongelma tai kohde on harvoin eristetty muun maailman ilmiöiden vaikutteista. Esimerkiksi sovelluksen Web-pohjainen käyttöliittymä on interaktiivinen eli edellyttää molemminpuolista vuorovaikutusta. Onnistunut vuorovaikutus asiakkaan ja järjestelmän välillä edellyttää sitä, että asiakkaat osaavat käyttää käyttöliittymän toiminnallisia ominaisuuksia oikein. Tässä tapauksessa tarvittavaa osaamista voidaan taataa käyttöopastuksella, joka on yleensä toiminnallisen tutkimuksen vallan alainen. Hanna Vilkka ja Tiina Airaksinen (2003) omassa työssä sanovat, että toiminnallisuus opinnäytetyössä tavoittelee käytännön toiminnan ohjeistamista, esimerkiksi käyttöohjeella. Toteutustapa on aika laaja, kirjasta tapahtumaan – toteutuu viestinnällisellä tai visuaalisella keinolla. Sovellukselle tehdään käyttöohje, joka on suunnattu loppukäyttäjille eli asiakkaille. Käyttöohje tulee olemaan integroituna sovelluksen käyttöliittymän lisäkomponenttina. Toiminnallisen tutkimuksen tiedonkeruumenetelmänä valitsin laadullisen haastattelun. Aineistoa kerätään joko yksilö- tai ryhmähaastattelun aikana riippuen siitä, millaista tietoa selvityksellä halutaan oman idean sisällöksi tai tueksi. Yksilöhaastatteluna suositeltavia aineiston keräämisen tapoja ovat joko lomake- tai teemahaastattelu. Teemahaastattelu on puolistrukturoituna vapaampi tapa kerätä aineistoa ja toimii toiminnallisena opinnäytetyössä silloin, kun tavoitteena on kerätä tietoa jostakin tietystä teemasta (Vilkka & Airaksinen 2003, 63). Tämä vapaamuotoinen teemahaastattelu oli suoritettu toimeksiantajani kanssa ensimmäisen tapaamisen vaiheessa, kun me keskustelimme sovelluksen käyttöliittymän tehtävistä. Haastattelun aikana saadut tiedot on hyödynnetty vaatimusmäärittelyssä.

1.4 Käsitteitä

CSS (Cascading Style Sheets) HTML-sivuihin liittyä ulkoinen tyylitiedosto, joka määrittelee sivuilla käytetyt muotoiluasetukset. Kun asetuksiin tehdään muutoksia (esimerkiksi tekstin kokoa tai väriä muutetaan), vaikutus näkyy kaikilla niillä HTML-sivuilla, joissa viitataan kyseiseen CSS-tiedostoon. (Järvinen 2003, 94.)

Demoversio on esittelyversio jostain tietokonepelistä tai -ohjelmasta.

Digitaalinen valokuva eroaa filmillä otetusta valokuvasta lähinnä vain tallennustavaltaan. Filmikameran kuva on filmillä negatiivina, dialla tai vedostettuna paperikuvaksi. Digitaalikameran kuva on numeerisena tietona jollain muistilla.(Rinne 2011.)

Komponenttikaavio (component diagram) kuvaa riippuvuudet ohjelmistokomponenttien välillä mm. lähdekoodi, objektikoodi ja dokumentit. Järjestelmä rakentuu komponenteista. Komponenttikaavion tekeminen kuuluu ohjelmistoprosessin suunnitteluvaiheeseen. (Oulun seudun ammattikorkeakoulu 2011.)

Käyttötapauskaavio (use case diagram) kuvaa järjestelmän käyttötapausten väliset suhteet ja käyttötapauksiin osallistuvat järjestelmän ulkoiset toimijat. Käyttötapauskaavioita käytetään järjestelmän käyttöyhteyden ja vaatimusten korkean tason kuvaukseen.(Koskimies 2000, 127.)

Luokkakaavio (class diagram, static structure diagram) kuvaa järjestelmään kuuluvia luokkia ja niiden välisiä suhteita. Luokkakaavioita käytetään järjestelmän staattisen rakenteen kuvaukseen.(Koskimies 2000, 128.)

Panoraamakuva on yhdestä pisteestä koottu kuvasarja, jonka kuvakulma on 360 astetta ja kattaa näkymän horisontista horisonttiin (Haggren, 1999).

Sekvenssikaavio (sequence diagram) koostuu tiettyyn vuorovaikutukseen liittyvistä olioista ja niiden välisistä sanomista (message). Sekvenssikaaviota käytetään kuvaamaan oliojoukon välistä vuorovaikutusta tietyssä tilanteessa.(Koskimies 2000, 129–130.)

UML (Unified Modeling Language) on nopeasti yleistynyt graafinen ohjelmistojen mallinnuskieli. UML:ää käytetään sekä ihmisten väliseen kommunikointiin, että ihmisen ja koneen väliseen vuorovaikutukseen. Monet työkalut tukevat UML:n käyttöä mallien laatimisessa, koodin tuottamisessa malleista, sekä mallien tuottamisessa koodista. (Koskimies, Koskinen, Maunumaa, Peltonen, Selonen, Siikarla & Systä 2010.)

XML (Extensible Markup Language) Kuvauskieli, jolla määritellään WWW-sivulla olevan datan sisältö. HTML-kieli määrittelee vain tavan, jolla data esitetään; XML antaa datalle merkityksen, mikä auttaa hakukoneita ja erilaisia tietokantasovelluksia. Kieli on laajennettavaa niin, että kuvauskoodeja (kuten <HINTA> 100,00 </HINTA>) voidaan luoda sovellus- ja tapauskohtaisesti. (Järvinen 2003, 783.)

XMLSocket - yhteys on kaksisuuntainen ja pysyvä yhteys käyttäjän ja palvelimen välillä. Käyttäjän muodostettua yhteyden palvelimeen yhteys pysyy auki, kunnes käyttäjä tai palvelin sulkee yhteyden. Tämän yhteyden läpi voidaan lähettää rajaton määrä viestejä.(Manninen & Marttila 2006, 395.)

2 ARKKITEHTUURIMALLIN VALINTA

2.1 Web - sovelluksen suunnitteluperiaatteet

Haikala & Märijärvi (2004, 312 - 319) hyvin laajasti käsittelevät ohjelmistosuunnittelun yleisiä periaatteita, josta erotin lyhyesti omiin sanoin tärkeät kohdat käsiteineen:

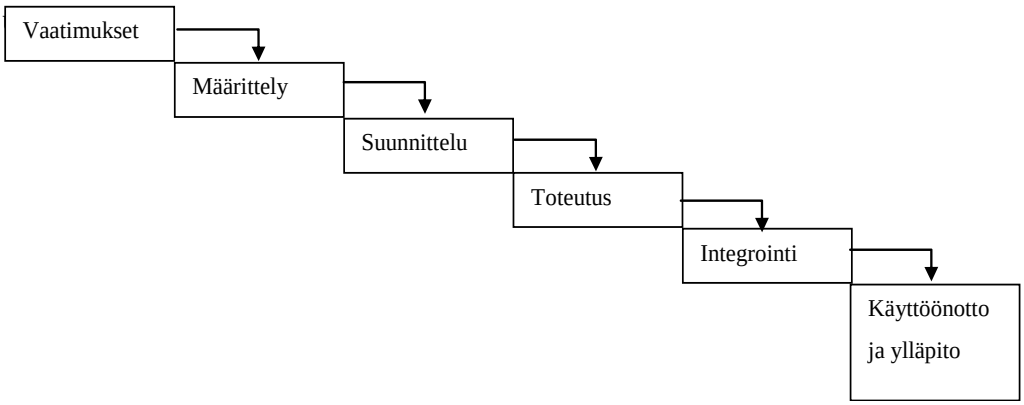
Ohjelma usein käsittelee monimutkaisia ongelmia, joita kannattaa ratkaista mahdollisimman yksinkertaisilla ja suoraviivaisilla tavoilla. Monimutkaisuuden voidaan hallita niin sanotulla **osittamisella**. Osittaminen tapahtuu siten, että kokonaisuus jaetaan osiin tai moduuleihin, joita tarvittaessa on mahdollista muuttaa, hajottamatta kokonaisen ohjelman kokoonpanoa.

On myös tärkeä suunnitella moduulit siten, että tulevien tapahtuvien muutoksien sijainti olisi mahdollisimman **lokalisoitu**. Lokaalisointi mahdollistaa toteuttaa, testata tai muuttaa erikseen ohjelmiston osia. Myös moduulin sisäinen toteutus on lokalisoitava niin, että sen sisäisen rakenteen ratkaisut toistuvat mahdollisimman harvoin ohjelman eri kohdissa. Huonon lokaalisuuden yhtenä esimerkkinä voi olla saman koodipätkän toistuminen monessa kohdassa. Toteuttamalla sovellus PHP ohjelmointikielellä, asian voidaan korjata esimerkiksi include – funktiolla.

Ohjelmiston moduuleiden suunnittelussa käytetään **abstraktioita**. Abstraktio on malli, jolla kuvataan asiasta oleellisen tiedon käsitteiden avulla. Abstraktion esimerkkinä voi olla autoa kuvaava tietorakenne, joka sisältää auton merkin, värin ja valmistumisvuoden. Abstraktio näkyy sovelluksen käyttäjälle rajapinnalla. Oikein laadittujen abstraktioiden on oltava sekä ymmärrettäviä että muunneltavia. Abstraktion ymmärrettävyys johtaa rajapinnan toimintalogiikan selkeyteen ja muunneltavuus perustuu siihen, että abstraktion toteutus tarvittaessa voidaan vaihtaa rajapintaan koskematta.

Nykyään ohjelmistoja tehdään tietyn **toteutusfilosofian** eli arkkitehtuurityylin mukaan. Toteutusfilosofia on tavallaan yhdenmukaisien toteutusperiaatteiden kokoelma, joiden avulla ohjelmiston piirteet toteutetaan. Esimerkkejä toteutusfilosofioista on olemassa runsaasti. Web-sovellukset jaotellaan arkkitehtuurityyliltään kerrosarkkitehtuurin ja sen takia valitsin omalle sovellukselle kerroksittaisen arkkitehtuurimallin.

Ohjelmistotuotannossa on olemassa yksi käsite, jota Haikalan & Märijärven (2004, 36 - 37) mukaan sanotaan ohjelmiston kehitystyön elinkaariksi. Elinkaarta voidaan jakaa vaiheisiin vesiputousmallilla. Mallista on olemassa useita muunnelmia, mutta yleiset vaiheet ovat kuitenkin aina samat: määrittely-, suunnittelu- ja toteutusvaihe. Seuraava tekemäni kuva esittelee vesiputousmallin muunnelman, jota käytetään opinnäytetyössäni.



Kuva 1. Sovellustuotannon vesiputousmalli

2.2 Kerroksittainen arkkitehtuurimalli

Kerroksittainen (layered) ohjelmistoarkkitehtuuri tarkoittaa ohjelmiston rakennetta, joka koostuu loogisesti yhtenäisistä eri käsitetasoilla olevista kerroksista siten, että ylempi kerros käyttää alemman kerroksen palveluja (Koskimies 2000, 212). Eli ensimmäisellä tai alemmalla tasolla oleva kerros tarjoaa palveluja kaikille siitä ylimmällä oleville kerroksille. Seuraava tekemäni kuva auttaa havainnollistaa nuolien avulla kerroksesta kerrokseen tarjoavien palvelujen määrän vähenemisen.

	Käyttöliittymä - esitys kerros	
	Sovelluslogiikkakerros	
	Tietokerros	

Kuva 2. Kerroksittainen arkkitehtuurimalli

Koskimiehen (2000, 215) mukaan, kerroksittaisen arkkitehtuurin etu johtuu sen selkeydestä ja yksinkertaisesta rakenteesta. Luetellut ominaisuudet tekevät mahdolliseksi ohjelmiston kehittämisen ja ymmärtämisen käsitetasoltaan kohoavissa kerroksissa. Samoin vähenee ohjelmanosien riippuvuus toisistaan. Kerroksittainen arkkitehtuuri tukee myös ylläpitoa ja muunneltavuutta, jos vain kerrosten välillä on hyvin hallitut rajapinnat. Kerroksia voidaan muuttaa tai vaihtaa toiseksi, kunhan ne noudattavat edelleen samoja käyttö- ja toteutusrajapintoja. Kerroksittaisella arkkitehtuurimallilla on olemassa vielä yksi arvo-ominaisuus: malli tukee mahdollisuuden rakentaa tietyn kerroksen tarjoaman palvelun varaan uusi järjestelmä. Toisin sanoen, rajapinnan kahden eri järjestelmien välillä.

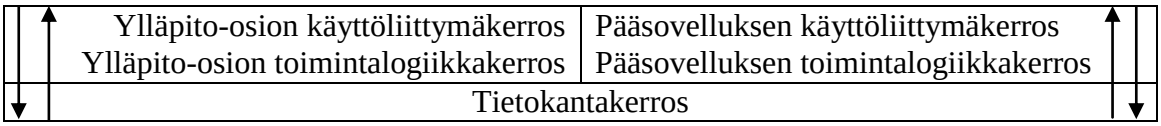
Valitsin sovelluksen lopulliselle versiolle kerroksittaisen arkkitehtuurimallin koska sen rakenne koostuu viidestä osiosta, jotka periaatteessa ovat itsenäisiä toisistaan. Ne osat ovat tietokanta, ylläpito-osion käyttöliittymä, ylläpito-osion toimintalogiikkakerros, pääsovelluksen käyttöliittymä ja pääsovelluksen toimintalogiikkakerros.

