Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Sähköisen liiketoiminnan järjestelmät

2010

Titta Korhonen

TOIMEKSIANTOJÄRJESTEL-MÄN MÄÄRITTELY JA TOTEUTUS DIANTI OY:LLE

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Sähköisen liiketoiminnan järjestelmät

Kesäkuu 2010 | Sivumäärä: 46 + 4 liitettä

Ohjaaja: Minna-Kristiina Paakki

Titta Korhonen

TOIMEKSIANTOJÄRJESTELMÄN MÄÄRITTELY JA TOTEUTUS DIANTI OY:LLE

Tämän opinnäytetyön tarkoituksena on kertoa toimeksiantojärjestelmän määrittelystä ja toteutuksesta osaksi Dianti Oy:n asiakkaalle suuntaamaa Focusa-oppilaitosohjelmistoa.

Dianti Oy on pieni web-ohjelmointiyritys, jonka Focusa-tuotteet on toteutettu Oraclen tietokantaan ja sovelluskehitysvälineenä yritys käyttää Oraclen Application Expressiä. Toimeksiantojärjestelmä toteutettiin vaatimusmäärittelyn perusteella Dianti Oy:n olemassa olevaan tietokantaan yrityksen käyttämin välinein.

Toimeksiannolla tarkoitetaan tässä työssä määriteltyä tehtävää tai työtä, jolla on toimeksiantaja ja tekijä, status ja prioriteetti. Toimeksiantojärjestelmän tarkoituksena on listata oppilaitoksessa tehtävät työt työjonoon ja helpottaa töiden valmistumisen ja määrän seurantaa tekijäkohtaisesti. Järjestelmä koostuu interaktiivisesta raportista, lomakkeesta ja kalenterisivusta. Raportissa tietueita voidaan suodattaa ja etsiä käyttäjän antamien kriteerien mukaan. Lomakkeen avulla toimeksiantoja lisätään, muokataan ja poistetaan. Kalenterisivun kuukausinäkymässä toimeksiantoja voidaan katsoa tekijäkohtaisesti.

Työn alussa kerrotaan ohjelmistotuotannon peruskäsitteistä, kuten vaihejakomalleista, sekä tukitoiminnoista. Ohjelmistokehitysprosessit voidaan jakaa vaiheisiin, joita voidaan painottaa eri lailla käytettävän menetelmän ja mallin mukaan. Tukitoiminnoilla varmistetaan ohjelmistoprojektin onnistuminen ja ne käsittävät mm. vaiheiden dokumentoinnin ja laadunvarmistustoimet. Ennen toimeksiantojärjestelmän määrittely- ja toteutusvaiheiden selvitystä, kerrotaan lyhyesti Application Express –sovelluskehitysvälineestä, jolla järjestelmä toteutettiin.

Toimeksiantojärjestelmän toteutus vastaa hyvin vaatimusmäärittelyä ja se toimii sulavasti ja varmatoimisesti. Palaute on ollut positiivista. Oppilaitosasiakas ei ole vielä käyttänyt järjestelmää, joten palautetta järjestelmän käytettävyydestä ei vielä ole olemassa.

ASIASANAT:

Ohjelmistotuotanto, Järjestelmäkehitys, Oracle Application Express, Apex

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology | e-Business Systems

June 2010 | Total number of pages: 46 + 4 appendices

Instructor: Minna-Kristiina Paakki

Titta Korhonen

SPECIFYING AND IMPLEMENTING AN ASSIGNMENT SYSTEM FOR DIANTI OY

The purpose of this thesis is to describe the requirement specification and implementation phases of an assignment system for Dianti Oy. The assignment system will later be attached to Dianti's Focusa product, which is an application directed to educational institutions like adult educational centers.

Dianti is a small company, which produces Oracle based Focusa web applications. Focusa applications are implemented with Oracle Application Express web application development tool. Dianti's existing database was used and modified in the implementation of the assignment system and the system was implemented with the development tool used in Dianti.

Here an assignment refers to an assigned task, which has a client and a realizer, a status and a priority. The purpose of the assignment system is to list assigned tasks in a queue and make it easier to follow the progress of status of the tasks' and the amount of the task of a realizer. The assignment system consists of an interactive report, a form and a calendar page. With the interactive report the user can filter and search records by giving parameters. With the form, the user can add, modify and delete tasks. In the calendar page, the user can select whose tasks (s)he wants to see in a monthly calendar view.

At the beginning the basic concepts of the software production like software development models and auxiliary activities are discussed. The software development processes are divided in phases, which can be weighted differently depending on the method or the model used. The auxiliary activities ensure the success of the development project and it consist for instance of the documentation of the development phases and the quality assurance issues. In addition to the requirement specification and implementation phases of an assignment system, Oracle Application Express, the development tool is also briefly introduced.

The assignment system corresponds well to the requirement specification and it functions accurately and smoothly. The feedback from Dianti has been positive. Since any of Dianti's clients has not used the system yet there is no feedback concerning the usability.

KEYWORDS:

Software Production, System Development, Oracle Application Express, Apex

SISÄLTÖ

KUVAT

KUVIOT

LIITTEET

1 JOHDANTO

Dianti Oy on pieni web-ohjelmointialan yritys, jossa työskentelee kolme vakinaista työntekijää ja lisäksi opiskelijaharjoittelijoita. Työntekijöillä on mahdollisuus työskennellä yrityksen tiloissa tai etätyönä Internetin kautta. Diantin asiakkaille suuntaamat Focusa-ohjelmistot ovat oppilaitosohjelmisto ja julkiselle hallinnolle suunnattu vuokrausjärjestelmä. Lisäksi yritys tekee asiakkaille tilauksesta Internet-kotisivuja. Yritys on Oracle-ohjelmistotalon yhteistyökumppani ja sovelluksissa käytetään Oraclen tietokantaa ja sovelluskehitysvälineenä on Oraclen Application Express, eli Apex.

