Menetelmiä portlettipohjaisen verkkosovelluksen kehittämiseen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, 24.11.2010

Juhani Jaakkola

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Tietojenkäsittelyn koulutusohjelma
Hämeenlinna

Työn nimi Menetelmiä portlettipohjaisen verkkosovelluksen kehittämiseen

Tekijä Juhani Jaakkola

Ohjaava opettaja Lasse Seppänen

Hyväksytty _____._____.20_____

Hyväksyjä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Visamäki
Tietojenkäsittelyn koulutusohjelma

Tekijä Juhani Jaakkola **Vuosi** 2010

Työn nimi Menetelmiä portlettipohjaisen verkkosovelluksen kehittämiseen

TIIVISTELMÄ

Opinnäytetyöllä oli kaksi tavoitetta; selvittää kirjallisuuden pohjalta portlettitekniikan käsitteet, menetelmät ja tekniikan tuomat rajoitteet verkkosovellusten suunnittelussa. Toisena tavoitteena oli kehittää portlettipohjaisen verkkosovelluksen suunnitteluprosessia ensimmäisen tavoitteen tulosten perusteella.

Työssä käytettiin kvalitatiivista tutkimusmenetelmää, jossa teoriaa luodaan aineistolähtöisesti. Ensimmäisen tavoitteen tuloksia verrattiin Painonnostoliitolle tehdyn portlettipohjaisen projektin kulkuun ja johtopäätösten perusteella teoriaa hiottiin käytännönläheisemmäksi ja tarkemmaksi.

Työssä todettiin, että toteutustekniikka pitää valita ennen käyttöliittymäsuunnittelun aloittamista, jotta se voidaan ottaa huomioon suunnittelussa. Portaali kannattaa ottaa toteutustekniikaksi vain, jos kootaan sisältöä monesta eri järjestelmästä yhden käyttöliittymän alle.

Portletti tuo sisällön suorakaiteen muotoiseen ikkunaan, jonka sijainnin ja koon portaali määrittelee. Suunnittelussa tulee huomioida, ettei portlettiin voida mahduttaa kokonaisen verkkosovelluksen toiminnallisuutta. Portletit kuitenkin parantavat verkkosovellusten käytettävyyttä tuomalla osan toiminnallisuudesta käyttäjän helposti saataville. Portlettitekniikka mahdollistaa ainoastaan pienten tietomäärien siirtämisen portlettien välillä. Suunnittelussa tämä pitää ottaa huomioon minimoimalla portlettien välinen tiedonsiirto.

Portlettipohjaisen käyttöliittymän suunnittelu helpottuu, kun määritellään verkkosovelluksen käsittelemä tieto ja jaetaan se portletteja vastaaviksi kokonaisuuksiksi ennen käyttöliittymän näkymien suunnittelua.

Avainsanat Ohjelmistokehitys, portaali, portletti, prosessimallit

Sivut 44 s. + liitteet 1 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme in Business Information Technology

Author Juhani Jaakkola **Year** 2010

Subject of Bachelor's thesis Methods for developing Portlet-based web application

ABSTRACT

This thesis has two objectives; based on literature, to define the concepts and methods of portlet technology as well as the constraints caused by technology in web application design. The second objective is to develop the designing process of web applications based on results from the first objective.

In this thesis the qualitative research method was used, where theory is created based on material. The theory was refined and made more practical by comparing the results from the first objective to the Portlet-based project made for Painonnostoliitto.

In this thesis it was concluded that implementation technology must be chosen before the beginning of the user interface design, so it could be considered in the design. Portal technology should be chosen as implementation technology only if there is a requirement to aggregate content from many sources to a single user interface.

The portlet brings the content to a rectangle shaped window, the size and location of which the portal determines. In the design it should be considered that the portlet can not contain all the functionalities of the entire web application. The portlet improves the usability of web applications by bringing part of the functionality easily available to the user. Portlet technology enables portlets to transfer only small amounts of information between them. In the design this should be considered by minimizing intercommunication between portlets.

Designing of portlet based user interfaces becomes easier, when defining information used by the web application and dividing it to entities corresponding to portlets before designing the view of the user.

Keywords Portal, portlet, process models, software development

Pages 44 p. + appendices 1 p.

Abstraktiluokka	Abstraktit luokat ovat yläluokkia, joilla määritellään alaluokille haluttuja omaisuuksia ja metodeja, mutta niistä ei ole mahdollista luoda olioita.
Ankkuroitu teoria	Eng. ”grounded theory”, laadullinen tutkimusmenetelmä, jossa teoriaa lähdetään kehittämään aineistopohjaisesti.
Esityspyyntö	Eng. ”render request” esityspyynnöllä portletti tuottaa sisältöpalan senhetkisen tilansa perusteella. Esityspyyntö voidaan suorittaa samanaikaisesti monella portletilla.
Hibernate	Hibernate mahdollistaa luokista muodostuvien objektien suoran tallentamisen tietokantaan nopeuttaen tietokantayhteyden toteuttamista.
Kertakirjautuminen	Eng. ”single sign-on”, mahdollisuus kirjautua yhdellä kerralla moneen eri järjestelmään.
Liferay	Avoimen lähdekoodin portaali, jossa on monia sisäänrakennettuja omaisuuksia, kuten rekisteröinti, dokumenttikirjasto ja uutisvirtat.
Metatieto	Tietoa tiedosta, kuvailevaa ja määrittelevää tietoa tietovarannosta.
Muunneltavuus	Eng. ”customize”, muuntaa tarpeiden mukaiseksi, räätälöidä (MOT). Mahdollisuus muuttaa/valita sivuston tarjoamaa sisältöä.
MySQL	Suosittu SQL-tietokantojen hallintajärjestelmä, jota hyvin usein käytetään web-palveluita toteutettaessa.
Personointi	Eng. ”personalize”, tehdä jostakin asiasta henkilökohtainen (MOT). Mahdollisuus muuttaa sivuston ulkoasua omia mieltymyksiä vastaavaksi.
Portaali	Eng. ”portal”, verkkosovellus, joka koostaa tietoa ja palveluita yhden käyttöliittymän alle.
Portletti	Eng. ”portlet”, ikkunoitu verkkosovellus, jolla tuodaan sisältöä portaalisivulle jo olemassa olevista järjestelmistä ja muista tietolähteistä
Rajapinta	Ohjelmien väliseen tiedon siirtoon ja pyyntöjen tekemiseen tarkoitettu käyttöliittymä, jolla mahdollistetaan ohjelmien keskustelu keskenään.

Relation Management Methodology (RMM)	Rakenteisen verkkosovelluksen suunnitteluun tarkoitettu malli, joka jakaa suunnitteluprosessin vaiheisin vesiputousmallin tavoin.
Scrum	Ketteriin menetelmiin kuuluva prosessimalli, joka sisältää projektin hallintaan liittyviä käytäntöjä ja arvoja. Menetelmä ei ota kantaa projektin tuotoksiin.
Sisällön koostaminen	Eng. ”content aggregation”, sisällön keräytyminen tai ryhmittäytyminen (MOT). Yhden käyttöliittymän alle monesta eri lähteestä tuotettu sisältö.
Sisältöpala	Eng. ”fragment”, osa merkkauskielestä (esimerkiksi HTML, XHTML, WML), josta Internet-sivu koostuu. Sisältöpala on portletin tuottama osuus sivusta.
Spesifikaatio	Eng. ”Spefication”, valmistusta tms. koskevat määräykset, teknistä tuotetta koskevat tiedot (MOT).
Spring	Avoimen lähdekoodin Java-ohjelmistokehys, joka muodostaa kehyksen toteutettavalle ohjelmalle.
Toimintapyyntö	Eng. ”action request” toimintapyynnöllä portletilla voidaan suorittaa toimintoja ennen esityspyynnön suorittamista.
Unified Process	Ketterä menetelmä, joka toimii usein yritykselle luotavan prosessimallin pohjana. Se tarjoaa projektin käyttöön noin 50 erilaista vapaastivalittavaa tuotosta ja ohjeistaa projektin vaiheistamisessa.
Velocity	Java-pohjainen sivupohjamoottori, jonka avulla sivustosuunnittelijat voivat viitata Java-koodissa määriteltyihin metodeihin.
Vesiputousmalli	Yleinen ongelmaratkaisumalli, jossa ensin analysoidaan ongelma perinpohjaisesti, suunnitellaan ratkaisu, toteutetaan se ja testataan ratkaisua. Prosessimalli, joka etenee vaihe vaiheelta kuin vesiputous.
Extreme Programming (XP)	Yleisesti käytetty ketterä menetelmä, jonka tavoitteita ovat muutokseen reagointi ja asiakastyytyväisyys. Se sisältää paljon erilaisia työskentelytapoja määritteleviä tekniikoita.

SISÄLLYS

Liite 1 KETTERÄ MANIFESTI

1 JOHDANTO

Portletti on ikkunoitu verkkosovellus, jolla tuodaan sisältöä portaalisivulle jo olemassa olevista järjestelmistä ja muista tietolähteistä. Portlettitekniikalla sivustopohja ja jokainen portletti voidaan toteuttaa omana kokonaisuutenaan. Standardoitu tekniikka nopeuttaa kehittämistä ja mahdollistaa portlettien käytön uudelleen portaaleissa riippumatta valmistajan teknologiasta. Portlettitekniikka tuo kuitenkin mukanaan uusia käsitteitä ja myös teknisiä rajoitteita, jotka pitää ottaa huomioon portaalisuunnittelussa.

Ohjelmistotuotannossa käytettävillä ohjeistoilla, prosessimalleilla, on tarkoitus hallita monimutkaista ohjelmistokehitysprosessia. Prosessimallit määrittelevät ohjelmistokehittämisen eri vaiheet, suosittelevat, millä tavoin asiat pitäisivät tehdä ja ehdottavat erilaisia tuotoksia tehtäväksi vaiheiden aikana. Prosessimallin valinnalla on suuri vaikutus koko prosessin kulkuun. Monista eri prosessimalleista ketterät ohjelmistokehitysmenetelmät pyrkivät vastaamaan nykyaikaisten ohjelmistoprojektien vaatimuksiin. Ketterissä menetelmissä projektiin liittyvien riskien minimoimisessa korostetaan viestinnän ja nopean muutokseen reagoinnin merkitystä.

Opinnäytetyö paneutuu portlettipohjaisen verkkosovellusprojektin kehittämiseen keskittyen prosessimalleissa ainoastaan ketteriin menetelmiin. Työssä pyritään kirjallisuuden ja opiskelijaprojektin pohjalta löytämään parhaat tavat ja periaatteet portlettipohjaisuuden erityispiirteiden huomioimiseksi hyvin toimivien portlettipohjaisten verkkosovellusten kehittämisessä. Työssä halutaan vastaus seuraaviin kysymyksiin: Mitä käsitteitä ja erityispiirteitä portlettitekniikka tuo mukanaan? Miten portlettitekniikan erityispiirteet voidaan ottaa huomioon ohjelmistosuunnittelun eri vaiheissa?

Opinnäytetyön tilaaja on hämeenlinnalainen sähköiseen liiketoimintaan ja yhteisöllisiin ratkaisuihin erikoistunut asiantuntijayritys Ambientia. Opinnäytetyön aikana yritys toteutti opiskelijaprojektina portlettipohjaisen verkkosovelluksen Painonnostoliitolle. Opinnäytetyössä verrataan kirjallisuuden pohjalta tehtyjä päätelmiä parhaista tavoista ja periaatteista Painonnostoliitolle tehdyn portlettipohjaisen projektin kulkuun ja johtopäätösten perusteella teoriaa hiotaan käytännönläheisemmäksi ja tarkemmaksi. Työssä käytetään kvalitatiivista tutkimusmenetelmää, jossa teoriaa luodaan aineistolähtöisesti. Työelämän yhteyshenkilönä toimii Ambientian tuotanto- ja teknologiajohtaja Henri Sora.

Opinnäytetyön aiheenvalintaan vaikutti aiheen ajankohtaisuus ja hyödynnettävyys. Monet nykypäivänä toteutettavista verkkosivustoista käyttävät portlettitekniikkaa, eikä tekniikan tuomista erityispiirteistä ole vielä suomenkielistä materiaalia saatavilla. Lisäksi aihe on opinnäytetyön tekijän mielestä kiinnostava ja työn tekemisen aikana oppii paljon uutta tulevaa työuraa ajatellen.

2 PORTLETIT

Portletti on ikkunoitu verkkosovellus, jolla tuodaan sisältöä portaalisivulle jo olemassa olevista järjestelmistä ja muista tietolähteistä (Sarin 2009, 4–5).

Sarin (2009) kertoo verkkosivujen rakentamisen painopisteen siirtyneen sisällöstä käyttäjäkokemuksiin. Nykyään käyttäjä voi itse muuttaa verkkosivujen ulkoasua ja kontrolloida sivuston sisällön näkymistä. Ennen portlettien yleistymistä sivustojen muunneltavuutta ja muita ominaisuuksia toteutettiin tilapäisratkaisuilla, jotka vaikeuttivat sivujen ylläpitoa, pidensivät toimitusaikoja ja heikensivät sivustojen toteuttamisen tuottavuutta. Portlettitekniikalla voidaan kaivattuja ominaisuuksia saada sivustoille standardoidun spesifikaation mukaan. (Sarin 2009, 4–5.)

Linwood ja Minter (2004) kertovat portlettispesifikaation syntyneen tarpeesta luoda yleismäärittely, jonka avulla eri valmistajien sovellukset voidaan liittää portaaliin ja mahdollistaa sovellusten välinen yhteistyö. Portlettispesifikaatio tekee mahdolliseksi kenen tahansa valmistaman osan liittämisen kenen tahansa valmistamaan portaaliin.

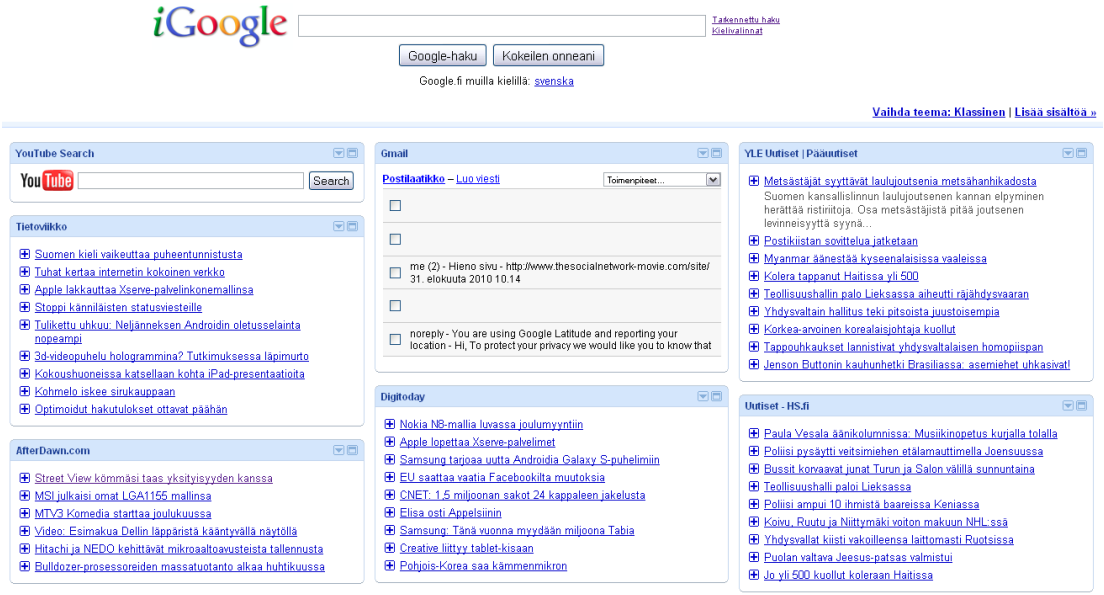
Nykyään suuri osa portaaliohjelmistojen valmistajien omista ratkaisuista on yhteensopivia portlettispesifikaation kanssa, joten niihin voi liittää minkä tahansa spesifikaatiota noudattavan portletin. Ennen spesifikaation kehittämistä jokaiselle eri valmistajan portaalille täytyi tehdä omat portletit käyttäen valmistajan omia ratkaisuja. (Linwood & Minter 2004, 1–2.)

2.1 Portaalit

Portlettispesifikaatio kuvailee portaalin verkkopohjaiseksi sovellukseksi, joka koostaa sisältöä monesta lähteestä yhden käyttöliittymän alle. Portaali toimii esityskerroksena muille informaatiojärjestelmille. Se antaa käyttäjälle mahdollisuuden personoida sivujen ulkoasun ja sisällön muunteluun. (Hepper 2008, 17.)

Sarinin (2009) mukaan portaali on kokoelma portleteiksi kutsuttuja minisovelluksia, jotka tukevat erilaisia ominaisuuksia, kuten personointi, sisällön koostaminen eri lähteistä, käyttäjän tunnistaminen ja muunneltavuus. Portletit toimivat ikkunoituina verkkosovelluksina portaalisivulla tuoden sisältöä sivustolle. Portletteja voidaan kutsua minisovelluksiksi, sillä ne tarjoavat tavallisesti vain rajoitetun määrän ominaisuuksia ja informaatiota käyttäjälle verrattuna muihin verkkosovelluksiin.

Kuvassa 1 on esimerkki portaalisivusta; iGoogle-etusivu sisäänkirjautumisen jälkeen. Sivulla näkyvistä ikkunoista, joissa on vaaleansininen otsikko, jokainen edustaa yhtä portlettia. Ne tuovat sivustolle sisältöä erilaisista tietolähteistä. Kuvassa näkyvä Gmail-portletti tuo sähköpostin tärkeimmät ominaisuudet helposti saataville, Youtube-portletilla voidaan hakea nopeasti videoita ja monia muita portletteja.

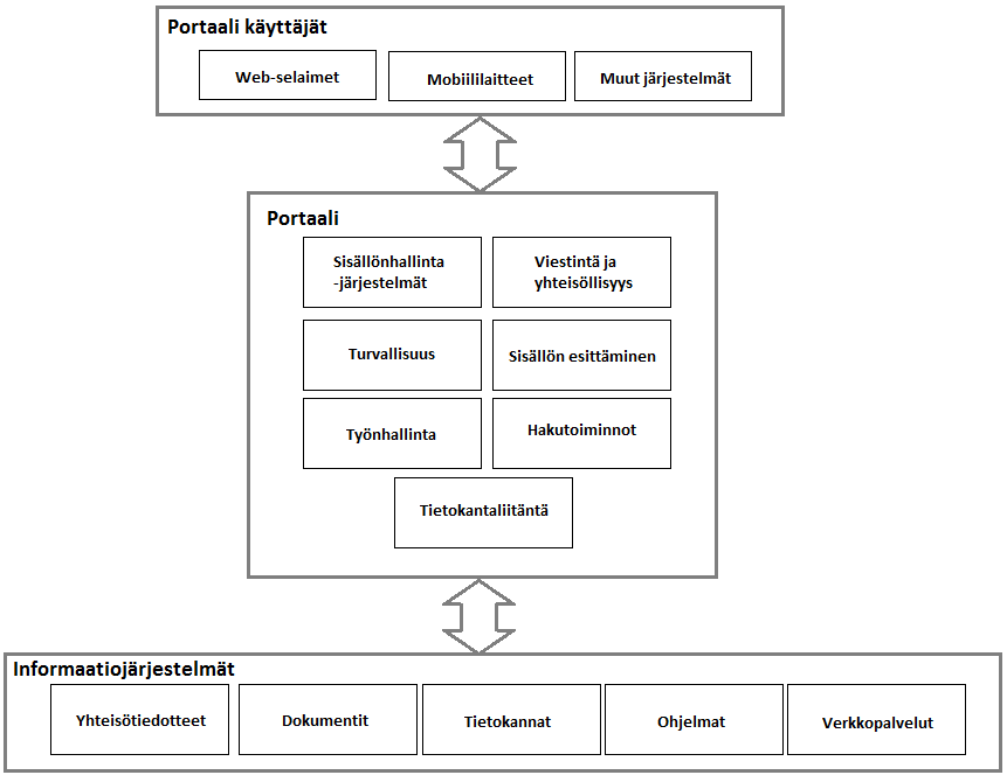


KUVA 1 *iGoogle aloitussivu sisäänkirjautumisen jälkeen (iGoogle, 2010).*

Portleteilla ei ole kaikkia samoja ominaisuuksia kuin alkuperäisillä ohjelmilla. Esimerkiksi Gmail-portletti näyttää viestit, sillä voi luoda ja poistaa viestejä, mutta siitä puuttuvat muut alkuperäisessä Gmail-verkkosovelluksesta löytyvät ominaisuudet. (Sarin 2009, 5–9.)