2.2.1 Kerroksen hierarkkisuus ja ohituksettomuus

Koskimiehen (2000, 212 - 213) mukaan, kerroksittainen arkkitehtuurimalli muodoltaan on hierarkkinen ja ohitukseton. Hierarkkisuus tarkoittaa kutsujen kulkemista joko samalla tasolla tai korkeammalta alempaan. Ohituksettomuus tarkoittaa sitä, että kutsut eivät ohita kerroksia vaan kohdistuvat aina välittömästi seuraavaan kerrokseen. Koska UML mallinnuskielen työkaluissa ei ole tarkoitettuja välineitä juuri kerrosarkkitehtuurimallille, niin malli kuvataan luokkien, rajapintojen tai komponenttien välisillä riippuvuuksilla.

Tosielämässä käytetyt kerroksittaiset arkkitehtuurimallit ovat harvoin puhtaita. Ja yksi piirteistä, joka rikkoo sen, on kerrosten välinen takaisinkutsu. Vaatimusmäärittelyn mukainen pohjakarttasovellus on aika yksinkertainen ja sen rakennuksessa ei ole paljon kerroksia, mutta sekin ei ole mallin mukainen (katso kuvaa 2.). Käyttöliittymäkerros jakautuu kahteen osaan: pääsovellus- ja ylläpito-osion käyttöliittymiin, joilla ei ole yhteistä rajapintaa. Samoin sovelluslogiikkakerros jakautuu kahteen osaan: ylläpito-osion- ja pääsovelluksen toimintalogiikkakerrokseen. Niiden alla on yhteinen tietokantakerros.

Kerroksittaista arkkitehtuuria käytetään normaalisti siten, että käyttäjä tai jokin toinen ohjelmisto kutsuu jotakin ylimmän kerroksen palveluja. Tämä puolestaan kutsuu alemman tason palveluja, ja kontrolli siirtyy näin alaspäin palatakseen alempien tasojen palvelujen suorituksen jälkeen jälleen ylimmälle tasolle. (Koskimies 2000, 213.) Niin pohjakarttasovellusta käyttävä asiakas pystyy lähettämään tiedusteluja käyttöliittymän kautta (kuvalla 3. nuoli alaspäin) ja sitten vastaanottaa sovelluksen tietovarastossa olevia tiedostoja kuvallisessa tai tekstimuodossa (kuvalla 3. nuoli ylöspäin). Vastaavasti tapahtuu ylläpidonkin tapauksessa: nuoli alaspäin kuvailee käyttäjätunnuksen oikeellisuuden tiedustelua kirjautumissivulta tietokantaan. Nuoli ylöspäin kuvailee takaisin reagointia sisään kirjautumisella taikka virheilmoituksella. Seuraava telemäni kuva esittelee pohjakarttasovelluksen kerroksittaista arkkitehtuurimallia kutsuineen molempiin suuntiin.



Kuva 3. Pohjakarttasovelluksen kerroksittainen arkkitehtuurimalli

2.2.2 Kerroksittaisen arkkitehtuurimallin suunnittelu

Koskimiehen (2000, 216) mukaan kerroksittaisen arkkitehtuurimallin suunnittelu tapahtuu siten, että suunnittelija pohtii seuraavia kymmenen suunnittelu askelta ja etsii niille vastauksia. Kysymykset ovat seuraavia:

1. Määritellään abstrahointikriteerit, joihin kerrosjako perustuu.

2. Määrätään kerrokset ja niiden nimet perustuen abstrahointikriteereihin, sekä määritellään kerrosten tehtävät karkeasti.

3. Määritellään kerrosten tarjoamat palvelut.

4. Määritellään kerrosten tarvitsemat palvelut ja tarkistetaan, että ne löytyvät alemmilta kerroksilta.

5. Tarvittaessa tehdään kerrosjakojen korjaus.

6. Jaetaan kukin kerros komponentteihin, toisiin sanoin moduuleihin, ja määritellään näiden kommunikointi kerroksen sisällä. Komponentilla tarkoitetaan yleensä kokoelmaa toisiinsa liittyviä olioita/luokkia, jotka suorittavat ohjelmassa jotain tehtäväkokonaisuutta (Luukkainen 2010).

7. Määritellään tiedon kulku vierekkäisten kerrosten välillä.

8. Varmistetaan, että alempi kerros ei riipu ylemmästä.

9. Varmistetaan ettei ylempi kerros riipu alemman kerroksen toteutusratkaisusta.

10. Suunnitellaan poikkeustilanteiden käsittely.

Niille kysymyksille sain seuraavat vastaukset:

1. Pohjakarttasovelluksessa on seuraava tehtävien jaottelu kerroksien mukaan:

- asiakas, sovelluksen käyttöliittymä, painike, katsominen
- tietoliikenne, kutsu, vastaus
- tietovarasto, valokuvataulukko, panoraamataulukko, tekstitiedostotaulukko, valokuva, panoraamavalokuva, tekstitiedosto
- ylläpitäjä, superuser, ylläpito-osion käyttöliittymä, kirjautuminen sisään, kirjautuminen ulos, käyttäjätunnus, salasana, lisääminen, poisto, muokkaaminen.

2. Kerrosten nimet ja tehtävät karkeasti:

- tietokantakerros sisältää kuva- ja tekstitiedostot sekä ylläpitäjän tunnukset
- pääsovelluksen käyttöliittymäkerros on tarkoitettu asiakkaiden käyttöön, palauttaa asiakkaalle hänen kutsutut tiedostot
- ylläpito-osion käyttöliittymäkerros on tarkoitettu ylläpitäjän ja käyttöön, kirjauttaa ylläpito-osioon sisään sekä mahdollistaa sisällön ylläpitoa.
- ylläpito-osion toimintalogiikkakerros mahdollistaa kutsujen ja vastauksien liikennettä ylläpito-osion käyttöliittymän ja tietokannan välillä

3. Kerrosten tarjoamat palvelut:

Tietokantakerros tarjoaa **pääsovelluksen käyttöliittymälle** seuraavat palvelut:

- päivitäTiedosto (kuva, panoraama, teksti)
  sen jälkeen, kun ylläpitäjä tai superuser oli päivittänyt olevan tietokannassa sisällön tiedot, niin ne automaattisesti heijastuvat sovelluksen käyttöliittymällä.

Tietokantakerros tarjoaa **pääsovelluksen toimintakerrokselle** seuraavat palvelut:

- vastausTarvittavaTieto (kuva- tai tekstitiedosto)
  XML-dokumentti lähettää kutsun tietokannalle; tietokanta vastauksena palauttaa tarvittavan tiedon XML-dokumentille.

Tietokantakerros tarjoaa **ylläpito-osion käyttöliittymälle** seuraavat palvelut:

- tarkistaOikeellisuus (tunnus, salasana)
  sen jälkeen, kun ylläpitäjä tai superuser oli syöttänyt omat tunnukset ylläpito-osion käyttöliittymällä olevaan kirjautumiskenttään, niin tietokannassa olevat ”oikeat” tunnukset tavallaan ovat avaimella osioon sisään pääsylle.

Tietokantakerros tarjoaa **ylläpito-osion toimintakerrokselle** seuraavat palvelut:

- lataaTiedosto(kuva, panoraama, teksti)
- lisääTiedot(tunnus, salasana, nimi, sukunimi).

Pääsovelluksen käyttöliittymäkerros tarjoaa **asiakkaalle** seuraavat palvelut:

- avaa (valokuva, panoraama, teksti)
  painamalla hiirellä painikkeen, avautuu joko valokuva, panoraamavalokuva tai selostusteksti
- lähennä (valokuva, panoraama)
  painetaan tiettyä painiketta

- loitonna (valokuva, panoraama)
  painetaan tiettyä painiketta
- sovelluksen käyttöliittymäkerros toiminnakseen tarvitsee tietokantakerroksessa olevan päivitetyn sisällön (katso tietokantakerros tarjoaa sovelluksen käyttöliittymälle).

Ylläpito-osion käyttöliittymäkerros tarjoaa **ylläpitäjälle** seuraavat palvelut:

- kirjauduSisään(tunnus, salasana)
  kerroksen käyttöliittymällä olevan kirjautumiskentän avulla päästään tehtävien valikkoon
- valitseTehtävä(kuva- tai tekstitiedoston muokkaus)
  painamalla vastaava linkki, valitaan tehtävää.

Pääsovelluksen toimintalogiikkakerros tarjoaa **pääsovelluksen käyttöliittymäkerrokselle** seuraavat palvelut:

- vastausTarvittavaTieto (kuva- tai tekstitiedosto)
  saatu tarvittava tieto tietokannasta, XML-dokumentti palauttaa sen kutsujalle.

Ylläpito-osion toimintalogiikkakerros tarjoaa **ylläpito-osion käyttöliittymäkerrokselle** seuraavat palvelut:

- valitseTehtävä (valokuvienHallinta, panoraamakuvienHallinta, tekstienHallinta, ylläpittäjäTietojanHallinta)
- suoritaTehtävä ()
- poistaTiedosto(kuva, panoraama, teksti)
- muokkaaTiedosto(teksti)
- poistaTiedot(tunnus, salasana, nimi, sukunimi)
- muokkaaTiedot(tunnus, salasana, nimi, sukunimi)
- kirjauduUlos(käyttäjätunnus, salasana)

4. Kerrosten tarvitsemien palveluiden määrittely ja saannin tarkistus

- tietokantakerros toiminnakseen ei tarvitse mitään palveluja
- pääsovelluksen käyttöliittymä toiminnakseen tarvitsee tulostukseen tarvittavaa tietoa, jota toimittaa XML-dokumentti pääsovelluksen toimintalogiikkakerroksen kautta
- pääsovelluksen toimintalogiikkakerros toiminnakseen tarvitsee eteenpäin välittävän tarvittavan tiedon saantia ”varastosta”, jota saa tietokannasta
- ylläpito-osion käyttöliittymä toiminnakseen tarvitsee syötetyn käyttäjätunnusten oikeellisuuden vahvistusta, jota ”myöntää” tietokanta; mahdollisuuden lisätä, muokata ja poistaa tiettyä tietoa ylläpito-osion toimintalogiikkakerroksen avulla
- ylläpito-osion toimintalogiikkakerros toiminnakseen tarvitsee mahdollisuuden tallentaa muokatut tiedot tietokantaan
- käyttäjä oman toiminnan vastineeksi tarvitsee tulostettuna tiettyä tietoa pääsovelluksen käyttöliittymän kautta
- ylläpitäjä ylläpito-osion käyttöliittymän kautta pääsee ensin kirjautua sisään ja sitten suorittaa tarvittavat tehtävät.

6. Kerrokset komponentteihin ja niiden kommunikointi kerroksen sisällä

- tietokantakerros koostuu riippumattomista toisistaan taulukoista: valokuva, panoraamakuva, teksti, käyttäjä – ei ole mitään välistä kommunikointia
- pääsovelluksen käyttöliittymäkerros: swf-muotoinen esitys
- pääsovelluksen toimintalogiikkakerros: XML-dokumentti
- ylläpito-osion käyttöliittymäkerros: HTML-sivut ja ”sisällä” toimivat PHP-scriptit – PHP-ohjelmointikielen koodipätkät ovat sääntöjen mukaisesti sisäänrakennetut HTML-sivulle
- ylläpito-osion toimintalogiikkakerros: PHP-scriptit.

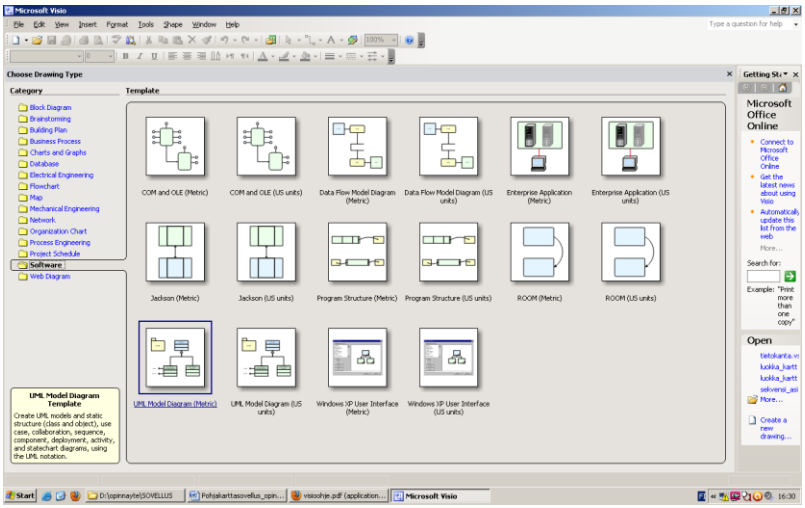
3 PÄÄSOVELLUKSEN ARKKITEHTUURISUUNNITTELU

3.1 UML ja Microsoft Visio 2003

Alustavan ratkaisumallin hahmottamisen vaiheen jälkeen, alkoi mallin hahmottamisen vaihe. Toimeksiantajan kanssa keskustelun perusteella tein vaatimusmäärittelyn ja UML-mallinnuskielen avulla kuvailin kaikki pohjakarttasovelluksesta odotetut ominaisuudet ja toiminnot. Sitten valitsin sopivat ohjelmointikielet ja ohjelmat.