Yrityksen asiakkaille suuntaamat Focusa-ohjelmistot on rakennettu moduuleista eli osioista, joita yhdistelemällä saadaan koottua jokaisen asiakkaan tarpeita vastaava ohjelmisto. Focusa-oppilaitosohjelmiston asiakkuus-osion keskeisiä asioita ovat oppilaitoksen asiakkaiden tietojen tallentaminen, yrityksen ja asiakkaan välisen vuorovaikutuksen kirjaaminen sekä markkinointitoimen suunnittelun ja toteuttamisen kirjaaminen. Tehtävänäni on määritellä ja toteuttaa Diantin asiakkaille tarjoaman Focusa-oppilaitosohjelmiston asiakkuus-osioon toimeksiantojärjestelmä Dianti Oy:n käyttämin välinein.

Tässä työssä toimeksiannolla tarkoitetaan määriteltyä tehtävää tai työtä, jolla on toimeksiantaja ja tekijä. Toimeksiantojärjestelmän tarkoituksena on voida tallentaa oppilaitoksessa henkilöiden, opintoalojen ja tiimien toteutettaviksi suunnitellut tehtävät, eli toimia tehtävien töiden työjonolistana. Järjestelmän avulla on tarkoitus voida seurata mm. tehtävien tilaa, eli miten työt edistyvät ja minkälaiset tehtävät työllistävät eniten. Tavoitteena on myös helpottaa tehtävien jakamista niin, että yksittäisiä resursseja ei kuormiteta liikaa. Tein

toimeksiantojärjestelmän Dianti Oy:n Focusa-oppilaitosohjelmistoa käyttävän asiakkaan termejä käyttäen.

Tehtävänäni oli toteuttaa toimeksiantojärjestelmä, joka sisälsi lomakesivun, jonka avulla käyttäjä voi lisätä, muokata ja poistaa toimeksiantoja sekä raportti- ja kalenterisivun, joiden avulla käyttäjä voi seurata toimeksiantojen määrää ja tilaa tekijäkohtaisesti. Järjestelmän käyttäjiä varten kirjoitin sivu- ja kenttäkohtaiset käyttöohjeet. Dianti Oy:n henkilökunta huolehtii sekä järjestelmän liittämisestä Focusa oppilaitosohjelmiston asiakkuus-osioon että sen käyttöönotosta ja ylläpidosta.

Kerron tässä työssä Diantin asiakkaille tarjoaman Focusa-oppilaitosohjelmiston asiakkuus-osion osaksi tekemäni toimeksiantojärjestelmän määrittely- ja toteutusvaiheista. Kerron myös ohjelmistotuotantoon liittyvistä peruskäsitteistä, koska tein työn ohjelmistoalan yritykselle ja Oraclen Application Expressistä (Apex), jolla toteutin järjestelmän. Lopuksi pohdin oman työni onnistumista.

2 MÄÄRITELMÄT JA KÄSITTEET

Toimeksianto tarkoittaa tässä työssä yksittäistä työtä tai tehtävää. Focusa-oppilaitosohjelmistoon kirjautunut oppilaitoksen työntekijä voi osoittaa toimeksiannon oppilaitoksen työntekijän, opintoalan tai tiimin tehtäväksi. Henkilö voi osoittaa toimeksiannon tarvittaessa myös itselleen.

Toimeksiantojärjestelmän avulla Diantin asiakasoppilaitoksen Focusa-oppilaitosohjelmistoon kirjautuneen henkilön on mahdollista tallentaa, muokata ja poistaa toimeksiantoja sekä seurata esimerkiksi niiden määrää ja tilaa.

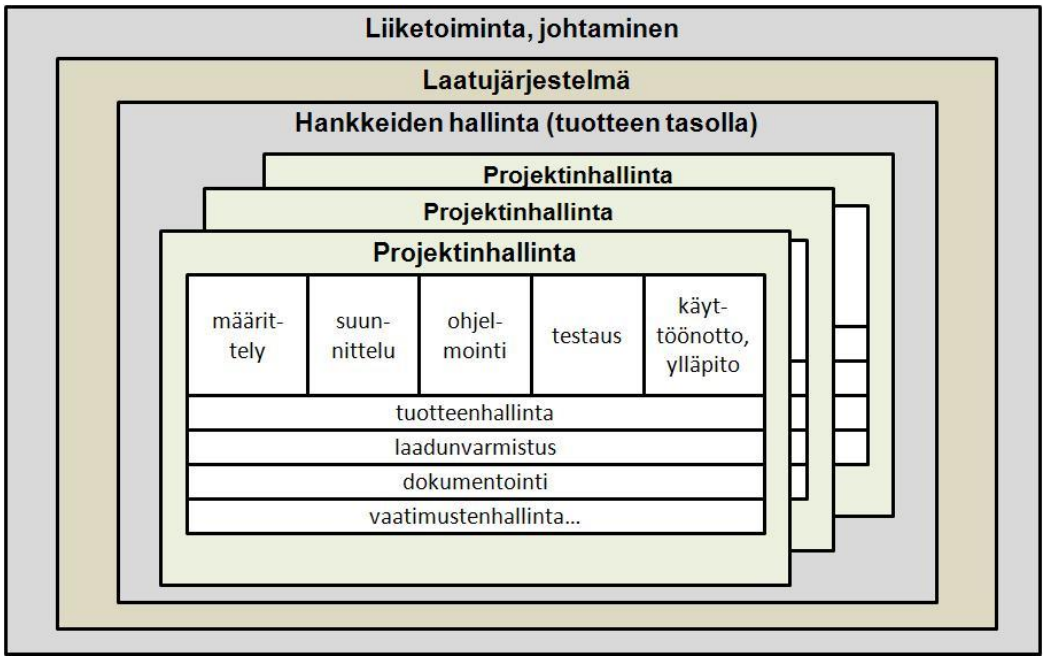
Määrittely tarkoittaa asiakasvaatimusten analysointia. Asiakasvaatimusten pohjalta määritellään ohjelmistolle asetettavat vaatimukset. Määrittely vastaa kysymykseen: mitä järjestelmä tekee? (Haikala & Märijärvi 2002, 38.)