Personoinnilla tarkoitetaan käyttäjän mahdollisuutta muokata sivuston ulkoasua. Portletit mukautuvat portaalisivulle tehtyihin muutoksiin. Käyttäjä voi valita sivustolla näkyvät portletit, joista jokainen tuo sisältöä eri lähteistä. Portaalisivut ovat yleensä personoitu ja räätälöity tietyille käyttäjille tai käyttäjäryhmälle tarjoten eri kokoelman sisältöä eri käyttäjille (Linwood & Minter 2004, 1–2; Sarin 2009, 5–9).

Sarin (2009) kertoo portaalien käytön lisäävän tehokkuutta ja parantavan käyttökokemusta, koska käyttäjä ei tarvitse montaa eri ohjelmaa päästäkseen käsiksi haluamaansa sisältöön. Yleensä portaalit tarjoavat tavallisten verkkosovellusten eniten käytettyjä toimintoja käyttäjälle. Kun verkkosovelluksen harvemmin käytettyjä toimintoja tarvitaan, portaali ohjaa käyttäjän alkuperäiseen verkkosovellukseen, jotta tarvittavat toiminnot saadaan suoritettua. Portaalit eivät ole perinteisten verkkosovellusten korvaajia, mutta ne sopivat laajentamaan olemassa olevien verkkosovellusten toimivuutta keräämällä oleellista sisältöä olemassa olevista lähteistä helposti käyttäjän saataville. Kuvassa 2 on yleiskuva portaalista. Yläosassa näkyy portaalin käyttäjät, keskellä portaalin tarjoamia ominaisuuksia ja toimintoja ja alhaalla mahdollisia tietolähteitä, joista portaalin sisältöä voidaan koostaa. (Sarin 2009, 7–10.)

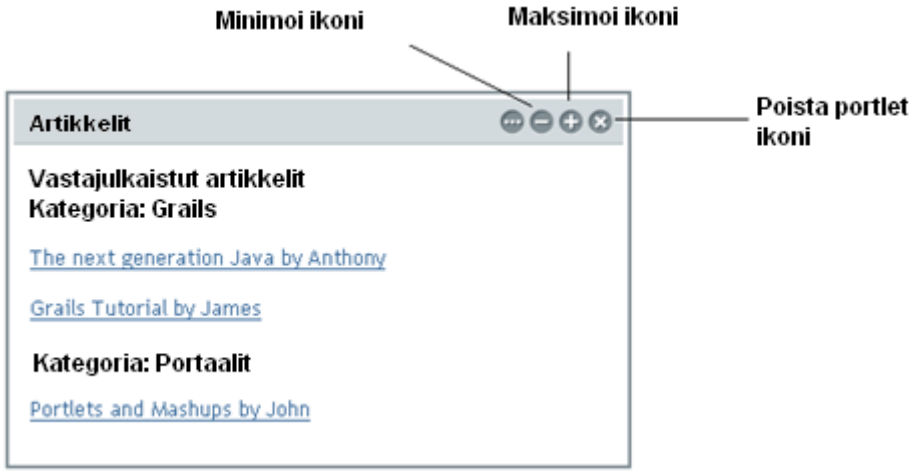


KUVA 2 *Kuluttajat, portaalin ominaisuudet, tietolähteet. Mukaeltu lähteestä Richardson, Avondolio, Vitale, Len & Smith 2004, Introduction.*

2.2 Portletit

Portlettispesifikaatio (Hepper, 2008) määrittelee portletin seuraavasti; portletti on verkkosovellus, joka tarjoaa tietyn palan sisällöstä lisättäväksi osaksi portaalisivua. Portlettia hallinnoi portlettisäiliö, joka käsittelee sivupyynnöt ja suorittaa tarvittavat toiminnot portleteilla. Portaalit käyttävät portletteja liitettävinä käyttöliittymäkomponentteina.

Portletin tuottamaa sisältöä kutsutaan sisältöpalaksi. Sisältöpala koostuu merkkauskielestä, esimerkiksi HTML, XHTML, WML, joka noudattaa tiettyjä sääntöjä. Portaali näyttää jokaisen sisältöpalan sisällön portlettiikkunoissa (Kuva 3), joista portaalisivu koostuu. Portaali voi lisätä ikkunaan otsikon, kontrollinäppäimet ja muotoiluja. (Hepper 2008, 17.)



KUVA 3 *Portletti-ikkuna, jossa portletin tuottama sisältöpala näytetään. Mukaeltu lähteestä Sarin 2009, What is Porlet?*

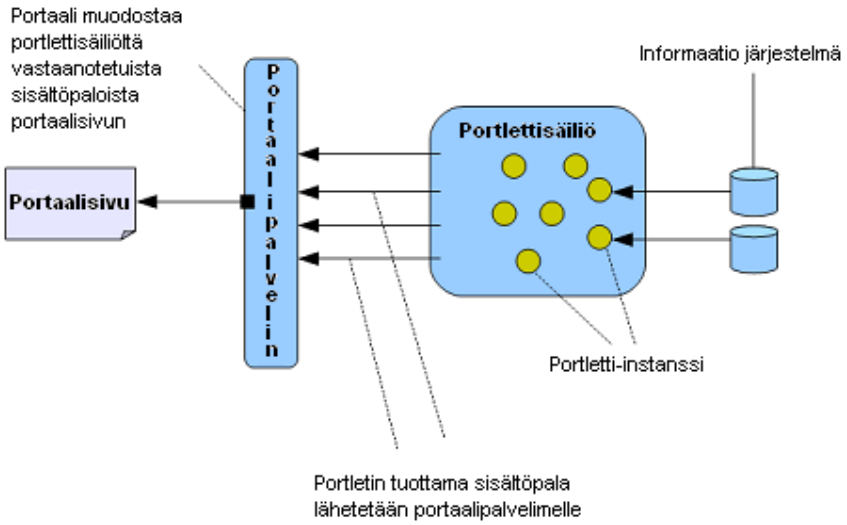
Internet-selain on yhteydessä portletteihin pyyntö–vastaus- toimintamallilla, jonka portaali toteuttaa. Tavallisesti käyttäjä toimii portlettien tuottaman sisällön kanssa, esimerkiksi seuraamalla linkkiä tai täyttämällä lomakkeen. Sen jälkeen portaali vastaanottaa toiminnon ja lähettää sen eteenpäin sisällöstä vastaavalle portletille, joka tuottaa vastauksen toimintoon. Portletin tuottama sisältö voi vaihdella riippuen käyttäjän asetuksista sekä käyttäjäryhmästä. (Hepper 2008, 17.)

Linwood ja Minter (2004) jatkavat aiheesta teknisemmällä näkökulmalla. Heidän mukaansa portletit ovat *Java 2 Enterprise Edition* (J2EE) sovelluksia, jotka noudattavat portlettispesifikaatiota toteuttaen *javax.portlet*-paketissa sijaitsevan portlettirajapinnan. Portlettiluokat pakataan *portlet.xml* sijoittelukuvauksen kanssa verkkosovellustiedostoksi *WAR*-tiedostomuotoon. Sijoittelutiedostossa määritellään portletin nimi, kuvaus, tuetut toimintatilat ja muita alkuarvoja sekä asetuksia.

Portletti vastaanottaa pyyntöjä portlettisäiliöltä ja palauttaa vastauksena sisältöpalan. Vastaus pitää sisällään osan portaalisivun sisällöstä, jonka portaali näyttää käyttäjälle. Portletin antama vastaus voi sisältää JSP-sivuja, Velocity-pohjia tai muita esityskerroksen tekniikoita. (Linwood & Minter 2004, 6, 33–32.)

2.3 Portlettisäiliö

Hepperin (2008) mukaan portlettisäiliö on portletteja hallinnoiva, niiden elinkaaresta vastaava portaalin komponentti. Portlettisäiliö mahdollistaa portlettien välisen tiedonsiirron säiliön sisällä ja portlettiasetusten pysyvän tallentamisen. Kuva 4 näyttää portaalin rakenteen ja eri osien suhteen toisiinsa.

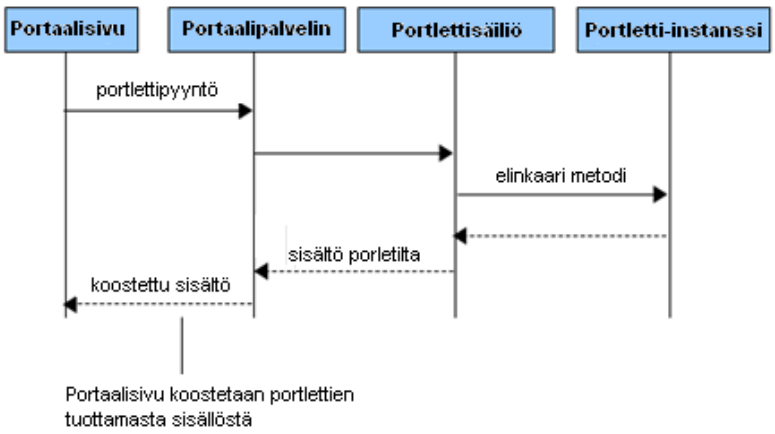


KUVA 4 *Portaalin infrastruktuuri, portlettisäiliö, portletit. Mukaeltu lähteestä Sarin 2009, Portlet Container.*

Portlettisäiliö vastaanottaa portaalilta saapuvat pyynnöt käynnistäen tarvittavan portletin. Portletti käsittelee pyynnön ja tekee tarvittavat toimenpiteet palauttaen sisältöpalan. Portlettisäiliö lähettää sisältöpalan portaalille, joka kokoaa sisältöpalasta yhdessä muiden palojen kanssa portaalisivun. (Hepper 2008, 17–18.)

Kuvassa 5 on tyypillinen tapahtumasarja käyttäjän tehtyä sivupyynnön portaalissa:

- Portaali vastaanottaa pyynnön ja tutkii, sisältääkö pyyntö toimintoja, jotka on kohdennettu portaalisivulla oleviin portletteihin.
- Jos pyyntö sisältää portletteihin kohdistettuja toimintoja, portaali pyytää portlettisäiliötä käynnistämään tarvittavat portletit.
- Portlettisäiliö pyytää portletteja tekemään tarvittavat toiminnot elinkaarimetodeilla ja palauttaa vastauksena saadun sisältöpalan eteenpäin.
- Portaali koostaa sisältöpaloista portaalisivun ja lähettää sivun takaisin käyttäjälle.



KUVA 5 *Portaalin ja portlettisäiliön roolit pyyntöjen hallinnassa. Mukaeltu lähteestä Sarin 2009, Portlet Container.*

2.4 Java portlettispesifikaatio

Richardsonin ym. (2004) mukaan portlettispesifikaatio on laajennus servlettispesifikaatiolle, joten näiden kahden tekniikan suhde toisiinsa ja eroavaisuudet on syytä tietää, jotta ymmärretään, miksi molemmat spesifikaatiot ovat olemassa.

Portletit ja servletit ovat molemmat Java 2 Enterprise Edition (J2EE) komponentteja, joita hallitsee joko portlettisäiliö tai servlettisäiliö. Molemmat toimivat pyyntö–vastausperiaatteella. Portletit palauttavat vastauksena vain osan sivun sisällöstä, sisältöpalan, kun taas servletit palauttavat kokonaisen sivun. Portletit tukevat erilaisia tiloja sekä kahta erilaita pyyntötyyppiä, joilla voidaan määritellä tarkemmin haluttua vastausta. Portletit tukevat käyttäjäkohtaisten asetusten pysyvää tallentamista, jolloin personointi ja muuntelu ovat mahdollisia.

Portletit ovat hiotumpi versio servleteistä, jotka on tarkennettu toimimaan juuri portaaleissa. Portletit asettavat enemmän rajoituksia toteutuksille, mutta mahdollistavat tämän ansiosta siirrettävyyden portaalien välillä. (Richardson, Avondolio, Vitale, Len & Smith 2004, 5–6.)

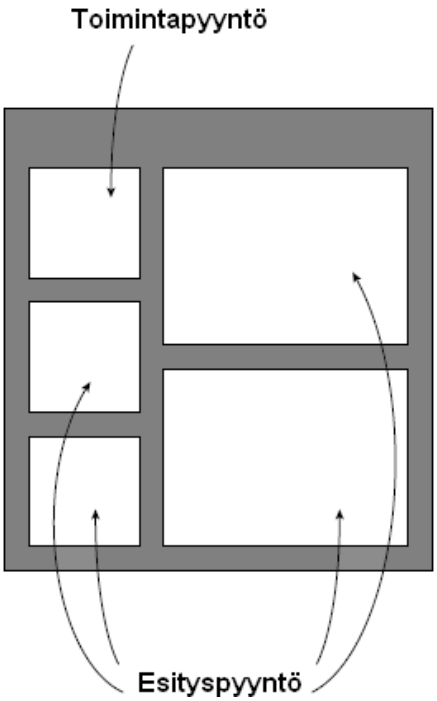
2.4.1 Portlettien elinkaari

Richardsonin ym. (2004) mukaan portlettisäiliö hallitsee portlettien elinkaarta neljällä metodilla. Niiden avulla määritellään, miten portletti alustetaan, miten se käsittelee pyyntöjä ja kuinka portletti poistetaan käytöstä. Metodeja toteuttamalla määritellään, mitä portletti palauttaa vastaukseksi portaalille. Pyyntöjen mukana siirtyvillä parametreillä valitaan käynnistettävä metodi, jos samalla elinkaarimetodilla on monta eri toteutusta.

Ennen kuin portletti voi vastaanottaa pyyntöjä, se pitää alustaa *init*-metodilla. Portlettisäiliö määrittelee *init*-metodin avulla portletin alkuarvot ja resurssit. Alustuksen aikana voidaan käynnistää kerran suoritettavia toimintoja, kuten avata yhteyksiä tietolähteisiin. Alustuksen jälkeen portletti on valmis käytettäväksi ja vastaanottamaan pyyntöjä. Alustamisen jälkeen portletti on ”käynnissä”, jolloin se voi vastaanottaa pyyntöjä. Portletti vastaanottaa kahdenlaisia pyyntöjä; toimintapyyntöjä ja esityspyyntöjä.

Toimintapyyntö suoritetaan *processAction*-metodilla. Tämän metodin voi suorittaa kerralla vain yksi portletti. Portlettisäiliö suorittaa toimintapyynnön sillä portletilla, johon käyttäjän toiminto kohdistui. Kun portletti on suorittanut toimintapyynnön, käskee portlettisäiliö muita portletteja suorittamaan esittämispyynnön *render*-metodilla. Esittämispyyntö käskee portlettia tuottaa oma sisältönsä uudelleen portletin sen hetkisen tilan perusteella.

Toimintapyynnön tarkoituksena on mahdollistaa tilan muutokset kyseisellä portletilla ennen kuin muut portletit tuottavat sopivaa sisältöä. Toimintapyyntö voi kohdistua vain yhteen portlettiin kerralla, koska käyttäjän toiminto voi kohdistua vain yhteen portlettiin (Kuva 6). Käyttäjä voi klikata vain yhtä linkkiä kerralla.



KUVA 6 *Toimintapyyntö ja esittämispyyntö. Mukaeltu lähteestä Richardson, Avondolio, Vitale, Len & Smith 2004, 10.*

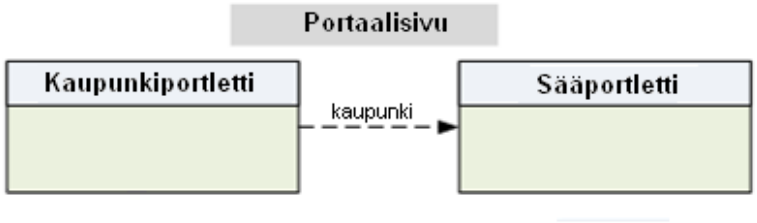
Kuvitellaan portaalisivu, joka näyttää kaupungin säätiedot valitun kaupungin perusteella. Yksi portletti toteuttaa kaupungin valinnan ja muut näyttävät säätietoja. Käyttäjän valitessa uuden kaupungin listasta, pitää muiden portlettien sivulla reagoida tilanmuutoksiin.

Neljäs metodi, jolla vaikutetaan portletin elinkaareen, on *destroy* eli tuhoametodi. Tällä metodilla vapautetaan alustuksessa jaetut resurssit, tallennetaan portletin senhetkinen tila ja voidaan suorittaa viimeisiä toimintoja. (Richardson, Avondolio, Vitale, Len & Smith 2004, 8–11.)

Portlettispesifikaation uusin versio lisää kaksi portletin elinkaareen vaikuttavaa metodia, *processEvent*, joka suoritetaan toimintapyynnön ja esityspyynnön välissä sekä *serveResource*, jolla voidaan jakaa resursseja portletille (Hepper 2008, 42–45).

2.4.2 Portlettien välinen tiedonsiirto

Portletit voivat keskustella toistensa kanssa käyttäjäkohtaisen portlettisession avulla. Aikaisemmin mainitussa tilanteessa, jossa yhden portletin avulla valitaan kaupunki (Kuva 7), toisen portletin näyttäessä valitun kaupungin säätiedot, pitää tieto valittuna olevasta kaupungista jakaa. Tämä voidaan toteuttaa portlettien välisen tiedonsiirron avulla. (Linwood & Minter 2004, 102.)



KUVA 7 *Portlettien välinen tiedonsiirto. Mukaeltu lähteestä Sarin 2009, Why Portlets?*

Portletit voivat käyttää sessioita tiedon tallennukseen ja välittämiseen hyödyntämällä portlettispesifikaation tarjoamaa *PortletSession*-rajapintaa. Sessiohin voidaan tallentaa avain-arvopareja, joissa on vapaasti määriteltyjä objekteja. Portlettien käyttämillä sessioilla on kaksi erilaista näkyvyysmääritystä ja sessioon tallennetut muuttujat voidaan asettaa näkymään kaikille portleteille tai ainoastaan kyseiselle portletille. (Richardson, Avondolio, Vitale, Len & Smith 2004, 18–19.)

Toinen mahdollinen keino portlettien väliseen tiedonsiirtoon on porlettiosoitteet, *portletURL*, jotka ovat samalla kutsukäskyjä portlettien elinkaarimetodien käynnistämiseen. Portlettiosoitteita on kahdenlaisia, toimintaosoitteita, *actionURL* ja esitysosoitteita, *renderURL*. Esitysosoite mahdollistaa toimintapyynnön ohittamisen kutsumalla ainoastaan portletin esityspyyntöä. Molempien osoitteiden mukana voidaan antaa parametreja metodeille. (Richardson, Avondolio, Vitale, Len & Smith 2004, 14.)