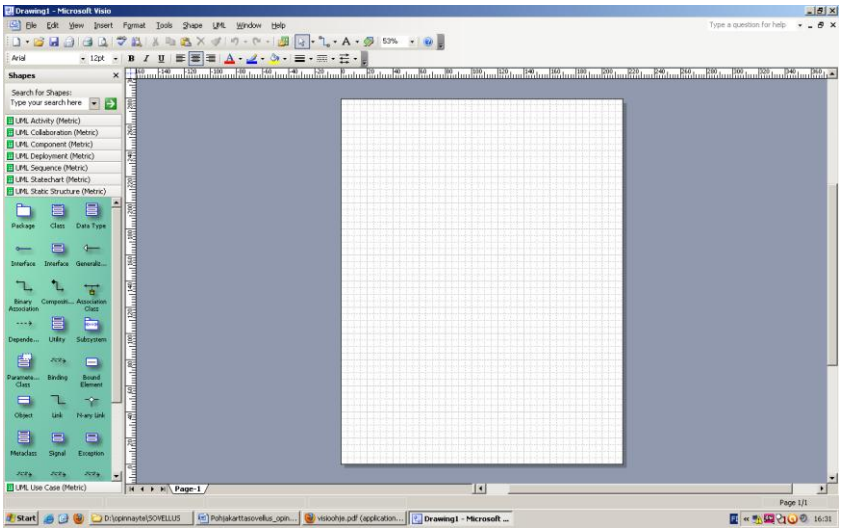
UML on mallinnuskieli (modeling language), ei menetelmä (method). Useimmat menetelmät sisältävät ainakin periaatteessa sekä mallinnuskielen että prosessin. **Mallinnuskieli** on (pääosin graafinen) kuvauskieli eli notaatio, joita menetelmissä käytetään suunnitelmien kuvaamiseen. **Prosessi** puolestaan on menetelmissä käytettävä ohjeistus suunnitelman laatimiseen tarvittavista vaiheista. (Fowler & Scott 2002, 2.) **Notaatio** (kuvauskieli) on se graafinen esitys, jota malleissa käytetään: se on mallinnus kielen syntaksi. Esimerkiksi luokkakaavion notaatio määrittelee, miten kohteet ja käsitteet, kuten luokka (class), yhteys (association) ja kerrannaisuus (multiplicity) esitetään. (Fowler & Scott 2002, 5.)

Microsoft® Office Visio® 2003 -ohjelmassa on malleja, muotoja ja piirustustyökaluja, joiden avulla voit luoda tehokkaita yrityskaavioita ja teknisiä kaavioita (Microsoft Corporation 2011). Seuraava kuva, jota tein toimivasta ohjelmasta Print Screen – avulla, esittelee ohjelman aloitussivun, joka mahdollistaa käyttäjää valita sopivan mallin Category ja Template – valikkojen kautta.



Kuva 4. Microsoft Visio 2003 ohjelman alkusivu

Kun sopiva Category on valittu, tässä esimerkissä se on Software, niin painamalla oikeata Template - painiketta (tässä tapauksessa - UML model diagram (Metric)), niin avautuu työskentelyruutu, jota voidaan nähdä seuraavalla kuvalla, jota tein toimivasta ohjelmasta Print Screen – avulla. Tästä lähtien, koko työ tapahtuu tässä piirustusympäristössä.



Kuva 5. Microsoft Visio 2003 ohjelman piirustusympäristö

3.2 Vaatimusmäärittely

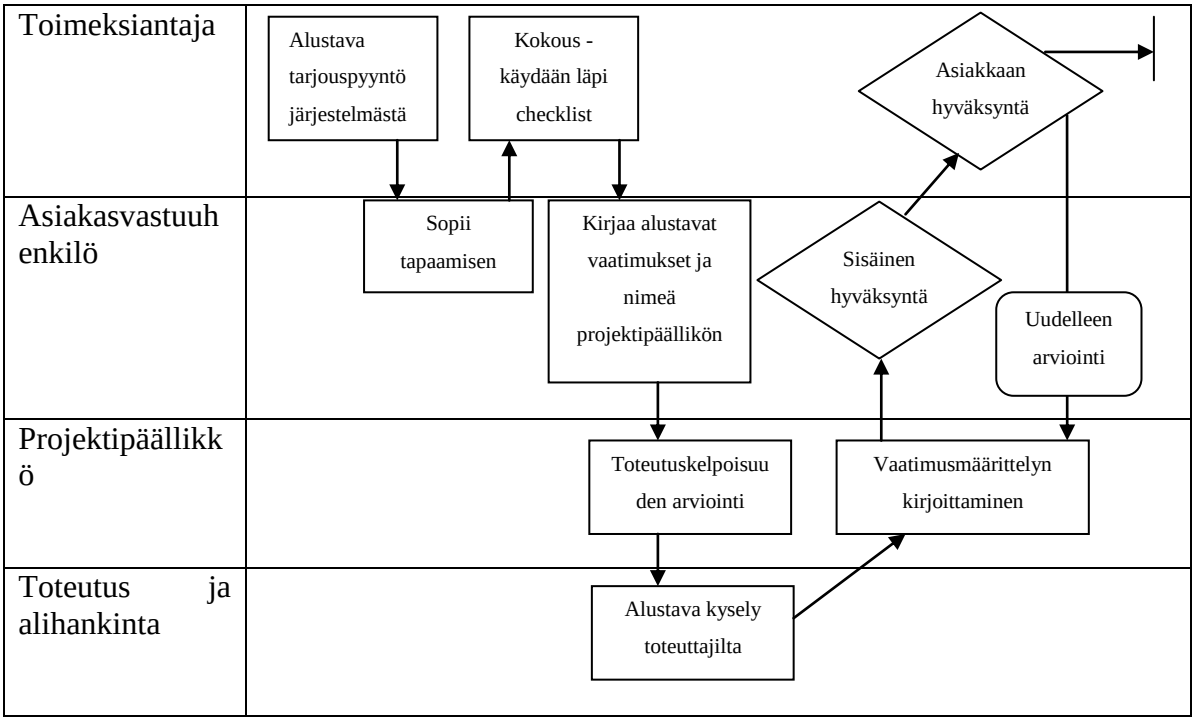
Vaatimusmäärittely kirjoitetaan yhtenä projektin ensimmäisistä dokumenteista. Monessa tapauksessa projektin teettäjä on kirjoittanut vaatimusmäärittelyn, jonka pohjalta pyydetään tarjousta. Vaatimusmäärittelyn primäärinä tavoitteena on kertoa, mitä projekti sisältää ja mitä ei. Vaatimusmäärittelyä käytetään jatkossa, kun katselmoidaan toiminnallista määrittelyä tai kun testataan sovelluksen kuormitusta tai muita teknisiä vaatimuksia. Vaatimusmäärittely kirjoitetaan usein ennen kuin projektia varsinaisesti on suunniteltu tai siihen on pyydetty rahoitus. (Kankaanpää 2004.)

Vaatimukset jaetaan usein käyttöliittymä, toiminnallisiin, liittymiin, teknisiin ja ei-toiminnallisiin vaatimuksiin. **Käyttöliittymävaatimuksissa** kuvataan, millaisessa ympäristössä sovellusta käytetään, esim. kuinka usein käyttäjä käyttää sovellusta, mikä resoluutio, millä käyttöliittymätyypeillä toimittava. **Toiminnalliset vaatimukset** listaavat sovelluksen toimintoja lyhyesti, esim. sisäänkirjoittautuminen, laskun

syöttäminen, URLin kirjoittaminen, sähköpostin lähettäminen jne. Liittymiin liittyvät vaatimukset kuvaavat järjestelmän vuorovaikutusta muiden järjestelmien kanssa. **Tekniset vaatimukset** liittyvät ratkaisuvaihtoehtoihin, vasteaikoihin jne. **Ei-toiminnalliset vaatimukset** kuvaavat käyttäjämääriä, tietoturvaa, palautumista vaikkapa virtakatkon jälkeen. (Kankaanpää 2004.)

Vaatimukset on hyvä formuloida taulukkomuotoon, jolloin saadaan selkeä lista vaatimuksista ja vaatimusten tärkeys näkyy selkeästi. Usein vaatimukset ryhmitellään niiden prioriteetin mukaan, tärkeimmät vaatimukset ensin ja vähemmän tärkeät, lopuksi. Jokaisella vaatimuksella on järkevä olla yksilöivä tunnus, johon voidaan viitata kun tehdään dokumenttien ristiviittauksia ja testataan vaatimusmäärittelyä vasten. (Kankaanpää 2004.)

Projektin alkuvaiheessa luotu lista vaatimuksista muuttuu useimmissa tapauksissa. Tällöin alkuperästä vaatimusmäärittelyn versiota käytetään hyväksi vertailukohteena projektin ajallisen keston ja kustannusten selvittämisessä. Dokumentoinnin yksi päätavoite on siis pitää kirjaa muutoksista ja huomioida ne hankkeen laajuuden arvioinnissa. (Kankaanpää 2004.) Seuraava kuva (Kankaanpää 2004) esittelee vaatimusmäärittelyn tekemisen prosessia, osallistuvia henkilöitä ja heidän vastuualueita.



Kuva 6. Vaatimusmäärittelyn tekemisen prosessi

Koskimiehen (2000, 161) mukaan, ensimmäisenä vaiheena sovelluskehitysprosessissa on vaatimusten kerääminen ja sitten vaatimusmäärittelyn tekeminen. Vaatimusmäärittely luo perustan hankinnalle: miksi ja mitä tarpeita hankinnan tulee tyydyttää? Vaatimusmäärittely keskittyy siihen, mitä järjestelmältä vaaditaan, eikä siihen, miten se toimii teknisesti. (Yritys-Suomi 2008.)

Keräsin ne tiedot keskustelemalla toimeksiantajani kanssa siitä, mihin vaatimuksiin pohjakarttasovelluksen on vastattava. Vaatimusten keräämisen ja analysoinnin jälkeen sain seuraavan vaatimuskuvauksen:

Tehdään interaktiivinen Web–pohjainen sovellus Kemin oppimiskeskuksen pohjakartan pohjalta. Sovellusta käytetään Kemi–Tornion ammattikorkeakoulun Kirjasto- ja tietopalvelun nettisivustolta. Kirjaston asiakas, toisin sanoin sovelluksen käyttäjä, käyttämällä sovellusta toimii seuraavasti: avaa kirjaston nettisivu; siirtää kursori pohjakartalla oleviin painikkeiden päälle ja sitten klikkaamalla sitä voi katsoa esineen tai paikan kuvaa, panoraamaa tai selostustekstiä; sulkee nettisivu. Sisällön tuottaminen, ja yhtenäiseen muotoon käsittely ei kuulu tämän opinnäytetyön tehtäviin.

Vaatimuksiin kuuluu myös ehto, että sovelluksessa oleva sisältö (selostustekstit, valokuvat ja panoraamavalokuvat) voidaan vaihtaa. Tällöin tarvitaan ylläpito-osiota, jonka kautta on mahdollisuus päivittää käyttöliittymän sisältöä. Ylläpitämiseksi tarvitaan vain yhtä tiliä, jonka käyttäjänimi ja salasana annetaan tämänhetkisen tehtävän suorittajalle. Henkilöt, jotka suorittavat ylläpitäjän tehtäviä, voivat vaihtua. Sen takia uuden suorittajan koulutusta varten tarvitaan opastusta. Ylläpitäjä toimii seuraavalla tavalla: kirjautuu sovelluksen ylläpito-osioon; valitsee tehtävää, suorittaa sen; kirjaa ylläpito-osiosta ulos. Toimeksiantaja toivoi myös mahdollisuuden vaihtaa ylläpitäjä-tilin käyttäjätunnus ja salasana. Ylläpito-osion toteuttaminen ei kuulu tämän opinnäytetyön tehtäviin.

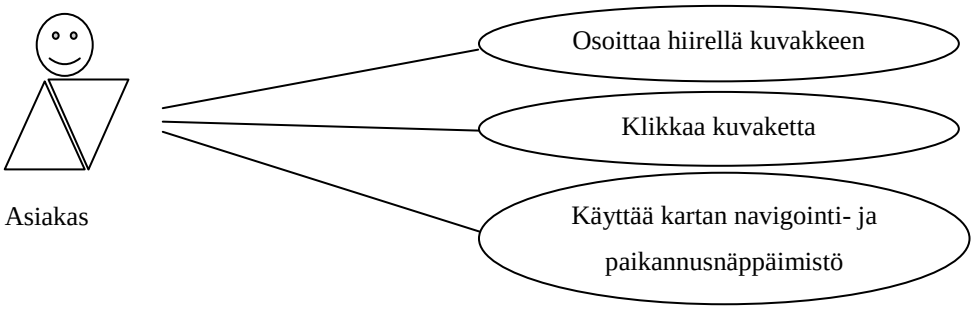
Vähän aikaisemmin minä mainitsin siitä, että käyttämällä sovellusta, asiakas pääsee tutustumaan Kemin oppimiskeskuksen tiloihin lukuisien valokuvien ja panoraamakuvien avulla. Samoin, sisältönä olevat kuvatiedostot halutaan ajoittain vaihtaa. Kaikki ne toivomukset edellyttävät tietokannan olemassa oloa. Sovelluksen käyttöliittymältä lähetetyt kutsut menevät tietokantaan ja palaavat vastauksina tiedostoineen. Manninen & Marttila omassa työssä (2006, 382) mainitsevat siitä, että

Flash–sovellus ei voi itsenäisesti ottaa tietokantaan yhteyttä. Siihen tarvitaan välittäjää. Tietoa Flash-sovellukseen voidaan toimittaa monilla tavoilla. Yksi niistä on XML-luokka, jota voidaan käyttää tässä tapauksessa, koska siirrettävä tieto on yksinkertainen ja ennalta ennustettu. Flash-sovelluksen ja tietokannan välisen yhteyden pystyttäminen ei kuulu tämän opinnäytetyön tehtäviin.