Ohjelmistokehitystyön termi **toteutus** pitää yleisnimityksenä sisällään kaikki määrittelyvaiheen jälkeen suoritettavat ohjelmistokehityksen vaiheet (Haikala & Märijärvi 2002, 41). Tämän työn otsikossa ja johdannossa tarkoitan juuri tätä yleiskäsitettä, myöhemmin työssä toteutus voi tarkoittaa rajatumpaa ohjelmistokehitystyön vaihejaon mukaista käsitettä. Asiayhteydestä, jossa käytän termiä, käy ilmi kumpaa edellä kerrottua määritettä tarkoitan.

3 OHJELMISTOTUOTANNON OSAT

Koska toteutin toimeksiantojärjestelmän ohjelmistoalan yritykselle, kerron tässä luvussa ohjelmistotuotannon keskeisistä osa-alueista, lähinnä ohjelmistokehitysprosessin vaiheista ja projektinhallinnan tukitoiminnoista.

Haikala & Märijärvi (2002, 35) esittävät ohjelmistotuotannon jakautuvan osa-alueisiin kuvion 1 mukaisesti. Laatujärjestelmä ohjaa yrityksen tuotantoa ja määrittelee yrityksen toimintatavat ja toimintaprosessit. Hankkeiden hallinta (tuotteenhallinta) kokoaa yhteen samaan laajaan tuoteprojektiin kuuluvat osaprojektit, joista jokainen voidaan jakaa vaiheisiin. Projektin tukitoimintoihin kuuluvat mm. tuotteenhallinta, laadunvarmistus ja dokumentointi. Tukitoiminnot kestävät koko projektin ja ohjelmiston elinkaaren ajan, eli ajan, joka alkaa ohjelmiston kehittämisen aloittamisesta ja päättyy sen poistamiseen käytöstä. (Haikala & Märijärvi 2002, 35 – 36.)



Kuvio 1. Ohjelmistotuotannon osa-alueet (Haikala & Märijärvi 2002, 35).

3.1 Ohjelmiston laatu

Ohjelmiston käyttäjä muodostaa mielikuvan ohjelmointiyrityksestä tuotteen käytettävyyden ja visuaalisuuden mukaan. Tuotteen tulisi viestiä luotettavuudesta, laadukkuudesta ja kokemuksesta. Mikäli tuote koetaan hyvin tehdyksi, ohjelmistoyritys koetaan laatutietoisena ja lahjakkaana tekemään työtä alallaan. (Sinkkonen ym. 2009, 28, 250.)

Hyvälaatuinen ohjelmisto täyttää käyttäjän sille asettamat kohtuulliset toiveet ja odotukset. Tuotteen laatuun voidaan parhaiten vaikuttaa toiminnan laatua kehittämällä. Yrityksen tuotantoa ohjaavat laatukäsikirjat ja ohjeistukset kootaan laatujärjestelmään, jonka tavoitteena on tuottaa suunniteltua laatutasoa olevia tuotteita aikataulun ja budjetin mukaan. Laatujärjestelmän mukaisesta toiminnasta tuotetaan kirjallisia todisteita, esim. pöytäkirjoja, raportteja ja laadunmittaustietoja. (Haikala & Märijärvi 2002, 48.)

Laadun varmistaminen tarkoittaa tuotteen ja toiminnan laadun mittaamista erilaisilla menetelmillä. Toiminnan laatua voidaan arvioida esimerkiksi auditoimalla, jolloin järjestelmää tai sen osia tarkistetaan, noudatetaanko niissä laatujärjestelmän mukaisia toimintatapoja ja ohjeistuksia. Tuotteen laadunvarmistuksen tavoitteena on edesauttaa mahdollisimman virheettömän ohjelmiston tuottaminen ja auttaa löytämään mahdolliset virheet testaamalla mahdollisimman aikaisessa vaiheessa. Testaaminen mielletään usein yhdeksi prosessivaiheeksi, eikä varsinaiseksi laadunvarmistustoiminnoksi. (Haikala & Märijärvi 2002, 49.)

Yritys voi hakea laatujärjestelmälleen sertifikaattia ja osoittaa näin noudattavansa standardoitua järjestelmää. Standardointi ei todista

laatujärjestelmän erinomaisuutta, vaan kertoo yrityksen noudattavan sitä. (Haikala & Märijärvi 2002, 49.)

Oman työni toteutuksessa noudatin Diantin toimintatapoja ja sääntöjä. Toimeksiantojärjestelmä noudattaa Diantin asiakkaille tarjoaman Focusa-oppilaitosohjelmiston ulkoasua ja logiikkaa. Toteutin työn vaatimusmäärittelyn mukaiseksi ja koko toteutusvaiheen ajan testasin järjestelmän toimintaa varmistaakseni sen toimivuuden.

3.2 Dokumentointi

Dokumentointi on olennainen osa ohjelmistotyötä. Se sisältää tiedot, joilla ohjelmistoa voidaan tehokkaasti ja onnistuneesti kehittää, käyttää ja ylläpitää. Käytännössä monet yritykset kokevat dokumentoinnin vievän liikaa aikaa ja jättävät sen liian vähälle huomiolle. Usein dokumentointityötä tehdään jälkikäteen, jotta asiakkaalle voidaan toimittaa sovitut dokumentit. (Bayer & Muthig 2006.)

