VERTAAVA ESITYS OHJELMISTOKEHITYKSESTÄ PROJEKTINA JA PROSESSINA

Case: Nurseus-hoitosuunnitelmajärjestelmä

SAMI LEINO

Opinnäytetyö
Joulukuu 2011
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon
suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

LEINO, SAMI: Vertaava esitys ohjelmistokehityksestä projektina ja prosessina
Case: Nurseus-hoitosuunnitelmajärjestelmä

Opinnäytetyö 44 sivua
Joulukuu 2011

Ohjelmistokehitys on mysteeri monelle tavalliselle ihmiselle. Yleinen käsitys on, että ohjelmistokehitys on vain ohjelmoijien istumista pienissä kopeissa toimistotiloissa. Harva tietää, että ohjelmistokehitykseen sisältyy paljon muutakin kuin vain ohjelmakoodin kirjoittamista.

Tämä opinnäytetyö avaa ohjelmistokehityksen sisältöä ja pyrkii Tampereen ammattikorkeakoululle tehtyä Nurseus–hoitosuunnitelmajärjestelmän kehitystä esimerkkinä käyttäen kertomaan ohjelmistokehityksestä sellaisin termein ja kuvauksin, joita tavallinen lukija voi ymmärtää.

Lukijaa johdatellaan ensin työssä käytettyyn termistöön, jotta asioista voidaan puhua myöhemmin ilman, että lukija jää miettimään, mistä on kyse. Termien kuvaamisen jälkeen esitetään keskeiset käsitteet, joita opinnäytetyössä käsitellään: projekti ja prosessi. Kun termit on esitelty, käydään työssä läpi Nurseus-hoitosuunnitelmajärjestelmän kehitysraportti. Raporttia hyödyntäen ohjelmiston kehitysvaiheet selitetään ja lukijalle kerrotaan, mitä ohjelmistokehityksen vaiheissa tulee huomioida ja miten ne on huomioitu.

Työssä pyritään kertomaan tavalliselle lukijalle, ettei ohjelmistokehitys ole vain ohjelmakoodin kirjoittamista. Ohjelmistokehitys esitetään vaiheittain korostaen huomioitavia asioita, kun ohjelmistokehitystä ajatellaan projektina ja kun ohjelmistokehitystä käsitellään projektia laajempana kokonaisuutena: prosessina. Käsittely aloitetaan alustavasta ohjelmiston tilauksesta ja päätetään, kun ohjelmisto on otettu käyttöön ja jätetty odottamaan jatkokehitystä.

Lopulta arvioidaan, onko Nurseus -hoitosuunnitelmajärjestelmän kehityksessä onnistuttu huomioimaan opinnäytetyössä esitettyjä suuntauksia.

Asiasanat: Ohjelmistokehitys, ohjelmointi, projekti, ohjelmistosuunnittelu, vaatimusmäärittely, projektisuunnitelma, kehitysprosessi

ABSTRACT

Tampere University of Applied Sciences
Bachelor's degree programme of
information and communications technology

LEINO, SAMI: Comparative Demonstration of Software Development as a Project and a Process
Case: Nurseus Treatment Planning System

Bachelor's thesis 44 pages
December 2011

Software development is a mystery to many normal people. A common idea is that software development is merely programmers sitting in small cubicles in offices. Few know that software development includes a lot more than just writing code.

This thesis unveils the contents of software development and using Nurseus - treatment planning system created for Tampere University of Applied Scienses as an example to tell about software development with such terms and descriptions, that a normal reader can understand.

The reader will first be guided into the terms used in the thesis, so things can be presented later without leaving the reader wondering clueless of what was just said. After describing the terms, the main ideas, project and process, used in the thesis are presented. After presenting the terms, the thesis goes through the development report of Nurseus -treatment planning system. Taking advantage of this development report, the development phases of the system are explained to the reader and the reader is told what should and what has been taken into account in different phases of development.

The thesis tries to tell the normal reader that software development is not merely writing code. Software development is presented in phases whilst highlighting things to be taken into consideration when software development is thought as a project and when software development is handled as a broader concept, a process. Presentation will start with the starting order of the software suite and end when the suite has been implemented into use and is waiting future development.

Finally the success of Nurseus -treatment planning system's development in following the guidelines given in the thesis is evaluated.

Keywords: Software development, programming, project, software design, requirement specification, project planning, development process

SISÄLLYS

1 JOHDANTO

Ohjelmistokehitys on monelle ihmiselle täysi mysteeri. Moni luulee ohjelmistokehityksen rajoittuvan vain koodin kirjoittamiseen. Näin ei kuitenkaan ole, koska ohjelmistokehittäjän tulee ottaa huomioon monia asioita voidakseen tuottaa ohjelmia, joita asiakkaat voivat käyttää ongelmitta ja joita voidaan tulevaisuudessa kehittää ilman suuria ajallisia investointeja.

Tämä opinnäytetyö pyrkii kertomaan ohjelmistokehityksestä tietämättömän ihmisen termein ja ilmauksin, mitä ohjelmistokehitys pitää sisällään ja mitä ohjelmistokehityksessä on otettava huomioon. Tarkoituksena on hieman raottaa kantta iät ja ajat mysteerinä pysyneen ohjelmistokehityksen mustasta laatikosta. Opinnäytetyö välttää käyttämästä termistöä, joka ei avautuisi ohjelmointia tuntemattomalle lukijalle ja tapauksissa, joissa termistöä joudu-taan käyttämään, termit pyritään selittämään riittävän kattavasti.

Vaikka johdanto tulee avaamaan tietojenkäsittelyn ja erityisesti ohjelmistotekniikan termistöä kokemattomalle, ei opinnäytetyön tarkoitus ole tehdä lukijasta ohjelmistokehittäjää vaan kertoa, mitä ohjelmistokehittäjän rooliin todellisuudessa kuuluu. Ohjelmistokehittäjän rooli ei välttämättä rajoitu niihin osiin, jotka tässä teoksessa eritellään, mutta nämä ovat kuitenkin pääasialliset osat ohjelmistokehityksestä. Vaikka esimerkkinä käytetyn ohjelmiston kehityksessä käytetyt tekniikat esitellään, ei lukijan kuitenkaan odoteta taitavan kyseisten tekniikoiden saloja, vaan ymmärtävän pinnallisesti, mitä tai, miten mainitut tekniikat toimivat ja, miten niitä hyödynnetään.

Esitystapa tukeutuu vahvasti Tampereen ammattikorkeakoululle kehitetyn ohjelmiston, Nurseus –hoitosuunnitelmajärjestelmän, vaiheisiin sekä esittelee ohjelmiston kehitysraportin. Kehitysraporttia hyödyntäen kehityksen vaiheet ja niihin liittyvät päätökset eritellään ja avataan käsiteltäväksi tavalla, joka toivottavasti valaisee ohjelmistokehitystä paremmin kuin esimerkeittä annettu lausunto.

Vaikka tämä työ on kirjoitettu maallikon ymmärrettäväksi, voi myös ohjelmistokehittäjä tai mainitun hoitosuunnitelmajärjestelmän jatkokehittäjä löytää opinnäytetyöstä osia, jotka on syytä huomioida ohjelmistoja kehitettäessä.

Kuten monella alalla, myös informaatio- ja kommunikaatiotekniikalla on oma termistönsä ja omat laitteensa ja tapansa. Ohjelmistokehityksestä puhuminen vaatii joidenkin termien selittämistä, jotta asiaa voidaan pohtia. Tietysti monella termillä on monia merkityksiä tai niiden tarkoitus voi muuttua riip-puen millä informaatio- tai kommunikaatiotekniikan osalla niitä käytetään. Laitteiden ja tekniikoiden yksityiskohtaista toimintaa ei ole tarkoitus kuvata vaan antaa yleiskuva termin tai laitteen tarkoituksesta. Joillakin termeillä voi olla laajempia merkityksiä, joten termit on kuvattu vain siltä osin kuin opinnäytetyön ja lukijan kannalta on olennaista. Kirjallisuudesta löytyviä kuvauksia on jouduttu soveltamaan, jotta kuvaukset on voitu kääntää suomeksi tai saatu yksinkertaistettua ja eroteltua termejä muista termeistä, joita opinnäytetyö ei käsittele. Esitetyt termit ovat kirjoittajan esittämiä näkemyksiä ja termejä käytettäessä tarkoitetaan tässä luvussa esitettyjä asioita.

2 KÄYTETYT TERMIT JA TEKNIIKAT

2.1 Käytettyä termejä

Tietokoneohjelma:
Tietokoneohjelma on tietokoneen suorittama tapahtuma, joka toteuttaa ennalta ohjelmoidun toiminnon saamiensa tietojen tai asetusten mukaan.

Tietokoneohjelmisto:
Ohjelmisto on kokoelma ohjelmia, jotka on ohjelmoitu suorittamaan yhtä suurempaa toimintokokonaisuutta. Ohjelmiston ohjelmat voidaan erottaa toisistaan niin että kukin ohjelma toimii myös erikseen. Tällöin kuitenkin ohjelma ei välttämättä palvele enää sitä tarkoitusta johon se on ohjelmistoon rakennettu.

Käyttöliittymä:
Käyttöliittymä on näkymä, joka toimii käyttäjän ja laitteen tai laitteiston välissä. Tällä voidaan tarkoittaa muun muassa tietokoneen käyttöjärjestelmän graafista ulkoasua tai tietokoneohjelman graafista näkymää. Myös web-sibun linkit ja painikkeet muodostavat web-sivun käyttöliittymän.

Tietokanta:
Tietokanta on tapa kirjata ja säilyttää tietoa ennalta määrätyllä tavalla. Esimerkiksi listausmenetelmässä tiedot on kerätty riveihin niin että yksi rivi tai "tietue" kattaa ennalta määrätyn määrän nimettyjä tietoja. Esimerkiksi yhden henkilön henkilötiedot voidaan tallentaa yhtenä tietueena (kuva 1).

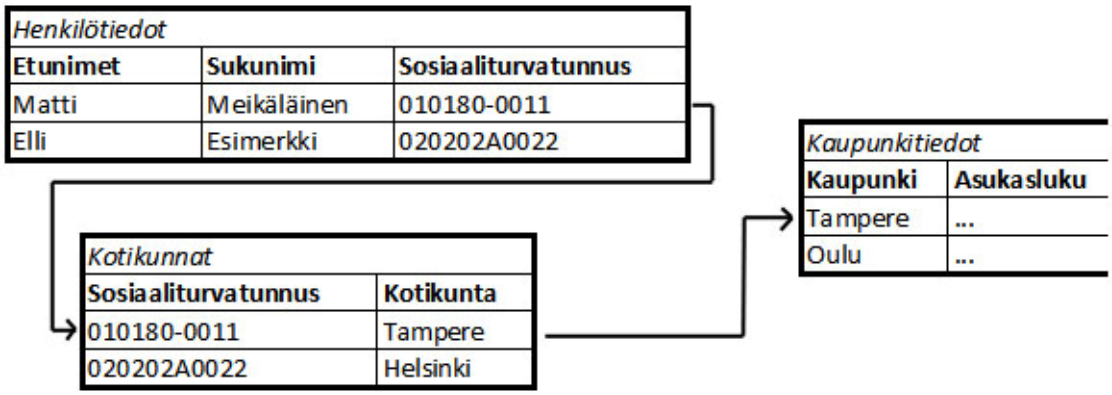
	Etunimet	Sukunimi	Sosiaaliturvatunnus
Tietue 1 -->	Matti	Meikäläinen	010180-0011
Tietue 2 -->	Elli	Esimerkki	020202A0022

Kuva 1 : Yksittäisistä tietueista koostuva tietokanta (huom: esitetyt tiedot ovat kuvitteellisia).

Jokainen tietue koostuu tietystä määrästä yhteneviä tietoja, jotka on tallennettu omalle rivilleen ja tietueen yksittäiset tietokentät tai rivin "sarakkeet" on nimetty kuvamaan nimenomaisen sarakkeen tietoa. Tätä kutsutaan tietokantatauluksi. Tietokantataulussa kaikki tiedot kirjataan listanomaisesti samassa järjestyksessä.