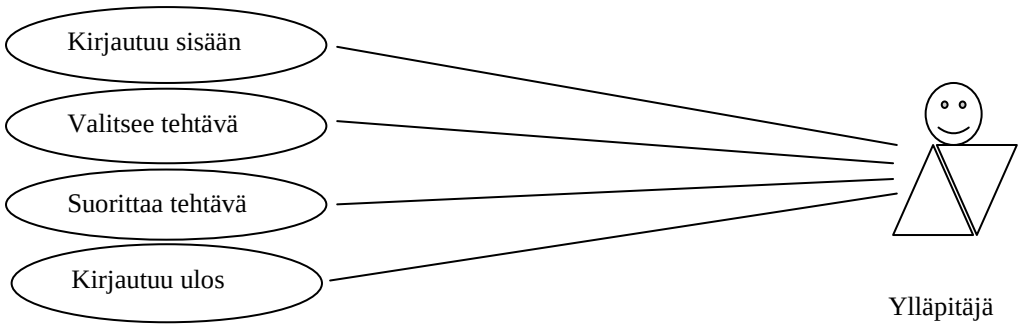
3.3 Vaatimusanalyysi

Vaatimusmäärittelyn jälkeen seuraavana vaiheena on vaatimusanalyysi, jonka tehtävänä on kuvata **sovelluksen ja sen käyttäjien väliset toiminnot**, eli tuottaa ymmärryksen siitä, mitkä ovat sovelluksen tehtävät. Tätä vaihetta kuvasin kaavioilla.

Aloitin kahdella käyttötapauskaaviolla. Koskimiehen (2000, 127) mukaan käyttötapaus eli englanniksi use case on kuvaus siitä, miten järjestelmän ja sen käyttäjien välinen vuorovaikutus näyttää käyttäjän näkökulmasta. Seuraavaa tekemäni kaksi kuvaa esittelee käyttötapauskaavioita, joissa näkyy ensin asiakkaan ja sitten ylläpitäjän toiminnot.



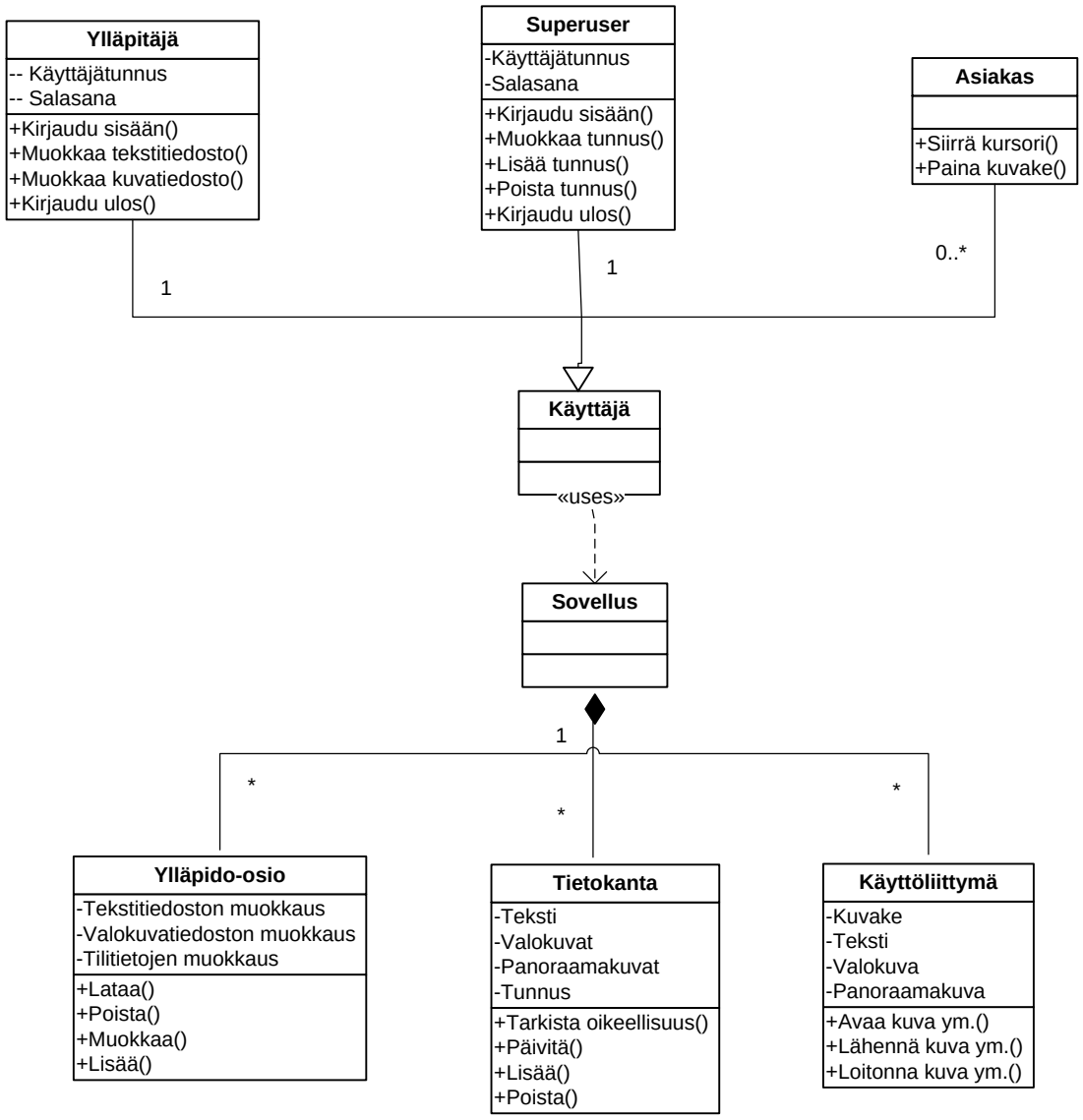
Kuva 7. Sovelluksen käyttö



Kuva 8. Sovelluksen ylläpito

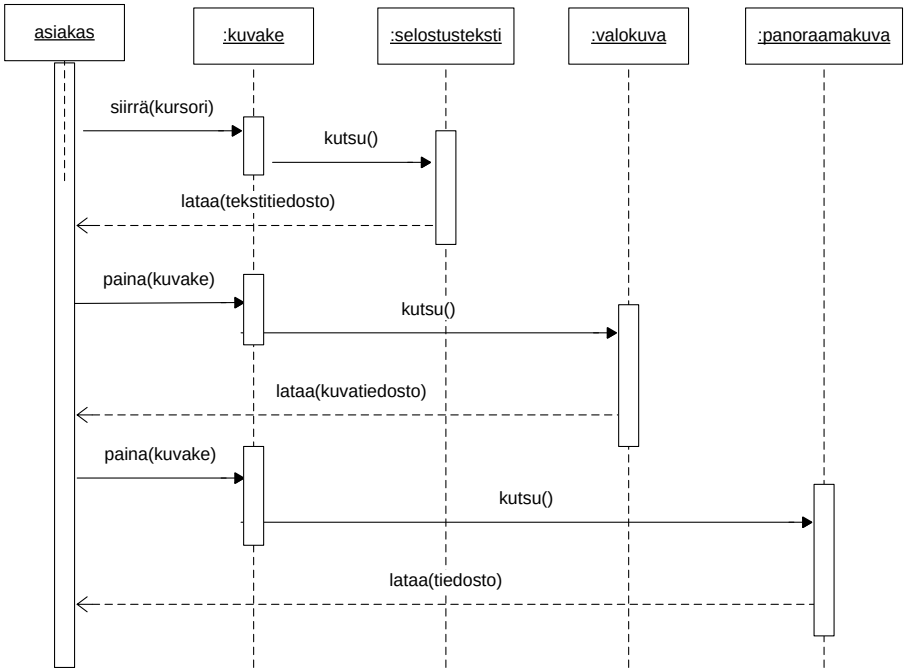
Käyttötapauskaaviot ovat kuvauksia siitä, mitä sovelluksen avulla käyttäjä ja ylläpito-osion avulla ylläpitäjä voivat saada tai tehdä.

Seuraavaksi, havannoilistin luokkakaavion avulla sovelluksen rakennetta, määrittelin mahdollisia olevia luokkia. Luokkakaavion avulla saadaan hahmoteltua ohjelmistolle rakenne, jopa arkkitehtuuri, joka auttaa ohjelmiston kehittämisessä laadukkaaksi (Juslin 2010). Seuraava tekemäni kuva esittelee pohjakarttasovelluksen määritetyt luokat ja niiden attribuutit.

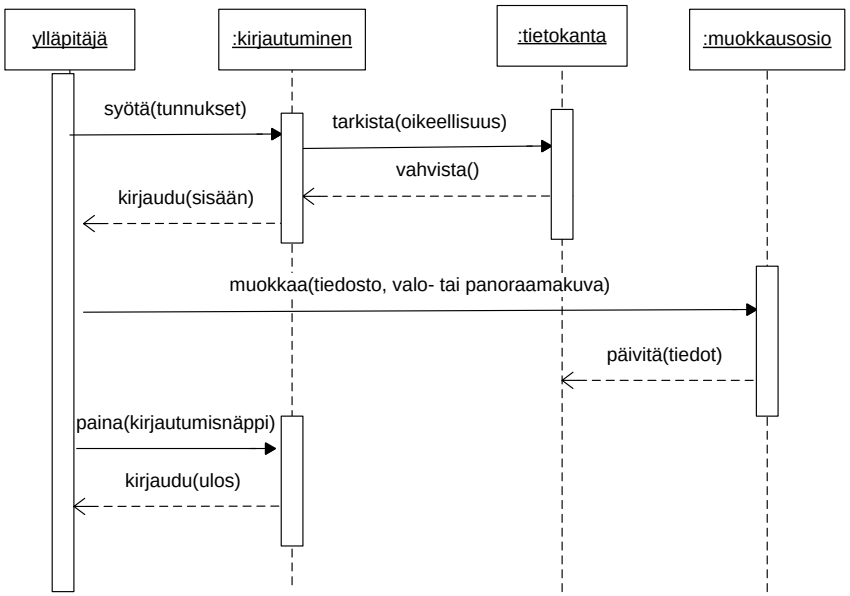


Kuva 9. Sovelluksen luokat

Vielä tarkemmin, **kuvaus tehtävistä avataan sekvenssikaaviolla**, jossa käyttäjän ja järjestelmän väliset suhteet katsotaan kokonaisuutena. Usein, sekvenssikaavio kuvaa **olioiden vuorovaikutusta tietyn käyttötapauksen toteutuessa**. Sekvenssikaaviossa aika kulkee ylhäältä alas, ja vuorovaikutustapahtumat kuvataan tässä aikaskaalassa vaakatasossa olevina nuolina, jotka kulkevat sanoman lähettäjältä sen vastaanottajalle (Koskimies 2000, 130). Seuraavaa kaksi tekemäni sekvenssikaaviota esittelee asiakkaan ja sovelluksen käyttöliittymän sekä ylläpitäjän ja ylläpito-osion vuorovaikutukset.