Portlettispesifikaation uusin versio määrittelee portleteille kaksi tiedonsiirtotapaa. Lisää *portletEvent*-käsky mahdollistaa toimintapyynnön vastaanottaneen portletin lähettä parametrejä eteenpäin. Julkiset esityparametrit -määrityksellä voidaan portletin preferenssiobjektiin määritellä julkisia parametreja, joihin myös muilla portleteilla on oikeus. (Hepper 2008, 41–45, 51–53.)

2.4.3 GenericPortlet

Sarinin (2009) mukaan *GenericPortlet* on abstraktiluokka, joka toteuttaa portlettirajapinnan. Yleisin tapa lähteä tekemään uutta portlettia on laajentaa *GenericPortlet*-luokkaa, joka pohjaluokkana tarjoaa valmiin perustoteutuksen portlettispesifikaatioon.

Kuvassa 8 nähdään ”HelloWorldPortlet”-luokka, joka laajentaa GenericPortlettia. Luokan alussa ilmoitetaan, missä paketissa luokka sijaitsee, minkä jälkeen tuodaan tarvittavat muut luokat. Kohdassa #3 ilmoitetaan annotaatiolla seuraavan metodin toteuttavan esityspyynnön. Kuvassa näkyvä portletti tuottaa esityspyynnön saapuessa sisältöpalan, joka sisältää tekstin ”Hello World”. Pakkaamalla luokka portlettispesifikaation mukaisesti voitaisiin portletti lisätä portaaliin. Kuvassa 8 näkyvä portletti ei ylikirjoita, *init*- ja *destroy*-metodeja, jolloin kyseiset metodit toimivat *GenericPortlet*-luokassa määritellyllä perustoteutuksella. (Sarin 2009, 35–37.)

```
1  package helloworldportlet.hamk.fi;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.portlet.*;
6
7  public class HelloWorldPortlet extends GenericPortlet
8  {
9      @RenderMode(name = "VIEW")
10     public void sayHello(RenderRequest request,
11         RenderResponse response) throws
12             PortletException, IOException {
13
14         PrintWriter out = response.getWriter();
15         out.println("Hello World");
16     }
17 }
```

KUVA 8 *”Hello World”-portletti. Mukaeltu lähteestä Sarin 2009, Hello World Portlet Example.*

2.4.4 Preferenssiobjekti

Portlettien ollessa ”käynnissä” jokaiseen portlettiin on yhdistettynä yksi tai useampi yksilöllinen preferenssiobjekti, johon voidaan tallentaa käyttäjän tekemiä asetuksia. Objektin käytöllä on mahdollista personointi ja muunneltavuus. Objektiin ladataan käynnistyksen yhteydessä sijoittelukuvauksessa määritellyt alkuarvot, mutta objektin arvoja on kuitenkin mahdollista muuttaa ohjelmakoodin avulla. (Richardson, Avondolio, Vitale, Len & Smith 2004, 8–9.)

Preferenssiobjektin avulla voidaan esimerkiksi sääportaalisivulla tallentaa käyttäjän kotikaupunki, jolloin kirjautuessaan käyttäjä näkee oletuksena oman kotikaupunkinsa sään, jos hän on sen aikaisemmin asettanut.

2.4.5 Portletin toimintatilat

Richardsonin ym. (2004) mukaan portleteilla on kolme toiminnallista tilaa, joilla voidaan määritellä vastaus esityspyyntöön. Portletti voidaan asettaa vastaamaan eri tavalla riippuen portletin senhetkisestä tilasta.

Näitä kolmea tilaa on tarkoitettu käytettäväksi seuraavasti; *View*-tila on portletin normaalitila, joka tuottaa sisältöpalan senhetkisten asetusten perusteella. View voidaan toteuttaa kirjoittamalla *doView*-metodi luokkaan. *Edit*-tila tuottaa sisältöpalan, jolla mahdollistetaan portletin asetusten muokkaaminen. Edit toteutetaan kirjoittamalla *doEdit*-metodi. *Help*-tila tuottaa sisältöpalan, jolla voidaan näyttää käyttäjälle ohjeita portletin käyttöön ja se toteutetaan kirjoittamalla *doHelp*-metodi. (Richardson, Avondolio, Vitale, Len & Smith 2004, 15.)

2.4.6 Portletti-ikkunan tilat

Portaali näyttää jokaisen portletin omassa portletti-ikkunassa. Richardsonin ym. (2004) mukaan portletti-ikkunalla on kolme tilaa, jotka määrittelevät, kuinka paljon tilaa ikkuna vie portaalisivulta. *Normal*–tilassa portletti jakaa portaalisivun sillä olevien muiden ikkunoiden kanssa. Portletin tulisi rajoittaa tuottamaansa sisältöpalan kokoa. *Minimixed*–tilassa portletti-ikkuna on minimoituna, jolloin sen tulisi tuottaa vähän tai ei ollenkaan sisältöä. *Maximixed*–tilassa portletti ei jaa portaalisivua muiden kanssa, jolloin sisältöpalan kokoa ei tarvitse rajoittaa. (Richardson, Avondolio, Vitale, Len & Smith 2004, 16.)

2.5 Portaali- ja portlettisuunnittelu

Richardson ym. (2004) kehottavat miettimään ennen aloittamista, onko portaali oikea ratkaisu ongelmaan. Usein riittävä ratkaisu on perinteinen verkkosovellus, jolta ei vaadita portaalin ominaisuuksia. Jos halutaan esimerkiksi uusi laskutusohjelma, sen tekemiseen ei tarvita portaalia, vaan ohjelma voidaan rakentaa verkkosovelluksena ottaen huomioon mahdollisesti myöhemmin rakennettava portaali.

Aina ei tarvita kokonaista portaalia ongelman ratkaisemiseksi. Jos verkkosovellus on tehty kunnolla, saadaan se helposti jälkikäteen liitettyä portaaliin. Portaalin etu on useiden ohjelmien tarjoaminen yhden käyttöliittymän alla. Kaikkia ohjelmia ei kuitenkaan voida laittaa portaaliin, sillä samalla menetettäisiin merkittävästi toiminnallisuutta. (Richardson, Avondolio, Vitale, Len & Smith 2004, xxviii–xxix.)

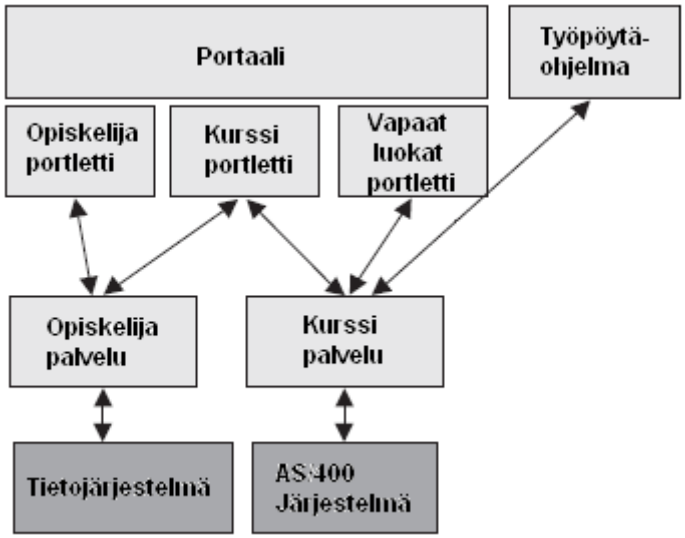
Sarin (2009) pitää portaalin kehittämistä kannattavana, jos halutaan tuoda sisältöä monesta tietolähteestä, tarjota se yhden palvelun kautta ja antaa käyttäjälle mahdollisuus muuntaa omakohtaisesti sisältöä ja ulkoasua (Sarin 2009, 11).

Linwoodin ja Minterin (2004) mukaan portaalin toteuttamispäätöksen jälkeen portaalisuunnittelun ensimmäinen vaihe on kerätä käyttäjien vaatimukset ja suunnitella informaatioarkkitehtuuri portaaliprojektia varten. Informaatioarkkitehtuurin suunnittelussa otetaan huomioon käyttäjän tarpeet. Informaatioarkkitehtuuri sisältää portaalin kautta näytettävän sisällön, käyttöliittymän, tulevat portletit, metatiedon ja hakutoiminnot.

Informaatioarkkitehtuurin suunnitteleminen on tärkeä vaihe, sillä portaali koostaa sisältöä jo olemassa olevista sisällönhallintaohjelmista, dokumenttien hallintajärjestelmistä, tietokannoista ja monista muista lähteistä. Portaalin sisällön suunnittelussa pitää ottaa huomioon mahdolliset liitäntä- ja käyttöoikeusongelmat. Jo olemassa olevasta järjestelmästä sisällön tuominen ei välttämättä onnistu suoraan, vaan järjestelmien väliin joudutaan rakentamaan muuntimia. Niitä rakennettaessa on selvitettävä tuotavan sisällön käyttöoikeudet ja yhdistäminen vanhan järjestelmän käyttöoikeuksiin.

Portaalit sopivat hyvin toteuttamaan palvelukeskeistä arkkitehtuuria, jossa toiminnallinen logiikka tuodaan palveluna portaaliin esimerkiksi Web Services-tekniikan avulla. Portaalin portletti sisältää ainoastaan käyttöliittymän ja kontrollilogiikan. Portletti kutsuu tarpeen mukaan etäpalvelulta tietoa tai toiminnon suorittamista.

Kuvassa 9 on esimerkki palvelukeskeisestä toteutuksesta koulun tietojärjestelmässä. Portaali kokoaa kolmella portletilla tietoa etäjärjestelmistä. Portaalin rinnalla on toiminnallisempi työpöytäohjelma kurssiaikataulujen hallintaan. (Linwood & Minter 2004, 25–33.)



KUVA 9 *Palvelukeskeinen arkkitehtuuri portaalissa. Mukaeltu lähteestä Linwood & Minter 2004, 32.*

3 PROSESSIMALLIT

Ohjelmistotuotannossa käytettävillä ohjeistoilla, prosessimalleilla, on tarkoitus hallita monimutkaista ohjelmistokehitysprosessia. Prosessimallit määrittelevät ohjelmistokehittämisen eri vaiheet ja suosittelevat, millä tavoin asiat pitäisi tehdä ja ehdottavat erilaisia tuotoksia tehtäväksi vaiheiden aikana. Prosessimallin valinnalla on suuri vaikutus koko prosessin kulkuun.

3.1 Prosessimallien vaiheet

Haikalan ja Märijärven (2006) mukaan kaikissa prosessimalleissa esiintyy ainakin määrittely-, suunnittelu-, ohjelmointi- ja testausvaihe. Projektin alussa ennen muiden vaiheiden alkamista tehdään esitutkimukseksi tai tarvekartoitukseksi kutsuttu vaihe. Kaikkiin vaiheisiin liittyy mukaan tarkistus- ja testaustoimenpiteitä, joilla pyritään poistamaan virheitä järjestelmästä mahdollisimman aikaisessa vaiheessa. Vaiheiden välissä pidettävissä katselmuksissa todetaan projektin tila ja tarkistetaan, onko kaikki vaiheen tavoitteet saavutettu.

Esitutkimuksella ennen projektin varsinaista aloittamista vastataan esimerkiksi kysymyksiin: Mikä on ratkaistava ongelma, onko ratkaisua olemassa, mitä se saa maksaa ja mitä reunaehtoja sillä on? Nämä asiakasvaatimukset määrittelevät asiakkaan tarpeet, mutta eivät ota kantaa siihen, minkälainen järjestelmä täyttää asiakkaan vaatimukset. Tässä vaiheessa voidaan vielä päättää, toteutetaanko projekti vai ei. Esitutkimus mielletään usein osaksi määrittelyä, koska asiakastarpeiden analysointi ja tarkentaminen jatkuvat koko määrittelyvaiheen ajan.

Määrittelyvaiheessa arvioidaan projektin tarpeellisuutta ja toteuttamiskelpoisuutta. Asiakasvaatimusten perusteella laaditaan ohjelmistovaatimukset, jotka kuvaavat toteutettavat ominaisuudet, toteutukselle asetetut eitoiminnalliset ja toiminnalliset vaatimukset sekä rajoitukset. Lisäksi kuvaillaan toteutettavat käyttöliittymät ja liittymät muihin järjestelmiin.

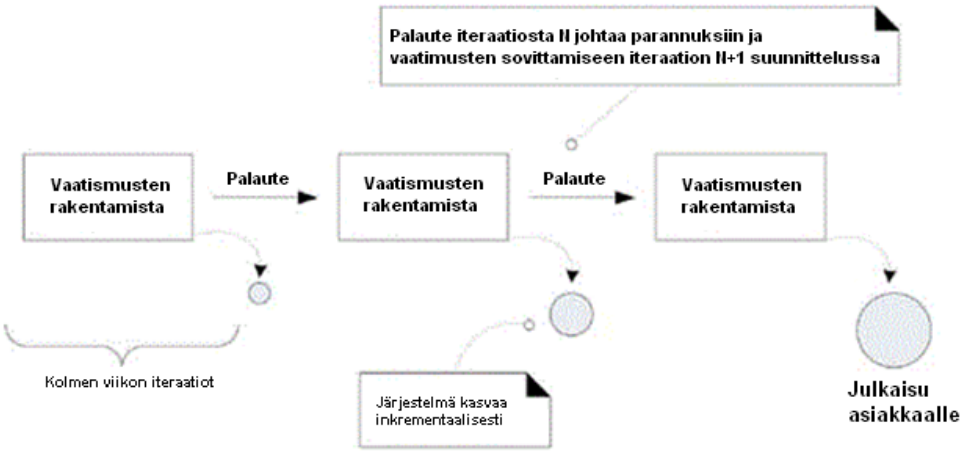
Suunnitteluvaiheessa suunnitellaan määrittelyssä kuvailtujen toimintojen toteutus. Suunnittelu on yleensä kaksivaiheinen, arkkitehtuurisuunnittelussa järjestelmä jaetaan toisistaan riippumattomiin osiin moduuleiksi. Moduulisuunnittelussa suunnitellaan jokaisen moduulin sisäinen rakenne.

Ohjelmointivaiheessa toteutetaan osa tai kaikki aikaisemmissa vaiheissa suunnitellut ja määritellyt osat.

Testausvaiheessa ohjelmistoa testataan mahdollisten ohjelmistovirheiden varalta. Testaus jaetaan yleensä osiin, moduulitestaukseen, integrointitestaukseen ja järjestelmätestaukseen. (Haikala & Märijärvi 2006, 37–40, 79.)

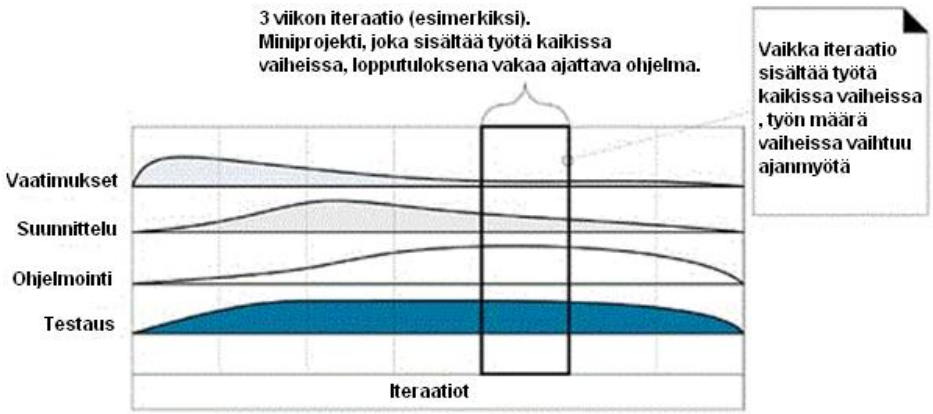
3.2 Iteratiivinen ohjelmistokehitys

Iteraatio on pieni projekti, joka sisältää määrittelyn, suunnittelun, ohjelmoinnin ja testauksen. Koko projekti koostuu useista iteraatioista. Iteraation tavoitteena on tuottaa vakaa, integroitu ja testattu, osittain valmis ohjelmisto. Iteraatio voi olla olemassaolevien ominaisuuksien säätämistä tai suorituskyvyn parantamista, mutta yleensä jokainen iteraatio kasvattaa ohjelmistoa uusilla ominaisuuksilla, kuten kuvassa 10. Iteratiivisessa ja inkrementaalisessa ohjelmistokehityksessä jokainen iteraatio kasvattaa ohjelmistoa. Tällainen ominaisuus on kaikkien ketterien menetelmien ydinajatuksena. (Larman 2004, Iterative Development.)



KUVA 10 *Iteratiivinen ja inkrementaalinen ohjelmistokehitys, jokainen iteraatio kasvattaa ohjelmistoa. Mukaeltu lähteestä Larman 2004, Iterative Development.*

Nykyiset iteratiiviset menetelmät suosittelevat iteraatioden pituudeksi yhdestä kuuteen viikkoa. Kaikki iteraatiot sisältävät myös ohjelmointia, eikä vain esimerkiksi vaatimusmäärittelyä. Jokaisen iteraation tuotos lisätään kokonaisuuteen, josta lopulta muodostuu lopputuote. Kuva 11 kuvaa eri työvaiheiden määrän muutosta iteraatioden edetessä. Projektin alussa iteraatiot sisältävät enemmän vaatimusmäärittelyä ja suunnittelua, kun taas projektin loppupuolella on enemmän ohjelmointia ja testausta. (Larman 2004, Iterative & Evolutionary.)



KUVA 11 *Iteraatioiden sisältö projektin aikana. Mukaeltu lähteestä Larman 2004, Iterative Development.*

3.3 Ketterät menetelmät

Ketterät menetelmät ovat prosessimalleja, jotka pyrkivät nopealla reagoinnilla hallitsemaan projektin aikana esiintyvää epävarmuutta. Ketterissä menetelmissä korostuvat lisäksi projektin henkilöstö, sujuva vuorovaikutus, yhteistyö asiakkaan kanssa, yksinkertaiset käytännöt sekä palaute. Ketterille menetelmille yhteisiä käytäntöjä ovat aikalaatikoidut iteraatiot ja evolutiivinen kehittäminen. (Larman 2004, Agile.)

Ketterä manifesti on kaikkia ketteriä menetelmiä koskeva, arvoperusteinen ohjeisto, jossa on neljä menetelmille tyypillistä arvoa ja 12 periaatetta (Liite 1).