Ohjelmistoprosessin dokumentoinnin laajuus riippuu prosessin laajuudesta ja siitä, kuinka monimutkaisesta järjestelmästä on kyse. Tuotteen perusdokumentaatioon kuuluvat vähimmillään määrittely-, suunnittelu- ja testausdokumentit. Määrittely- ja suunnitteludokumenttien puuttuminen voi johtaa myöhemmin jopa koko tuotteen uudelleenkoodaukseen. (Haikala & Märijärvi 2002, 51,70 - 71.)

Dokumentoinnin korkea laatu riippuu siitä, mitkä ovat sille asetetut vaatimukset sekä mihin tarkoitukseen dokumentointia halutaan käyttää. Dokumentointia tehdessä tulee miettiä ketä varten sitä tehdään, eli ketkä ovat dokumentin lukijoita ja käyttäjiä. Dokumentoinnin yhdenmukaistamiseksi ja helpottamiseksi yrityksessä olisi hyvä olla valmiit dokumenttimallit, jolloin kokematonkin tekijä

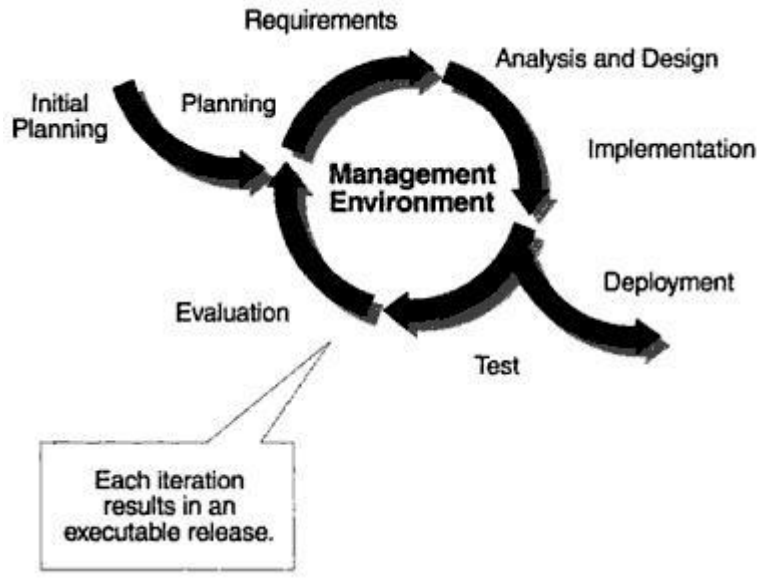
saa niistä tukea. Dokumentointi koostuu yleensä dokumentin laatijoiden tiedoista, muutoshistorian tiedoista, havainnollisista kaavioista ja niitä selventävistä teksteistä. (Bayer & Muthig 2006.)

Dokumentoin Diantin asiakkaalle tarjoaman Focusa-oppilaitosohjelmiston osaksi tekemäni toimeksiantojärjestelmän vaatimusmäärittelyn, suunnitteluvaiheen, jota päivitin toteutusta vastaavaksi ja testitapaukset. Koska Diantissa ei ole käytössä dokumentointimalleja, sovelsin eri lähteistä löytämiäni ohjeita riittävän dokumentoinnin tekemiseksi.

3.3 Vaihejakomalleja

Ohjelmiston kehitysprosessit jaetaan yleensä vaiheisiin alkaen esitutkimuksesta ja päättyen ohjelmiston käyttöönottoon ja ylläpitoon. Vaihejakomallit auttavat prosessin kokonaisuuden ja etenemisen hahmottamisessa sekä dokumentoinnin systemaattisessa tekemisessä. Riippuen käytettävästä mallista, painotetaan joitakin suunnittelu- ja toteutusvaiheita sekä vaiheiden suhteita toisiinsa nähden (Sinkkonen ym. 2009, 41). Seuraavaksi kerron malleista ja menetelmistä, joista löydän yhteneväisyyksiä oman työni tekemisessä. Lopuksi kuvaan mallin, joka kuvaa tekemäni toimeksiantojärjestelmän kehitysprosessia.

Vaihejakomallit voidaan jakaa karkeasti kahteen ryhmään, lineaarisesti etenevään vesiputousmalliin ja evolutionäärisiin malleihin, joissa menetelmät ovat iteratiivisia ja inkrementaalisia. Iteratiivisuus tarkoittaa, että vaiheita toistetaan, minkä myötä muutoksia ja lisäyksiä tehdään tarvittaessa. Inkrementaalisuus tarkoittaa, että alussa tehdään ydinjärjestelmä, johon tehdään lisäyksiä iteratiivisten kierrosten myötä. Kuviossa 2 esitetään iteratiivisen ja inkrementaalisen prosessin vaiheiden sykli, jota toistetaan, kunnes lopputulos vastaa määritettyjä vaatimuksia.