Relaatiotietokanta:
Relaatiotietokanta poikkeaa edellä esitetystä listausmenetelmän tietokantamallista. Relaatiotietokannassa saattaa olla useita tietokantatauluja ja

eri taulujen tiedot voidaan liittää toisiinsa "viiteavaimella". Kuva 2 antaa esimerkin relaatiotietokannasta ja tietojen liittämisestä.



Henkilötiedot

Etunimet	Sukunimi	Sosiaaliturvatunnus
Matti	Meikäläinen	010180-0011
Elli	Esimerkki	020202A0022

Kaupunkitiedot

Kaupunki	Asukasluku
Tampere	...
Oulu	...

Kotikunnat

Sosiaaliturvatunnus	Kotikunta
010180-0011	Tampere
020202A0022	Helsinki

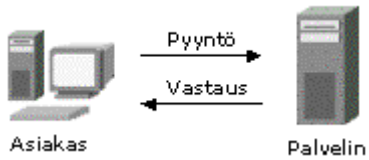
Kuva 2 : Esimerkki relaatiotietokannasta (huom: esitetyt tiedot ovat kuvitteellisia).

Viiteavaimen avulla yhden tietokantataulun tietue voidaan liittää toisen taulun tietueeseen. Näin voidaan yhdellä tietokantakyselyllä saada tietää, eri asioita ilman, että koko tietokantaa joudutaan tarkastelemaan kerralla. Viiteavain myös mahdollistaa tietokannan yhtenäisyyden, jos johonkin tietueeseen tehdään muutoksia. Esimerkiksi, jos edellä esitetyn kaltaisessa tietokannassa kaupunki "Tampere" vaihtaisi nimeään, kertaalleen tallennettu tieto päivittyisi suoraan muihin tauluihin, jossa kaupunkiin viitataan viiteavaimella ilman, että muita tietokantatauluja tarvitsee päivittää.

Web-selain:
Tietokoneohjelma tai -ohjelmisto, jolla on mahdollista selata web-sivuja. Tunnetuimpia web-selaimia ovat 28.6.2011 w3schools.com:in mukaan Mozilla Firefox (keskimäärin 42,5% kaikista käytetyistä selaimista), Google Chrome (keskimäärin 25,5% kaikista käytetyistä selaimista) ja Microsoft Internet Explorer (keskimäärin 25,0% kaikista käytetyistä selaimista).

Asiakaspääte:
Asiakaspääte on laite ja ohjelmisto, jolla tavallinen käyttäjä käyttää ja yhdistää ohjelmistoon. Laitteena saattaa olla tietokone, matkapuhelin, televisio tai muu laite, joka voi saada sisältönsä ulkoistetusta lähteestä.

Palvelin:
Palvelin on laite tai ohjelmisto, johon asiakaspääte voi yhdistää saadakseen tietyn palvelun, tietoa tai muuta sisältöä. Yksi palvelinlaite voi sisältää useamman palvelinohjelmiston. Nykyaikana yleisimpiä palvelimia ovat web-, tietokanta, ja ohjelmistopalvelimet. Esimerkiksi web-sivuja lukiessa sivu ladataan web-palvelimelta. Web-palvelin puolestaan saattaa lukea osan sisällöstään tietokantapalvelimelta.



Kuva 3 : Palvelin - asiakas.

Palvelin siis vastaa asiakaspäätteen pyyntöön sen mukaan, mitä asiakaspääte on palvelimelta pyytänyt. Esimerkiksi asiakaspääte voi pyytää palvelinta näyttämään web-sivun, jolloin palvelin lähettää web-sivun tiedot takaisin asiakaspäätteelle. Ennen kuin palvelin palauttaa vastauksensa, se saattaa suorittaa esikäsittelyä saattaakseen sivun haluttuun muotoon ennen kuin se lähetetään asiakkaalle.

Vaatimusmäärittely:
Vaatimusmäärittely on ohjelmistokehityksen vaihe, jonka tuloksena syntyy asiakkaan tai tilaajan esittämä listaus ja kuvaus niistä ominaisuuksista, toiminnoista ja suorituskyvystä, joita asiakas tai tilaaja toteutettavalta ohjelmistolta vaatii. Vaatimusmäärittelyn tarkoitus on kertoa ohjelmiston toteuttajalle, mitä asiakas ohjelmistolta haluaa, miten ohjelmiston tulisi toimia ja miltä ohjelmiston tulisi näyttää.

Mikäli vaatimusmäärittely ei määrää jotakin ohjelmiston ominaisuutta tai toimintoa, ei ohjelmiston toteuttajan tule sitä toteuttaa resurssien säästämiseksi. Asiakas ei myöskään voi hylätä projektia, mikäli projektin tuotoksesta puuttuu ominaisuuksia, joita vaatimusmäärittelyssä ei mainita.

Useimmissa tapauksissa vaatimusmäärittely laaditaan yhteistyössä asiakkaan ja toteuttajan kanssa, jolloin (asianmukainen) toteuttaja voi opastaa asiakasta ominaisuuksista ja toiminnallisuudesta, joita ohjelmistoon on hyvä tuoda. Tällöin myös toteuttajan on mahdollista neuvotella asiakkaan kanssa ominaisuuksista, jotka saattavat asiakkaasta vaikuttaa yksinkertaisilta, mutta voivat toteuttajan kannalta olla hyvinkin vaikeita tai jopa mahdottomia. Vaatimusmäärittelyneuvottelut alkavat usein hyvinkin suurpiirteisistä vaatimuksista, jotka neuvottelun edetessä tarkentuvat siihen muotoon, jossa ne tullaan toteuttamaan tilattuun ohjelmistoon.

Myös professorit Frank Tsui ja Orlando Karam tukevat tätä näkemystä kirjassaan: (Essentials of software engineering, 2007, sivu 12). Professorit puhuvat kuitenkin toteuttajasta ohjelmistoinsinöörinä ja esittävät, että insinööri tekee aina tekniset valinnat vaatimusmäärittelyssä. Esitetyssä näkemyksessä ei mainita, että asia kuitenkin riippuu asiakkaasta ja yleensä asiakas esittää, että toteutettavan uuden ohjelmiston on toimittava asiak-kaalla jo olemassa olevien ohjelmistojen ja laitteiden kanssa. Tällainen vaatimus yleensä rajaa hyvin paljon mahdollisia teknisiä valintoja ja tällöin vaatimus toiminnasta olemassa olevassa järjestelmä- ja laitteistorakenteessa kirjataan vaatimusmäärittelyyn.

2.2 Käytettyjen tekniikoiden kuvaukset.

XML

Eng: eXtensible Markup Language. XML ei ole ohjelmointikieli eikä konekieltä vaan systemaattinen tapa ilmaista tietoa ja tiedon tyyppiä tekstimuodossa. XML on tapa kirjata tietoa tietyin ehdoin, niin että tieto kirjataan "elementteinä", joilla on tarvittaessa myös attribuutteja eli elementtiä kuvaavia

lisätietoja. XML -dokumentti siis muodostuu elementeistä, joilla on tarvittaessa attribuutit ja lapsielementit tai tekstisisältö. Esimerkki kuva 4:

```
<viesti luokitus="tarkea">
    <vastaanottaja>
        <nimi>Matti meikäläinen</nimi>
        <osoite>Jossain tuolla</osoite>
    </vastaanottaja>
    <lahettaja>
        <nimi>Maija Meikäläinen</nimi>
        <osoite>Jossain täällä</osoite>
    </lahettaja>
    <otsikko>Tärkeä viesti</otsikko>
    <sisalto>Tämä on viestin sisältö</sisalto>
</viesti>
```

Kuva 4 : xml viesti.

Tässä esimerkissä 'viesti' on yksi elementti, jolla on attribuutti 'luokitus', jonka arvo on 'tarkea'. 'Viesti' -elementillä on sisältönään neljä muuta elementtiä: 'vastaanottaja', 'lahettaja', 'otsikko' ja 'sisalto'. 'Vastaanottaja' -elementillä puolestaan ei ole attribuutteja ja elementin sisältö koostuu 'nimi' ja 'osoite' -elementeistä. 'nimi' ja 'osoite' -elementeillä taas on sisältönään vain tekstiä. On tärkeä huomata, että elementti alkaa aina '<elementinnimi>' ja päättyy '</elementinnimi>'. Lisähuomio on, että elementin on päätyttävä aina sen elementin sisällä, jossa se on. Näitä elementtirakenteita hyödyntämällä voidaan saada aikaan kompleksejakin tietorakenteita, joita on puumaisesta rakenteesta johtuen helppo käsitellä.

XHTML

Eng: eXtensible HyperText Markup Language. XHTML on XML rakennetta noudattava, tarkkaan määritelty kokoelma elementtejä, joiden avulla voidaan rakentaa verkkosivuja. Selaimet tulkitsevat XHTML -dokumentin tiedot ja

esittävät ne XHTML -dokumentin kertomalla tavalla. Kuvan 5 esimerkissä on kyse XHTML Strict 1.0 suosituksen mukaisesta dokumentista.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Sivun otsikko</title>
    </head>
    <body>
        <div>Ja sivun teksti (tai vastaava)</div>
    </body>
</html>
```

Kuva 5 : XHTML -sivu.

"<!DOCTYPE ... " -julistus on selaimia varten, jotta ne tietävät minkä standardin mukaan XHTML -dokumenttia tulee tulkita. "<head>" -elementin sisällä olevat elementit kertovat lisätietoja siitä, kuinka sivu tulee näyttää sekä muita lisätietoja sivusta. Muun muassa elementti "<title>" kertoo sivun nimen tai otsikon, joka näytetään selaimen otsikkopalkissa. "<body>" -elementin sisältö taasen kertoo, kuinka ja, mitä itse sivulla näytetään. Käyttämällä 'body' ja 'head' -elementtien sisällä sopivia lapsielementtejä, on mahdollista rakentaa hyvinkin komplekseja verkkosivuja. XHTML -dokumenttiin pätevät samat säännöt kuin XML -dokumenttiin, sillä erotuksella, että käytettävissä olevat elementit ja paikat, joissa niitä voi käyttää on määritelty tarkasti.

CSS

Eng: Cascading Style Sheet. CSS -muotoilu on tapa koota web-sivun ulkoasun asetuksia yhteen paikkaan, josta niitä voidaan kutsua XHTML -dokumentista. Tavallisesti ulkoasun määritykset pitäisi kertoa jokaisessa XHTML -dokumentissa aina kun tyyliä haluttaisiin käyttää. CSS -muotoilun avulla tyyli voidaan muotoilla yhteen paikkaan, josta kaikki XHTML -dokumentit voivat kutsua tyyliä. Tällöin myös mahdolliset tyylin muutokset voidaan tehdä vain yhteen paikkaan. Lisäämällä XHTML -dokumentin "<head>" -elementtiin viittaus CSS -

tiedostoon (kuva 6), voidaan XHTML -dokumentissa viitata tiedoston sisällössä kuvailtuihin tyylien asetuksiin.

```
<head>
    <title>Sivun otsikko</title>
    <link href="tyyli.css"
    rel="stylesheet" type="text/css" />
</head>
```

Kuva 6 : XHTML 'head' elementti jossa CSS -tyylitiedosto.

CSS -tiedostossa voidaan määritellä useita eri tyyliyhdistelmiä, joihin XHTML dokumentti voi viitata. CSS -tiedosto sisältäisi useampia kuvan 7 kaltaisia merkintöjä erilaisista yhdistelmistä tyylejä tai tyylien osia.

```
#content {
    position: absolute;
    background-image: url(kuva.jpg);
    font-family: Arial, Helvetica, Sans-serif;
}
```

Kuva 7 : Esimerkki CSS tiedoston sisällöstä.

Kun XHTML -dokumentissa on oikea viittaus CSS -tiedostoon (kuva 8), voidaan minkä tahansa elementin attribuutilla kertoa selaimelle, kuinka sivu pitäisi näyttää.