3.3.1 Ketterien menetelmien käytäntöjä

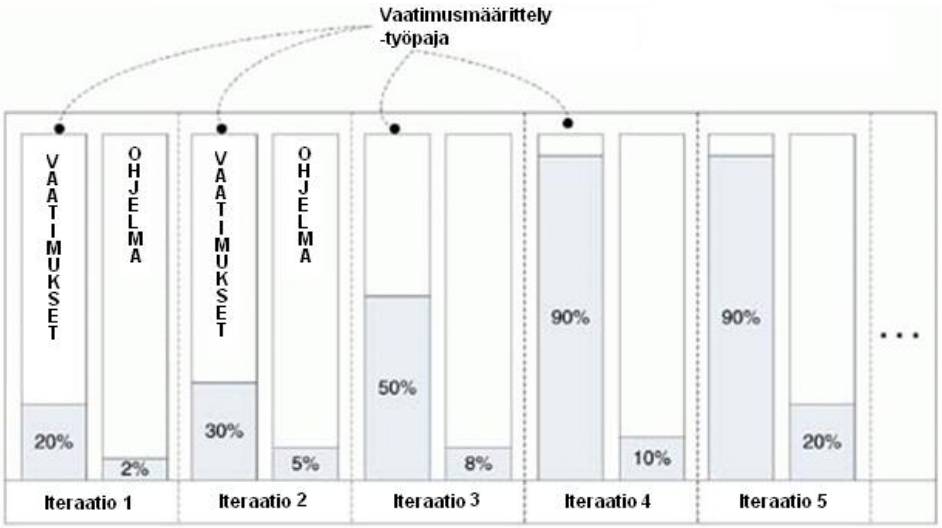
Menetelmät suosittelevat *riskiohjautuvan* kehittämisen ja *asiakasohjautuvan* kehittämisen yhdistämistä. Ensimmäisessä tapauksessa vaikeimmin toteutettavat, erittäin riskialttiit ominaisuudet työstetään aikaisissa iteraatioissa. Toisessa tapauksessa seuraavan iteraation ominaisuudet toteutetaan asiakkaan toiveiden mukaan. Kumpikin lähtökohta on tarpeen huomioida, sillä asiakas ei välttämättä aina tiedä, mikä on teknillisesti hankalinta tai riskialtteinta toteuttaa. Ohjelmistokehittäjä ei välttämättä taas tiedä, millä omaisuudella on eniten arvoa asiakkaalle. (Larman 2004, Risk-Driven and Client-Driven Iterative Planning.)

Aikalaatikointi on iteratiivisten menetelmien käytäntö, jossa iteraatiolle on sovittu kiinteä lopetuspäivä, jonka ei anneta muuttua, vaikka kaikkia aiottuja ominaisuuksia ei olisi toteutettu. Jos lopulta huomataan, ettei kaikkia toteutettavaksi valittuja ominaisuuksia ehditä toteuttaa sovittuissa aikatauluissa, karsitaan vähemmän tärkeitä ominaisuuksia. (Larman 2004, Timeboxed Iterative Development.)

Evolutiivinen kehittäminen on lähestymismalli ohjelmistokehitykseen, jossa vaatimusmäärittely, suunnitelma, toteutusarviot ja ratkaisu eli ohjelmistokoodi kehittyy iteraatioiden edetessä, toisin kuin tarkoissa etukäteismäärittelyissä, joissa määrittely on valmis ja ”jäädytetty” ennen toteuttamisen aloittamista. Evolutiivisen ohjelmistokehittämisen ideana on rakentaa ensimmäisessä julkaisussa ydinjärjestelmä, johon seuraavissa julkaisuissa lisätään uusia ominaisuuksia. (Larman 2004, Evolutionary and Adaptive Development; Haikala & Märijärvi 2006, 40.)

Evolutiiviseen kehittämiseen liittyy *joustava kehittäminen*, jolla tarkoitetaan kehittämisen mukautumista työstä saatuun palautteeseen. Päämäärä on sama kuin evolutiivisessa kehittämisessä, mutta käsitteellä korostetaan mukautumista käyttäjiltä, testaajilta, kehittäjiltä ja muilta projektiin liittyviltä ryhmiltä saatuun palautteeseen. (Larman 2004, Evolutionary and Adaptive Development.)

Evolutiivinen ja joustava kehittäminen painottavat vaatimusten muutostarvetta iteraatioiden aikana. Suurin osa vaatimusten määrittelystä ja tarkentamisesta tapahtuu yleensä aikaisten iteraatioiden aikana. Niissä keskitytään tärkeiden rakenneratkaisujen ja yritystoiminnan kannalta tärkeimpien vaatimusten löytämiseen. Esimerkiksi kokonaisuudessaan 20 iteraation projektissa on tavallista, että suurin osa vaatimusmäärittelystä tapahtuu ensimmäisen kolmen tai neljän iteraation aikana. Kuvassa 12 on vaatimusmäärittelyn määrä esimerkki projektissa. Ensimmäisen iteraation aikana määritellään 20 % vaatimuksista, toisessa 30 % ja jne. (Larman 2004, Evolutionary and Adaptive Development.)



KUVA 12 *Vaatimusmäärittelyn valmistuminen iteraatioden aikana. Mukaeltu lähteestä Larman 2004, Evolutionary and iterative requirements.*

Jokaisessa iteraatiossa on yhdestä kahteen päivään kestävä vaatimusmäärittelytyöpaja, jossa määrittelyä laajennetaan ja tarkennetaan analyysitulosten ja palautteen perusteella. (Larman 2004, Iterative & Evolutive.)

Inkrementaalinen julkaisu tarkoittaa käytäntöä, jossa kehitysprojekti on jaettu iteraatiosarjoihin. Ohjelmisto tuodaan julkisuuteen kolmesta kahteentoista kuukauteen kestävän iterointisarjan jälkeen. Seuraavassa iteraatiosarjassa ominaisuuksien määrää kasvatetaan aikaisemmasta ja julkaistaan tuotteen seuraava versio. Kuvassa 13 nähdään esimerkkiprojekti, joka on jaettu kahteen iteraatiosarjaan, joista ensimmäinen sisältää 10 iteraatiota ja toinen 7. Molempien sarjojen jälkeen ohjelmisto julkaistaan. Inkrementaalinen julkaisu sekoitetaan usein iteratiiviseen kehittämiseen. (Larman 2004, Incremental delivery.)

KUVA 13 *Inkrementaalinen julkaisu ja iteraatiot. Mukaeltu lähteestä Larman 2004, Incrementel Delivery.*

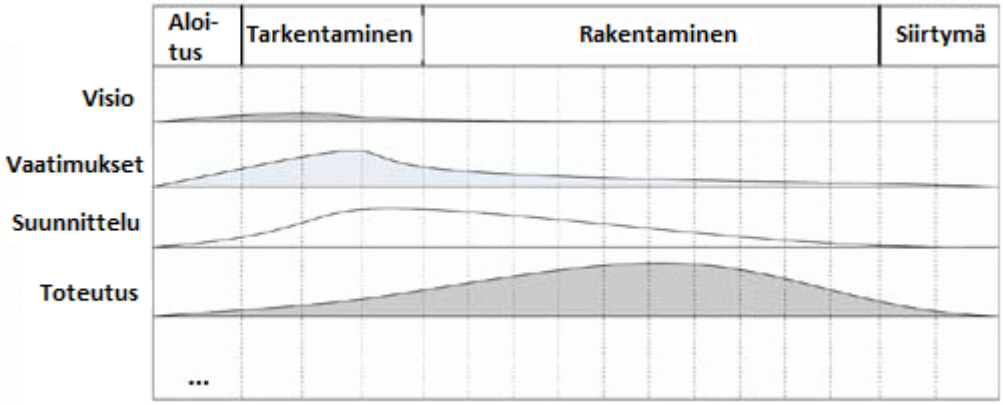
Evolutiivinen julkaisu on pidemmälle hiottu versio inkrementaalisesta julkaisusta, jossa painotetaan voimakkaasti jokaisen julkaisun jälkeen saadun palautteen merkitystä seuraavan julkaisun tiennäyttäjänä. (Larman 2004, Evolutionary Delivery).

Ketterä projektin hallinta painottaa projektin vision luomista, vuorovaikutuksen lisäämistä ja pyrkimystä välttää tiukkoja määräyksiä. Projektin johtamista ja suunnittelun vastuuta pyritään hajauttamaan koko projektiryhmälle. Projektipäällikön tehtävänä ryhmän valmentaminen, tarvittavien resurssien hankkiminen ja projektia ohjaavan vision ylläpitäminen. (Larman 2004, Agile Project Management.)

3.3.2 Unified Process

Larmanin (2004) mukaan Unified Process on kattava iteratiivisen ja inkrementaalisen prosessin viitekehys, joka on pohjana yrityksen tarpeisiin luotavassa prosessimallissa. Sen tunnetuin pidemmälle hiottu versio on kaupallinen Rational Unified Process. Menetelmän pääpiirteinä ovat käyttötapauslähtöisyys ja arkkitehtuurikeskeisyys. Käyttötapauksilla mallinnetaan toiminnallisia vaatimuksia ja määritellään sisältöä iteraatioihin. Arkkitehtuurikeskeisyydellä tarkoitetaan rakenteen suunnittelemisen korostamista. Menetelmässä arkkitehtuurin suunnittelu ja rakentaminen on erotettu omaksi vaiheekseen projektissa. Huomattavin ero muihin menetelmiin verrattuna on sen skaalatutuvuus. Se tarjoaa projektin käyttöön noin 50 erilaista vapaasti valittavaa tuotosta. Menetelmää on mahdollista soveltaa muutaman henkilön projekteista suuriin, yli satojen ihmisten äärimmäistä toimintavarmuutta vaativiin projekteihin. Unified Process edellyttää prosessin räätälöintiä. Jokaiselle projektille tulisi valita käytännöt ja projektin aikana tehtävät tuotokset. Kaikissa projekteissa suositellaan käytettäväksi ainakin riskilistaa ja Visio-dokumenttia, josta selviää sidosryhmien todelliset tarpeet.

Unified Process jakaa projektin neljään osaan: Aloitus on lyhyt parin päivän jakso, jonka tarkoituksena kerätä noin 10 % vaatimuksista tarkasti ja tehdä luonnokset tärkeimmistä projektin tuotoksista. Tarkentamisvaiheen iteraatioissa luodaan ydinarkkitehtuuri valmiiksi ja testataan se. Vaiheen aikana pidetään työpajoja, joissa vaatimuksia tarkennetaan. Loput järjestelmästä toteutetaan rakentamisvaiheen iteraatioissa. Tässä vaiheessa vaatimukset voi vielä muuttua, mutta suurimmat muutokset ovat toivottavasti saatu selville. Siirtymävaiheen iteraatioissa ohjelmisto julkaistaan. Kuvassa 14 havainnollistetaan Unified Process-vaiheita ja iteraatioiden määrää projektissa. Viivan paksuudella kuvataan vaiheen työmäärää. Huomioitavaa on, että tarvittavien iteraatioiden jako ja määrä riippuvat projektista. (Larman 2004, Unified Process.)



KUVA 14 *Iteraatoiden ja vaiheeseen tarvittavien resurssien määrä projektissa. Mukaeltu lähteestä Larman 2004, Unified Process.*

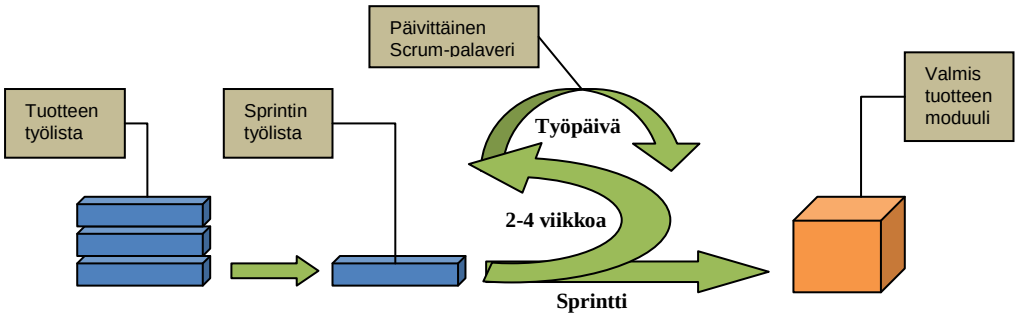
Unified Process listaa kuusi tärkeintä käytäntöä. Ensimmäinen on kahdesta kuuteen viikkoa kestävien aikalaatikoitujen iteraatioiden käyttäminen. Ohjelmointi tulee aloittaa heti tärkeimpien vaatimusten määrittelyn jälkeen. Muut vaatimukset tarkentuvat saadusta palautteesta työn edetessä. Toinen käytäntö on suurimman riskin omaavien elementtien toteuttaminen aikaisissa iteraatioissa ja olemassa olevien komponenttien uudelleenkäyttö. Kolmas on jatkuva laadunvarmistaminen, joka pitää sisällään testaamisen useasti aikaisessa vaiheessa. Jokaisen iteraation jälkeen koko järjestelmä tulee testata muiden tuotosten lisäksi. Neljännessä käytännössä neuvotaan ennen ohjelmoinnin aloittamista mallintamaan edes tyydyttävä määrä, esimerkiksi tunnin ajan kirjoitustaululle käyttäen vapaamuotoista UML-merkintää. Viides käytäntö koskee vaatimusten hallintaa. Projektissa tulisi olla keinoja, joilla voidaan järjestää ja lajitella vaatimuksia riskin, prioriteetin ja statuksen perusteella. Vaatimuksia tulisi lisätä ja tarkentaa hyödyntämällä käyttötapauksia. Viimeinen käytäntö on versionhallinnalla ja muutospyyntöprotokollalla tehtävä kurinalainen muutoksenhallinta. (Larman 2004, Unified Process.)

3.3.3 Scrum

Larmanin (2004) mukaan Scrum-menetelmä sisältää projektin hallintaan liittyviä käytäntöjä ja arvoja. Menetelmä ohjeistaa projektiryhmän työskentelyä ottamatta kantaa tuotoksiin. Tämän ansiosta Scrum voidaan yhdistää muihin menetelmiin tai sillä voidaan täydentää muita. Itseohjautuvien ryhmien, päivittäisten arviointien ja ohjailun välttämisen korostaminen erottaa Scrum-menetelmän muista ohjelmistonkehitysmenetelmistä. Itseorganisoituvan ryhmän projektipäällikkö pyrkii poistamaan esteitä projektin tieltä ja kannustaa ryhmää itsenäisiin ratkaisuihin, mutta ei neuvo ryhmää ongelmien ratkaisuissa. Iteraation aikana pidettävät palaverit ovat jokaisena päivänä samassa paikassa samaan aikaan. Iteraation ollessa käynnissä siihen ei lisätä tehtäviä. Tällä tavoin pyritään säilyttämään herpaantumaton keskittyminen iteraation toteuttamiseen. Ihanteena pidetään koko projektiryhmän työskentelyä samassa työtilassa. Jokaisen iteraation jälkeen seuraa Sprint-tarkistus, jossa käydään läpi iteraation kulku ja tulokset.

Scrum-menetelmä suosittelee iteraation pituudeksi 30 päivää, mutta ajan suhteen voidaan joustaa. Projektiryhmä päättää sopivasta määrästä dokumentointia, katselmointeja ja muita vaiheita. Scrumin kehittäjät neuvovat karsimaan muodollisuudet projektissa mahdollisimman vähiin. Projektiryhmän kooksi suositellaan seitsemää henkilöä tai pienempää määrää, mutta projektiin voi kuulua useita ryhmiä. Jokaisesta ryhmästä valitaan yksi jäsen projektipäälliköksi, jonka tehtävänä on huolehtia ryhmän toiminnasta ja palaverien etenemisestä.

Scrum-menetelmä jakaa projektin neljään vaiheeseen (Kuva 15), joista kahta ensimmäistä kutsutaan esipeliksi, joka sisältää suunnittelu- ja valmisteluvaiheet. Kaksi jälkimmäistä ovat sprintiksi ja julkaisuksi nimettyä toteutusvaihetta. Suunnitteluvaiheessa luodaan projektin työlista vaadituista ominaisuuksissa. Valmisteluvaiheessa valitaan seuraavassa sprintissä toteutettavat ominaisuudet sprintin työlistalle ja pohditaan ominaisuuksien parhaita toteutustapoja. Näissä kahdessa vaiheessa sidosryhmien edustajat ovat vahvasti mukana. (Larman 2004, Scrum.)



KUVA 15 *Scrum-prosessi. Mukaeltu lähteestä Huttunen 2006, 25.*

Sprintti-vaiheessa työlistan ominaisuudet toteutetaan ja pidetään lyhyt Scrum-palaveri jokaisena työpäivänä. Sen aikana jokainen ryhmäläinen vastaa vuorollaan Scrum-kysymyksiin:

1. Mitä olet tehnyt viime palaverin jälkeen?
2. Mitä teet työlistalta ennen seuraavaa palaveria?
3. Mitä esteitä sinulla on työsi edistymisessä?

Scrum-palaveria johtaa projektipäällikkö, joka samalla huolehtii, että jokainen jäsen vastaa kaikkiin kolmeen kysymykseen. Scrum-palaverien tarkoituksena on löytää mahdolliset ongelmakohdat mahdollisimman nopeasti, jotta niihin voidaan reagoida. Sprintti-vaiheen jälkeen pidetään saavutettujen tavoitteiden katselmointi. (Larman 2004, Scrum.)

3.3.4 Extreme Programming (XP)

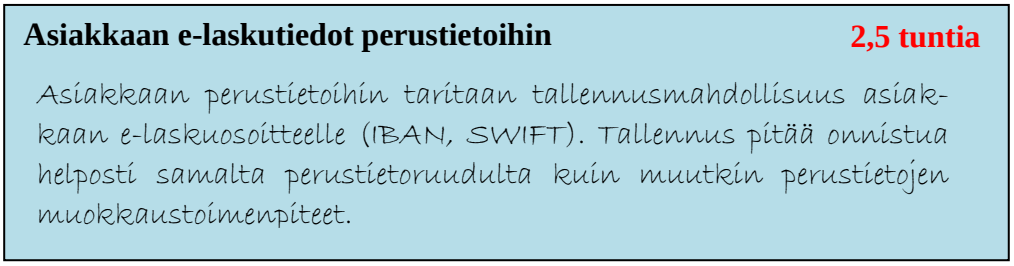
Larmanin (2004) mukaan Extreme Programming on yleisesti käytetty ketterä menetelmä, jonka tavoitteita ovat muutokseen reagointi ja asiakastyytyväisyys. Muihin menetelmiin verrattuna XP on erittäin käytännönläheinen. Se sisältää paljon erilaisia työskentelytapoja määritteleviä tekniikoita, kuten pariohjelmointi, tarinakorttien käyttö, testivetoinen kehittäminen, yhteinen työskentelytila koko ryhmälle ja jatkuvasti paikalla oleva asiakkaan edustaja.

Extreme Programming pohjautuu neljälle arvolle; *kommunikaatio*, *yksinkertaisuus*, *palaute ja rohkeus.* Suurin osa projektin ongelmista johtuu huonosta kommunikaatiosta. Ohjelmoijien välistä kommunikaatiota edistetään pariohjelmoinnilla, päivittäisillä katselmuksilla ja läsnäolevalla asiakkaalla. Asiat tulisi tehdä mahdollisimman yksinkertaisella, toimivalla tavalla. Yksinkertaisuuden periaatetta tulee soveltaa suunnittelun lisäksi vaatimusmäärittelyyn ja projektin hallintaan. Laatu ja joustavuus pohjautuvat yksikkötestauksesta, asiakkailta ja päivittäisistä katselmuksista saatuun palautteeseen. Rohkeutta tarvitaan nopeaan kehittämiseen ja muutoksien tekemiseen projektissa.

Pariohjelmointi, tarinakortit ja testivetoinen kehittäminen ovat käytäntöjä, jotka edesauttavat esiteltyjen arvojen toteutumista projektissa. Pariohjelmoinnilla tarkoitetaan työparia, joista toinen kirjoittaa lopulliseen ohjelmaan päätyvän koodin ja toinen ohjelmoija seuraa vieressä kirjoitettavaa koodia. Ohjelmointipari vaihtaa välillä kirjoittajan ja tarkkailijan roolia. Tarkkailijan tehtävänä on etsiä virheitä kirjoitettavasta koodista ja samalla ajatella koodia osana laajempaa kokonaisuutta.