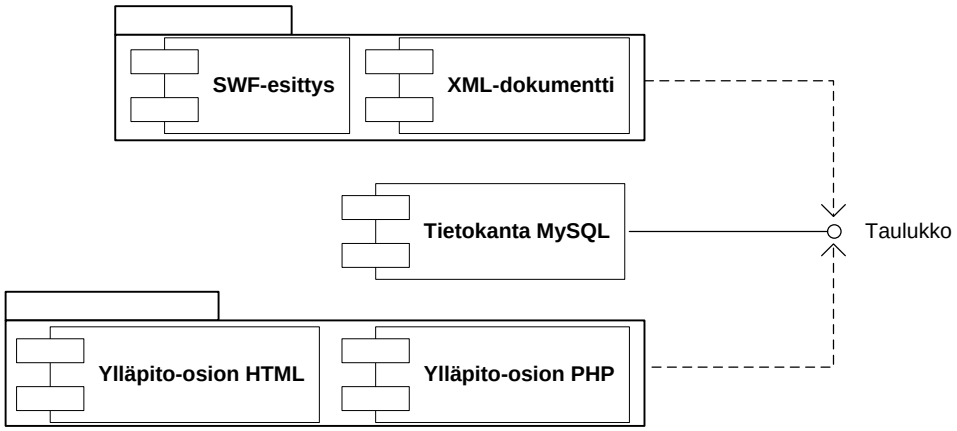


Kuva 10. Asiakas ja käyttöliittymä



Kuva 11. Ylläpitäjä ja ylläpito-osio

Vaatimusanalyysin jälkeen, Koskimiehen (2000, 163) mukaan, täytyy tarkastella, miten haluttu toiminnallisuus saadaan aikaan. Tietyn järjestelmän arkkitehtuurisuunnittelussa kiinnitetään järjestelmän kerrokset, merkittävimmät komponentit, ohjelmistojen sijoittelu laitteistoihin, tietokantaohjelmistot, prosessit ja niiden kommunikointi, käyttöliittymäratkaisut sekä muut keskeiset ohjelmiston arkkitehtuuriin vaikuttavat ratkaisut. Seuraavaa tekemäni komponenttikaaviota (kuva 12) katsoen, voidaan huomata, miten pääsovelluksen ja ylläpito-osion komponentit ovat riippuvaisia tietokannasta. Esimerkiksi XML-dokumentin ja tietokannan välillä muodostuu yhteys XMLSocket-luokan avulla ja ylläpito-osion PHP-scripti muodostaa ensin TCP - yhteyden MySQL - palvelimelle ja sitten mysql_connect()-funktiolla yhteys MySQL - tietokantaan. Kaksi naapuria - pääsovellus ja ylläpito-osio eivät omista yhteistä rajapintaa, koska nämä osat ovat kokonaan riippumattomia toisistaan.



Kuva 12. Pohjakarttasovelluksen komponenttikaavio

4 PÄÄSOVELLUKSEN TOTEUTUS

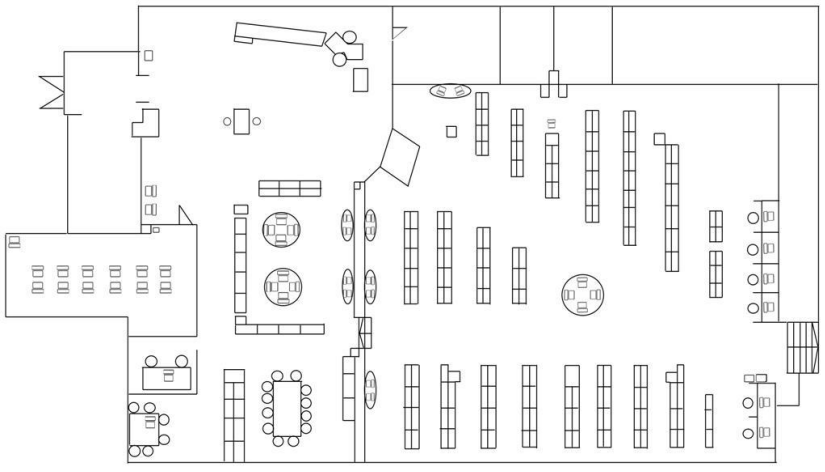
4.1 Toteutuksen kuvaus

Aloitin sovelluksen toteutuksen piirtämällä kuvakäsittelyohjelmalla kartan, joka pohjautuu oppimiskeskuksen floor mapiin.

Sitten ohjelmoin Adobe Flash 8 ohjelmointiympäristössä ActionScript 2.0-ohjelmointikielellä sovelluksen alustavan demoversion. Sovelluksen käyttöliittymän tärkeänä osana ovat painikkeet, joiden avulla vuorovaikutus tapahtuu sovelluksen ja käyttäjän välillä. Koska tämän opinnäytetyön aikana en toteuta sovelluksen koko sisältöä, niin tein vain kaksi esimerkki – painiketta, niiden toimintaperiaatteen tutustumista varten. Toimeksiantaja haluaa sovelluksella esitellä oppimiskeskuksen tiloja ja toimipisteitä, samoin kirjahyllyjen sisältöä. Tiedostoja ei voi upota Flash-sovellukseen, koska vaatimusmäärittelyssä on määritetty että sisältötietoa halutaan ajoittain vaihtaa. Kaikki tiedot pidetään tietokannassa ja käyttöliittymälle siirretään XML-merkintäkielellä.

4.2 Käyttöliittymä ja Adobe Photoshop CS2

Adobe Photoshop CS2 free 30-Day trial versio ohjelmalla tehty rasteritiedosto, on sovelluksen käyttöliittymän pohja, joka on esitetty seuraavalla tekemäni kuvalla.



Kuva 13. Pohjakarttasovelluksen käyttöliittymä

Itse käyttöliittymä on swf-muotoinen esitys, jolloin painiketta klikkaamalla avautuu tulostusalueelle pohjakartan vieressä joko teksti tai valokuva.

Tällä hetkellä sovelluksen koko näytöllä on 1500x614 pikseliä ja esitysnopeus on 24 kuvaa sekunnissa. Koska opinnäytetyönä teen sovelluksen alustavan demoversion, en ole miettinyt käyttöliittymän visuaalista ilmettä.

Jokainen oleva käyttöliittymällä painike tulostaa tulostusalueelle oman kuva- tai tekstiobjektin. Painikkeet ovat tehty Button:ksi ja painikkeen aikajanalla voidaan toteuttaa eri Up-, Over- ja Down-tilat. Päänäyttämöllä käyttöönotetulla painikkeella pitää olla oma ainutlaatuinen instanssinimi, joka auttaa sidottaa painike ActionScript-ohjelmointikoodiin.

4.3 XML, CSS ja lähdekoodieditori

Extensible Markup Language 1.0 (XML 1.0) on World Wide Web Consortiumin virallinen suositus yleiskäyttöiseksi tekstimuotoisten dokumenttien merkintäkieleksi. XML 1.0 määrittelee XML-dokumenttien luokan kieliopin sekä tarjoaa tyyppimäärityskielen, jonka avulla dokumenttien loogista rakennetta voidaan mallintaa. Keskeisenä ideana on yhteisesti sopia, kuinka tekstimuotoista tietoa kannattaa tallettaa tietorakenteina dokumenteiksi siten, että tieto on helposti ja yksikäsitteisesti luettavissa. (Nykänen 2001, 2.)

Tämän opinnäytetyön piireissä, käytän XML-merkintäkieltä siten, että XML-dokumentissa olevaa tekstiä voidaan tulostaa tekstimuotoisena dokumenttina.

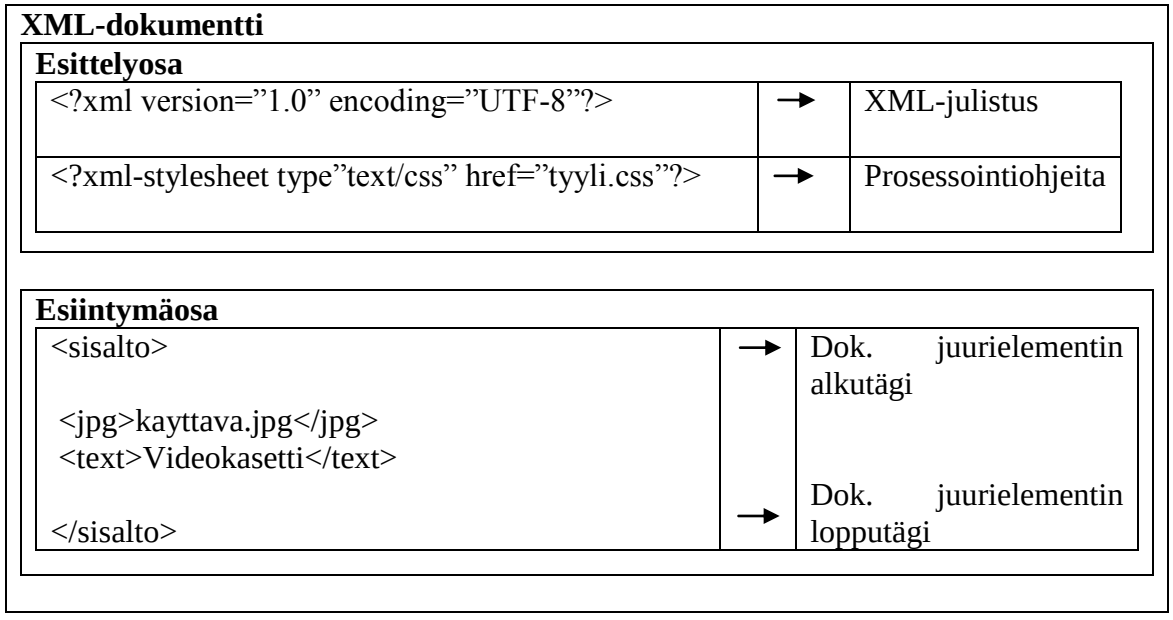
XML 1.0 – spesifikaatio on varsin lyhyt dokumentti, jossa päähuomio on hyvin muodostettujen XML-dokumenttien kieliopin määrittelyssä. Erityisesti XML 1.0 ei määrittele:

- yhdenkään XML - elementin tulkintaa
- yhdenkään XML – elementin ulkoasua
- millä ohjelmalla XML – dokumentteja tulisi käsitellä
- miten rakenteisia XML – dokumentteja tulisi suunnitella
- millä tavalla XML – dokumentti talletetaan massamuistiin.

Osaan näistä löytyy määrityksiä XML – standardiperheen sisältä, osaan ratkaisu pitää keksiä sovelluskohtaisesti. (Nykänen 2001, 96.)

XML – ydinspesifikaation lukemista ei myöskään voi aloittaa puhtaalta pöydältä: XML 1.0 sisältää viittauksia muihin alan standardeihin, joita se itse ei määrittele (Nykänen 2001, 96).

XML-dokumenteilla on XML-kieliopin osoittama väljä yleisrakenne. XML-dokumentti alkaa valinnaisella esittelyosalla, joka tiedoston koodauksen ohella voi esitellä myös dokumentin tyypin. Esittelyosaa seuraa dokumentin esiintymä, joka sisältää dokumentin varsinaisen merkatun tietosisällön. Dokumentin looginen rakenne on merkkausrakenteen käsittämä kokonaisuus. Dokumentin fyysinen rakenne taas määräytyy dokumenttien tiedot säilövän entiteettirakenteen perusteella. (Nykänen 2001, 104.) Nykänen (2001, 105) kuvaa XML-dokumentin osia ja niiden sisältöä. Tämän kuvauksen perusteella tein kuvan (kuva 14), joka auttaa hahmottaa XML-dokumentin yleisrakennetta visuaalisesti. Seuraava tekemäni kuva antaa kuvauksen siitä, miten pohjakarttasovelluksen sisältö kuvataan XML-merkintäkielellä.



Kuva 14. XML-tiedoston rakenne

Esitetyssä esimerkissä XML-tiedoston ulkoasua määritettiin CSS -tyylikielen avulla. Siihen viittaa XML-tiedostosta linkki tyylitiedostolle ”tyyli.css”. Itse tyylitiedosto luodaan erikseen teksti- ja lähdekoodieditorilla. Minä käytin ilmaista GPL - lisenssiin pohjautuvaa ohjelmaa notepad++, joka tukee useita eri ohjelmointikieliä. Koska tekstitiedoston rakenne on yksinkertainen, niin sen tyylimäärittelykin ei ole monimutkainen, kuten seuraava tekemäni kuva esittelee.

```
sisalto{
  display: block;
  background: white;
}
text {
  display: block;
  color: #33CC33;
  font-family: sans-serif;
  font-size: 12px;
}
```

Kuva 15. CSS-tyylitiedoston rakenne

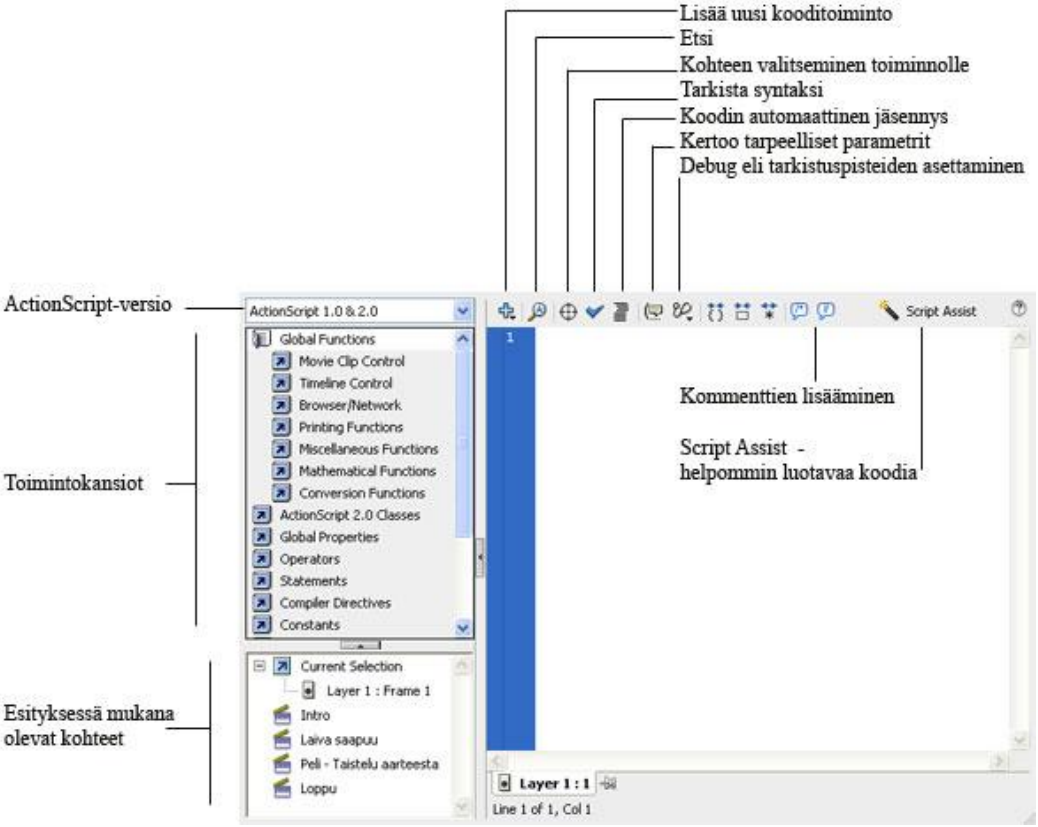
Nykänen (2001, 67) omassa kirjassa kuvailee CSS siten, että kielellä on oma erikoinen syntaksi, joka käsitteellisesti perustuu sääntöihin ja formatointiominaisuuksiin. Säännön mukaan valitaan se elementti, jota halutaan formatoida eli muuttaa visuaalisia ominaisuuksia. Esimerkiksi **text** - elementtiin kuuluvan tekstin fonttivärin on oltava vihreä ja fonttikoon on oltava 12 pikseliä.