```
<body>
    <div class="content">
    Ja sivun teksti (tai vastaava)
    </div>
</body>
```

Kuva 8 : CSS -tyylin käyttö XHTML -dokumentissa.

PHP

Eng: Hypertext PreProcessor. Kirjainlyhenne PHP on jäänne kielen alkuajoista. PHP ei varsinaisessa määrityksessään ole ohjelmointikieli, vaan komentosarja-

kieli. PHP:tä ei siis työstettäessä käännetä konekielelle, toisin kuin varsinaiset ohjelmointikielet. PHP-ohjelmaa ajettaessa suorittava alusta tai laite siis tulkitsee PHP:n tekstimuotoiset komennot ja toimii niiden mukaan. Usein-miten Web-palvelin suorittaa PHP-komentosarjoja ja näyttää lopuksi asiak-kaalle (käyttäjälle) tuloksenaan web-sivun, joka on rakennettu PHP:n suorit-teen tuloksena. PHP:llä on siis mahdollista muodostaa Web-ohjelmistoja, jotka ottavat tietoja käyttäjältä ja käsittelevät saamansa tiedon ja lopulta näyttävät uuden sivun saamiensa tietojen perusteella. Esimerkiksi PHP:llä voidaan rakentaa ohjelmisto, joka pyytää XHTML web-sivun avulla tietoja käyttäjältä ja tietojen avulla tulostaa uuden XHTML sivun.

SQL

Eng: Structured Query Language. SQL on relaatiotietokantoja varten standardoitu kyselykieli. Sen avulla on mahdollista välittää eri ohjelmista kyselyitä SQL-tietokantapalvelimelle ilman, että SQL-kielellä tehtyjä osia tarvitsee muuttaa, jos ohjelmiston käyttämä tietokanta muuttuu. SQL-kielellä kirjoitetaan tietokantaohjelmalle lähetettäviä lauseita, jotka sisältävät ohjeet ja käskyt tietokantaohjelmalle sekä kriteerit, joiden mukaan tietokanta palauttaa tuloksena halutut tiedot (kuva 9).