Testivetoinen kehittäminen on ohjelmointitapa, jossa kirjoitettavalle koodille suunnitellaan yksikkötestaus ennen koodin varsinaista kirjoittamista. Ohjelmoija kirjoittaa uuden testin testausohjelmistoon ja ajaa kaikki testit läpi todetakseen uuden testin virheet. Ajamalla testi ennen kyseisen koodin kirjoittamista auttaa varmistamaan testin toimivuutta. Testien ajamisen jälkeen ohjelmoija kirjoittaa tarvittavan koodin ja testit ajetaan uudelleen. Extreme Programming suosittelee kaiken kirjoitetun koodin yksikkötestausta.

Tarinakortit ovat yksinkertainen keino asiakasvaatimusten kirjaamiseen. Projektin suunnitteluvaiheessa asiakas kirjoittaa paperille lyhyitä kuvauksia järjestelmän keskeisimmistä ominaisuuksista. Kuvassa 16 nähdään tarinakortti, joka kuvaa yhden toteutettavan ominaisuuden ja arvioidun toteutusajan. Extreme Programming-menetelmässä dokumentoinnin määrää kompensoidaan jatkuvasti läsnä olevalla asiakkaan edustajalla. (Larman 2004, Extreme Programming.)

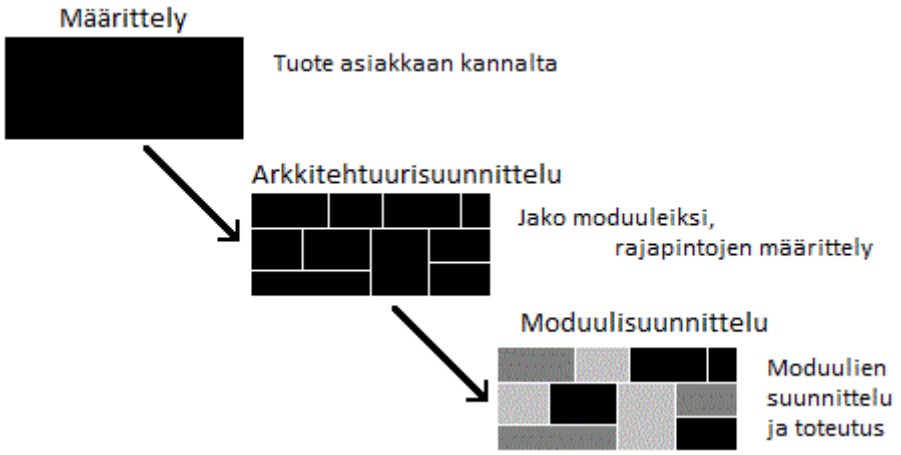


KUVA 16 *Esimerkki Tarinakortista, joka kuvaa yhden ominaisuuden järjestelmästä. Mukaeltu lähteestä Huttunen 2006, 19.*

4 VERKKOSOVELLUKSEN SUUNNITTELU

”Yleensä todetaan, että määrittelyn tulisi vastata kysymykseen *mitä* ja suunnittelun kysymykseen *miten*.” (Haikala & Märijärvi 2006, 62).

Haikalan ja Märijärven (2006) mukaan ohjelmistosuunnittelun voidaan yksinkertaistettuna ajatella menevän kuvan 17 mukaisesti. Määrittelyn valmistuttua näkökulma muuttuu toteutustekniseksi. Järjestelmää jaetaan osiin niin kauan, kunnes osat ovat tarpeeksi pieniä toteutettaviksi ohjelmointikielellä. Jakaminen toteutetaan yleensä kahdessa vaiheessa. Ensimmäiseksi jaetaan ohjelmisto moduuleiksi arkkitehtuurisuunnittelussa. Tämän jälkeen suunnitellaan moduulien sisältö moduulisuunnittelussa. Arkkitehtuurisuunnittelu on tarpeen tehdä huolella, sillä huonot ratkaisut kostautuvat myöhemmin korkeina toteutus- ja ylläpitokustannuksina. (Haikala & Märijärvi 2006, 303–310.)



KUVA 17 *Yksinkertaistettu näkemys ohjelmiston laatimisesta. Mukaeltu lähteestä Haikala & Märijärvi 2006, 303.*

4.1 Tavoitteet ja periaatteet

Ennen kuin suunnittelu voidaan aloittaa, täytyy miettiä, minkälaisia tavoitteita suunnittelulla halutaan saavuttaa. Yleinen tavoite suunnittelussa on saavuttaa asiakkaan haluama lopputuotteen laatu. Laatu kuitenkin on useiden osatekijöiden summa; siihen vaikuttavat käytetyt menetelmät, ohjelmistotalon infrastruktuuri ja sidosryhmät. Laatutavoitteet yleensä sovitaan jo määrittelyvaiheen aikana. (Taina 2009, 1–6.)

Suunnittelussa kannattaa ottaa huomioon, että ohjelmiston suunnittelu ei ole vain tekninen ongelma, vaan se on ohjelmistojen koon kasvaessa yhä suuremmassa määrin myös oppimis- ja kommunikointiprosessi. Esimerkiksi valittujen suunnitteluratkaisujen tulee olla sellaisia, että ne on helppo kommunikoida projektin muille työntekijöille. (Haikala & Märijärvi 2006, 303–310.)

Haikala ja Märijärvi (2006) listaa selkeisiin ja ymmärrettäviin ratkaisuihin johtaviksi periaatteiksi seuraavat: *yksinkertaisuus* ja *suoraviivaisuus*, *osittaminen* ja *lokaalisuus*, *abstraktioiden hyödyntäminen* ja *yhdenmukainen toteutusfilosofia* eli arkkitehtuurityyli. Ohjelmistojen rakenteessa on olennaista monimutkaisuuden hallinta, koska monimutkaisuutta ei voi välttää laajoissa ohjelmistoissa. Ainoa keino monimutkaisten kokonaisuuksien hallintaan on osittaminen osiin eli moduuleihin. Osittamista voi tapahtua kahdella tavalla. Kokonaisuus voidaan jakaa samalla tasolla oleviin osakokonaisuuksiin tai hierarkkisesti osiin.

Lokaalisuudella tarkoitetaan suunnitteluratkaisujen kapselointia mahdollisimman hyvin moduulien sisälle, jolloin muutosten tekeminen lokalisoituu. Onnistuneen lokalisoinnin ansiosta ohjelmiston yksittäisiä osia tai muutaman osan muodostavaa kokonaisuutta on mahdollista toteuttaa ja testata muista osista erillään. Myös moduulien sisäiseen rakenteeseen liittyvät ratkaisut on syytä lokalisoida siten, että ne näkyvät mahdollisimman harvoissa kohdissa.

Onnistunut osittaminen perustuu abstraktioiden käyttöön. Abstraktiolla tarkoitetaan mallia, joka kuvaa esittämästään asiasta oleellisen. Epäoleelliset asiat voidaan jättää pois ja tarpeettomat yksityiskohdat kapseloidaan abstraktion sisälle. Abstraktiosta voidaan käyttää myös nimitystä informaation kätkeminen.

Abstraktiot toteutetaan ohjelmamoduuleina, joista näkyy käyttäjälle vain rajapinta, jonka toteutus on kätketty moduulin sisään. Abstraktioiden hyödyllisyys perustuu juuri tähän. Oikein laaditun abstraktion käyttäjä ei tarvitse tietoa, miten toteutus on laadittu. Erityisen tärkeää suunnittelussa on abstraktoida suunnitelmaratkaisut, joiden muuttumistodennäköisyys on suuri.

Rajapintaa voidaan pitää moduulin käyttäjän ja toteuttajan välisenä sopimuksena, jossa määritellään moduulin tarjoamat palvelut. Rajapintoja määriteltäessä kiinnitetään huomiota moduulin koheesioon, jolla tarkoitetaan loogisesti yhteenkuuluvien osien ryhmittelyä samaan moduuliin sekä moduulien välisiin kytkentöihin. Rajapinnat tulisi määritellä siten, että sisäinen yhteenkuuluvuus olisi mahdollisimman suuri ja moduulien välisten yhteyksien mahdollisimman vähäisiä.

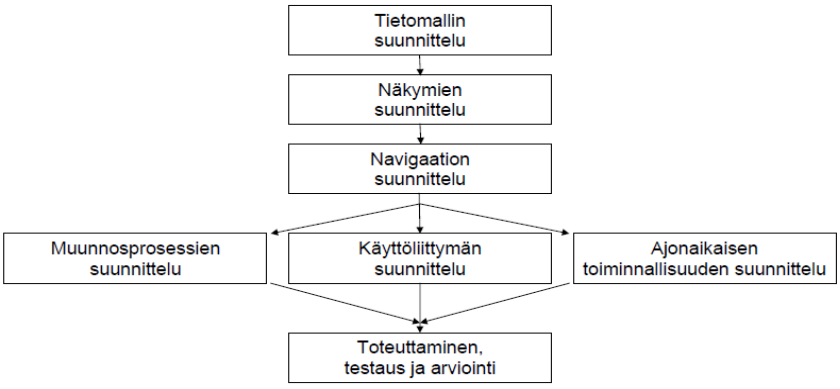
Toteutusfilosofialla tarkoitetaan yhdenmukaisia toteutusperiaatteita, joilla järjestelmän piirteet toteutetaan. Toteutusfilosofia kertoo ne periaatteet ja rakenteet, joiden ajatellaan pysyvän muuttumattomina koko ohjelmiston elinkaaren ajan. Toimiviksi toteutusfilosofiaksi voidaan sanoa esimerkiksi MVC-arkkitehtuuria tai palvelukeskeistä arkkitehtuuria. (Haikala & Märijärvi 2006, 310–316, 320.)

4.2 Käyttäjäkeskeinen suunnittelu

Käyttäjäkeskeisessä suunnittelussa lähtökohtana on käyttäjän tuominen keskeisesti mukaan ohjelmistokehitysprosessiin. Tulevia käyttäjiä haastatellaan, tarkkaillaan heidän työskentelyään ja analysoidaan käyttäjäryhmän tietoja. Käyttäjille näytetään käyttöliittymämalleja tulevasta järjestelmästä ja pyydetään heitä kokeilemaan niitä. Samalla tarkkaillaan käyttäjien tekemiä toimintoja. Tilanteista saadun palautteen, tietojen ja havaintojen perusteella kehitetään parannuksia käyttöliittymään. Ketteriin menetelmiin verrattuna käyttäjäkeskeinen suunnittelu hakee palautteen tulevilta käyttäjiltä eikä asiakkaan yhteyshenkilöltä. Ennen muuta toteuttamista on oltava ensimmäiset käyttöliittymämallit valmiina ja tulevat käyttäjät selvillä. Käyttäjäkeskeinen suunnittelu projektin alussa mahdollistaa käyttöliittymän suunnittelun tekemättä suuria ohjelmistokehitystä rajoittavia päätöksiä. (Juhola 2007.)

4.3 Relation Management Methodology (RMM)

RMM on rakenteisten verkkosovellusten suunnitteluun tarkoitettu malli, joka jakaa suunnitteluprosessin vaiheisiin vesiputousmallin tavoin. Se soveltuu erityisesti datakeskeistä tietoa sisältävän sovelluksen suunnitteluun. Datakeskeisyydellä tarkoitetaan tietoa, joka voidaan luontevasti jakaa palasiin. RMM jakaa suunnittelun seitsemään (Kuva 18) vaiheeseen. (MATHM-37000 Hypermedian perusteet, 2007.)



KUVA 18 *RMM-mallin vaihejako, mukaeltu lähteestä MATHM-37000 Hypermedian perusteet, 2007.*

Ensimmäinen vaihe on tietomallin suunnittelu eli sovelluksen tietoalkioiden ja niiden välisten suhteiden määrittely. Vaihe vastaa pitkälti tietokannan suunnittelua esimerkiksi ER-kaavioilla. Toisessa vaiheessa suunnitellaan sovelluksen näkymät ja kolmannessa vaiheessa navigaatio tietomallissa kuvattujen tietoalkioiden perusteella. Muunnosprosessin suunnittelulla tarkoitetaan toteuttamisen suunnittelua, esimerkiksi miten toteutetaan listat ja taulukot. Käyttöliittymän suunnitteluvaiheessa mietitään komponenttien asettelu käyttöliittymään ja näkymien ulkoasu. Ajonaikaisella toiminnallisuudella tarkoitetaan navigointihistoriaan, käyttäjän seurantaan tai esimerkiksi hakuihin liittyvän toiminnallisuuden suunnittelua. Viimeinen vaihe on toteuttaminen, testaus ja arviointi. (MATHM-37000 Hypermedian perusteet, 2007.)

4.4 Suunnittelun työkalut

Wiki-työkalut, mahdollistavat tekstin, kuvien ja dokumenttien välisten linkkien tekemisten ja helpon julkaisun käyttämällä hyväksi ainoastaan Internet-selainta. Wiki-työkalut ovat raskaampiin työkaluihin verrattuna helppoja ja nopeita. (Larman 2004, Practice Tips.)

Kirjoitustauluilla ja digitaalikameralla tallennetaan malleja luovaa käsinmallintamista korostavassa ketterissä menetelmissä, jossa on käytössä esimerkiksi UML–tyyppinen mallintaminen (Larman 2004, Practice Tips).

Käyttöliittymäprototyypit voidaan piirtää seinälle suurelle paperille, jossa paperi on verkkosivu ja liitettävät muistilaput käytettäviä linkkejä, painikkeita ym. Menetelmää kuvataan nimellä ”GUIs with glue” eli liimatut käyttöliittymät. Käyttöliittymäprototyyppien tekemiseen löytyy myös tietokoneohjelmia, jotka on tarkoitettu verkkosivustojen ”rautalankamallien” tekemiseen. Ohjelmilla voidaan nopeasti luoda tulevan käyttöliittymän suuntaviivat ja mallintaa käyttöliittymän toiminnallisuutta. (Larman 2004, Practice Tips; Wikipedia, Website wireframe.)

CASE-työkalut (Computer Assisted Software Engineering) ovat ohjelmistoja, jotka ovat kehitetty helpottamaan ohjelmistotyön määrittely- ja suunnitteluvaiheita. Yksinkertaisimmillaan nämä työkalut ovat jotain graafista mallinnuskieltä, kuten UML:ää tukevia piirtotyökaluja tai dokumentointivälineitä. Pisimmälle viedyt työkalut sisältävät projektin hallintaan liittyviä ominaisuuksia ja pystyvät riittävän tarkasta kuvauksesta generoimaan koodia. (Haikala & Märijärvi 2006, 83–86.)

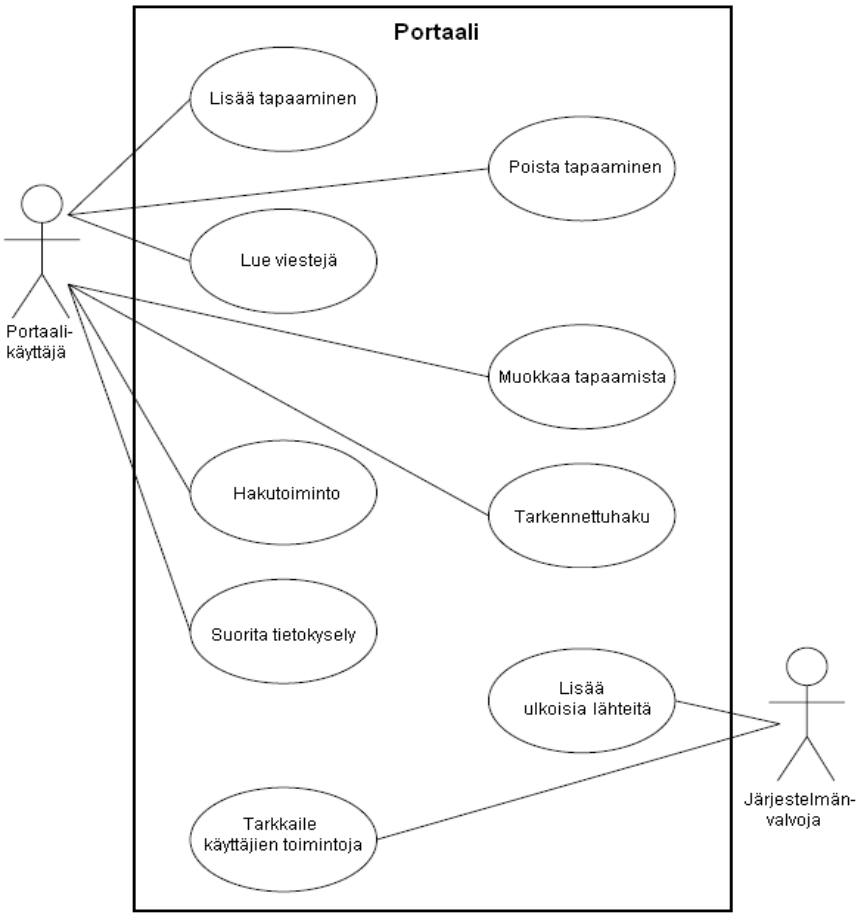
4.5 UML-mallinnuskieli

UML on olioperustainen graafinen mallinnuskieli, joka määrittelee standardoidun notaation kaavioille. UML-mallinnuskieli sisältää 13 erilaista kaaviota, jotka tarjoavat ohjelmistokehitysprojektiin osallistuville yhteisen kielen näkemysten vaihtoon. UML ei sisällä mitään ohjeita, miten kaavioita pitäisi käyttää, joten kielen käyttäjälle on annettu paljon mahdollisuuksia soveltaa kaavioita omien tarpeidensa mukaan. (Vesterholm & Kyppö 2004, 51.)

4.5.1 Käyttötapaukset

Käyttötapauskaaviolla voidaan mallintaa yksinkertaisella tavalla, mitä ohjelmalla voidaan tehdä ja ketkä ohjelmaa käyttävät. Käyttötapauskaavio (Kuva 19) koostuu käyttäjistä, käyttötapauksista ja käyttötapausten välisistä suhteista. Yksi käyttötapaus kuvaa yhtä käyttäjän kannalta mielekästä toimintoa. Käyttötapaukset yleensä laajennetaan sanallisiksi kuvauksiksi kaavion laatimisen jälkeen. Sanallinen kuvaus kertoo lyhyesti ja yksityiskohtaisesti, mitä kyseisessä toiminnossa tapahtuu. (Vesterholm & Kyppö 2004, 51–55.)

Käyttötapauskaaviosta ymmärtää helposti käyttäjälle tarkoitetut toiminnot, kuten lisää tapaaminen, poista tapaaminen tai lue viestiketjua. Toisella puolella on portaalin ylläpitäjälle tarkoitetut toiminnot, lisää ulkoisia lähteitä sekä seuraa käyttäjien toimintoja.



KUVA 19 *Portaalin käyttötapauskaavio. Muokailtu lähteestä Richardson, Avondolio, Vitale, Len & Smith 2004, 183.*

Kuva 20 on sanallinen käyttötapauskuvaus Excel-listan tiedoista. Kuvassa näkyvä käyttötapauskuvaus on tehty Ambientialla käytössä olevalla CA-SE-työkalulla.