4.4 ActionScript ja Adobe Flash 8

ActionScript2. 0 -ohjelmointikielenä perustuu ECMAScript 4 -standardiin, samoin kuin esimerkiksi JavaScript. Yhtäläisyyksiä moneen muuhunkin kieleen löytyy, esimerkiksi

PHP-ohjelmoinnissa käytetään usein pitkälti samoja koodirakenteita kuin ActionScript:ssä. (Manninen & Pasila 2006, 20.)

Actions-ikkuna on Actionscriptin alusta, johon skripti muodostetaan. Action-ikkuna löytyy Window-valikosta tai painamalla F9. Actions-ikkunassa vallitsevana alueena on itse kirjoitusalue. Vasemmasta reunasta löytyy luettelo eri mahdollisuuksista ja toiminnoista sekä lista esityksessä mukana olevista kohteista. Koodialueen yläpuolelta löytyy työkaluja ja apuvälineitä kirjoittamiseen. Perustilassa koodin kirjoittaminen vaatii luonnollisesti ymmärtämystä actionscriptin muodosta ja kirjoitustavasta. Yksi erinomaisista apuvälineistä niin vasta-alkajalle kuin ammattilaisellekin on Check Syntax-työkalu, joka tarkistaa koodin virheiden varalta.



Esimerkkiä ActionScript - kielioppia:

Kirjainherkkyys

ActionScript tunnistaa pienet ja suuret kirjaimet erillisiksi merkeiksi. Tästä syystä kaikki kielen sanat tulee kirjoittaa yhtenevällä tavalla. Esimerkiksi komentosana if tulee kirjoittaa "if", ei "If "tai "IF". Vastaavasti seuraavat muuttujat ovat kukin erillisiä:

abstotal
absTotal
AbsTotal
ABSTOTAL

ActionScript ei välitä edellä mainituista merkeistä, jos ne ovat koodiin kuuluvien sanojen välissä. Poikkeuksena ovat merkkijonot, joita ei saa katkaista välilyöntien kohdalta. Lauseet päätetään puolipisteeseen (;), kuten muissakin C-tyylisissä kielissä. Periaatteessa puolipisteen voi jättää pois jos lauseke on omalla rivillään. Tätä tapaa ei kuitenkaan mitenkään voi suositella. Kaikki kaarisulkeiden {} sisään kirjoitetut lauseet muodostavat ryhmän ja ne suoritetaan yhdessä. (Kemppainen 2009.) Alla oleva tekemäni kuva esittelee ohjelmointikielen koodiesimerkkiä pohjakarttasovelluksen toimintaympäristössä:

```
var myXML:XML = new XML();
myXML.ignoreWhite=true;
myXML.onLoad = loadXML;
myXML.load("sisalto.xml");
                btn.onRelease = function() {
                _root.createEmptyMovieClip("my_jpg", 10, 1050, 40);
                my_jpg.loadMovie(myXML.firstChild.childNodes[3].firstChild);
                function startLoading(whichImage) {loadMovie(whichImage, "my_jpg");
                                _root.onEnterFrame = function() {
                                infoLoaded = my_jpg.getBytesLoaded();
```

```
infoTotal = my_jpg.getBytesTotal();

percentage = Math.floor(infoLoaded/infoTotal*100); }};}
```

Kuva 16. ActionScript-koodi

5 POHDINTA

5.1 Tulokset ja jatkokehittäminen

Opinnäytetyön tuloksena on interaktiivisen pohjakarttasovelluksen alustava demoversio, joka antaa kuvauksen siitä, miltä toimeksiantajan mielestä pääsovelluksen pitäisi näyttää ja toimia. Työskentelyn aikana osoittautui, että vaatimusmäärittelyn mukainen työn määrä on todella iso yhden opinnäytetyön mittaan. Sen takia, minä jouduin osioimaan koko pääprojekti osiin, joita voidaan tehdä tulevaisuudessa itsenäisinä alihankintaprojekteina.

Toteutettu pohjakarttasovelluksen alustava demoversio on vain yksi osa koko projektista. Jatkokehitysehdotukset ovat seuraavia:

- Ylläpito-osiolla laajennus. Osio käyttää tietokannan tarjoamia palveluja. Toteutus PHP-ohjelmointikielellä.
- Sisällön tuottaminen ja yhtenäiseen muotoon käsittely. Prosessi koostuu valokuvien ottamisesta ja käsittelystä sekä panoraamakuvien valokuvista koostamisesta.
- Tietokannan rakenteen suunnittelu ja sisällön syöttö. Toteutus MySQL-hallintajärjestelmällä.
- Tietokannan ja Flash – sovelluksen välillä yhteyden pystyttäminen sopivalla tekniikalla.

5.2 Haasteet työskentelyn aikana

Tekemällä tätä opinnäytetyötä, olen oppinut paljon uutta. Pohjakarttasovelluksen suunnittelua ja demoversion toteutusta kaltaista työtä en ole tehnyt aikaisemmin. Työskentelyni alussa en osannut arvioida todellista työmäärää oikein. Sitten pikku hiljaa alkoi tulla ymmärrys siitä, että tämä projekti on pakko paloitella osiin, koska muutoin aikaa ei riitä kaikkeen.

Työskentely oli täynnä haasteita eri puolelta. Esimerkiksi jouduin itsenäisesti ja ihan alusta lähtien opettelemaan ActionScript 2.0 ohjelmointikielen sekä XML-merkintäkielen perusteita. Minun piti käydä monta tutorial – esimerkkiä läpi, jotta minä

ymmärtäisin ohjelmointikielen toimintaperiaatteita. Tämä vaati aikaansa. Internetistä kyllä löytyi aika paljon toimintakoodiesimerkkejä, mutta AS 2.0 ja XML koodipalojen yhdistäminen tarkoituksenmukaisesti oli vaatinut minulta aika paljon sinnikkyyttä, periksiantamattomuutta sekä oma - aloitteisuutta. Niinä hetkinä kun minusta tuntui, että olin jumiutunut paikalle, niin ongelmille aikalisän antaminen ja muiden tehtävien suorittaminen auttoi.

Koska suomen kieli ei ole minun äidinkieleni, niin raportin kirjoittaminen vieraalla kielellä lisäsi ylimääräisiä vaikeuksia. Yksi niistä on esimerkiksi tehokas ajankäyttö - vaikeus. Ongelmana on se, että oikeinkirjoittamiseen ja ajatusten ilmaisemiseen sanoiksi vieraalla kielellä tarvitaan kaksi kerta enemmän aikaa kuin omalla äidinkielellä kirjoittamiseen. Mutta riittävä aika ja säännöllinen työskentely veivät kuitenkin eteenpäin. Per aspera ad astra!

LÄHTEET

Painetut

Fowler, Martin & Kendal, Scott 2002. UML. 1. painos lokakuu 2002. Docendo Finland Oy, Jyväskylä

Haikala, Ilkka & Märijärvi, Jukka 2004. Ohjelmistotuotanto. 10. uudistettu painos. Karisto Oy, Hämeenlinna.

Järvinen, Petteri 2003. IT - tietosanakirja. Docendo Finland Oy, Jyväskylä.

Koskimies, Kai 2000. Oliokirja. Gummerus Kirjapaino Oy, Jyväskylä.

Nykänen, Ossi 2001. XML.1. painos marraskuu 2001. Docendo Finland Oy, Jyväskylä.

Vilkka, Hanna & Airaksinen, Tiina 2003. Toiminnallinen opinnäytetyö. Gummerus Kirjapaino Oy, Jyväskylä.

Painamattomat

Haggren, Henrik 1999. Panoraamakuvien kartoituskäyttö - unta vai kuvien huumaa? Teknillinen korkeakoulu. Luettu 16.4.2011. <http://foto.hut.fi/publications/paperit/hhaggren/Maankaytto_1_99/Maankaytto_99.html>.

Juslin, Jukka 2010. Luokkakaavio. Luentokalvot. Haaga-heila ammattikorkeakoulu. Luettu 4.4.2011. <http://myy.haaga-helia.fi/~ict1tn006/vko11/luentokalvot/viikko11.pdf>.

Kankaanpää, Timo 2004. Ohjelmiston määrittely 2op. Kurssit. Vaasan AMK. Luettu 20.4.2011. <http://www.cc.puv.fi/~tka/kurssit/Ohjelmiston_maarittely/maarittelydokumentit.html#Johdanto>.

Kasanen, Eero & Lukka, Kari & Siitonen Arto 1991. Konstruktiivinen tutkimusote liiketaloustieteessä. Liiketaloustieteellinen Aikakauskirja

3:1991. Luettu 9.10.2010.
<http://lille.

haaga-helia.fi/ampedatk/menetelmapaletti/konstruktiivinen.html>.

Kemppainen, Lea 2009. Tietokoneanimaatiot 2, 3op. Verkkokurssi. Kymenlaakson AMK. Luettu 12.4.2011.

<http://www2.kyamk.fi/~zlelj/flash2/lessons/lesson4.html>.

Koskimies, Kai & Koskinen, Johannes & Maunumaa, Mika & Peltonen, Jari & Selonen,

Petri & Siikarla, Mika & Systä, Tarja 2010. UML työvälineenä ja

tutkimuskohteena. Ohjelmistotekniikan laitos. Tampereen teknillinen

yliopisto. Luettu 20.4.2011. <http://www.cs.tut.fi/~ohar/kirjallisuutta/UML%20tyovalineena%20ja%20tutkimuskohteena.pdf>.

Luukkainen, Matti 2010. Helsingin yliopisto. Tietojenkäsittelyntieteen laitos. Luentomateriaalia. Ohjelmistojen mallintaminen. Luento 9, 30.11. Diakuva 23. Luettu 17.10.2010.

< http://www.cs.helsinki.fi/u/mluukkai/ohmas10/luentokalvot/luento9.pdf >.

Microsoft Corporation 2011. Microsoft Office Visio 2003:n käyttöopas. Luettu 9.2.2011

<http://ope.hayo.fi/~jatu/oppis/dtsuu/visioohje.pdf>.

Oulun seudun ammattikorkeakoulu. Raahen tietokonetekniikan ja liiketalouden yksikkö. Ohjelmistotekniikka teema 6. Luettu 17.4.2011. <http://www.students.oamk.fi>.

Rinne, Olli 2011. Mikä on digitaalinen valokuva? Pikseli – verkkolehti. Luettu 22.4.2011. <http://www.pikseli.fi/digifaq/2_digivalokuva.html>.

Yritys-Suomi 2008. Vaatimusmäärittely. Luettu 26.4.2011. <http://www.yrityssuomi.fi/default.aspx?nodeid=16290>.

LIITE 1

VAATIMUSMÄÄRITTELY

POHJAKARTTASOVELLUS

Taltio:	
Luontipäivämäärä:	23.9.2010
Tekijä(t):	Inga Hiltunen 07TIE/WEB
Muutospäivämäärä:	20.4.2011
Viimeisin tallentaja:	Inga Hiltunen 07TIE/WEB
Kommentit:	Muutoksen syy on dokumentin sisällön päivitys
Tarkastuspäivämäärä:	
Tarkastaja:	
Hyväksymispäivämäärä:	
Hyväksyjä:	Anitta Örn / KTAMK Kirjasto- ja tietopalvelut
Dokumentin omistaja:	KTAMK, Kirjasto- ja tietopalvelut

VERSIOHISTORIA 2(15)

Versio	Päiväys	Muutokset	Tekijä(t)
0.1	23.9.2010	Alustava versio	Inga Hiltunen 07TIE/WEB
0.2	20.4.2011	Sisällön päivitys	Inga Hiltunen 07TIE/WEB

1 JOHDANTO 3(15)

1.1 Dokumentin tarkoitus

Tämä dokumentti on interaktiivisen pohjakarttasovelluksen (Web-sovellus) vaatimusmäärittely.