```
SELECT id, nimi, osasto FROM tyontekijat WHERE osasto='rekrytointi'
```

Kuva 9 : Esimerkki SQL lauseesta.

Esimerkissä isoin kirjaimin kirjoitetut osat ovat käskyjä tietokantapalvelimelle ja pienellä kirjoitetut ovat käskyjen ehtoja ja tietoja siitä mitä tietoa tietokannan halutaan palauttavan. SQL siis sisältää sarjassa käskyjä ja käskyjen lisätietoja, toisin kuin muut tietokonekielet, joissa on yksi komento, joka saa jälkeensä lisätietonsa. SQL:llä on tästä johtuen mahdollista rakentaa hyvinkin monimutkaisia tietokantahakuja yhdellä lausekkeella.

Integraatio

Kaikki edellä esitetyt tekniikat on mahdollista yhdistää toisiinsa lähes saumattomasti. Näin saadaan aikaan dynaaminen, XHTML -suositusta noudattava

web-sivusto, joka käyttää tietojen tallennukseen sekä SQL-tietokantaa että XML-tiedostoja.

2.3 Prosessi ja projekti

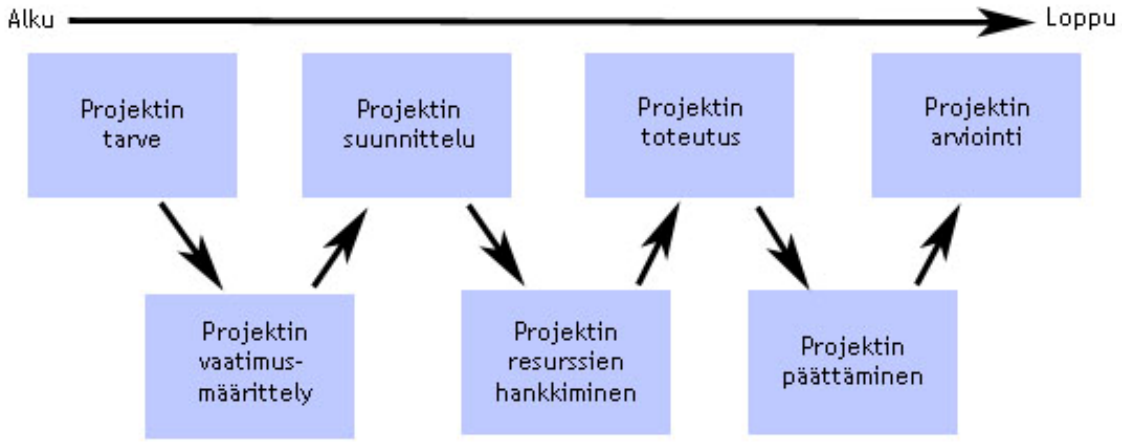
Ohjelmistokehitystä voidaan pitää sekä projektina että prosessina. Näillä käsityksillä on kuitenkin suuri ero. Erosta huolimatta kumpaakaan näkemystä ei voi jättää huomiotta, kun ohjelmistoja kehitetään. Tässä opinnäytetyössä on tarkoitus esittää ohjelmistokehitys kummastakin näkökulmasta. Aluksi on siis hyvä selvittää projektin ja prosessin merkitys sekä kertoa niiden eroista ja yhtäläisyyksistä.

2.3.1 Projekti

Projekti on kehys työlle tai työmäärälle, joka on ennalta rajattavissa. Projektilla on aina selvä alku, kun projekti suunnitellaan ja sen resurssit ja päämäärä määritellään sekä loppu, jolloin projektin tarkoitus tai määränpää on saavutettu ja projektin onnistuminen arvioidaan.

Projekti alkaa, kun on nähtävissä tarve toteuttaa jotakin. Ohjelmistokehityksessä tämä tarkoittaa hetkeä, jolloin on nähtävissä tarve kehitettävälle ohjelmistolle. Ohjelmistoa kehittävässä yrityksessä taas tarve syntyy, kun asiakas tilaa ohjelmiston. Tarpeen ollessa ilmeinen laaditaan asiakkaan tai ohjelmistotarpeen esittävien tahojen kanssa vaatimusmäärittely, jonka ehdot projektissa toteutettavan ohjelmiston tulee täyttää. Kun vaatimusmäärittely on tehty ja hyväksytty, voidaan siirtyä suunnittelemaan projektia. Projektisuunnitelmassa arvioidaan, mitä resursseja projektin toteuttaminen vaatii ja kuinka paljon aikaa projekti tarvitsee. Ennen projektin tuotoksen varsinainen toteutus alkaa, suunnitellut resurssit varataan projektin käyttöön. Jos resursseja ei varata ajoissa, voi projekti kaatua resurssien puutteeseen. Projekti päätetään, kun ohjelmisto voidaan antaa asiakkaalle tai ottaa käyttöön, kun vaatimusmäärittelyn esittämät ehdot on täytetty. Lopulta, kun projektin tuo-

tos on valmis, sen onnistuminen arvioidaan. Arvioinnissa katsotaan, onko projekti toteutunut suunnitellusti ja kuinka tulevia projekteja voidaan parantaa, jotta ne onnistuisivat (kuva 10). Kaiken tämän jälkeen projekti on päättynyt eikä siihen tavitse enää palata.



Kuva 10 : Projektin kulku.

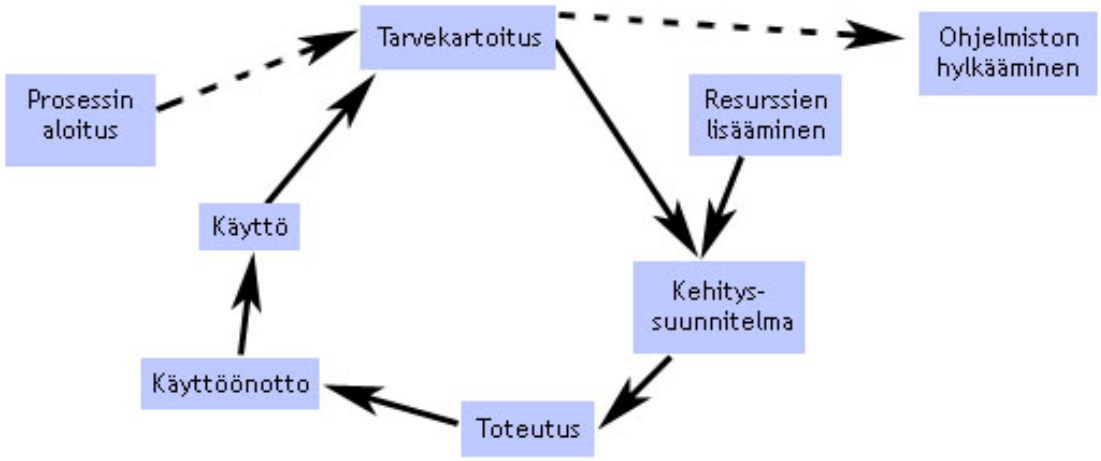
Projekti voi myös olla työ, jolla ohjelmiston kehitysprosessi käynnistetään. Tällöin projektiin ei sisälly varsinaista ohjelmiston toteutusta, vaan projektilla luodaan tila, jossa ohjelmistoa voidaan toteuttaa prosessin seuraavana projektina.

2.3.2 Prosessi

Todellinen prosessi alkaa kerran ja sitä jatketaan aina niin, että jotkin prosessin vaiheet toistetaan. Prosessin alun jälkeen varsinaiseen alkuun ei enää palata vaan prosessi jää kiertämään omaa "silmukkaansa".

Ohjelmistokehityksen prosessi alkaa kerran ja loppuu vain, jos kyseisen ohjelmisto hylätään kannattamattomana. Ohjelmiston kehitysprosessi alkaa, kun ohjelmaa aletaan kehittää. Tällöin kartoitetaan, millainen ohjelmistosta pitäisi tehdä. Tämän kartoituksen mukaan laaditaan kehityssuunnitelma ja prosessille annetaan ne resurssit, jotka kehityksen toteuttamiseen tarvitaan tai aiotaan käyttää. Kun käytettävissä olevat resurssit tiedetään, voidaan suunnitel-

la, kuinka ohjelmiston kehitys toteutetaan. Tämän jälkeen kehitys toteutetaan suunnitellusti ja tuotettu ohjelmisto otetaan käyttöön. Käytön aikana suoritetaan jatkuvasti tarvekartoitusta, jonka avulla nähdään, onko ohjelmistoa kehitettävä ja miten ohjelmistoa pitäisi kehittää. Tarvekartoi-tuksen jälkeen kierto alkaa uudestaan (kuva 11). Kehitysprosessi kestää niin kauan kuin tarvekartoituksessa ei nähdä tarvetta hylätä tai lopettaa ohjel-miston kehitystä.



Kuva 11 : Prosessin kulku. (Katkoviivoilla esitetty kulkusuuntaa käytetään vain kerran prosessin alkaessa ja päättyessä)

Yksi prosessin kierto voidaan käsitellä projektina eli projektoida, jolloin käytön ja tarvekartoituksen aikana luodaan vaatimusmäärittely uudelle kehitysprojektille. Kehityssuunnitelmana toimii yhteinen vaatimusmäärittely ja projektisuunnitelma. Kehitysprojekti päätetään, kun jatkokehitetty ohjelmisto on otettu käyttöön. Tämän jälkeen voidaan aloittaa uusi projekti keräämällä tietoa käytöstä ja suorittamalla uusi tarvekartoitus.

Prosessin päättyessä kannattamattomana, aloitetaan uusi kehitysprosessi, jonka avulla pyritään luomaan kokonaan uusi, parempi ohjelmisto.

2.3.3 Projekti osana prosessia

Vaikka prosessi ja projekti on käsitteinä erotettava toisistaan, on projektointi usein osa ohjelmistokehityksen prosessia. Esimerkiksi ohjelmistokehityksen aloitus voidaan projektoida, jolloin ohjelmiston kehitysprosessi alkuun saattaminen on oma projektinsa. Tämä projekti noudattaa edellä esitetyn projektin kaavaa ja päättyy, kun ohjelmiston kehitys on aloitettu uudella kehitysprojektilla. Prosessi jatkuu ohjelmiston käytöllä. Käytön aikana kerätään tietoa ohjelman toiminnasta ja käyttäjäpalautteesta. Käyttöä ja sen aikana tapahtuvaa tiedonkeruuta voidaan pitää ylläpitoprojektina. Tiedon perusteella laaditaan uusi kartoitus kehityksen tarpeesta. Jos kriteerit kehityksen tarpeelle täyttyvät, voidaan aloittaa uusi kehitysprojekti, jonka vaatimusmäärittely muodostuu asiakaspalautteesta ja käyttökokemuksien perusteella kerätyistä tiedoista muodostuneista puutteista ja parannusehdotuksista. Projekti noudattaa esitettyä kaavaa kunnes taas päättyy uuteen käyttöönottoon ja ylläpitoprojektiin. Tämä kierto jatkuu kunnes ohjelmiston kehitysprosessi päätetään hylätä.

3 NURSEUS -HOITOSUUNNITELMAJÄRJESTELMÄ

"Nurseus" on Tampereen ammattikorkeakoululle tilauksesta harjoittelusuhteessa suunniteltu ja toteutettu hoitosuunnitelmajärjestelmä opetuskäyttöön. Järjestelmän avulla opiskelijoiden on mahdollista kirjata järjestelmään potilaan tietoja FinCC-luokituksen mukaisilla luokkatunnuksilla sekä kirjata potilaalle hoitotyön toteutuksia sitä mukaan, kun potilasta hoidetaan. Järjestelmä luotiin opetuskäyttöön korvaamaan aiemmin käytetty 'kynä ja paperi' -menetelmä, jota käyttäen opiskelijat ovat aiemmin tehneet opintojen vaatimia harjoittelutehtäviä. Järjestelmä myös mahdollistaa opettajille töiden tarkistamisen ilman irrallista työn palautusta.

Tämä kappale on kyseisen ohjelmiston kehitysraportti ja kattaa ohjelmiston kehityksen vaatimusmäärittelystä, lopulliseen toimeksiantajalle luovutettuun

ohjelmistoon. Koska ohjelmisto on kehitetty kahdessa jaksossa, tullaan nämä kehitysjaksot käsittelemään erikseen tässä kehitysraportissa.

3.1 Vaatimusmäärittely

Nurseus -hoitosuunnitelmajärjestelmän vaatimusmäärittelyn tekeminen alkoi hyvin väljällä hakuilmoituksella, josta kävi ilmi vain pinnallisesti mitä ohjelmistolta haluttiin. Tämä vaatimusmäärittely esitettiin sähköpostiviestissä, jolla ohjelmistolle haettiin toteuttajaa. Tämän viestin avulla käynnistettiin projekti, jolla Nurseus -järjestelmän kehitys saatettiin aloittaa.

Alkuperäinen vaatimusmäärittely oli sähköpostissa muotoa: "*Tehtävänä on luoda TAMKin terveysalan opetuksen käyttöön sähköinen potilastietojen kirjaamisalusta. Kyseisen kirjaamisalustan tulee sisältää kansallisesti sovitun suomalaisen hoitotyön luokituksen (FinCC). Valmiita kaupallisia sähköisiä potilastietojärjestelmiä on olemassa useita, entisessä PIRAMKissakin niitä on kaksi.Harjoittelijan toivotaan kehittävän näihin verrattuna yksinkertaisempi ja helppokäyttöisempi kirjaamisen alusta sairaanhoitajaopiskelijoiden opetuskäyttöön.* "

Tästä vaatimattomasta kuvauksesta kasvoi asiakasneuvotteluissa lopulta se vaatimusmäärittely, jonka mukaan Nurseus -hoitosuunnitelmajärjestelmä kehitettiin. Allekirjoittaneen saatua sovittua osallisuutensa ohjelmiston kehityksessä sovittiin resurssipuutteesta johtuen heti, että ohjelmiston kehitys käyttäisi "progressiivista vaatimusmäärittelyä" tai "agile" -kehitystapaa lähestyvällä tavalla, eli vaatimusmäärittelyä päivitettäisiin ja laajennettaisiin sitä mukaan, kun ohjelmiston ominaisuudet saavuttaisivat voimassa olevan vaatimusmäärittelyn. Näin voitiin myös antaa asiakkaalle mahdollisuus vaikuttaa ohjelmiston kehitykseen, haluttujen toimintojen selkeytyessä asiakkaalle. Alustavan vaatimusmäärittelyn valmistuttua allekirjoittanut saattoi laatia asiakkaalle alustavan projektisuunnitelman, jonka mukaan projektia alet-taisiin toteuttaa.

3.2 Projektisuunnitelma

Projektisuunnitelma kertoo, missä ajassa ja missä järjestyksessä projektin osia tehdään. Lisäksi projektisuunnitelma kertoo myös ne resurssit, joita projektin toteutuksen oletetaan kuluttavan. Koska Nurseus projekti toteu-tettiin palkattomana harjoitteluna, ei projektisuunnitelmaan kirjattu kulu-tettuja henkilöstöresursseja vaan ainoastaan aikasuunnitelma, jossa järjes-telmän oli oltava sellaisessa vaiheessa, että se saatettiin esitellä ensimmäistä kertaa.

Koska projekti oli alun alkujaan mitoitettu kahdelle toteuttajalle, oli projektisuunnitelman laadinnassa käytettävä porrastettua "agile" -mallia, jolloin projektisuunnitelmaa laajennettiin asteittain aina asiakasesittelyjen jälkeen, kun asiakkaan uudet tai tarkennetut toiveet oli kirjattu vaatimusmäärittelyyn. Projektisuunnitelmaan siis kirjattiin aina aikaväli jona tietyt ominaisuudet toteutettaisiin ja esiteltäisiin asiakkaalle arvioitavaksi. Arvioinnin jälkeen projektisuunnitelmaa päivitettäisiin sisältämään uusi aika-arvio uusien ominaisuuksien toteuttamisesta.

Kokonaisuutena projektisuunnitelma sisälsi asiakkaan vaatimat ominaisuudet sekä summittaisen aika-arvioin niiden toteuttamisen vaatimasta ajasta. Alustaviin ominaisuuksiin kuuluivat seuraavat ominaisuudet:

- Hallinnollinen käyttöliittymä, jolla oli mahdollista lisätä järjestelmään käyttäjiä sekä muokata ohjelmiston käyttämää FinCC tietokantaa.
- Järjestelmään kirjautuminen, jolla järjestelmän toimintoihin pääseminen vaatii kirjautumisen. Ilman kirjautumista järjestelmää ei ole mahdollista käyttää tai hallinnoida.