UC-11 Make an excel list from the choosed data P1

User can make different kind of excel lists from the system. Lists are

Details	
Parent: Painonnostoliiton use cases	
Primary Actors:	**Supporting Actors:**
Preconditions:	**Success Guarantee:**
Level:	**Complexity:**
Use Case Status:	**Implementation Status:**
Assigned To:	**Release:**

Flow of Events

Main Success Scenario:

1. User choose different options and clicks Make a list -button
2. The system makes an ecxel list of the choosed options

Extensions:

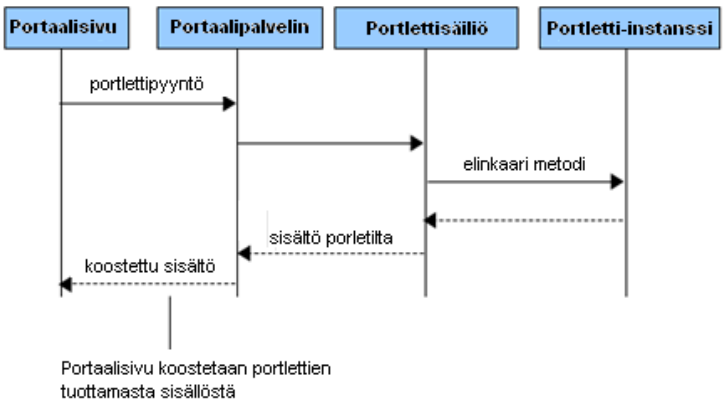
2.a Required fields are empty
    1. The system alerts about the empty fields

KUVA 20 *Sanallinen käyttötapauskuvaus. Painonnostoliiton projekti.*

4.5.2 Sekvenssikaavio

Sekvenssikaavioilla kuvataan erilaisia tapahtumaketjuja ja niissä tapahtuvaa tietojen vaihtoa. Eri osapuolten välistä vuorovaikutusta kuvaavalla kaaviolla voidaan laajentaa käyttötapausta tarkemmaksi. Sekvenssikaavio yleensä laaditaan lisäämään havainnollisuutta hankalasti ymmärrettäviin toimintoihin. (Haikala & Märijärvi 2006, 149–152.)

Sekvenssikaaviolla voidaan esimerkiksi kuvailla tapahtumaketjua käyttäjän sisäänkirjautumisen jälkeen (Kuva 21). Käyttäjä klikkaa portlettia, jolloin portaali välittää pyynnön portlettisäiliölle, joka suorittaa tarvittavan toiminnon oikealla portletilla. Portletin suoritettua tarvittavat toiminnot se palauttaa sisältöpalan ja portaali kokoaa sisältöpaloista vastauksen, joka välitetään käyttäjälle.

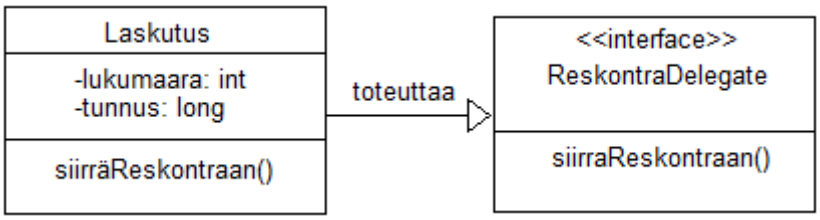


KUVA 21 *Sekvenssikaavio käyttäjän tekemästä pyynnöstä. Mukaeltu lähteestä Sarin 2009, Portlet Container.*

4.5.3 Luokkakaavio

Luokkakaaviolla kuvataan ohjelman keskeisiä käsitteitä ja niiden välisiä suhteita ohjelman eri tasoilla. Luokkakaaviolla kerätään kokonaisuuksia, *luokkia*, jotka sisältävät ominaisuuksia ja toimintoja. Luokkien välisiä suhteita voidaan mallintaa koostumusta ja lukumäärää kuvaavilla merkinnöillä. Luokkakaavio voi olla myös ohjelman niin sanottu domain-kaavio, jolloin se kuvaa yleensä tietokantaan tallennettavia *käsitteitä*, kuten henkilötiedot, osoite, asiakastiedot ym. sekä näiden välisiä suhteita. Domain-kaavio on siis hyvin lähellä tietokantakaaviota, joka kuvaa tallennettavia *tietoja* ja niiden suhteita. Domain-kaavion luokka voi sisältää usean tietokannan luokka voi sisältää usean tietokannan tietueita. (Vesterholm & Kyppö 2004, 52–59.)

Kuvassa 22 on toiminnallisuutta ja luokkien välisiä suhteita kuvaava luokkakaavio. Laskutus-luokka toteuttaa ReskontraDelegate-rajapinnan, jossa on määritelty siirraReskontraan-toiminto, Laskutus-luokka on rajapinnan toteuttava luokka, jolloin sen pitää toteuttaa määritelty toiminto. Laskutus-luokalla on myös ominaisuudet lukumäärä ja tunnus.



KUVA 22 *Toiminnallinen luokkakaavio.*

5 PAINONNOSTOLIITON PROJEKTI

Opinnäytetyön aikana Ambientia toteutti opiskelijaprojektina portlettipohjaisen verkkosovelluksen Painonnostoliitolle. Opinnäytetyössä verrataan kirjallisuuden pohjalta tehtyjä päätelmiä parhaista tavoista ja periaatteista Painonnostoliitolle tehdyn portlettipohjaisen projektin kulkuun ja johtopäätösten perusteella teoriaa hiotaan käytännönläheisemmäksi ja tarkemmaksi.

5.1 Tutkimusmenetelmät

Tässä opinnäytetyössä pyrittiin selittämään tutkittavaa ilmiötä kvalitatiivisellä lähestymistavalla. Työssä aineistoa analysoitiin ankkuroidulla teorialla, joka on teoriapainotteinen kvalitatiivinen menetelmä. Siinä teoria syntyy aineistonkeruun ja jatkuvan analysoinnin kanssa vuorovaikutuksessa. (Hirsjärvi & Hurme 2000, 164, 165; Hirsjärvi, Remes, & Sajavaara 2009, 156–161.)

Kvalitatiiviset menetelmät ovat käyttökelpoisia aineiston ollessa numeerisesti hankalasti mitattavissa. Menetelmän valintaa tuki myös tavoite saada tutkimuksen kohteesta kattava, kokonaisvaltainen kuva ja ymmärtää tutkimuskohteen merkityksiä. Ankkuroitu teoria on käyttökelpoinen tutkimusaineiston olleessa kirjallisia lähteitä ja käytännönosuuden projekti.

5.2 Projektin toteutus

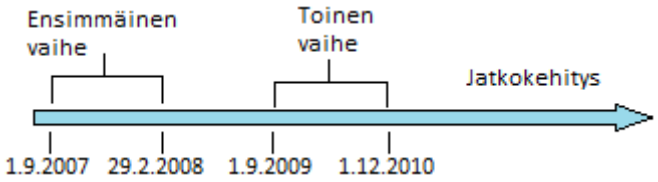
Painonnostoliiton verkkosovellus toteutettiin kahdessa osassa: Ensimmäinen vaihe aloitettiin vuonna 2007, jolloin Ambientia antoi Hämeen ammattikorkeakoulun kolmelle opiskelijalle erikoistumisprojektin aiheen. Tavoitteena oli tehdä Painonnostoliitolle verkkosovellus niin sanottuun ”mock-up” -vaiheeseen asti. Käyttöliittymässä olisi vähintään perusominaisuudet ja valmis ulkoasu. Projektin toisen vaiheen tavoitteena oli rakentaa toiminnallisuus aiemmin suunniteltuun käyttöliittymään. Projektin molemmissa vaiheessa opiskelijat toimivat itsenäisesti Ambientian tarjotessa tukea lähinnä teknisessä toteuttamisessa.

5.2.1 Projektin ensimmäinen vaihe

Projektin ensimmäinen vaihe (Kuva 23) kesti 1.9.2007–29.2.2008 ja vaiheen keskeisimmät tehtävät olivat seuraavat:

- Selvittää Puntti-sovelluksen käyttöönottomahdollisuudet Painonnostoliiton koneiden ulkopuolella ja vanhan Solid-tietokannan siirtomahdollisuudet MySQL:ään.
- Tutkia Suomen Liikunta ja Urheilu Ry:n lisenssitietokannan käyttäminen heidän järjestelmänsä ulkopuolella.
- Laatia toteutettavan verkkosovelluksen käyttötapausdokumentointi ja suunnitella käyttöliittymä tulevaan verkkosovellukseen.

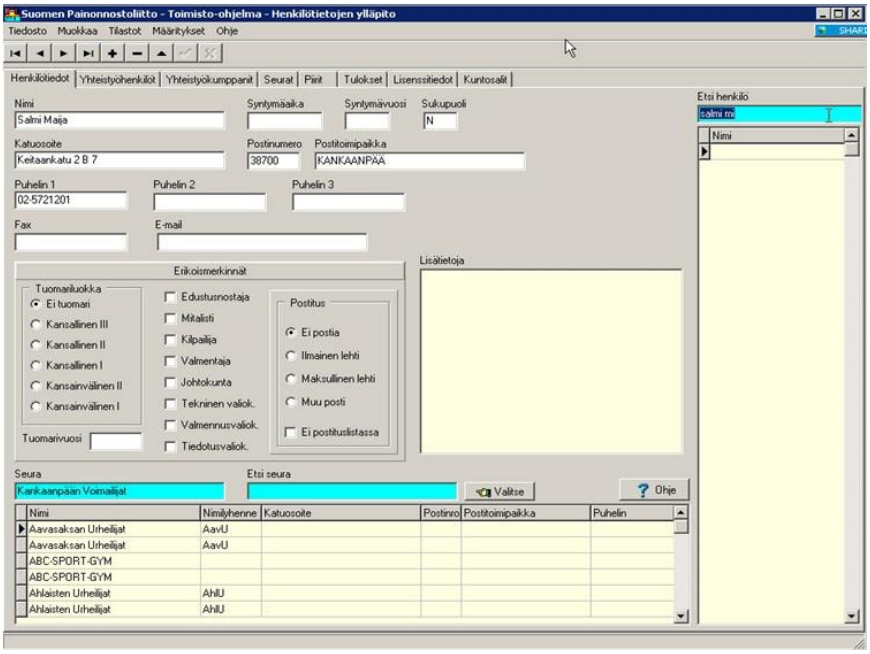
- Perustaa Painonnostoliiton projektille versionhallinta Ambientian versionhallintajärjestelmään.
- Toteuttaa malli kuvaamaan tulevan sovelluksen kokonaisuuden kannalta oleellisimmat näkymät ja rakentaa malliin perusominaisuudet, kuten henkilötietojen hallinta.



KUVA 23 *Painonnostoliiton projektin vaiheet.*

Painonnostoliiton vanhalla Puntti-sovelluksella (Kuva 24) hallittiin kaikkia liiton ylläpitämiä tietoja, muun muassa jäsenten henkilötietoja, lisenssejä ja tuloksia. Sitä käytettiin myös esimerkiksi postituslistojen ja vuosittaisten yhteenvetojen tulostamiseen. Sovellus oli työpöytäohjelma, jota voitiin käyttää ainoastaa yhdeltä tietokoneelta. Sen toiminnallisuudessa oli myös muita puutteita, henkilöä ei voinut poistaa vanhasta järjestelmästä eikä ohjelmasta saanut ulos kuin pdf- tiedostomuodossa olevia dokumentteja. Sovelluksessa oli myös kokonaaan käyttämättä jääneitä toimintoja.

Painonnostoliiton tietojen ylläpidosta vastaa muutama henkilö, joiden käyttöön ohjelma ensisijaisesti suunniteltiin. Käyttö kuitenkin haluttiin mahdollistaa miltä koneelta tahansa. Vanhanaikaisen toimistosovelluksen tilalle toivottiin korvaajaksi verkkosovellusta, jossa vanhan järjestelmän puutteet olisi korjattu.

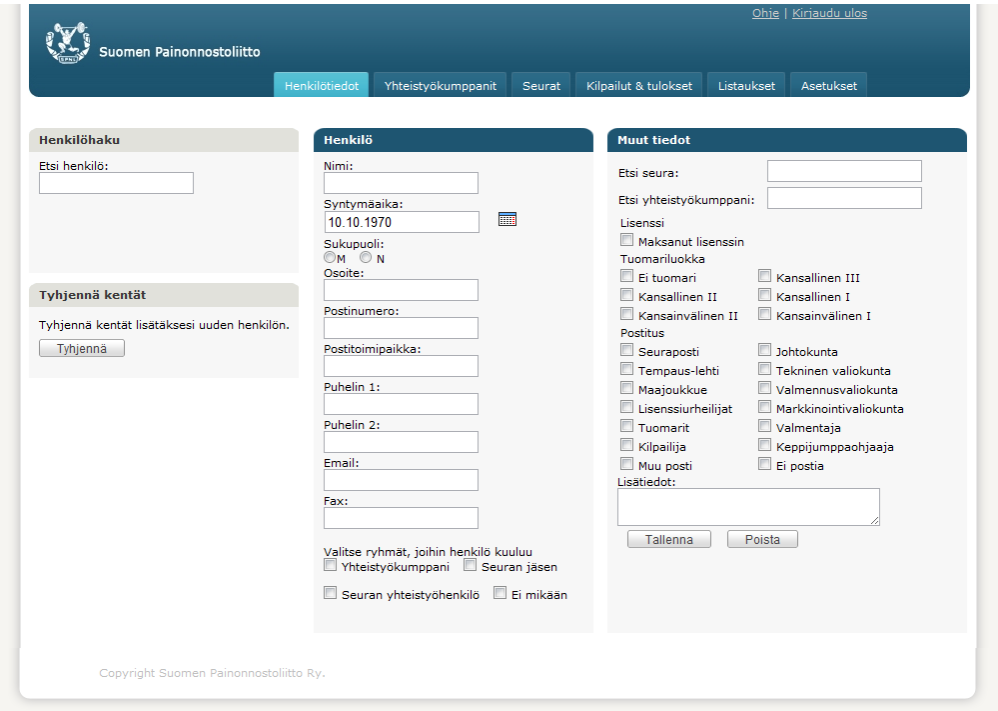


KUVA 24 *Painonnostoliiton vanha toimistosovellus.*

Uutta verkkosovellusta lähdettiin määrittelemään ja suunnittelemaan vanhan toimistosovelluksen perusteella sekä asiakastapaamisissa Painonnostoliiton kanssa. Asiakastapaamisissa kerättiin asiakkaan vaatimuksia uu-

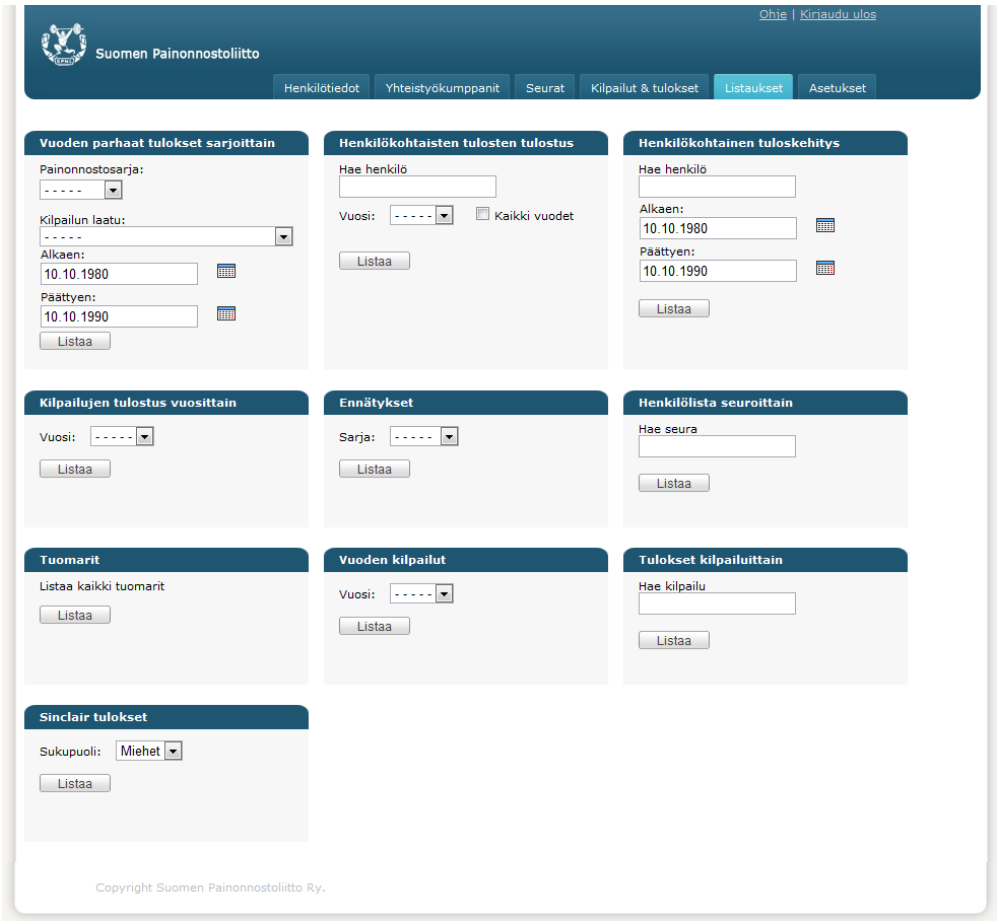
delta järjestelmältä ja saatiin kopioita listauksista, joita vanhalla toimistosovelluksella pystyttiin tulostamaan. Uutta verkkosovellusta lähdettiin laatimaan pitkälti asiakastapaamisissa kerättyjen tietojen ja vanhasta sovelluksesta otettujen kuvankaappausten perusteella. Pian projektin alkamisen jälkeen selvisi, että vanhan toimistosovelluksen siirtäminen käytettäväksi muualla kuin Painonnostoliiton tietokoneilla olisi vaikeaa. Tuleva verkkosovellus oli tarkoitus rakentaa käyttämällä Velocity-sivupohjamoottoria ja siirtää vanha Solid–tietokanta MySQL–tietokantaan.

Ensimmäisen vaiheen yhteydessä ei kuitenkaan laadittu tulevaa verkkosovellusta ”mock-up” -vaiheeseen asti epärealistisen aikataulun takia, vaan päädyttiin toteuttamaan vain tulevalle verkkosovellukselle käyttöliittymä ulkoasulla (Kuva 25), joka mukailee Ambientian aikaisemmin Painonnostoliitolle toteuttamaa ulkoasumallia. Ensimmäisen vaiheen aikana laadittiin myös käyttötapaukset ja käyttötapausten sanalliset kuvaukset, joiden perusteella tulevaa järjestelmää oli tarkoitus lähteä toteuttamaan.



KUVA 25 *Uuden verkkosovelluksen käyttöliittymä, Henkilötiedot-sivu avoinna.*

Tavoitteena oli, että uudella verkkosovelluksella Painonnostoliiton tietoja voisi hallinta miltä tahansa Internet-yhteydellä varustetulta tietokoneelta. Uuden sovelluksen olisi pystyttävä kaikkiin vanhalla toimistosovelluksella tehtäviin ylläpitotoimintoihin, kuten henkilötietojen, yhteistyökumppanien, seurojen sekä tulosten hallintaan ja tietojen listaamiseen (Kuva 26) Excel-tiedostomuotoon. Uudesta verkkosovelluksesta jätettäisiin pois vain vanhassa järjestelmässä käyttämättä jääneet toiminnot.