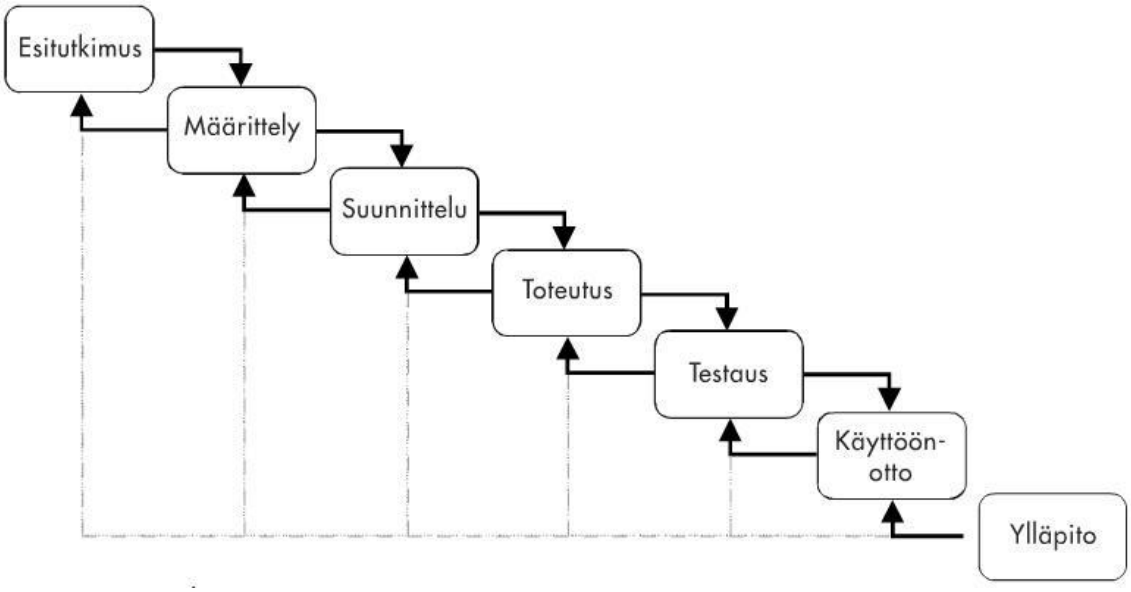


Kuvio 2. Iteratiivinen ja inkrementaalinen prosessi (Kruchten 2004, 7).

3.3.1 Vesiputousmalli

Vesiputousmalli kuvaa perinteistä suunnittelua, jossa prosessi on jaettu kuvion 3 mukaisiin vaiheisiin. Vaiheet suoritetaan järjestyksessä ylhäältä alaspäin kuvion 3 osoittamalla tavalla ja jokainen vaihe dokumentoidaan. Uuden vaiheen työstäminen aloitetaan vasta, kun edellinen on tehty kokonaan valmiiksi. Edellisen vaiheen dokumentti ohjaa uutta vaihetta.

Vesiputousmalli kuvaa iteratiivisuuden huonosti, mikä tekee edellisten vaiheiden puutteiden ja virheiden korjaamisesta työlästä, sillä muutokset aiheuttavat usein kaikkien edeltävien vaiheiden uusimista. (Pohjonen 2002, 40.) Käytännössä edellisiin vaiheisiin pitää aina palata, mitä kuvaavat kuviossa 3 olevat ylävirtaan osoittavat nuolet. Malli soveltuu käytettäväksi parhaiten silloin, kun tavoitteet ovat selkeät ja kehitystiimi on pieni. (Sinkkonen ym. 2009, 41.)



Kuvio 3. Vesiputousmalli (Pohjonen 2002, 40).

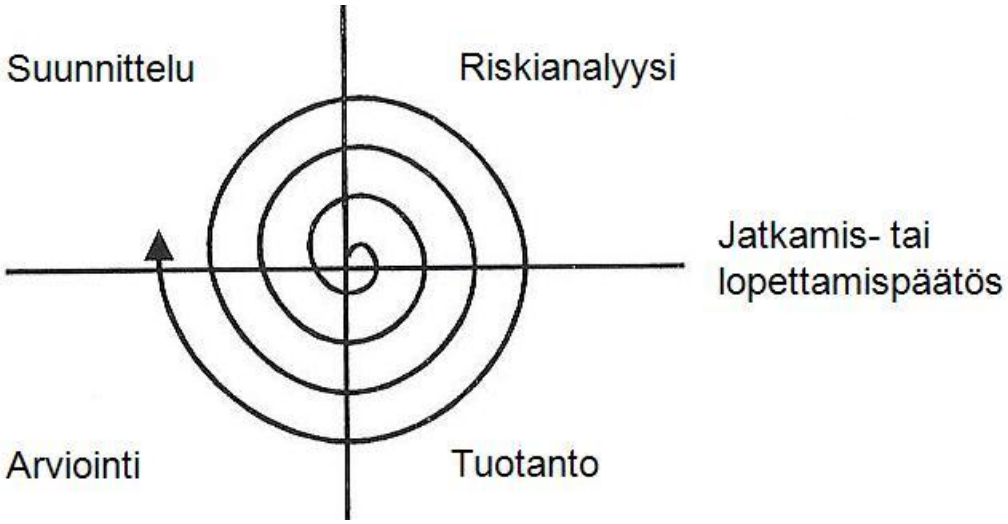
Diantin asiakkaalle tarjoaman Focusa-oppilaitosohjelmiston asiakkuus-osion osaksi tekemäni toimeksiantojärjestelmän kehitysprosessi oli melko suoraviivaisesti etenevä, eikä vaatimusmäärittelyyn tullut muutoksia työn kuluessa. Työn olisi voinut toteuttaa vesiputousmallia käyttäen. Koen vesiputousmallin mukaisen kehitystyön suunnitteluvaiheen haasteelliseksi, sillä koko järjestelmän toiminta tulisi hahmottaa tarkasti etukäteen ja osata ennalta varautua mahdollisiin ongelmiin.

3.3.2 Spiraalimalli

Spiraalimallissa edetään spiraalimaisesti suunnittelu-, riskianalyysi-, tuotanto- ja arviointivaiheita toistaen, kunnes järjestelmä on valmis. Kuviossa 4 esitetyn spiraalimallin mukaan toteutetussa kehitysprosessissa toteutettava tuote on sitä valmiimpi ja yksityiskohtaisempi, mitä kauemmas spiraalin keskipisteestä edetään. (Pohjonen 2002, 43.)