Vaatimusmäärittely on tarkoitettu Web-sovelluksen toiminnallisen määrittelyn lähtökohdaksi. Se kuvaa toimeksiantajan määrittelemät tavoitetoiminnan mukaisia prosesseja ja järjestelmäarkkitehtuuria sekä asiakaskohtaisia Web-sovelluksen vaatimuksia.

1.2 Web-sovellus

Pohjakarttasovellus on tarkoitettu Kemi–Tornionlaakson koulutuskuntayhtymän Lappian opiskelijoiden sekä muiden kohderyhmien käyttöön. Se on interaktiivinen Kemin oppimiskeskuksen pohjakartta, joka on varustettu panoraamakuvilla, valokuvilla sekä tekstillä. Sovellus toteutetaan selainpohjaisena palveluna, joka sijoitetaan Kemi–Tornion ammattikorkeakoulun Kirjasto- ja tietopalvelujen nettisivulle. Pohjakarttasovellus pystytetään Lappian omalle Web-palvelimelle.

1.3 Tavoitteet

Kemi–Tornion ammattikorkeakoulun Kirjasto- ja tietopalvelujen tarpeisiin tarvitaan sovellusta, joka tulee helpottamaan Kemi–Tornionlaakson koulutuskuntayhtymä Lappian uusien opiskelijoiden sekä muiden kirjasto- ja tietopalveluiden käyttäjien Kemin oppimiskeskuksen tilojen ja palvelupisteiden sijainnin hahmottamista ennen varsinaista käyntiä. Samoin sovelluksen interaktiivisuus toimii Lappian toiminnan mainoskanavana.

1.4 Määritelmät, termit ja lyhenteet. Dokumentissa esiintyvät termit ja lyhenteet:

Web-sovellus	Internetin kautta jaeltava ohjelmisto, jolla on web-käyttöliittymä
Sovelluksen ylläpitäjä	Henkilö, jolla on oikeus lisätä, poistaa ja muokata tekstit, valokuvat ja panoraamakuvat. Sillä hän päivittää sovelluksen sisältö.
Sovelluksen käyttäjä	Henkilö, joka kommunikoi sovelluksen käyttöliittymän kautta.
Lappian opiskelijat	Kemi – Tornion ammattikorkeakoulun sekä ammattiopisto Lappian opiskelijat.
Muut kohderyhmät	Kaikki muut ihmiset
KTAMK	Kemi – Tornion ammattikorkeakoulu
Kirjasto- ja tietopalvelut	Se on Kemi – Tornion ammattikorkeakoulun kirjasto. Kaikki yksiköt.
Kemin oppimiskeskus	Kirjaston yhdestä yksiköstä, sijainti on Kemissä.
Lan/Wlan	Lähiverkko (Local Area Network); Langaton lähiverkko (Wireless)
W3C validator	World Wide Web Consortium kehittää yhteisiä ja yhteensopivia Webbin pelisääntöjä ja teknologioita.
Adobe Flash 8	Ohjelma, jonka avulla luodaan interaktiivisen käyttöliittymän sisältö

ActionScript 2.0	Kieli, jolla ohjelmoidaan Flash-tiedoston toiminnot	5(15)
HTML	Hypertext Markup Language. Ohjelmointikieli, jolla rakennettaan Web – sivustot.	
PHP	Hypertext Preprocessor. Ohjelmointikieli, jota käytetään Web-palvelinympäristöissä dynaamisten Web-sivujen luonnissa.	
MySQL	SQL-tietokannan hallintajärjestelmä.	
phpMyAdmin	Selaimen kautta käytettävä MySQL-tietokannan hallintatyökalu.	
XAMPP	Alustariippumaton Web - palvelin	
.swf	.swf-päätteinen tiedosto on Flash-tiedosto, joka saa auki Flash player - ohjelmalla	
Flash player	Adobe® Flash® Player eri käyttöjärjestelmissä toimiva selainpohjainen sovellus-runtime, joka tarjoaa ilmeikkäiden sovellusten, sisällön ja videoiden ehdottoman katselukokemuksen eri näytöissä ja selaimissa.	

1.5 Yleiskatsaus dokumenttiin

Tämä dokumentti on rakennettu seuraavasti:

- Kappale 1 sisältää tietoa dokumentista ja sen käyttöä koskevia yleisiä asioita.
- Kappaleessa 2 annetaan yleiskuva tavoitetoiminnan prosesseista, käyttötapaukset ja liiketoimintaluokkakaavio.
- Kappaleessa 3 kuvataan tietojärjestelmäarkkitehtuuri,
- Kappaleessa 4 voi tutustua suunnitterajoitteisiin.

1.6 Lukijakunta 6(15)

Vaatimusmäärittely on tarkoitettu Web - sovelluksen vaatimusten kuvaamiseen. Se on laadittu Web - sovelluksen toiminnallisen määrittelyn pohjaksi. Tilaaja (Anitta Örn / KTAMK Kirjasto- ja tietopalvelut) tekee sen yhdessä Web - sovelluksen toteuttajan (Inga Hiltunen 07TIE/WEB) kanssa.

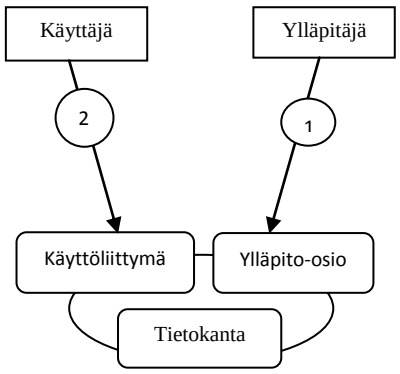
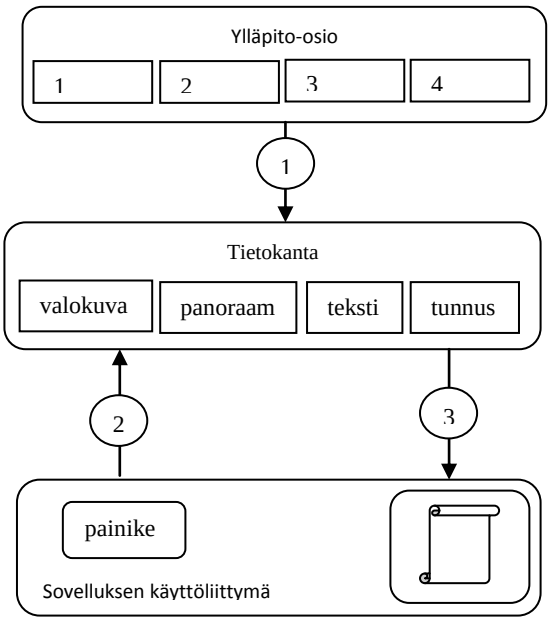
Vaatimusmäärittelyn pääasiallinen lukija on tilaaja ja Web-sovelluksen toteuttaja. Eräs kohderyhmä on myös testauksen toteuttajat.

2 TAVOITETOIMINTA 7(15)

2.1 Ympäristömalli

Toiminnan yleiskuvaus.

Web-sovellus ja sen ympäristön rajapinta: Palvelun käyttö ja sisällön ylläpito

	1. Ylläpitäjä kirjautuu syöttämällä tunnukset. 2. Käyttäjä etsii tarvittavat tiedot.
	Ylläpitäjä valitsee tehtävä: 1-valokuvien hallinta 2-panoraamakuvien hallinta 3-tekstitiedostojen hallinta 4-tunnuksien hallinta (superuser) 1. Syötetty/muokattu tieto tallennetaan tietokantaan 2. Käyttäjä painikkeen avulla lähettää kutsun tietokantaan 3. Tietokannasta vastauksena palaa kutsuttu tieto sovelluksen tulostusalueelle

2.2 Toiminnot 8(15)

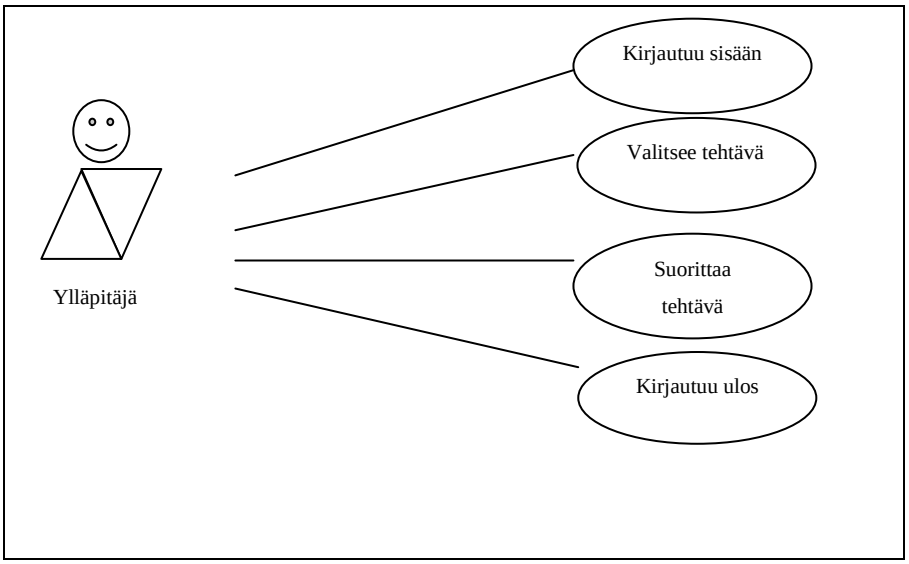
2.2.1 Ylläpitäjän tunnistaminen

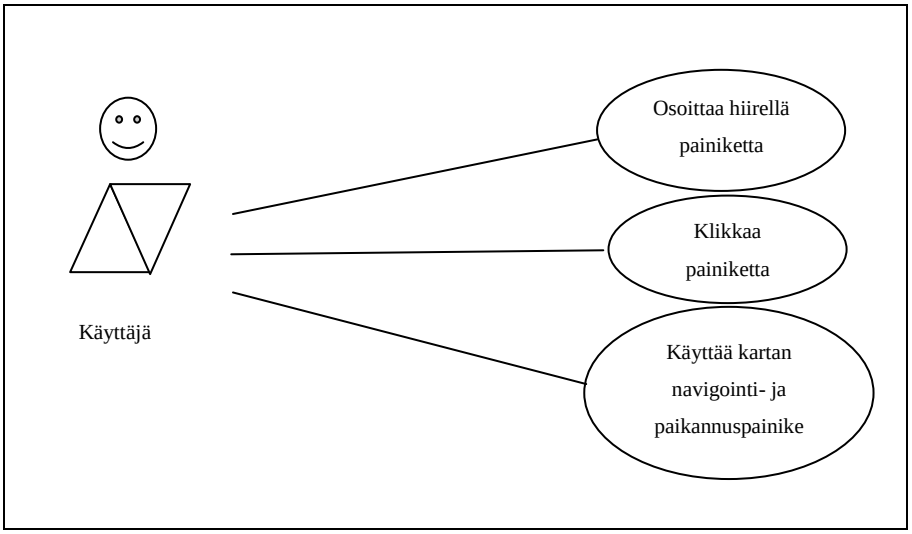
Esiehdot	Ylläpito-osion aloitussivu selaimen ikkunassa on auki
Syötetiedot	Kursori on valmiiksi käyttäjätunnus kentässä. Ylläpitäjä syöttää käyttäjätunnuksen ja salasanan. Enterin tai kirjaudu - painikkeen painallus lähettää tunnistamispyynnön palvelimelle.
Toiminto	Salasana kaiutetaan näytölle tähti-merkkeinä. Onnistuneen salasanan tarkistuksen jälkeen näytetään ylläpito-osion tervetuloa -sivu.
Palaute	Avataan ylläpito-osion pääikkuna.
Poikkeustilanteet	Tunnustietoja ei lähetetä palvelimelle, mikäli käyttäjätunnus tai salasana puuttuu. Mikäli tunnustiedot eivät täsmää, annetaan varoitusviesti "Käyttäjätunnus tai salasana väärin, yritä uudestaan."