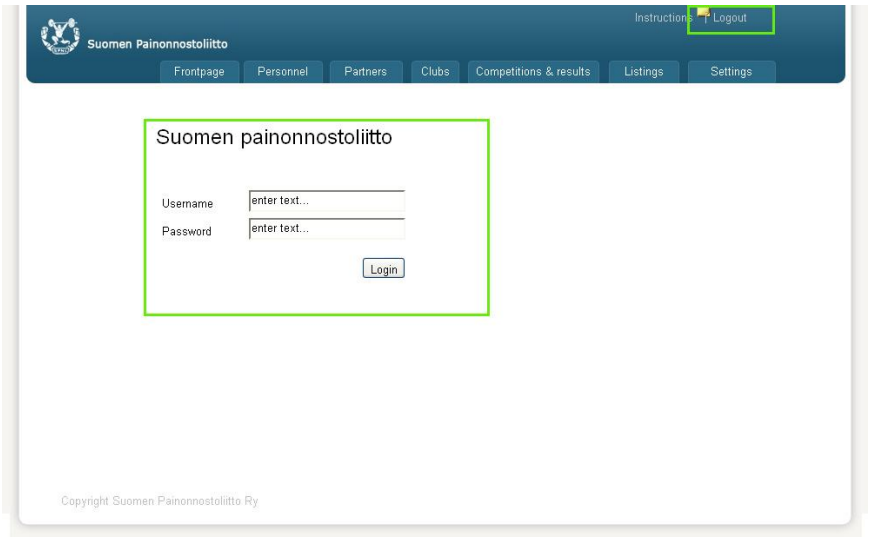


KUVA 26 *Uuden verkkosovelluksen käyttöliittymä, Listaukset-sivu avoinna.*

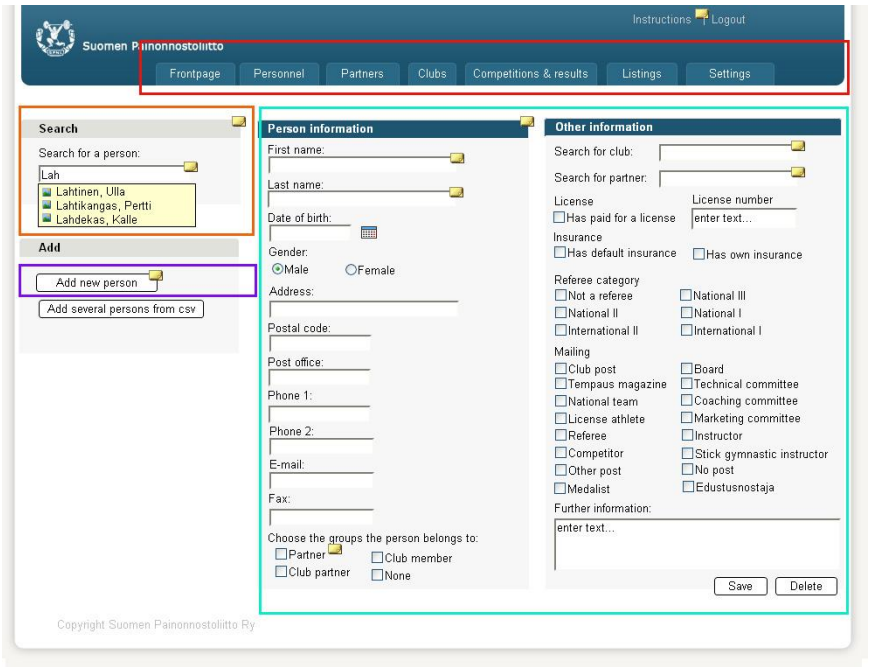
5.2.2 Muutoksia välivaiheessa

Ensimmäisen ja toisen vaiheen välissä Ambientia päätti vaihtaa toteutustekniikan portlettipohjaiseksi, samalla dokumentoinnin työvälineeksi otettiin Ambientialla käytössä oleva CASE-työkalu ja dokumentoinnin kieli muutettiin englanniksi. Verkkosovellukseen haluttiin lisäksi Excel-tiedostojen lukutoiminto, jolloin kilpailujen tuloksia voitaisiin lukea uuteen järjestelmään suoraan tiedostosta syöttämättä niitä yksi kerrallaan järjestelmään.

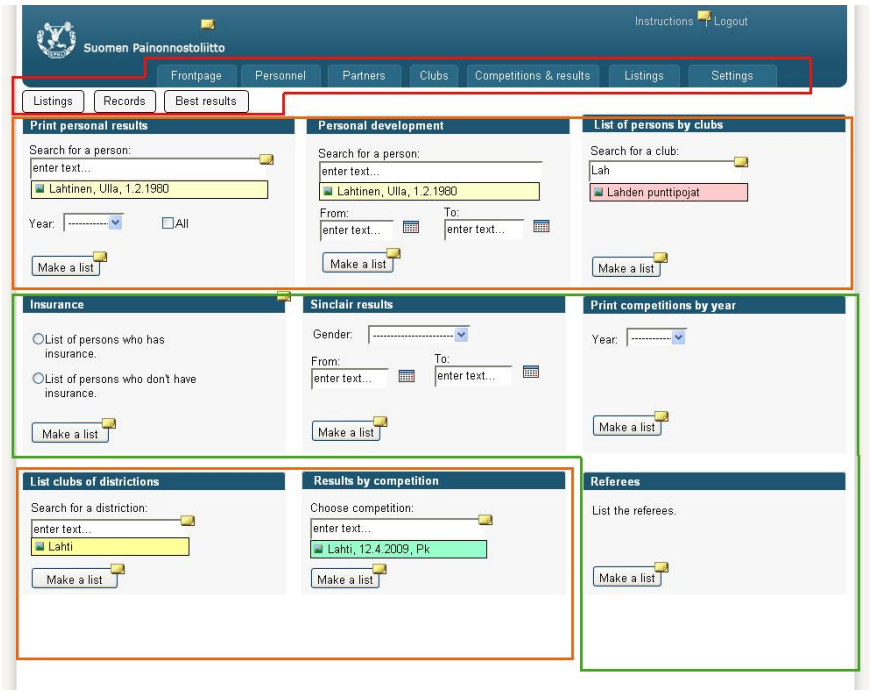
Aikaisemmin tehty käyttöliittymä jaettiin portleteiksi, jotka ajateltiin toteutettavan toisessa vaiheessa. Ensimmäisessä vaiheessa toteutettu käyttöliittymämalli muutettiin ”rautalankamalliksi”, johon voitiin liittää toiminnallisuudesta kertovia viestejä. Käyttöliittymä jaettiin portaaleiksi siten, että malleista kaapattuihin kuviin merkattiin ne alueet käyttöliittymästä, joiden ajateltiin edustavan yhtä portlettia. Kuvissa 27, 28 ja 29 on kuvankaappaukset rautalankamallista, Etusivu, Henkilötiedot-sivu ja Listaukset-sivu, joissa suunnitellut portletit näkyvät erivärisiksi merkittyinä alueina. Yksi väri edustaa yhtä ajateltua portlettia ja pienet keltaiset merkit avaavat toiminnallisuudesta kertovan kuvauksen.



KUVA 27 *Painonnostoliiton uuden verkkosovelluksen Frontpage-sivu, portleteiksi jakaminen.*



KUVA 28 *Painonnostoliiton uuden verkkosovelluksen Personel-sivu, portleteiksi jakaminen.*



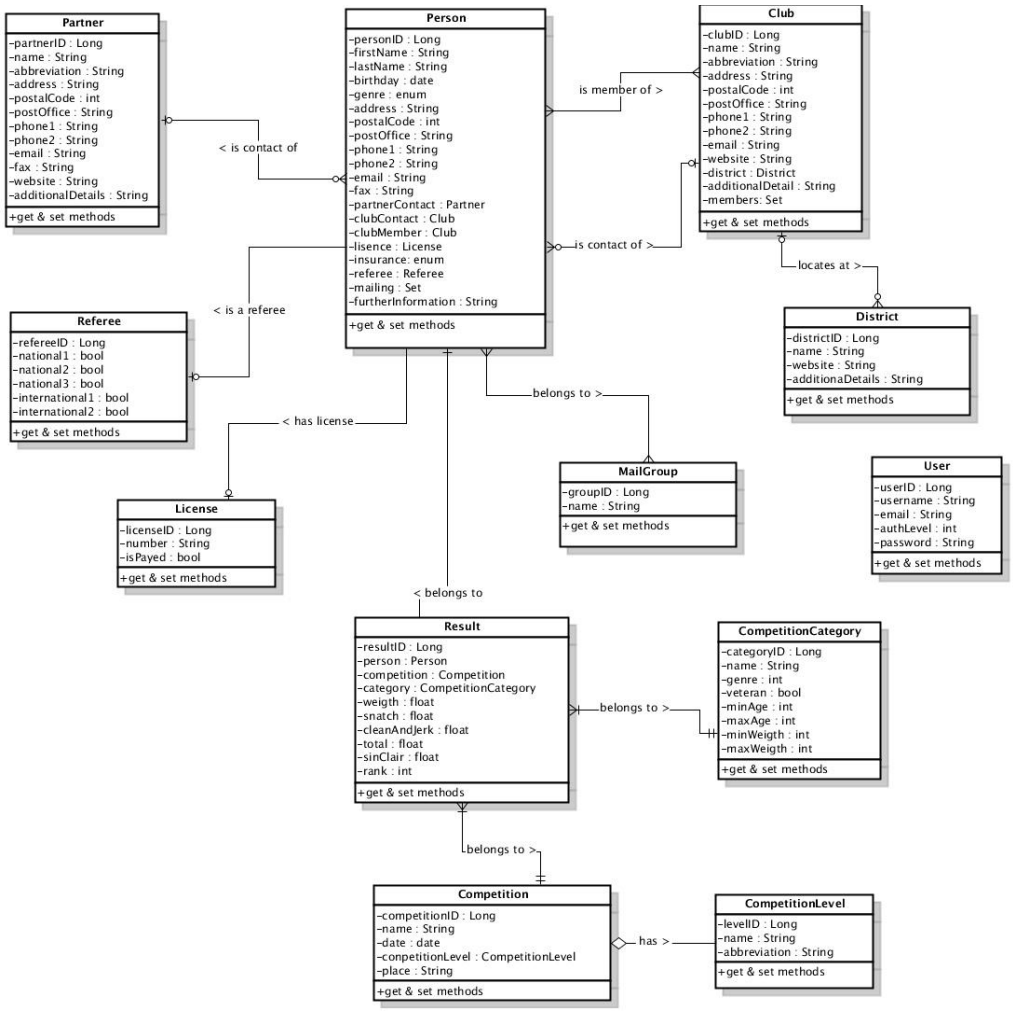
KUVA 29 *Painonnostoliiton uuden verkkosovelluksen Listing-sivu, portleteiksi jakaminen.*

5.2.3 Projektin toinen vaihe

Ambientia etsi Painonnostoliiton projekin toiseen vaiheeseen eli toteutukseen jatkajaa Hämeen ammattikorkeakoululta syksyllä 2009. Projektissa aloitti kaksi opiskelijaa, joista toinen teki erikoistumisprojektin ja toinen opinnäytetyönsä. Kummallakaan opiskelijalla ei ollut aikaisempaa kokemusta porlettipohjaisista verkkosovelluksista ja vain vähän kokemusta Java-ohjelmoinnista.

Projektin toisen vaiheen tavoitteena oli rakentaa toiminnallisuus aiemmin suunniteltuun käyttöliittymään. Toisen vaiheen kestolle ei annettu aikarajaa, vaan työtä oli tarkoitus jatkaa niin pitkälle, kuin erikoistumisprojektin ja opinnäytetyön puitteissa ehditään.

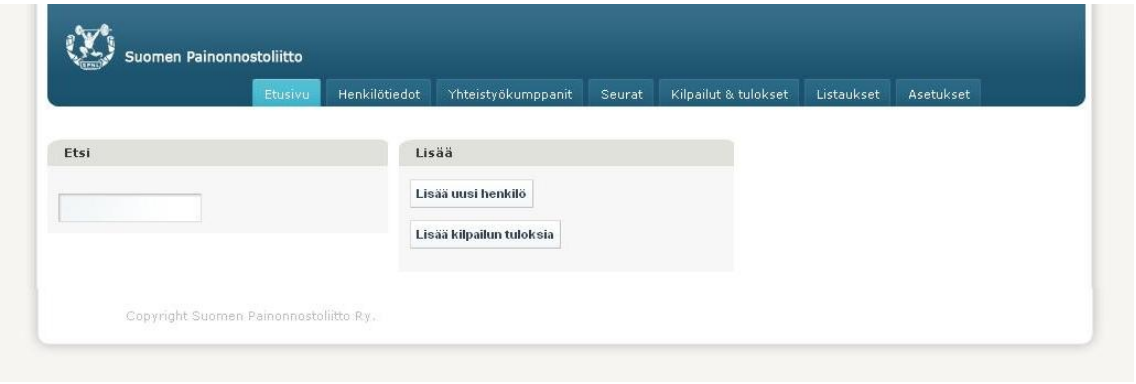
Projekti aloitettiin suunnittelematta toteutuksen kulkua ja siinä käytettäviä menetelmiä. Käytössä oli dokumentointiin tarkoitettu wiki-työkalu ja tehtävienhallintaohjelmisto projektin koordinointia varten. Käyttöliittymämallin perusteella laadittiin luokkakaavio (Kuva 30) tallennettavista tiedoista. Mallia käytiin läpi kohta kohdalta miettien, mitä tietoja sivuilla on ja mitä tietoja pitää tallentaa tietokantaan. Esimerkiksi kuvan 28 Henkilötiedot-sivulta kerättiin luokkakaavion *person*-luokan tietoja, kuten etunimi, sukunimi ja syntymäaika. Verkkosovelluksen alustana käytettyyn Liferay-portaaliohjelmistoon tehtiin tarvittavat portletit ja käyttöliittymä rakennettiin vastaamaan ensimmäisessä vaiheessa laadittua mallia. Portlettien rakentamisessa käytettiin Hibernate- ja Spring-tekniikoita.



KUVA 30 *Vapaamuotoinen Painonnostoliiton verkkosovelluksen tietokanta- ja luokkakaavio.*

Ensimmäisiksi toteutettaviksi portleteiksi valittiin ne, joilla hallittiin mahdollisimman itsenäisiä tietoja. Sellaisia olivat piirien ja yhteystietojen hallintaan tarkoitetut. Ne rakennettiin helpoimmista kohti monimutkaisempia, monia tietoja hallitsevia portletteja, kuten Henkilötiedot-porletti ja Tulokset-portletti. Kaikkien luokkakaaviossa näkyvien luokkien tietojen hallintaan ei tehty niitä, sillä osalla portleteista hallittiin useampaa kuin yhtä luokkaa. Osalla niistä toteutettiin vain toiminnallisuuksia, kuten hakutoiminto ja listaukset. Portletteja rakennettiin lopulta 23 kappaletta toteuttamaan tarvittavat toiminnallisuudet käyttöliittymään.

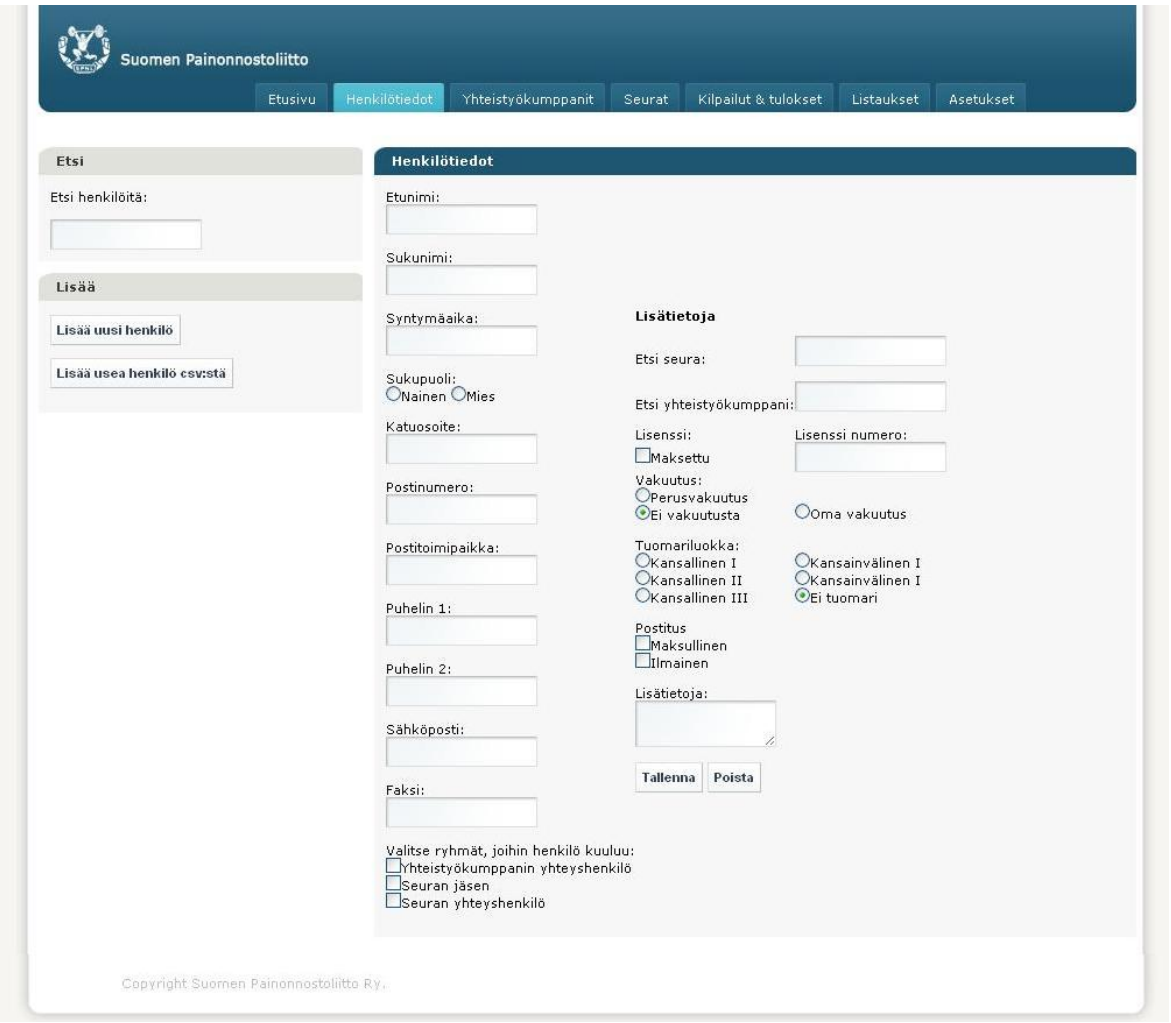
Kuvassa 31 on portaalin Etusivu, jossa on kaksi portlettia. Etusivu aukeaa käyttäjälle kirjautumisen jälkeen. Etsi-portletilla voidaan hakea tietoa kaikista liiton tiedoista. Lisää on sivun toinen portletti. Siinä on kaksi painiketta, toinen avaa Henkilötiedot-sivun ja toinen Kilpailut & tulokset-sivun. Yläosan navigaatiopalkki on osa portaalia eikä sen toteuttamiseen tarvittu portlettia.



KUVA 31 *Uuden verkkosovelluksen Etusivu, muutamia muotoiluja vailla.*

Kuva 32 on portaalin Henkilötiedot-sivu liittoon kuuluvien henkilöiden tietojen hallintaan. Sivulla on kolme portlettia; Etsi-portletti, jolla voidaan hakea henkilöä. Henkilötiedot-sivun ollessa auki Etsi-portletti hakee ainoastaan liiton henkilötiedoista. Toinen portletti on Henkilötiedot-portletti, joka näyttää haetun henkilön tiedot. Se mahdollistaa vanhojen henkilötietojen muokkaamisen ja uusien lisäämisen. Kolmas sivulla oleva portletti on kaksinappinen Lisää-porletti, joista toisella voidaan tyhjentää Henkilötiedot-portletin kentät ja toinen ohjaa käyttäjän Excel-tiedoston lataamissivulle, jossa voidaan lisätä monia henkilöitä kerralla järjestelmään.