- Käyttäjähallinta, jolla käyttäjän oli mahdollista muokata omia tietojaan ja pääkäyttäjien (opettajien) mahdollista tarvittaessa muuttaa myös opiskelijakäyttäjien tietoja.
- Työympäristö, jolla annettiin opiskelijakäyttäjille annettiin mahdollisuus luoda vaaditunkaltaisia työtiedostoja, järjestelmän olemassa olevaa tietokantaa ja valmiiksi laadittuja kaavakkeita hyödyntäen.
- Tulostusnäkymä, jota käyttäen opiskelija- tai opettajakäyttäjät voivat tulostaa työnsä soveltuvassa muodossa.

- Ryhmähallinta, jossa opiskelijoilla on mahdollisuus ilmoittautua ryhmiin ja palauttaa töitään ryhmiinsä sekä opettajakäyttäjille mahdollisuus tarkastaa ja arvostella sekä nähdä palautettuja töitä tulostettavassa muodossa.

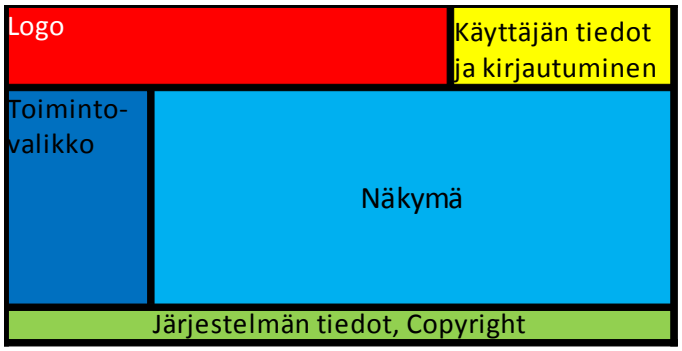
Projektisuunnitelman valmistuttua oli Nurseus -hoitosuunnitelmajärjestelmän kehitysprosessi käynnistetty ja saatettiin aloittaa ensimmäiseen kehitysprojekti. Resurssien puutteesta johtuen projektisuunnitelmaa ja projektin alkukehitykseen käytettyä aikaa jouduttiin venyttämään alustavasti sovitun kolmen kuukauden ulkopuolelle, jotta ohjelmisto oli mahdollista saada luovutettavaan muotoon. Luovutuksen yhteydessä sovittiin, että ohjelmiston kehitystä jatkettaisiin.

3.3 Alkukehitys ja varsinainen koodaus

Ohjelmiston fyysinen kehitys eli varsinainen koodaus alkoi toteuttajan koodirepositorion tarkastelulla. Koodirepositorio on tallenne toteuttajan tekemistä töistä tai töiden osista, joita voidaan uudelleen käyttää muissa projekteissa. Repositorion tarkastelulla on tarkoitus löytää mahdollisesti sopivia osia, jottei kaikkea tarvitse tehdä alusta alkaen. Samalla järjestelmälle luotiin oma keskitetty toimintokirjasto eli osa, joka sisältäisi toimintoja, joita käytettäisiin useasti järjestelmän eri osissa. Toimintokirjastoon liitettiin koodirepositoriosta löydetyt hyödylliset osat, jotka muokattiin vastaamaan järjestelmän tarpeita.

Repositorion läpikäynnin jälkeen oli päätettävä ohjelmiston yleinen ulkoasu. Tässä yhteydessä mielipide pyydettiin asiakkaalta ennen toteutusta. Toteut-

taja esitti ehdotuksen asiakkaalle hyvin karkealla kuvalla ulkoasusuunnitelmasta (kuva 12).



Kuva 12 : Kaavio ulkoasusta.

Kun ulkoasu oli hyväksytty, toteutettiin ensimmäisenä hallinnollinen osa järjestelmään. Hallinnollinen osa sisälsi tässä vaiheessa vain mahdollisuuden lisätä järjestelmään käyttäjiä. Asiattomien pitämiseksi poissa hallinnollisista toiminnoista, valittiin .htaccess suojaus vahvistamaan ylläpitäjä salasanalla ja käyttäjätunnuksella. .htaccess-tiedosto on asetustiedosto web-palvelimelle, jolla voidaan asettaa joitakin palvelimen asetuksia uudestaan. Muun muassa web-palvelimella olevaa sisältöä voidaan suojata salasanalla.

Kun käyttäjien lisääminen tietokantaan oli mahdollista, kyettiin järjestelmää varten luomaan kirjautuminen, jolla käyttäjät oli mahdollista varmentaa ja ohjelman toiminnot suojata tuntemattomilta käyttäjiltä. Kirjautuminen noudatti kahden yhdistettävän tiedon vertailua käyttäjien varmentamiseksi. Käyttäjätunnusta ja salasanaa käyttämällä käyttäjä voitiin sitoa riittävän luotettavasti käyttäjän muihin tietoihin. Kirjautuminen toteutettiin niin, että onnistuneen kirjautumisen jälkeen käyttäjälle annettin pääsy järjestelmän toimintoihin.

Kun käyttäjien oli mahdollista kirjautua järjestelmään, oli olennaista luoda käyttäjähallinta, jonka avulla käyttäjä saattoi muuttaa tietojaan. Muutettavat tiedot rajattiin kuitenkin käyttäjän tason mukaan. Opiskelijakäyttäjän sallittiin muuttaa vain yhteystietojaan ja salasanaansa. Opettajakäyttäjän sallittiin muuttaa kaikkia tietojaan, käyttäjätunnustaan lukuunottamatta. Kun käyttä-

jien omien tietojen muokkaus oli saatettu valmiiksi, oli olennaista mahdollistaa käyttäjien salasanojen nollaus, koska oli odotettavissa etteivät käyttäjät muistaisi salasanojaan. Nollaus toteutettiin niin, että opettajakäyttäjät voivat nollata opiskelijakäyttäjien salasanoja, samalla järjestelmän ylläpito-osioon liitettiin toiminnallisuus käyttäjien salasanojen nollausta varten ilman rajoitusta käyttäjätasoon, jotta ylläpitäjä voisi tarvittaessa nollata opettaja-käyttäjien salasanoja.

Käyttäjätietojen hallinan valmistuttua oli mahdollista aloittaa järjestelmän ydintoiminnallisuuden eli varsinaisen työympäristön toteuttaminen. Tätä varten luotiin kaksi toiminnallista ominaisuutta: Ensimmäisenä mahdollisuus luoda kaavakkeella järjestelmään työtiedosto, joka sisältäisi työn- ja potilaan perustiedot. Toisena ominaisuutena luotiin kaavakkeet työtiedoston- ja potilaan perustietojen muokkaamiseen sekä mahdollisuuteen muokata työtiedostoa lisäämällä ja poistamalla työtiedoston sisältämiä merkintöjä. Muokkaamista varten luotiin omat käyttöliittymänsä, joilla merkintöjä oli mahdollista lisätä ja muokata. Yhteensä muokkausnäkymiä luotiin neljä:

- Hoidon tarpeiden lisäys ja muokkaus.
- Hoidon tavoitteiden muokkaus.
- Hoitotyön toimintojen lisäys ja muokkaus.
- Hoitotyön toteutusten lisäys ja muokkaus.

Näiden avulla oli mahdollista muokata kaikkia työn tietoja potilastietoja lukuunottamatta. Käyttöliittymän terminologiaa ei tässä kohtaa tarkistettu asiakkaalta. Käyttöliittymän nimet pyydetyille tiedoille eivät siis vastanneet hoitotyön tarkoin määriteltyjä ammatillisia nimikkeitä tai seuranneet eettisesti oikeita nimityksiä.

Kun työtietoja oli mahdollista muokata kokonaisuudessaan, tuli tehtäväksi kehittää näkymä, jossa työtiedosto oli mahdollista tulostaa ilman käyttöliittymän osia. Näkymä rakentui käyttäjän valitseman työtiedoston mukaan ja kävi läpi kaikki työtiedostoon kirjatut osat. Lisäksi näkymä kirjasi tarkistussumman eli määrätyistä tiedoista lasketun merkkisarjan jokaiselle työtiedoston erilliselle osakokonaisuudelle, joita olivat potilastiedot, hoidon tar-

peet, hoitotyön toiminnot ja hoidon toteutukset. Tarkistussumma mahdol-listi tyon osakokonaisuuksien yksilöllistämisen, niin ettei eri töiden osia voitu sekoittaa keskenään.

Viimeisenä toteutettin ryhmähallinta, jolla pääkäyttäjien oli mahdollista luoda opiskelijoita varten ryhmiä ja lisätä opiskelijoita ryhmiin. Samalla opiskelijoille luotiin mahdollisuus jättää ohjelmiston käyttämään tietokantaan merkintä työnsä palautuksesta siihen ryhmään, johon opiskelija oli lisätty. Palautusmerkinnän avulla toteutettiin ominaisuus, jolla pääkäyttäjiien oli mahdollista tarkastaa ja arvostella palautettuja töitä. Töistä ei kuitenkaan palautuksen tai arvostelun yhteydessä otettu tallennettavaa kopiota. Näiden ominaisuuksien toteuttamisen jälkeen pääkäyttäjille luotiin mahdollisuus poistaa ryhmiä ja käyttäjiä ryhmistä sekä poistaa ryhmään palautettujen töiden merkintöjä.

Tässä vaiheessa projektille säädetty aika oli loppu. Projekti palautettiin toimeksianjalle ja ohjelmistoa varten kirjoitettiin luottamuksellinen tekninen dokumentaatio. Projektin tuotoksen asiakkaalle jättämisen jälkeen kuitenkin havaittiin, että ohjelmistossa oli virheitä, jotka olisi pitänyt havaita testauksessa. Virheiden korjaukset toimitettiin erikseen projektin palautuksen jälkeen. Oli kuitenkin selvää että projekti ei vielä vastannut niitä tavoitteita, joita sille oli asetettu. Koska projekti oli toteutettu alustavaa suunnitelmaa pienemmillä resursseilla. Projekti oli mitoitettu kahdelle ohjelmoijalle kolmen kuukauden ajalle. Niinpä sovitiin, että projektia jatkettaisiin myöhempänä ajankohtana. Kehitysprosessi siis jatkui tällä kohden ylläpitoprojektina, jolloin ohjelma oli käytettävissä, mutta ei vielä siinä pisteessä, että ohjelmisto voitaisiin ottaa täysimittaisesti käyttöön.

3.4 Jatkokehitys

Kehitysprosessin oltua jäissä 3 kuukautta, saatiin lopulta aikaa sopia uusi harjoittelusuhde ja sen myötä uusi vaatimusmäärittely ja projektisuunnitelma joiden laatimisella käynnistettiin uusi kehitysprojekti, jonka aikana Nurseus -

hoitosuunnitelmajärjestelmää kehitettäisiin. Samalla päätettiin, että järjestelmä otettaisiin osittain käyttöön, jotta käyttöliittymästä ja ohjelman yleisasusta sekä toimivuudesta sataisiin kattavampaa käyttäjäpalautetta. Eli samanaikaisesti toimisi kaksi projektia: kehitysprojekti ja ylläpitoprojekti.

Vaatimusmäärittelyyn kuuluivat:

- Ulkoasun parantaminen: Entinen taulukkopohjainen käyttöliittymä oli epäesteettisen näköinen ja tulisi muuttaa.
- Ryhmähallinnan parantaminen: Ryhmähallintaa tulisi muuttaa niin että opiskelijat voisivat itse ilmoittautua siihen ryhmään, johon kuuluvat.
- Käyttäjähallinnan laajentaminen: Pääkäyttäjien käyttöliittymään tuli lisätä toiminnallisuus opiskelijakäyttäjien tietojen selaamiseen.
- Ylläpidollisten tehtävien rajoitettu siirto, kirjautuneiden pääkäyttäjien hallittaviksi.
- Käyttöoikeuksien muuttaminen: Toiminnallisuuksien laajentuessa tuli myös pääkäyttäjien oikeuksia rajata osalta pääkäyttäjistä.
- Töiden arvioinnin muuttaminen: Opiskelijoiden töistä tulisi arvioinnin yhteydessä ottaa kopio, johon arvostelun tiedot tallennettaisiin, muun työtiedon ohella.
- Käyttäjien pyytämien muutosten toteutus: Toteuttajan tuli toimittaa järjestelmään ne muutokset ja/tai parannukset, joita käyttävät asiakkaat pyytäisivät, niissä määrin kuin se olisi projektin resurssien kannalta mahdollista.
- Yleisominaisuuksien hiominen/parantaminen: Olemassa olevia tai toteutettuja ominaisuuksia tulisi hioa ja parantaa vastaamaan asiakkaan toiveita.

Järjestelmän ulkoasu oli alustavasti toteutettu luomalla näkymän pohjaksi taulukko, jonka ruutuihin käyttöliittymän osat liitettiin. Toivomuksesta käyttöliittymä muutettiin noudattamaan XHTML ja CSS muotoilua. Samalla myös sovelluksen ulkoasun muuttamisesta saatiin dynaamisempaa ja mahdol-liset myöhemmät ulkoasumuutokset olisi mahdollista toteuttaa helpommin. Samal-

la osa käyttöliittymän näkymästä oli mahdollista yhdistää yhdeksi toiminnoksi, joka oli mahdollista ladata keskitetystä toimintokirjastosta.

Vaatimusmäärittelyyn tehtyjä vaateita toteutettiin projektin aikana nopealla aikataululla. Valmiissa järjestelmässä useiden ominaisuuksien muuttaminen tai parantaminen oli suhteellisen helppoa, koska perustoiminnallisuus oli jo olemassa. Kuitenkin joidenkin ylläpidollisten toimintojen siirto kirjautuneille pääkäyttäjille, eli silloisille opettajakäyttäjille tuotti ongelman, koska kaikki opettajakäyttäjät eivät tarvitsisi pääsyä näihin toimintoihin. Päätettiin jakaa opettajakäyttäjät kahteen ryhmään:

- Opettajat: rajattu toiminnallisuus. Ei mahdollisuutta täyteen ryhmähallintaan tai FinCC tietokannan muokkaamiseen.