Mallissa painotetaan riskien analysointivaihetta ja mikäli riskit ovat liian suuria, voidaan prosessi keskeyttää. Riskianalyysin tavoitteena on minimoida riskit

seuraavalla iteraatiokierroksella. (Pohjonen 2002, 43.) Riskeiksi voidaan määritellä asiat tai tilanteet, jotka uhkaavat järjestelmälle tai sen kehitystyölle asetettuja tavoitteita (Pohjonen 2002, 80).



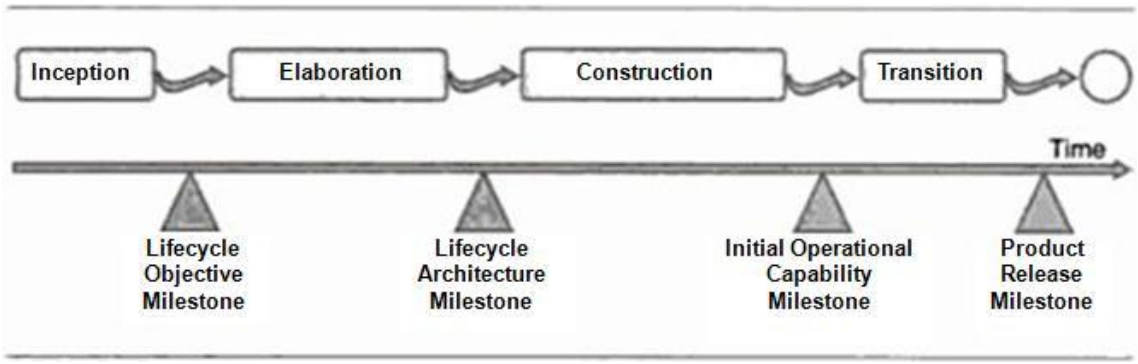
Kuvio 4. Spiraalimalli (Pohjonen 2002, 42).

Tekemäni toimeksiantojärjestelmän suunnittelu- ja toteutusvaiheessa arvioin jatkuvasti riskejä, joita lisättävä uusi toiminto mahdollisesti toi tullessaan. Pyrin arvioimaan kaikki tilanteet, joihin käyttäjä saattaisi joutua järjestelmää käyttäessään ja estämään hämmentävät ja virheelliset tapahtumat. Lisäsin järjestelmään uusia toimintoja sykleissä ja jokaisen syklin jälkeen olin lähempänä valmista tuotetta.

3.3.3 RUP-malli

RUP-malli, eli Rational Unified Process, perustuu olio-tekniikalle ja sen vaiheiden kuvailemiseen ja dokumentointiin käytetään UML-mallinnuskieltä (Sinkkonen ym. 2009, 42). Prosessin neljä päävaihetta ovat kuviossa 5 kuvatut aloitusvaihe (Inception), tarkennusvaihe (Elaboration), toteutusvaihe (Construction) ja luovutusvaihe (Transition). Jokainen vaihe koostuu yhdestä tai useammasta iteraatiosta, joista jokainen pitää sisällään määrittely-, suunnittelu-,

toteutus- ja testausvaiheet. (Fowler & Scott 2002, 15.) Kuviossa 5 näkyvät myös virstanpylväät, jotka saavutetaan, kun päävaiheen katsotaan olevan valmis ja voidaan siirtyä seuraavaan päävaiheeseen.

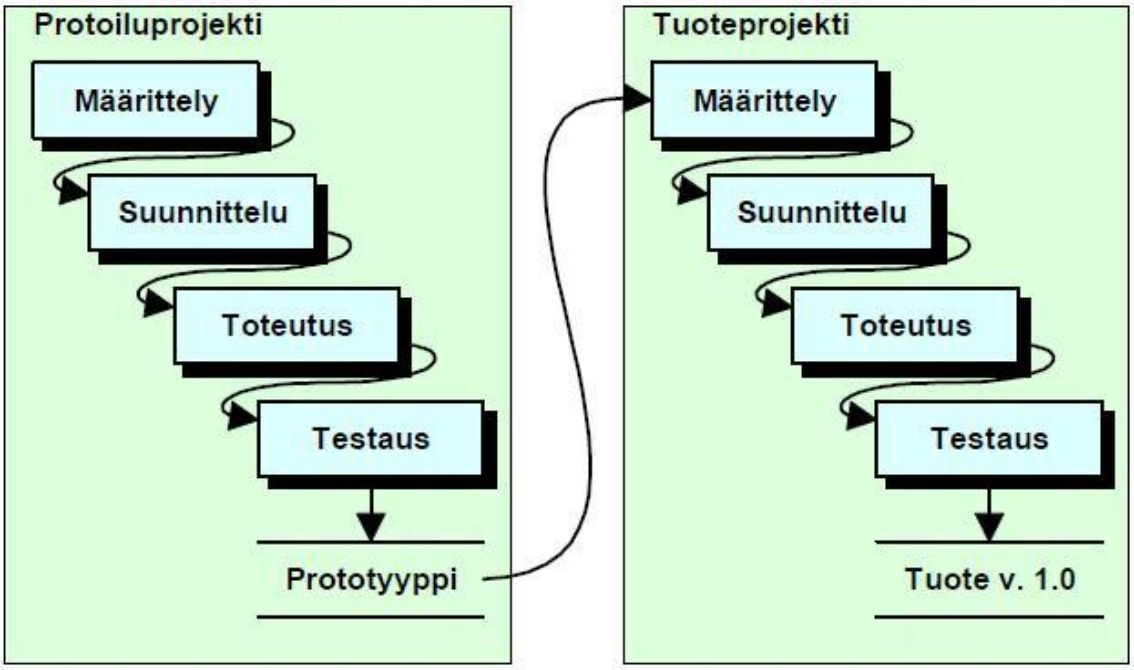


Kuvio 5. RUP-mallin vaiheet ja virstanpylväät (Kroll & Kruchten 2004, 12).