2.2.2 Valokuvan, panoraamakuvan, tekstin lisääminen

Esiehdot	Ylläpitäjä on kirjautunut ylläpito-osion sisään ja tehtävien valikko on auki. Valitaan tehtävä.
Syötetiedot	Ylläpitäjä täyttää lomakkeen kentät tiedoilla ja lisää – painikkeen painallus lähettää syötetyt tiedot tietokantaan.
Toiminto	SQL - skripti tallentaa syötetyt tiedot tietokannan taulukkoon
Palaute	Onnistuneesta tallentamisesta viestii ilmoitus esim. ”Tiedot on nyt tallennettu tietokantaan”
Poikkeustilanteet	Jos tiettyyn kenttään syötetyn tiedon muoto tai muu ominaisuus ei vasta ennalta määritetylle muodolle, niin annetaan virheilmoitus esim.”Valokuvan tiedostokoko ei saa olla isompi kuin 30 kilotavua”

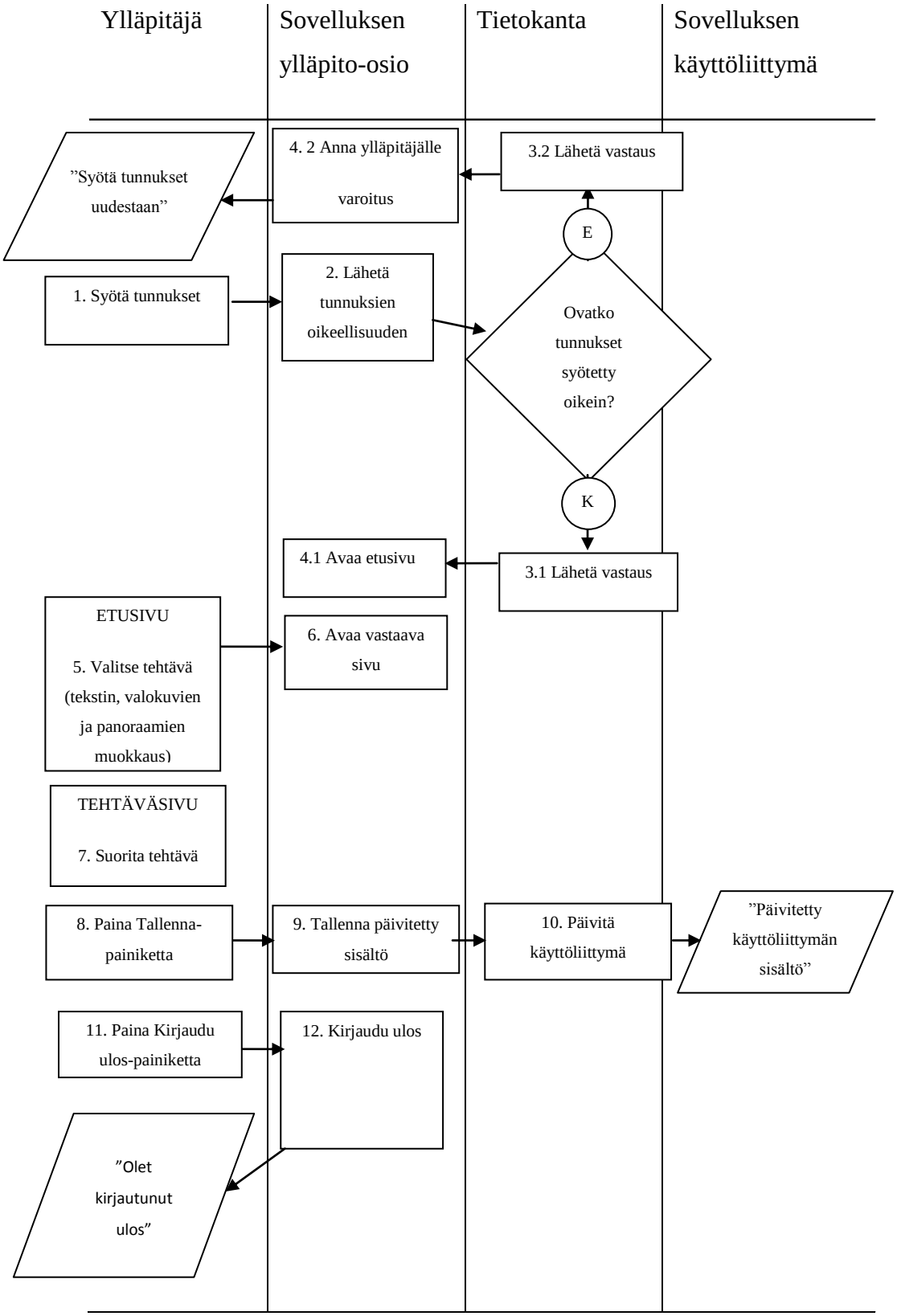
Käyttötapausmallit: Web-sovelluksen ylläpito ja käyttö





2.4 Prosessien kuvaukset 10(15)

2.4.1 Prosessikartta: Sovelluksen ylläpito

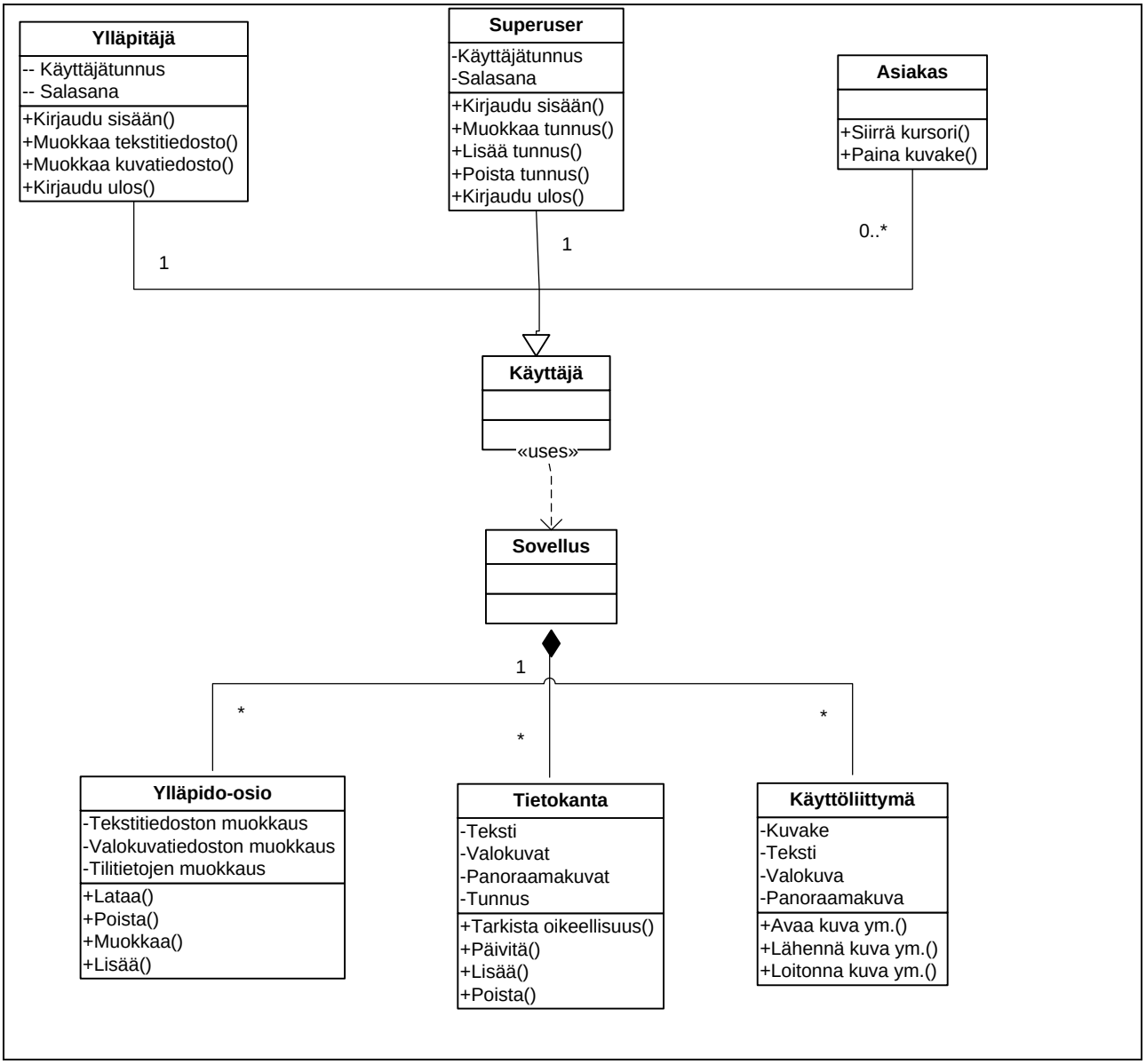


2.4.2 Prosessikuvaukset: Sovelluksen ylläpito 11(15)

Prosessi	Kuvaus
1. Syötä tunnukset	Ylläpitäjä avaa sovelluksen ylläpito-osion etusivun, jolla on kirjautumiskenttä. Kenttiin syötetään käyttäjätunnus ja salasana.
2. Lähetä tunnuksien oikeellisuuden tarkistuspyyntö	Sovellus ”tarkistaa” syötettyjen tunnuksien oikeellisuus vertaamalla niitä omassa tietokannassa oleviin.
3.1 Lähetä vastaus	Syötetyt tunnukset täsmäävät sovelluksen tietokannassa olevien tunnuksien kanssa, ylläpitäjän kirjautuminen sisään onnistui.
4.1 Avaa etusivu	Sisään kirjautunut ylläpitäjä pääse sovelluksen ”sisään” tehtävien valikkoon.
5. Valitse tehtävä	Painamalla vastaava linkki ylläpitäjä valitsee tehtäväksi: tekstin muokkaaminen, kuvien lataaminen, panoraamakuvien lataaminen.
6. Avaa vastaava sivu	Ylläpitäjä pääsee suorittamaan tehtävää.
7. Suorita tehtävä	Ylläpitäjä muokkaa tekstitiedostot, lataa/poistaa valokuvat, lataa/poistaa panoraamakuvat.
8. Paina Tallenna tulokset – painiketta	Työstämisen tulos tallentuu sovelluksen tietokantaan.
9. Tallenna päivitetty sisältö	Koodi tallentaa sovelluksen tietokantaan uusi sisältö.
10. Päivitä käyttöliittymä	Päivitetty tietokannan sisältö heijastuu sovelluksen käyttöliittymälle.
11. Paina Kirjaudu ulos -painiketta	Sovelluksen ylläpitäjä turvallisesti poistuu ylläpito-osiosta.
12. Kirjaudu ulos	Koodi kirjaa ylläpitäjän turvallisesti ulos.

2.5 Luokat 12(15)

Luokkakaavion hahmotelma: Web-sovelluksen ylläpito ja käyttö



3 TAVOITETILAN JÄRJESTELMÄARKKITEHTUURI 13(15)

3.1 Tietojärjestelmät

Pohjakarttasovellus tehdään Kemi–Torniolaakson koulutuskuntayhtymä Lappian opiskelijoita sekä muita kohderyhmiä varten. Se pystytetään KTAMK Kirjasto- ja tietopalvelujen nettisivuille Lappian olemassa olevalle Web-palvelimelle (esim. http://www.tokem.fi/?deptid=14043 linkki kartalle). Web-sovellus suunnitellaan siten, että se vaatii minimaalisen ylläpidon. Käyttö on mahdollista Lappian sekä ulkopuolisten työasemapäätteiden ja verkkojen avulla. Mitään muuta/uutta tietojärjestelmä tai tietojärjestelmän osia ei tule tarvitsemaan.

3.2 Laitteistot ja valmisohjelmistot

Työasemapäätteinä toimivat pöytätietokoneet ja kannettavat. SWF-muotoisen esityksen katseluun käyttäjän käyttöjärjestelmällä pitää olla asennettu Flash player – ohjelman viimeisin päivitetty versio.

3.3 Tietoliikenneyhteydet

Lappian omat Lan/Wlan verkot. Ulkopuoliset käyttävät omaa nettiyhteyttä.

3.4 Loppukäyttäjät

Web-sovelluksen käyttäjiä ovat Lappian opiskelijat, muut kohderyhmät.

3.5 Alustavat tietovarastot

Web-sovellus käyttää tietokantaa, jossa varastoidaan Web-sovelluksen sisältö kuten valokuvat ja panoraamakuvat. Kirjahyllyjen tietojen varastointipaikkaa ei ole vielä määritelty.

4 SUUNNITTELURAJOITTEET 14(15)

4.1 Viranomaismääräykset ja laatuvaatimukset

Suunnittelemalla pohjakarttasovellusta on mahdollista käyttää muita vastaavia tuotteita tai tuotteen osia ideointimalleina. Mutta niiden tuotteiden tiedostojen suora kopiointi, editointi, tulostus ja muu käsittely on kiellettyä. Eli valmiin tuotteen koko tietosisältö on suojattu tekijänoikeuslailla, jos julkisella kohdalla ei ole mainittu toisin.

Ohjelmoinnissa käytettyjen kaupallisten ohjelmistojen tai niiden komponenttien lupa-asioiden pitää olla kunnossa. Muissa tapauksissa käytetään trial- tai kevennyttä versiota.

- W3C validator
- Useiden selainten tuki
- Tarkka ja päivitetty sisältö
- Looginen navigointi
- Riittävä vuorovaikutteisuus
- Käyttäjäystävällinen värien käyttö

4.2 Standardit sekä mallinnuskielet ja menetelmät

Käytettyjen työkalujen alustava lista on:

Käyttöliittymän ympäristö	HTML
Dynaaminen sisältö	PHP
Tietokanta	MySQL
MySQL hallintatyökalu	phpMyAdmin
Alustariippumaton Web-palvelin	XAMPP
Flash-sovellukseen tietoja tulostusta varten toimitettava teknologia	XML-merkintäkieli
Palvelin ohjelma, joka käsittelee XMLSocket - yhteyksiä	Valinta ei ole tehty

Ohjelmoinnin ympäristö (ohjelma)	Adobe Flash 8
Käyttöliittymän interaktiivisuus	ActionScript 2.0
Ulkoinen tyylitiedosto, joka määrittelee sisällön käytetyt muotoiluasetukset.	CSS
Lähdekoodieditori	notepad++
UML – mallinnuksen kaaviot, vuokaavio, pseudokoodi	