- Pääkäyttäjät: Täysi toiminnallisuus. Ei kuitenkaan pääsyä ylläpidolle jääneisiin ominaisuuksiin kuten käyttäjien lisäykseen tai poistoon eikä myöskään mahdollisuutta nollata opettaja- tai pääkäyttäjien salasanoja.

Kun käyttöoikeudet oli jaettu uudelleen, päivitettiin ryhmähallintaa huomattavilla parannuksilla. Näihin kuuluivat muun muassa asiakkaan vaatima arvioitujen töiden säilytys lain vaatimaksi ajaksi. Ominaisuutta varten ryhmähallinnan toimintaa laajennettiin niin, että arviointi oli mahdollista antaa laajemmassa muodossa ja arviointi tallennettaisiin uuteen työtiedostoon, joka tallennettaisiin järjestelmään erillisenä kopiona. Samalla myös tulostusnäkymä päivitettiin niin, että sama näkymä kykeni käsittelemään myös arvosteltujen töiden näyttämisen. Muita järjestelmän komponentteja muokattiin riittävässä määrin, jotta uusi arviointijärjestelmä oli mahdollista integroida järjestelmään niin ettei se hajoittanut muita järjestelmän ominaisuuksia.

Järjestelmän toimintoja optimoitiin ja käyttöliittymää hiottiin niin, että käyttöliittymän termit saatiin vastaamaan terveydenhoitoalan hyväksymiä termejä. Opiskelijakäyttäjien toiveesta käyttöliittymään lisättiin myös joitakin toimintoja tai ominaisuuksia. Muun muassa joitakin "peruuta" ja "palaa" -painikkeita. Lisäksi toteutettiin joitakin hakutoimintoja, joilla oli yksinkertaista hakea järjestelmään kirjattuja tietoja, kuten käyttäjiä tai töitä.

Vaaditut ominaisuudet saatiin toteutettua projektin puitteissa. Varatusta ajasta niukasti yli jäänyt osa käytettiin käyttäjäkoulutukseen ja ylimääräisen asiakaspalautteen hankintaan.

3.5 Asiakaspalautteen huomiointi

Nurseus -hoitosuunnitelmajärjestelmän yhtenä määritelmänä oli "olla yksinkertainen ja saavutettava ilman suurta kokemusta tietokoneista tai vastaavista potilastietojärjestelmistä". Tätä ei kuitenkaan oltu huomioitu järjestelmän ensimmäisessä asiakkaalle esitetyssä versiossa, joka oli toteutettu englannin kielisenä. Ensimmäisen Tästä saatiin välittömästi palautetta, jonka mukaisesti järjestelmä muutettiin suomenkieliseksi. Käyttöliittymästä ei tämän lisäksi saatu muuta palautetta kuin aluksi käytetyn taulukkoasetelun rajojen näkymisestä. Rajat poistettiin, kun käyttöliittymä muutettiin toivomusten mukaan XHTML ja CSS muotoiluun. Muu käyttöliittymään liittyvä palaute rajoittui pääosin "palaa" ja "peruuta" painikkeiden puuttumiseen. Nämä ominaisuudet toteutettiin ensitilassa.

Palautteena pyydettiin myös, että käyttöliittymä näyttäisi ajastimen, joka kertoisi kuinka paljon aikaa on jäljellä ennen kuin käyttäjän istunto katkaistaan. Lisäksi opiskelijakäyttäjät esittivät joitakin toiveita järjestelmän FinCC tietokannan selaamiseen. Ominaisuudet kokonaisuuksina olisivat vaatineet JavaScript toiminnallisuuden tuomista järjestelmään, joten nämä käyttäjäpalautteen toiveet päätettiin ohittaa ilman suurempaa pohdintaa.

Osa saadusta palautteesta koski uusia ominaisuuksia, joita järjestelmä tarvitsi. Nämä lisättiin vaatimusmäärittelyyn ja toteutettiin käytössä olevien resurssien mukaan. Esimerkiksi ryhmähallinnan muuttamisen idea saatiin käyttäjäpalautteena. Myös joitakin uusia tietokenttiä lisättiin järjestelmän eri osiin.

Mutta tärkeimpänä asiakaspalautteena voidaan mainita asiakaskäyttäjien (oppilaat ja opettajat) antamat tiedot järjestelmän virheellisestä toimin-nasta,

jota ei oltu testauksessa huomattu. Tämä palaute mahdollisti järjes-telmän saattamisen muotoon, jossa se lopulta luovutettiin asiakkaalle.

3.6 Lopputulos

Nurseus -hoitosuunnitelmajärjestelmä on Tampereen ammattikorkeakoululle kehitetty kirjausjärjestelmä opetuskäyttöön. Järjestelmällä on mahdollista kirjata potilaan perustietojen lisäksi potilaan hoitokertomus, johon sisältyvät FinCC luokituksen mukaiset hoidon tarpeet ja hoitotyön toiminnot sekä hoitotyön toimintojen perusteella luotavissa oleva hoitotyön toteutusten listaus ja hoidon tavoitteet. Järjestelmä on helposti konfiguroitavissa toimimaan monenlaisissa ympäristöissä kunhan teknisen dokumentaation esittämät vaatimukset palvelinlaitteistolle ja -ohjelmistolle toteutuvat.

Järjestelmä täyttää sille esitetyt vaatimukset ominaisuuksista ja helppokäyttöisyydestä. Palautteen mukaan järjestelmä on helppokäyttöinen ja ymmärrettävä ilman kattavaa käyttäjäkoulutusta. Toiminnallisuudeltaan järjestelmä on vakaa, eikä itse ohjelmistosta johtuvia virheitä esiinny. Toteu-tettua järjestelmää siis voidaan käyttää opetustarkoitukseen, niin kuin alku-peräinen vaatimusmäärittely on esittänyt. Järjestelmä on myös riittävän yksinkertainen rakenteeltaan, jotta jatkokehitys on tarvittaessa helppo järjestää.

Järjestelmä valmistui siihen muotoon kuin projektien rajoissa oli määritelty. Jatkokehitystä ajatellen asiakkaalla oli jo tiedossa uusia toiminnallisia ominaisuuksia, jotka järjestelmän jatkokehittäjä voi integroida järjestelmään. Järjestelmässä siis on vielä kehitettävää, mutta lopputulosta voidaan pitää hyvin onnistuneena. Kehitysprosessi siis jatkuu ylläpitoprojektina, jonka aikana järjestelmästä kerätään palautetta ja sille etsitään uutta kehittäjää. Uuden kehittäjän löytyessä kehitysprosessi siirtyy taas uuteen kehitysprojektiin.

4 OHJELMISTON KEHITYS

Tämä luku kertoo Nurseus -hoitosuunnitelmajärjestelmän vaiheet ja esittää edellä kerrottujen tietojen pohjalta miksi valinnat on tehty. Osa näkemyksistä koskee päätöksiä, jotka on tehty, kun kyseessä oli ohjelmistokehitysprojekti. Osa taas tehty, kun kehitystä on ajateltu laajemmin prosessina. Prosessi ajatteluun kuuluu siis kehityksen ajattelu pitkälläkin aikajaksolla. Tästä johtuen vaikka jotkin valinnat voivat lyhyellä tähtäimellä vaikuttaa huonoilta ovat ne perusteltavissa, kun huomioon otetaan että ohjelmistoa kehitetään pidempään.

4.1 Toteutusalustan valinta

Toteutusalustan valinnalla viitataan siihen järjestelmäpohjaan, (käyttöjärjestelmä), jolle ohjelmisto tullaan toteuttamaan. Samalla valitaan myös ohjelmointikieli tai kielet, jolla ohjelmisto toteutetaan sekä mahdolliset muut järjestelmäpalvelut (kuten tietokannat), joita toteutettava ohjelmisto tulee käyttämään. Valintaan vaikuttavat oleellisesti vaatimusmäärittelyssä esitetyt toimintavaatimukset.

Kehitysprosessin alussa alustaksi tulee valita pohja, joka pysyy ennaltanähtävästi muuttumattomana pitkään jottei ohjelmistoa tarvitse tulevaisuudessa muuttaa kesken kehitysprosessin, kun toteutusalusta muuttuu. Valintaa tehtäessä kannattaa myös huomioida toteuttajan osaamisalueet ja valita alusta, jolle toteuttaja osaa tehdä ohjelmiston. Mikäli ohjelmisto päätetään toteuttaa asennettavana ohjelmistona, on kehityksessä huomioi-tava, että päätelaitteiden tulee täyttää ohjelmiston vaatimukset. Koska päätelaitteet kattavat hyvin laajan skaalan, yksinkertaisuutta haettaessa on syytä välttää järjestelmävaatimuksia, joista yksinkertaisen käyttäjän tulisi huolehtia.

Nurseus -hoitosuunnitelmajärjestelmän vaatimusmäärittely saneli yksiselitteisesti, että ohjelman tuli olla helppokäyttöinen. Ohjelmiston haluttiin olevan saavutettavissa kaikkialta ja ohjelmistotallenteiden tuli olla helposti hallinnoitavissa. Ohjelmisto siis ei voinut olla erikseen jokaiselle päätelait-teelle asennettava ohjelmisto. Tästä johtuen toteutustavaksi valittiin web-toteutus.

Tällöin ohjelmisto on saavutettavissa mistä tahansa web-kyke-nevästä laitteesta. Web-toteutusmalli poistaa myös tarpeen suunnitella ohjelmiston toimintaa riippuen siitä, mitä käyttöjärjestelmää ohjelmiston käyttäjä käyttää. Tällöin siis riittää, kun käyttäjän laitteessa on standardeja ja suosituksia noudattava web-selain.

Web-toteutus ei myöskään aseta vaatimuksia web-palvelimen käyttöjärjestelmälle, kunhan palvelimelta löytyvät vaaditut palvelut. Tällöin järjestelmää ei tarvitse muuttaa, vaikka toteutusalusta muuttuisi. Tästä esimerkkinä Nurseus -ohjelmiston kehitys, joka tehtiin Microsoft Windows käyttöjärjestelmässä, mutta sijoitettiin lopullisesti Linux -palvelimelle. Järjestelmän tekninen dokumentaatio tekee selväksi, mitä palveluja hoitosuunnitelmajärjestelmä palvelimelta vaatii eikä aseta vaatimuksia palvelimen käyttöjärjestelmälle. Siis niin kauan, kun nämä palvelut toimivat palveluiden dokumentaation esittämällä tavalla, niitä hyväkseen käyttävä ohjelmisto toimii.

4.2 Ohjelmistorakenteen suunnittelu

Ohjelmistorakenne on jo vaikeampia ohjelmistokehityksen käsitteitä. Ohjelmistorakenne voi myös muuttua ohjelmistoalustasta riippuen, mutta yliespiirteisessä suunnittelussa tällä tarkoitetaan ohjelmistoarkkitehtuurin osien, komponenttien, suunnitelua. Komponentti on osa ohjelmistosta, joka voidaan erottaa ohjelmistosta ilman, että ohjelmisto lakkaa toimimasta muilta, kuin poistetun komponentin osalta. Web-suunnittelussa tällä tarkoitetaan järjestelmän eri toimintojen eriyttämisen suunnittelua. Esimerkiksi järjestelmän hallinnolliset toiminnot voidaan eriyttää järjestelmästä niin etteivät ne vaikuta muun järjestelmän toimintaan.

Komponenttien eriyttämisen etu on, ettei koko järjestelmä lakkaa toimimasta, jos yhdessä komponentissa on virhe. Tämä myös mahdollistaa järjestelmän kehittämisen osissa. Tällöin järjestelmän komponentit voidaan antaa useammalle toteuttajalle tai mahdollinen kehitysprojekti voidaan rajata vain yhteen komponenttiin.

Kun rakennetta suunnitellaan, tulee toteuttajan huomioida toteuttajien osaamisalueet, jotta komponentit on jaettu niin, että kukin toteuttaja voi toimittaa oman osansa. Ja vaikka komponentit on eriytetty toisistaan, on ne jatkokehityksen kannalta kannattavaa toteuttaa samalla tavalla tai käyttäen samoja tekniikoita.

Kun eri komponentit käyttävät samanlaisia ominaisuuksia, on mahdollista siirtää ne omaan komentokirjastoonsa. Kun komentokirjasto on suunniteltu ajoissa ja hyvin, voidaan komponenteissa käyttää tähän komentokirjastoon toteutettuja toimintoja. Kun komentokirjasto on valmis projektin alkuvai-heessa, se säästää toteuttajien resursseja, kun eri toimintoja ei tarvitse toteuttaa uudestaan jokaiseen niinä käyttävään komponenttiin. Lisäksi, jos kehitysprosessissa toimintoihin on tarkoitus tehdä muutoksia, ne voidaan tehdä vain komentokirjastoon eikä kaikkia komponentteja tarvitse korjata tai muuttaa erikseen. Tämä helpottaa virheiden korjaamista tulevaisuudessa sekä helpottaa toimintojen muuttamista, jos vaatimusmäärittely muuttuu ja toimintaa pitää korjata vastaamaan uutta vaatimusmäärittelyä.

4.3 Visuaalisen ulkoasun suunnittelu