Käytin Diantin Focusa-oppilaitosohjelmiston osaksi tekemäni toimeksiantojärjestelmän suunnitteluvaiheessa toimintojen kuvaamiseen UML-mallinnuskieltä sekä iteroin suunnittelu-, toteutus- ja testausvaiheita, kunnes järjestelmä oli valmis.

3.3.4 Protoilumalli

Protoilumallissa pyritään tekemään asiakasvaatimusten perusteella mahdollisimman nopeasti ohjelmiston suunnittelu ja toteutus. Näin asiakas saa nopeasti arvioitavakseen prototyypin, jonka mukaan voidaan arvioida vastaavatko asetetut vaatimukset todellisia tarpeita. Protoiluprojektin jälkeen tehdään lopullinen tuote, jonka valmistaminen pitää sisällään määrittely-, suunnittelu-, toteutus- ja testausvaiheen. Kuviossa 6 on havainnollistettu prototyypin ja tuotteen kehitysprosessit protoilumallin mukaisesti.



Kuvio 6. Esimerkki protoilumallista (Haikala & Märijärvi 2002, 45).

Dianti Oy:n toimitusjohtaja oli aiemmin tehnyt omaan käyttöönsä ratkaisun, jonka voi katsoa olevan Focusa-oppilaitosohjelmiston asiakkuus-osion osaksi tekemäni toimeksiantojärjestelmän prototyyppi.

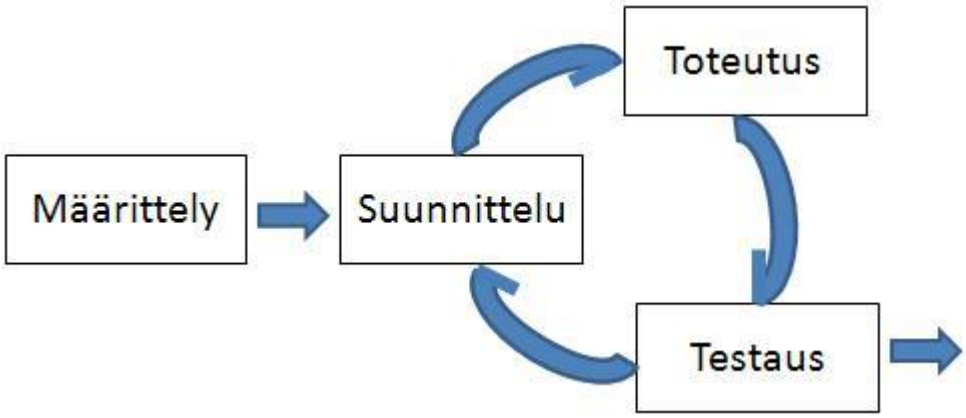
3.3.5 Ketterät menetelmät

Ketteriä (Agile), eli kevyitä menetelmiä on useita. Menetelmälle ominaisimpia piirteitä ovat ohjelmistokehityksen jakaminen lyhyisiin iteraatioihin, joiden tavoitteena on julkaisukelpoinen tuotos. Kehitysprosessi viedään läpi pari- ja tiimityöskentelynä. Ketterän kehityksen keskeiset ajatukset on koottu kannattajiensa tekemään julkaisuun, Agile Manifestoon, joka löytyy osoitteesta http://agilemanifesto.org/. Manifestin keskeinen sisältö: kommunikointi kasvokkain tiimin sisällä ja asiakkaan kanssa on ensiarvoisen tärkeää. Suunnittelutyötä tehdään koko ohjelmistokehitystyön ajan ja muutoksia tehdään, kunnes saavutetaan toimiva sovellus. (Manifesto for Agile Software Development.)

Dianti Oy:n henkilökunta oli minulle sekä tiimi että asiakas. Heidän kanssaan oli helppo työskennellä ja kommunikoida. Varsinkin vaatimusmäärittelyä tehdessä keskustelimme ja ideoimme tiiviisti, jotta järjestelmälle asetetut odotukset eivät jääneet epäselviksi. Työn toteutuksen aikaan tapasimme harvoin, mutta kommunikoimme tarvittaessa sähköpostitse tai puhelimitse.

3.3.6 Toimeksiantojärjestelmän kehitysprosessin vaihejakomalli

Kuten edellä kerroin vaihejakomallien esittelyssä, Diantin Focusa-oppilaitosohjelmiston asiakkuus-osioon tekemäni toimeksiantojärjestelmän kehitysprosessissa on monien määriteltyjen mallien piirteitä. Vaatimusmäärittelyvaiheen jälkeen iteroin suunnittelu-, toteutus- ja testausvaiheita kuvion 7 mukaisesti, kunnes järjestelmä täytti vaatimusmäärittelyn asettamat odotukset. Vaatimusmäärittely tarkentui prosessin aikana, kun keskustelimme Diantin toimitusjohtajan kanssa asioista, jotka vaativat tarkennusta. Osaan iteraatiokierroksista voi siis liittää myös määrittelyvaiheen.



Kuvio 7. Toimeksiantojärjestelmän kehitysprosessin vaihejakomalli.