Toisen vaiheen ensimmäinen palaveri Painonnostoliiton edustajien kanssa pidettiin projektin puolivälissä. Palaverissa käytiin läpi toteutusta ja haettiin vastausta toteutuksessa ilmenneisiin epäselvyyksiin. Selvisi, että tiedostojen lukeminen on yksi tärkeimmistä toiminnallisuuksista, joita uuteen ohjelmaan haluttiin. Kilpailun järjestäjille ympäri Suomea haluttiin käyttöön Excel-tiedosto, joka voitaisiin viedä järjestelmään tulosten täyttämisen jälkeen. Tiedostojen tuominen järjestelmään on vaikeimpia toteutuksen osia, koska uudet tiedot vaikuttavat kaikkiin muihin tietoihin. Ennen kuin uusi tulos voidaan laittaa järjestelmään, pitää esimerkiksi varmistaa, että henkilön tiedot löytyvät järjestelmästä.



KUVA 32 *Uuden verkkosovelluksen Henkilötiedot-sivu, muutamia muotoiluja vailla.*

Opinnäytetyön valmistumisen aikaan Painonnostoliiton verkkosovellus on muutamaa toiminnallisuutta vaille valmiina julkaisuun Painonnostoliitolle ensimmäisien palautteiden saamiseksi. Puuttuvia toimintoja ovat tiedoston lukeminen järjestelmään ja listaukset tietojen viemiseksi tiedostoon. Sovelluksen toimivuus on vielä testattava ennen julkaisua.

5.3 Projektin tarkastelu

Painonnostoliiton uutta verkkosovellusta lähdettiin rakentamaan vanhan sovelluksen puutteiden perusteella. Yhteyttä asiakkaaseen pidettiin harvoin Painonnostoliiton huonosti tavoitettavien yhteyshenkilöiden vuoksi. Ehkä yhteistyötä hidasti myös opiskelijoiden vaihtuminen projektin ensimmäisen ja toisen vaiheen välillä. Vasta projektin puolivälissä tiedettiin tarkkaan, mitä Painonnostoliitto halusi. Liiton kannalta tärkeää asiaa, tulosten ja muiden tietojen lukemista suoraan järjestelmään ei oltu suunniteltu projektin ensimmäisessä vaiheessa.

5.3.1 Prosessimallin valinta

Prosessimalleilla on tarkoitus hallita monimutkaista ohjelmistokehitysprosessia. Sen valinnalla on suuri vaikutus lopputulokseen. Toisaalta mikään malli ei auta, jos tehtävä on epäselvä. Prosessininhallinta oli tekijöille uutta eikä heillä ollut riittäviä tietoja ketteristä menetelmistä. Sen takia projekti olisi tarvinnut enemmän Ambientian henkilöstön ohjausta. Ketterissä menetelmissä pyritään nopealla reagoinnilla hallitsemaan projektin aikana esiintyvää epävarmuutta. Menetelmissä korostuvat osaava henkilöstö, sujuva vuorovaikutus, yhteistyö asiakkaan kanssa, yksinkertaiset käytännöt sekä palaute. (Larman 2004, Agile.) Jos ketteriä menetelmiä olisi käytetty edes osittain, olisi asiakkaan toiveet huomioitu paremmin ja projektin kulku olisi ollut hallitumpi.

Ketteristä menetelmistä Scrum sisältää projektin hallintaan liittyviä käytäntöjä ja arvoja ottamatta kantaa projektin tuotoksiin. Yksi menetelmän piirteistä on projektin jakaminen sprinteiksi kutsuttuihin 30 päivän aikalaatikoituihin iteraatioihin, joita ennen on valmisteluvaihe ja jälkeen projektipalaveri. Päivittäin pidetään seurantapalaverit ja projektiryhmä päättää sopivasta määrästä dokumentointia. (Larman 2004, Scrum.)

Painonnostoliiton projektin prosessimalliksi olisi sopinut Scrum-menetelmä yhdessä muutaman Unified Processista löytyvän tuotoksen kanssa. Projektiryhmän pienuuden takia projektissa ei ollut tarvetta laajalle dokumentoinnille, mutta säännöllisellä seurannalla ja palavereilla olisi projekti pysynyt paremmin hallinnassa. Sprinttiin olisi voitu valita muutama rakennettava portletti, valmisteluvaiheessa konsultoida Ambientiaa portlettien parhaasta toteuttamistavasta ja sprintin jälkeen toimitettu senhetkinen versio Painonnostoliitolle palautteen saamiseksi.

Extreme Programming on ketterä menetelmä, joka tavoittelee nopeaa reagointia muutokseen ja asiakastyytyväisyyttä. Siinä suositellaan paljon erilaisia työskentelytapoja, kuten jatkuvasti paikalla oleva asiakkaan edustaja ja pariohjelmointi. (Larman 2004, Extreme Programming.) Menetelmä edellyttää tiivistä yhteydenpitoa asiakkaaseen, mikä ei projektissa ollut mahdollista ja menetelmän ohjelmointikäytännöt olisivat olleet hidasteena toteutuksen ollessa samanaikaisesti toteutustekniikan opiskelua.

Unified Process on vahvasti laatua painottava prosessimalli, joka sisältää paljon testaamista ja laadunvarmistamista. Usein se toimii perustana yritykselle luotavassa prosessimallissa. Menetelmä on skaalattavissa pienistä suuriin projekteihin ja se tarjoaa projektin käyttöön paljon vapaasti valittavia tuotoksia. Projektin alussa voidaan tehdä Visio-dokumentti, josta selviää projektin päämäärät, tärkeimmät toiminnot ja sidosryhmien tarpeet. Dokumentti antaa projektille lähtökohdan ja ylläpitää oikeaa suuntaa koko projektin elinkaaren ajan (Larman 2004, Unified Process.)

Painonnostoliiton projektissa Unified Process-mallin käyttö kokonaisuudessaan olisi ollut kovin raskasta, mutta joltakin osalta sitä olisi voinut hyödyntää. Painonnostoliiton uuden ohjelman tärkein ominaisuus oli tiedostojen lukeminen järjestelmään, jolloin asiakkaan tarpeena oli käsin tehtävän työn vähentäminen. Visio-dokumentilla olisi asiakkaan tarpeet olisivat tulleet alussa selville ja projekti olisi voitu heti suunnata tähän päämäärään.

Ketterät menetelmät suosittelevat riskiohjautuvan kehittämisen ja asiakasohjautuvan kehittämisen yhdistämistä. Ensimmäisessä tapauksessa vaikeimmin toteutettavat, riskialttiit toiminnallisuudet rakennetaan ensin. Toisessa tapauksessa asiakkaan vaatimukset tuotteelle otetaan huomioon toteuttamalla asiakkaalle tärkeimmät ominaisuudet ensimmäiseksi. (Larman 2004, Risk-Driven and Client-Driven Iterative Planning.) Painonnostoliiton projektissa toteutettiin helpoimmat toiminnallisuudet ensimmäisenä, koska toteuttaminen vaati samanaikaisesti opiskelua. Projektin työntekijöille selvisi vasta toisen vaiheen puolivälissä Painonnostoliiton kannalta tärkeimmät ja samalla myös riskialtteimmat toiminnot. Ne olisivat todennäköisesti tulleet esille aikaisemmin, jos asiakkaaseen olisi pidetty yhteyttä useammin tai käytetty Unified Processin riskilistaa.

5.3.2 Toteutustekniikan valinta

Painonnostoliiton projektin ensimmäisen ja toisen vaiheen välissä päätettiin toteutustekniikka vaihtaa portlettipohjaiseksi. Verkkosovelluksessa tietoa ei kuitenkaan tarvinnut koota monesta eri lähteestä, joten portaalin tärkein ominaisuus jäi hyödyntämättä. Muitakin portaalin ja portlettien ominaisuuksia, kuten esimerkiksi mahdollisuus kertakirjautumiseen, käyttäjäryhmien muodostamiseen ja muunneltavuuteen jäi käyttämättä. Hyötyjen sijaan toteutustekniikan vaihtaminen sitoi toteutuksen portaalin ja portlettien aiheuttamiin rajoituksiin. Eri toteutustekniikoilla on omat tekniset rajoituksensa, minkä vuoksi toteutustekniikka pitää päättää ennen käyttöliittymäsuunnittelua.

Laaja verkkosovellus voidaan toteutetaa portlettipohjaisena jakamalla tarvittavat toiminnot monelle portletille ja sijoittamalla nämä portaalisivuille, kuten Painonnostoliiton projektin kohdalla tehtiin. Portlettipohjaista käyttöliittymää suunniteltaessa pitää kuitenkin ottaa huomioon tekniikan rajoitukset. Portletti-ikkunaa ei voi jakaa kahteen erilliseen osaan, sen täytyy olla suorakaiteen muotoinen ja sen koko määräytyy portaaliohjelmiston mukaan. Portlettien mukanaan tuomat rajoitukset eivät vaikuta Painonnostoliiton uuden verkkosovelluksen käytettävyyteen.

Toimiva ohjelmisto on onnistumisen ensisijainen mittari (Agile Manifesto 2001). Projektissa valmistuu Painonnostoliitolle verkkosovellus, jolla liiton tietoja voidaan hallita, mutta lopputuote olisi ollut vielä laadukkaampi ja valmistunut aikaisemmin, jos Painonnostoliitossa oltaisiin alusta asti tiedetty, mitä sovellukselta halutaan ja ensimmäisessä vaiheessa selvitetty, mitä asiakas haluaa.

5.4 Tulosten luotettavuus ja käyttöarvo

Tämän opinnäytetyön luotettavuutta voidaan arvioida kirjallisten lähteiden ja niiden pohjalta luodun teorian perusteella. Lähteitä on riittävästi ja ne ovat ylittäneet korkean julkaisukynnyksen. Opinnäytetyön aineisto on seulottu useista alan julkaisuista, joten aineiston pohjalta luodun teorian luotettavuus on monen lähteen varassa. Teoriaa voidaan arvioida vertaamalla sen toimivuutta toteutettuun portlettipohjaiseen projektiin. Vertailusta saadut johtopäätökset ovat linjassa teorian kanssa. Vertailun johtopäätöksissä on pyritty kriittisyyteen ja johtopäätöksissä on viitattu lähteisiin. Prosessimalleissa esitellyt periaatteet ja toimintatavat on yleisesti todettu toimiviksi ohjelmistoprojekteissa. Tässä työssä ei verrattu prosessimallien toimivuutta eri projektissa, mutta perustellusti voidaan olettaa niiden käytön auttavan ohjelmistokehitystä.

Opinnäytetyön hyödynnettävyyttä voidaan miettiä kaikkien työhön liittyvien tahojen kannalta. Painonnostoliitolle opinnäytetyön aikana valmistuu uusi verkkosovellus tietojen hallintaa. Vanhaan ohjelmaan verrattuna tietojen käsittely helpottuu ja nopeutuu, kun ohjelma saadaan otettua käyttöön. Työelämän ja Ambientian kannalta opinnäytetyötä voidaan soveltaa portlettipohjaisten verkkosovellusten suunnittelun perusohjeena ja antamaan näkemystä opiskelijaprojektien toteuttamisessa. Opinnäytetyön tekijä voi soveltaa työn tuloksia valmistumisensa jälkeen tulevassa työelämässä.

6 YHTEENVETO

Opinnäytetyöllä oli kaksi tavoitetta; selvittää kirjallisuuden pohjalta portlettitekniikan käsitteet, menetelmät ja tekniikan tuomat rajoitteet verkkosovellusten suunnittelussa. Toisena tavoitteena oli kehittää portlettipohjaisen verkkosovelluksen suunnitteluprosessia ensimmäisen tavoitteen tulosten perusteella. Työssä käytettiin kvalitatiivista tutkimusmenetelmää, jossa teoriaa luodaan aineistolähtöisesti. Ensimmäistä tavoitetta lähdettiin saavuttamaan keräämällä aineistoa portlettiaiheisesta kirjallisuudesta. Ensimmäisen tavoitteen tuloksia verrattiin Painonnostoliitolle tehdyn portlettipohjaisen projektin kulkuun ja johtopäätösten perusteella teoriaa hiottiin käytännönläheisemmäksi ja tarkemmaksi.

Tuloksina esitetään seuraavat portlettitekniikan tärkeimmät käsitteet, menetelmät ja tekniikan tuomat rajoitteet verkkosovellusten suunnittelussa. Portlettispesifikaatio on määrittely, joka helpottaa portaalikehittämistä. Jos valmistajan portaali tukee spesifikaatiota, voidaan siihen liittää sen mukaan rakennettuja portletteja. Portlettispesifikaatio edesauttaa koodin uudelleenkäyttöä ja siten nopeuttaa kehittämistä. Portletit kuitenkin tuovat mukanaan rajoitteita, johon erityisesti käyttöliittymäsuunnittelun täytyy mukautua.

Portaalin käyttöliittymän suunnittelussa pitää portletit jaotella sisällöllisesti loogisiin kokonaisuuksiin. Portaalin sisältö voidaan koota monesta eri lähteestä suorakaiteen muotoiseen portletti-ikkunaan, jonka koko ja sijainti määritellään portaaliohjelmistolla. Käyttöliittymää suunniteltaessa käyttöliittymään tulisi merkitä, mistä tietolähteestä kyseinen tieto tulee. Portlettien välinen kommunikaatio tulisi pitää mahdollisimman pienenä. Jokainen portletti tulisi ajatella omana kokonaisuutetaan, jolla on vain vähän yhteyksiä muualle. Portlettien välinen tiedonsiirto on mahdollista, mutta tietomäärien täytyy olla pieniä, koska portlettitekniikka ei mahdollista suurien tietomäärien tehokasta siirtämistä portletista toiseen.

Portaali on sisällön kokoamisen väline. Jos tiedon kokoamiseen ei ole tarvetta, on portaalialustalle rakentaminen enemmän rasite kuin apuväline. Portaalit tarjoavat jo olemassa olevien sovellusten eniten käytettyjä toimintoja portaalissa, jolloin niiden käyttö helpottuu. Sovelluksen kaikkia toimintoja on mahdotonta tuoda portlettiin.

Painonnostoliiton projektissa kokonainen verkkosovellus toteutettiin portleteilla. Portlettipohjaisuus ei ollut paras tekniikka tähän projektiin, koska tietoja ei tuotu monista eri lähteistä. Projektissa havaittiin, että toteuttamistekniikka on valittava ennen käyttöliittymäsuunnittelua, sillä toteutustekniikka vaikuttaa käyttöliittymän suunnitteluratkaisuihin.

Toisen tavoitteeseen haettiin lähtökohta ketteristä menetelmistä ja tutustumalla suunnittelun menetelmiin. Portlettipohjaisten verkkosovellusten suunnitteluun sopisi Relation Management Methodology, jossa suunnittelu jaetaan vaiheisiin. Ensimmäisenä vaiheena on tietomallin eli sovelluksen käsittelemän tiedon ja tiedon välisten suhteiden määrittely. Toisessa vai-

heessa tehdään jako portletteja vastaaviksi kokonaisuuksiksi ja seuraavaksi suunnitellaan käyttöliittymän näkymät. Menetelmä helpottaa portlettipohjaisuudesta johtuvien rajoitusten huomioimista projektissa.

Tässä opinnäytetyössä ei vertailtu prosessimallien toimivuutta projekteissa. Projektin perusteella voidaan kuitenkin todeta prosessimallien auttavan projektin koosta riippumatta lähtökohtien selvittämisessä ja asiakkaan tarpeiden huomioimisessa.

LÄHTEET

Agile Manifesto. 2001. Manifesto for Agile Software Development. Viitattu 23.9.2010. http://agilemanifesto.org/

Haikala, I., & Märijärvi, J. 2006. Ohjelmistotuotanto. Talentum Media Oy

Hepper, S. 2008. JSR-286: Java™ Portlet Specification Version 2.0. Sun Microsystems, Inc

Hirsjärvi, S., & Hurme, H. 2000. Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö. Helsinki: Tammi

Hirsjärvi, S., Remes, P., & Sajavaara, P. 2009. Tutki ja kirjoita. Helsinki: Tammi

Huttunen, J. 2006. Ketterän ohjelmistokehitysmenetelmän määrittely, vertailu ja käyttäjä-kysely. Helsingin Teknillinen Korkeakoulu. Diplomityö. http://lib.tkk.fi/Dipl/2007/urn007665.pdf

iGoogle. 2010. Etusivu. Viitattu 7.11.2010. http://www.google.fi/ig?hl=fi

Juhola, T. 2007. Käyttäjäkeskeinen suunnittelu ja ketteryys. Viitattu 7.11.2010. http://www.pcuf.fi/sytyke/lehti/kirj/st20074/ST074-14A.pdf

Larman, C. 2004. Agile & Iterative Development: A Manager's Guide. Person Education, Inc

Linwood, J., & Minter, D. 2004. Building Portals with the Java Portlet API. Apress

MATHM-37000 Hypermedian perusteet. 2007. syksy. Viitattu 12.11.2010. http://matriisi.ee.tut.fi/hmopetus/hmp/2007/pruju/hp2007-173-188.pdf

MOT 6.0. MOT® Dictionaries. MOT 6.0 Professional - 4.10.2005. Kielikone Ltd.

Richardson, C., Avondolio, D., Vitale, J., Len, P., & Smith, K. 2004. Professional Portal Development with Open Source Tools: JavaTM Portlet API, Lucene, James, Slide. Wiley Publishing, Inc

Sarin , A. 2009. Portlets in Action (MEAP). Manning Publications

Taina, J. 2009. Ohjelmistoprosessit ja ohjelmistojen laatu. Kurssi kalvot, Kevät 2009. Helsingin Yliopisto. http://www.cs.helsinki.fi/u/taina/opol/k-2009/

Vesterholm, M., & Kyppö, J. 2004. Java-ohjelmointi. Talentum Media Oy

Wikipedia. Website wireframe. Viitattu 4.11.2010.
http://en.wikipedia.org/wiki/Website_wireframe

LIITE 1

KETTERÄ MANIFESTI

Me etsimme parempia keinoja ohjelmistojen kehittämiseen itse ja auttamalla siinä muita.
Tässä työssämme olemme päätyneet arvostamaan

Yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja
Toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluita
Muutokseen reagoimista enemmän kuin suunnitelman noudattamista.

Vaikka oikeallakin puolella on arvoa,
me arvostamme vasemmalla olevia asioita enemmän.

Ketterän manifestin periaatteet:

1. Korkein prioriteettimme on tyydyttää asiakkaan tarpeet aikaisella ja jatkuvalla hyödyllisen ohjelmiston toimittamisella.
2. Vastaanotamme avosylin muuttuvat vaatimukset, myös pitkään kehityksen jo alettua. Ketterät prosessit hyödyntävät muutosta asiakkaan kilpailuedun kasvattamiseksi.
3. Toimitamme toimivaa ohjelmistoa säännöllisesti, parista viikosta pariin kuukauteen suosien lyhyempää aikajaksoa.
4. Liiketoiminnan ja kehittäjien tulee tehdä yhteistyötä päivittäin koko projektin ajan.
5. Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille ympäristön ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
6. Kaikkein tehokkain menetelmä välittää tietoa kehitystiimin sisällä on kasvokkain käytävä keskustelu.
7. Toimiva ohjelmisto on edistymisen ensisijainen mittari.
8. Ketterät prosessit kannattavat kestävää kehitystä. Sponsorien, kehittäjien ja käyttäjien tulee pystyä ylläpitämään kehityksen tahti pitkälle tulevaisuuteen.
9. Jatkuva huomio korkeaan tekniseen laatuun ja hyvään rakenteeseen edesauttaa ketteryyttä.
10. Yksinkertaisuus, tekemättä jätettävän työn maksimointi on oleellista.
11. Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvista tiimeistä.
12. Tiimi miettii säännöllisin väliajoin, kuinka parantaa tehokkuuttaan ja muuttaa toimintaansa sen mukaisesti.

(AGILE MANIFESTO)