Visuaalisella ulkoasulla tarkoitetaan sitä, miltä ohjelmiston tulisi näyttää, kuinka painikkeet tulisi sijoittaa ja kuinka värejä tai kuvioita tulisi käyttää. Vaatimusmäärittely usein sanelee ehdot ulkoasusta. Yleensä ulkoasu suunnitellaan ja osittain toteutetaan hyvin varhaisessa vaiheessa kehitysprojektia, jotta tuotetta voidaan tarvittaessa esitellä asiakkaalle myös ennen kehitysprojektin päättymistä.

Vaatimusmäärittely saattaa kuitenkin jättää ulkoasun kuvauksen hieman väljäksi. Näissä tilanteissa kehittäjä joutuu ajattelemaan ulkoasua käyttäjän ja kouluttajan näkökulmasta. Nurseus -ohjelmiston kehityksessä siis oli suunniteltava ulkoasu sopivaksi hoitoalan opiskelijoille. Tässä tilanteessa kuten Irmeli Sinkkonen ja muut kertovat kirjassaan (Käytettävyyden psykologia, 2002,

sivu 254) on visuaalinen asu suunniteltava sellaiseksi, että käyttäjät voivat omatoimisesti käyttää järjestelmää. Järjestelmässä tulisi käyttää samoja termejä kuin käyttäjien saamassa tehtävänannossa tai koulutuksessa. Hoitoalan opiskelijoiden ollessa kyseessä tämä tarkoitti, että visuaalinen ulkoasu piti suunnitella yksinkertaiseksi ja käyttämään sairaanhoidon termejä. Myös valikoissa tuli käyttää sellaisia termejä, joihin tulevat käyttäjät olivat jo aiemmin törmänneet toimiessaan järjestelmän tilaajan Tampereen ammattikorkeakoulun ympäristössä.

Lisäksi Nurseus -hoitosuunnitelmajärjestelmän vaatimusmäärittely esitti vaatimuksena, että järjestelmän tuli näyttää yksinkertaisemmalta kuin hoitoalalla käytettävät ohjelmistot. Näin ollen kehittäjän tuli tutustua hoito-alalla käytettäviin potilastietojärjestelmiin. Kehittäjän siis pitäisi ostaa ja asentaa tai muuten päästä käsiksi joihinkin ohjelmistoihin, jotka ovat yleisesti käytössä hoitoalalla. Tilaaja kuitenkin antoi mahdollisuuden käyttää organi-saatiossa olevaa Logica plc:n Pegasos -potilastietojärjestelmää. Pegasos on kokonainen, useissa sairaanhoitoalan organisaatioissa käytetty potilastietojen hallintajärjestelmä.

Visuaalisen ulkoasun siis tuli pyrkiä olemaan yksinkertaisempi kuin Pegasos -järjestelmän ulkoasu. Visuaalisesta ulkoasusta siis piti tehdä yksinkertasempi, eli kokonaisen, alalla käytössä olevan, potilastietojärjestelmän visuaalisista osista piti valita, mitä käytettäisiin. Pääosin tämä tarkoitti, että Nurseus -järjestelmässä tuli käyttää samoja termejä kuin Pegasos -järjestelmässä. Lisäksi Pegasos ohjelma muodostui useista yhtäaikaisistä näkymistä ja yksinkertaistuksessa tuli huomioida, miten nämä näkymät voitiin erottaa selkeästi, jottei käyttäjän tulisi törmätä kaikkiin näkymiin yhtäaikaa. Myös valikoiden määrää karsittiin huomattavasti ja joitakin ominaisuuksia yhdis-tettiin, jotta ne saatiin saman näkymän sisään. Tärkeintä oli kuitenkin erottaa valikot ja eri toiminnot toisistaan niin, että käyttäjät pystyivät erottamaan ne. Ohjelma siis siirtyi aina siihen näkymään, joka oli käyttäjän tarpeen mukainen.

4.4 Suunnittelussa huomioitavat erityispiirteet

Jo edellä mainittiin, että ohjelmiston tuli olla yksinkertaistettu kokonaisesta, alalla käytössä olevasta, järjestelmästä. Lisäksi piti huomioida, ettei järjestelmän käyttäjillä olisi vielä koulutusta tietokonejärjestelmien käytössä. Näin ollen itse ohjelman käyttö tai asennus ei saanut olla vaikeaa. Koska lähes mikä tahansa päätelaitteelle asennettava ohjelmisto olisi asettanut päätelaitteelle vaatimuksia, jotka sen tulisi täyttää, jotta ohjelmistoa voisi käyttää. Jo mainittu web-toteutus vähensi vaatimusten määrää niin, että ainoaksi vaatimukseksi jäi standardeja ja suosituksia noudattava web-selain ja verkkoyhteys.

Ohjelmiston liittyminen vahvasti opiskelijoiden alaan, oli myös huomioitava, että ohjelmisto käytti sitä termistöä, jota sairaanhoitoalalla käytetään. Vaikka tietokonejärjestelmien käyttöliittymien kehittäminen englanniksi on helpompaa kuin suomeksi olemassaolevan sanaston ansiosta, tuli järjestelmä kuitenkin toteuttaa suomenkielellä, jotta opiskelijoiden ei tarvitsisi työn-teossa keskittyä ohjelman kielen kääntämiseen vaan opiskelijat saattoivat käyttää ohjelmaa sillä kielellä, jolla tulisivat työskentelemään.

4.4.1 Standardit ja niiden noudattaminen

On olemassa useita standardeja ja suosituksia eli virallisesti hyväksyttyjä tapoja ja menetelmiä, joilla ohjelmistoja toteutetaan. Kehitysprojektia toteutettaessa tulee valita ne standardit ja käytännöt, joita halutaan noudattaa. Standardien ja suositusten noudattaminen johtaa suoraan siihen, että ohjelmistokehitysprosessin jatkuminen helpottuu, kun mahdollinen jatkokehittäjä tietää standardin ja näin ollen voi jatkaa ohjelmiston kehitystä samoja standardeja käyttäen. Tämä nopeuttaa jatkokehittäjän työtä, koska jatkokehittäjän ei tarvitse opetella alkuperäisen kehittäjän keinoja ja menetelmiä. Standardeja noudattamalla siis vähennetään ohjelmistokehitysprosessin jatkamisen vaatimia resursseja. Lisäksi, koska standardit ja suositukset on jo do-

kumentoitu, ei kehittäjän tarvitse dokumentoida käyttämiään menetelmiä samalla tarkkuudella kuin kehittäjän omaa menetelmää tulisi dokumentoida.

4.4.2 Asiakaslähtöinen ulkoasu

Järjestelmän tulisi aina näyttää siltä, mitä asiakas haluaa. Vaikka ohjelmoijalla voi olla omat näkemyksensä siitä, mikä "näyttää hyvältä", tulee ohjelmoijan tai visuaalisen ulkoasun suunnittelijan aina huomioida käyttäjät, jotka ohjelmaa lopulta tulevat käyttämään. Useimmiten lopullinen käyttöliittymä muodostuu kehitysprosessin aikana asiakkailta saadun palautteen perusteella. Projektissa käyttöliittymää voidaan tietysti muokata, jos ja kun tilaaja antaa palautetta projektin edetessä.

4.4.3 Erityisryhmien huomioiminen

Irmeli Sinkkonen ja muut kertovat kirjassa, (Käytettävyyden psykologia, 2002, luku 6.2) miten ihminen aistii ohjelman. He eivät kirjassa kuitenkaan kerro mitään tilanteesta, jossa jokin aisti saattaa puuttua kokonaan. Ohjelmistosuunnittelussa yleisimmin näköaistin puuttuminen on ainoa erityisryhmä, joka tulee ottaa huomioon. Ja vaikka näköaistinsa menettäneille on olemassa apuvälineitä, jotka mahdollistavat ohjelmistojen käytön, ei suunnittelussa kuitenkaan kannata unohtaa erityisryhmien tarpeita. Kirjallisuudessa on muutenkin puutteita, jos on hakemassa tietoa erityisryhmien tarpeiden huomioimisesta ohjelmistokehityksessä. Näkövamman ollessa kyseessä, erityisesti näköaistin täydellinen puuttuminen kannattaa ottaa huomioon suunnittelussa kahdesta syystä: 1. Jos ohjelmisto suunnitellaan käyttämään selkeitä graafisia osia, joita myös apuvälineet voivat käyttää, johtaa se suoraan selkeään käyttöliittymään ja parempaan käytettävyyteen myös tavallisille käyttäjille. 2. Monimutkaisia graafisia osia sisältävät käyttö-liittymät ovat alttiimpia virheille, kuin yksinkertaisemmat ja paremmin toteutetut käyttöliittymät.

Muutoinkin voidaan sanoa yksinkertaisuuden olevan kaunista. On huomattavasti tehokkaampaa suunnitella yksinkertainen käyttöliittymä hyvin, kuin kuluttaa resursseja hienoon ja raskaaseen käyttöliittymään. Lisäksi monimutkainen käyttöliittymä on vaikeampi lähestyttävä asiakkaille sekä hankalampi kouluttaa uusille käyttäjille.

4.5 Ohjelmointi ja toteutus

Kun kehitysprojektissa on olemassa hyvä projektisuunnitelma on ohjelmoinnissa ja toteutuksessa myös muistettava noudattaa niitä. Laajemmissa projekteissa on myös osattava jakaa projektin työmäärä tasaisesti projektia toteuttaville. Yleensä se on tehty toimivasti jo projektisuunnitelmassa, mutta tarvittaessa toteutusvaiheessa voidaan töitä delegoida muille. Lisäksi työn osat kannattaa jakaa niin, että jokainen toteuttaja saa osakseen, mahdollisuuksien mukaan, alueen, jonka hallitsee parhaiten.Toteuttajien on myös syytä sopia, kuinka kehitettävän ohjelman koodia kirjoitetaan, jotta lähdekoodi saadaan visuaalisesti näyttämään jokseenkin samalta ja yhtä luettavalta.

Kun käytetään ohjelmointikieltä, joka tulee aina kääntää konekielelle ennen kuin ohjelmaa voidaan testata, tulisi aina huomioida, että ohjelmakoodin kääntäminen voi laajoissa projekteissa viedä aikaa, jos koko projekti käännetään kerralla. Siksi projekti kannattaa jakaa osiin, jotka voidaan tarvittaessa kääntää erikseen. Tällöin tarvitsee kääntää vain ne osat, joihin on tehty muutoksia. Tällöin on myös syytä ottaa ajoittain kehitysnäytteitä. (eng: development snapshot) eli kopioita projektin kokonaisuudessa sellaisina aikoina, kun koko projekti on siinä muodossaan ollut käyttökelpoinen. Tällöin projektin edistymistä voidaan esitellä tilaajalle mahdollisen palautteen saamiseksi tai kehitysraportin esittämiseksi.

Kun käytetään komentosarjakieliä kuten PHP:tä, jolla Nurseus -järjestelmä kehitettiin, ei ohjelmakoodia tarvitse kääntää erikseen vaan järjestelmää voidaan käyttää heti, kun kooditiedostot on siirretty testauspalvelimelle. Tällöin

myös järjestelmä on heti esiteltävissä, kun testausympäristö, jossa järjestelmän sen hetkistä versiota testataan, ja kehitysympäristöt, joita ohjelmoijat käyttävät, on eriytetty toisistaan.

4.6 Jatkokehityksen huomiointi ja dokumentaatio

Koska mikään ohjelmisto ei koskaan ole "valmis" eli saavuta tilaa, jossa sitä ei enää tarvitse kehittää, pitää kehitystä ajatella jatkuvana prosessina. Tässä prosessissa ohjelmiston kehittäjät voivat vaihtua. Tästä syystä ohjelmiston kehitys on syytä dokumentoida hyvin, jotta mahdolliset uudet kehittäjät voivat sisäistää ohjelman helpommin, kun heidän ei tarvitse tulkita ohjel-miston lähdekoodia rivi riviltä, saadakseen selville, miten ohjelmisto toimii.

Dokumentaatioon ei kuulu vain ohjelmiston toimintojen kirjaaminen tekniseen dokumentaatioon, vaan myös itse ohjelmakoodin dokumentaatio eli koditiedostoihin "kommentteina" kirjoitetut kuvaukset koodin eri osien toiminnasta. Kommentointi ei tarkoita, että jokaiselle koodiriville tehdään kommentti, joka kertoo mitä rivi tekee, vaan että yhtenäiset koodin osakokonaisuudet eritellään kommentein. Näin tuleva ohjelmoija voi kommentit lukemalla löytää helpommin ne osat, joita on tarkoitus kehittää.

Koodi on myös syytä jättää helposti luettavaan muotoon. Tällä tarkoitetaan sitä, kuinka itse koodi on kirjoitettu. Ohjelmisto toimii vaikka sen lähdekoodi olisi kaikki kirjoitettu yhdelle riville. Tällaista ohjelmaa olisi kuitekin erittäin vaikeaa ja hidasta jatkokehittää, koska koodin lukeminen olisi todella vaikeaa. Kuvien 13 ja 14 koodit tekevät saman asian, samalla nopeudella. Jälkimmäinen on vain kirjoitettu muotoon, jota on helpompi lukea. Tällä tavalla kehittäjät säästävät aikaa, kun lähdekoodia on helpompi lukea.

```
if(var>2){reset(var);log("$timestamp : var overflow");echo "var out of bounds";}
```

Kuva 13 : Esimerkki huonosti luettavasta koodista.

Koodia kirjoitettaessa siis tulee pyrkiä pitämään rivipituus sopivana ja osat eriytettyinä. Asianmukaiset sisennykset ja kommenttirivit ovat siis tarpeen kun koodia työstetään.

```
//If var is greater than allowed:
//log and print error.
if(var>2){
    reset(var);
    log("$timestamp : var overflow");
    echo "var out of bounds";
}
```

Kuva 14 : esimerkki hyvin luettavasta koodista.

Kun ohjelmistoa kehitetään, on myös syytä kirjata ylös, miksi tiettyjä valintoja kehityksessä on tehty. Nämä kirjataan yleisimmin kehitysraporttiin, joka kertoo kehitysprojektin vaiheet. Lisäksi kirjaus on aiheellista tehdä myös tekniseen dokumentaatioon, joka kertoo ohjelmiston toiminnasta yksityiskohtaisesti. Tämä dokumentaatio myös kertoo ohjelmiston järjestelmävaatimukset eli mitä ohjelmisto vaatii toimiakseen. Lisäksi tiedot käytetystä testausympäristöstä on syytä kirjata ylös jatkokehittäjää varten.

4.7 Testaus

Kehitettävää ohjelmistoa on luonnollisesti aina testattava kehityksen yhteydessä, jotta mahdolliset ongelmat voidaan havaita ennen kuin järjestelmä otetaan käyttöön tai luovutetaan asiakkaalle.

Testauksessa on pyrittävä huomioimaan, että ohjelmistoa käytetään asiakkaan organisaatiossa ja asiakkaan laitteilla. Aina ei kuitenkaan voida järjestää testausta varten ympäristöä, joka vastaisi asiakkaan käyttämää järjestelyä.

Ajoittain kuitenkin voidaan asiakaan kanssa sopia, että ohjelmistoa testataan asiakkaan toimintaympäristössä sovittuna aikana ja näin saatu palaute ja testausmateriaali siirretään takaisin ohjelmistoa kehittävälle taholle.

Testauksen aikana on syytä pyrkiä tuottamaan ohjelmistolla kaikki ne tilanteet, jotka saattavat varsinaisen ohjelmiston käytön aikana esiintyä. Ohjelmistoa testattaessa ei tule olettaa, että käyttäjät käyttävät ohjelmistoa oikein, vaan että käyttäjät tekevät useita virheitä ja saattavat niillä aiheuttaa ohjelmiston viallisen toiminnan. Testaamalla virhetilanteet voidaan niiden aiheuttamiin ongelmiin varautua ja mahdollisesti estää niiden tapahtuminen kokonaan.

Inhimillisistä syistä johtuen on mahdotonta nähdä ennalta kaikkia tilanteita, joihin ohjelmisto saattaa joutua. Lisäksi kaikkia nähtyjä tilanteitakaan ei välttämättä voida suuresta määrästä johtuen testata. Tällöin on testauksessa ja sitä myöten kehityksessä pyrittävä löytämään ja korjaamaan yleisimmät ja katastrofaalisimmat virhetilanteet ja samalla kartoittaa kuinka ohjelmisto kykenee palautumaan katastrofaalisista virhetilanteista. Testauksen yhtey-dessä löydetyistä virhetilanteista kaikkia ei aina voida tai kannata korjata, koska niiden toteutumista ei pidetä niin todennäköisenä, että korjaukseen vaaditut resurssit kannattaisi kuluttaa. Tällaiset virheet on kuitenkin syytä kirjata tekniseen dokumentaatioon, jotta niistä on olemassa tieto, jos ja kun tilanne tulee esiin. Näin jatkokehityksessä tiedetään, mistä ongelma johtuu ja siihen voidaan puuttua ilman uutta testausta.

Lopulta myös korjaukset on syytä testata, jotta tiedetään että korjaukset itsessään eivät aiheuta uusia virhetilanteita.

4.8 Asiakaskäyttäjän ja asiakasylläpitäjän materiaalin toimittaminen

Koska järjestelmää ylläpidetään jatkuvasti eikä ohjelmiston toimittaja voi olla aina tavoitettavissa, kun ohjelmiston ylläpitäjällä tai käyttäjillä on kysy-

myksiä, on syytä jättää loppukäyttäjille ja -ylläpitäjille riittävästi materiaalia, jota konsultoimalla suurimmasta osasta kysymyksistä selvitään.

Materiaalia laadittaessa tulee huomioida tilanteet, joissa käyttäjillä voi olla kysyttävää. Peruskäyttäjän ongelmista selvitään useinmiten kirjoittamalla ohjelmistoon omat ohjeet, joita konsultoimalla kokemattomammankin käyttäjän tulisi selvitä ohjelman käytöstä.

Ylläpitäjän materiaali on kuitenkin hankalampaa kirjoittaa, koska sen sisältöön kuuluvat yleisempien ennaltanähtävissä olevien ongelmatilanteiden ratkaisujen kartoitus. Näiden tilanteiden ratkaisut tulee kirjata niin yksityiskohtaisesti kuin on tarpeellista ylläpitäjän kannalta. Dokumentaation tulee olla muotoa, jota sovittu ylläpitäjä kykenee käyttämään.

Nurseus -järjestelmän tapauksessa ohjelmaan sijoitettiin ohjeet tavalliselle käyttäjälle. Ylläpitäjältä kuitenkin odotetaan hieman oma-aloitteisuutta, koska ylläpitäjän ohjeet ovat hyvin suppeat ja ne on kirjoitettu suoraan ylläpitäjän toimintojen yhteyteen eikä omaksi ohjeekseen.

4.9 Projektiviimeistely ja tuotteen luovutus

Kun sovellus lopulta on valmis ja esitelty asiakkaalle, on aika suorittaa projektin viimeistely ennen kuin lopullinen tuote luovutetaan asiakkaalle. Projektiviimeistelyyn kuuluvat laadunvalvonnan onnistumisen tarkistus eli tuote käydään vielä kerran läpi ja varmistetaan, että se vastaa toteuttajan omia laadullisia vaatimuksia lopullisesta tuotteesta. Myös dokumentoinnin tarkistus eli olemassa olevan dokumentaation tarkastaminen ja mahdollinen korjaaminen tarvittaessa, kuuluvat olellisena osana projektin viimeistelyyn. Projektista kirjoitetaan myös loppuraportti, joka kertoo, mitä ja, miten projektilla saavutettiin ja vertaa lopputulosta projektisuunnitelmaan ja arvioi miten projekti on onnistunut, mikä projektissa epäonnistui ja miten vastaavia projekteja voidaan tulevaisuudessa parantaa. Loppuraportti sellaisenaan jää yleensä toteuttajalle, mutta joissain tilanteissa myös asiakkaalle laaditaan oma kappaleen-

sa. Asiakkaan kappale kuitenkin poikkeaa toteuttajan kappa-leesta, koska siinä ei kerrota toteuttajan organisaation sisäisistä asioista.

Tuotetta luovutettaessa pitää huomioida miten laadittu sopimus määrittää tuotteen luovutuksen. Kuten tuleeko asiakkaan saada täydellinen kopio varsinaisesta ohjelmiston lähdekoodista vai jääkö se toteuttajan haltuun. Mikäli toimitus tapahtuu kerta-asennuksena asiakkaalle, on mietittävä tuleeko ohjelmiston asennuspaketti toimittaa asiakkaalle, mikäli sovittu sopimus ei sitä määrittele. Kun taas on kyseessä ohjelmisto, joka saatetaan asentaa useita kertoja on hyvä toimittaa asiakkaalle useampia kopioita ohjelmiston asennustiedostoista, jottei yhden kopion tallessa pitäminen muodostu liian tärkeäksi.

Lopulta myös dokumentaatio (kuten ohjeet ja tekninen dokumentaatio) toimitetaan asiakkaalle sopimuksen määrittelemällä tavalla. On hyvä kuitenkin toimittaa sekä elektroniset, että paperikopiot dokumentaatiosta ja ohjeista.

Web-ohjelmistossa, kuten Nurseus –hoitosuunnitelmajärjestelmässä, lähdekoodia ei käännetä varsinaiseksi ohjelmistoksi, joten lähdekoodi on toimitettu asiakkaalle kokonaisuudessaan. Lisäksi toimitus sisälsi asennuksen asiakkaan ympäristöön. Asennuksesta huolimatta asiakkaalle toimitettiin myös kopio ohjelmistosta. Ohjelmistokopio sisälsi myös täydellisen elektronisen teknisen dokumentaation. Tekninen dokumentaatio toimitettiin myös paperikopiona.

5 KEHITYKSEN ONNISTUMISEN ARVIOINTI

Kuten hyvässä projektissa, on tässäkin syytä tarkastella, miten projekti on onnistunut tämän työn esittämissä avainkohdissa. Tämä arvio on aina toteuttajan itsensä tekemä eikä asiakas ole oikeutettu arvioinnin tulosten näkemiseen. On tietysti toteuttajan valittavissa toimitetaanko arvio asiakkaalle. Myös asiakas suorittaa arvioinnin nähdäkseen vastaako tuote sitä, mitä toteuttajalta on tilattu.

Nurseus –hoitosuunnitelmajärjestelmä on asiakkaan antaman palautteen mukaan noudattanut sille esitettyä vaatimusmäärittelyä juuri siinä muodossa

kuin vaatimusmäärittely on esittänyt. Myös käyttäjiltä saatu palaute on antanut ymmärtää, että ohjelmisto on ollut vaatimusten mukaisesti riittävän helppokäyttöinen. Tietysti joitakin ongelmatilanteita on testauksen aikana ilmennyt, mutta ne on tuoreemman palautteen mukaan onnistuttu korjaa-maan onnistuneesti. Asiakas on myös esittänyt kiitoksen joistakin ominaisuuk-sista, jotka kehittäjä on nähnyt tarpeellisiksi toteuttaa vaikkei niitä ole kirjattu vaatimusmäärittelyyn. Ohjelmisto on siis esitetysti kyetty toteut-tamaan noudattaen periaatetta yksinkertaisuudesta ja käytettävyydestä. Ohjelmisto on myös testattu käyttämällä rajoitteisten käyttäjien apuvälineitä ja saadun palautteen mukaan toiminut moitteetta ja ajoittain ollut yksin-kertaisempi käyttää kuin monet muut ohjelmistot.

Projektin päättämisen jälkeen on asiakas esittänyt kyselyn mahdollisuuksista laajentaa ohjelmiston toimintaa tulevaisuudessa vastaamaan tulossa olevia vaatimuksia hoitoalan kehittyessä. Tässä tapauksessa toteuttaja näkee tehneensä ohjelmiston niin, että esitetty toiminnallisuus on toteutettavissa suhteellisen helposti, kun vaadittujen toimintojen lopullinen muoto varmistuu. Voidaan siis sanoa ohjelmiston noudattavan huomiota jatko-kehityksen huomioimisesta.

Rakenteellisesti ohjelmiston ohjelmakoodi on tarkistettu ja muokattu käyttämään kuvaavia kommentteja osien erottamiseen ja lähdekoodi on pyritty kirjoittamaan mahdollisimman luettavaksi. On kuitenkin nähtävissä, että luettavuudesta on paikoin tingitty, kuitenkin niin, että lähdekoodi on ymmärrettävää, kun osiot, joissa luettavuudesta on kiristetty, on toteutettu aina samalla tavalla. Ohjelmakoodi on myös pystytty jakamaan selkeisiin komponentteihin, jotka eivät ole riippuvaisia toisistaan. Ohjelmiston kehitys useita kehittäjiä ajatellen on siis onnistunut.

Nurseus –hoitosuunnitelmajärjestelmä noudattaa tiukasti olemassa olevia suosituksia ja standardeja eikä ohjelmistoon ole rakennettu osia, joiden tarkoitus on korjata standardien noudattamisen tai noudattamatta jättämisen aiheuttamia ongelmia. Kehityksessä on siis onnistuttu erinomaisen hyvin huomioimaan olemassa olevat standardit ja suositukset.

Resurssien puutteesta johtuen kaikkia Nurseus –hoitosuunnitelmajärejstelmän virhetilanteita ei ole kyetty kartoittamaan tai dokumentoimaan. Yleisimmät virhetilanteet on kuitenkin kyetty estämään eikä ole nähtävissä tilanteita, joissa yksi virhe saattaisi vaarantaa koko ohjelmiston käytön. Voidaan siis sanoa ohjelmiston onnistuneen tässä suhteessa siinä määrin kuin resursseihin nähden on toivottavaa.

Lopullisesta tuloksesta voidaan hyvällä omatunnolla sanoa, että Nurseus – hoitosuunnitelmajärjestelmä on kyetty toteuttamaan tavalla, jota voidaan jatkokehittää helposti ja jota on tulevaisuudessa mahdollista ottaa käyttöön myös muissa hoitoalan oppilaitoksissa. Vaikka ohjelmisto ei esitetyn kehitysprosessiajattelun kannalta ole valmis, voidaan sen sanoa suoriutuvan tehtävästään toistaiseksi toivotulla menestyksellä.

LÄHTEET

Kirjallisuus:

Rantala Ari, 2005, Web-ohjelmointi

Sinkkonen Irmeli, Kuoppala Hannu, Parkkinen Jarmo, Vastamäki Raino : 2002 : Käytettävyyden psykologia

Talentum : 2008 : ATK-sanakirja 1

Tsui Frank, Karam Orlando, 2007, Essentials of software engineering

Valtiovarainministeriö , 2008, Valtiohallinnon tietoturvasanasto : Vahti 8/2008

Wiio Antti , 2004, Käyttäjäystävällisen sovelluksen suunnittelu

Verkkolähteet

w3schools, Browser statistics , [viitattu 28.6.2011]
Saatavissa:
http://www.w3schools.com/browsers/browsers_stats.asp

Ei julkiset lähteet

Leino Sami, Nurseus –hoitosuunnitelmajärjestelmä : käyttäjäpalaute

Leino Sami, Nurseus –hoitosuunnitelmajärjestelmä : projektisuunnitelma

Leino Sami, Nurseus -hoitosuunnitelmajärjestelmä, Tekninen dokumentaatio [viitattu 10.7.2011]

Leino Sami, Nurseus –hoitosuunnitelmajärjestelmä : vaatimusmäärittely

Leino Sami, Stanger Jennifer: Nurseus –hoitosuunnitelmajärjestelmä : apuvälinetestauksen palaute

Leino Sami, Vesaluoma Helena : Nurseus -hoitosuunnitelmajärjestelmä, sähköpostiviestintä ja elektroninen palaute [viitattu 10.7.2011]