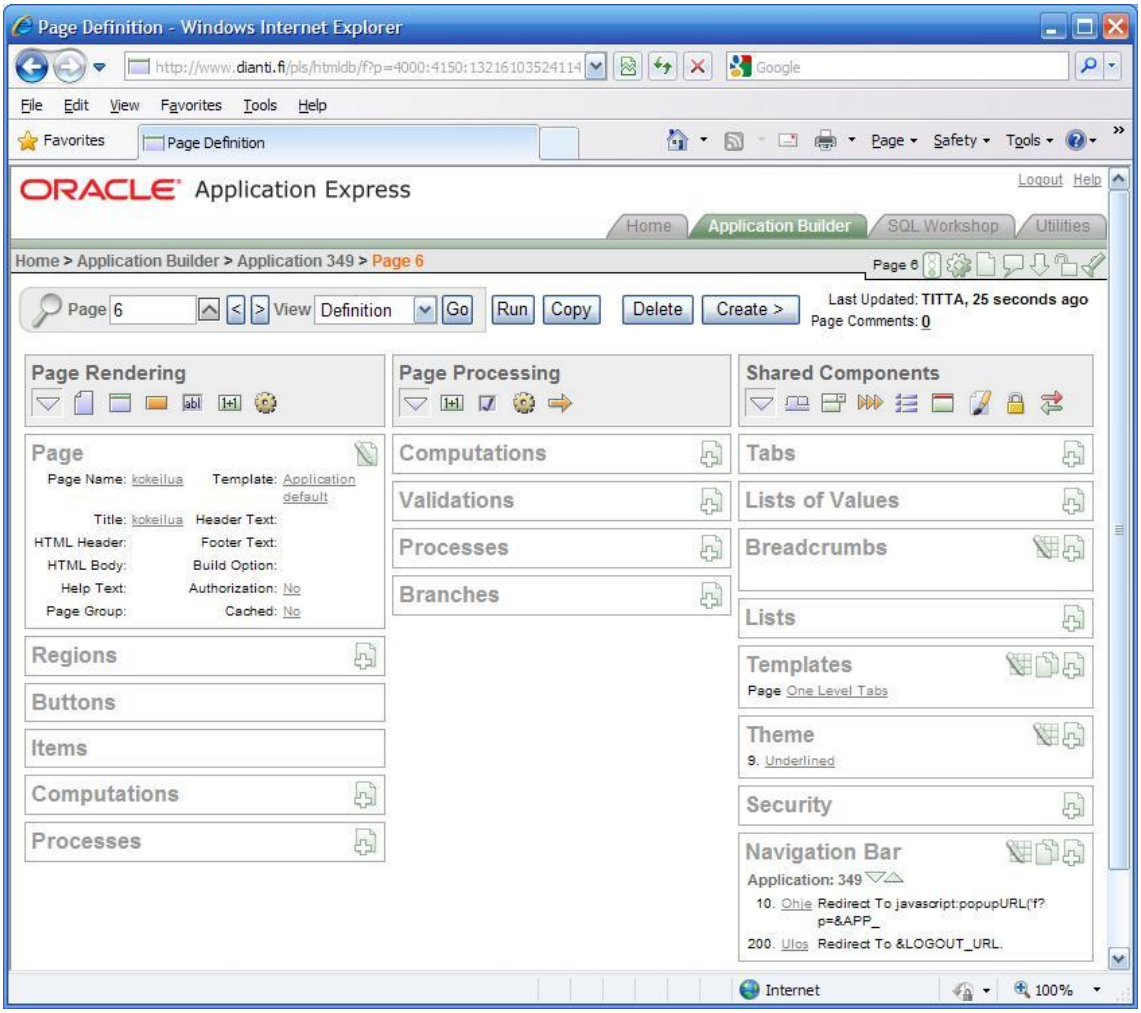
4 APPLICATION EXPRESS –SOVELLUSKEHITIN

Toteutin toimeksiantojärjestelmän Dianti Oy käyttämin välinein, eli sovelluskehittimenä oli Oraclen Application Express, jota kutsutaan lyhyesti nimellä Apex. Oraclen mukaan Apex on Oraclen tietokantaan integroitu web-pohjainen sovelluskehitysväline, jonka avulla voidaan melko helposti luoda nopea, monipuolinen ja turvallinen sovellus. Ainoina vaatimuksina välineen käyttöön ovat Internet-yhteys ja hieman ohjelmointitaitoja. Ohjelmointikielinä ovat PL/SQL ja JavaScript. (Oracle 2010.)

Apexin Application Builderiä käyttäen kehittäjä voi luoda uusia sovelluksia ja sovelluksen sisällöksi erilaisia sivuja. Luonti tapahtuu ohjatusti, joten kehittäjän tehtävänä on valita kussakin vaiheessa tarjolla olevista vaihtoehdoista haluamansa. SQL Workshopissa kehittäjä voi mm. lisätä, selata ja muokata objekteja, eli tauluja ja nimettyjä tietokantatoimintoja, tehdä SQL-kyselyitä ja ajaa skriptejä.

4.1 Sivunmärityksen osiot Apexissa

Kun sovelluksen sivu on luotu, kehittäjä muokkaa sivua ja sen toimintoja määrittelynäkymässä (Page Definition), joka on jaettu kolmeen osaan. Kuvassa 1 sivun vasemmalla näkyvässä osiossa (Page Rendering) määritellään, mitä tapahtuu, kun sivu ladataan selaimelle. Osiossa on eritelty sivun tiedot (Page), sivun fyysiset alueet (Regions), painikkeet (Buttons), sivun käyttämät kentät (Items), sekä latauksen yhteydessä tehtävät laskutoimitukset (Computations) ja prosessit (Processes).



Kuva 1. Application Expressin Page Definition –näkymä.

Kuvan 1 keskimmäisen osion (Page Processing) osuus käsitellään, kun sivua prosessoidaan, eli käyttäjä esimerkiksi tallentaa lomakkeen tiedot. Ensin tehdään mahdolliset laskutoimitukset (Computations), annetun tiedon oikeellisuuden tarkistukset (Validations), suoritetaan prosessit (Processes) ja viimeiseksi navigoidaan (Branches) määritetylle sivulle.

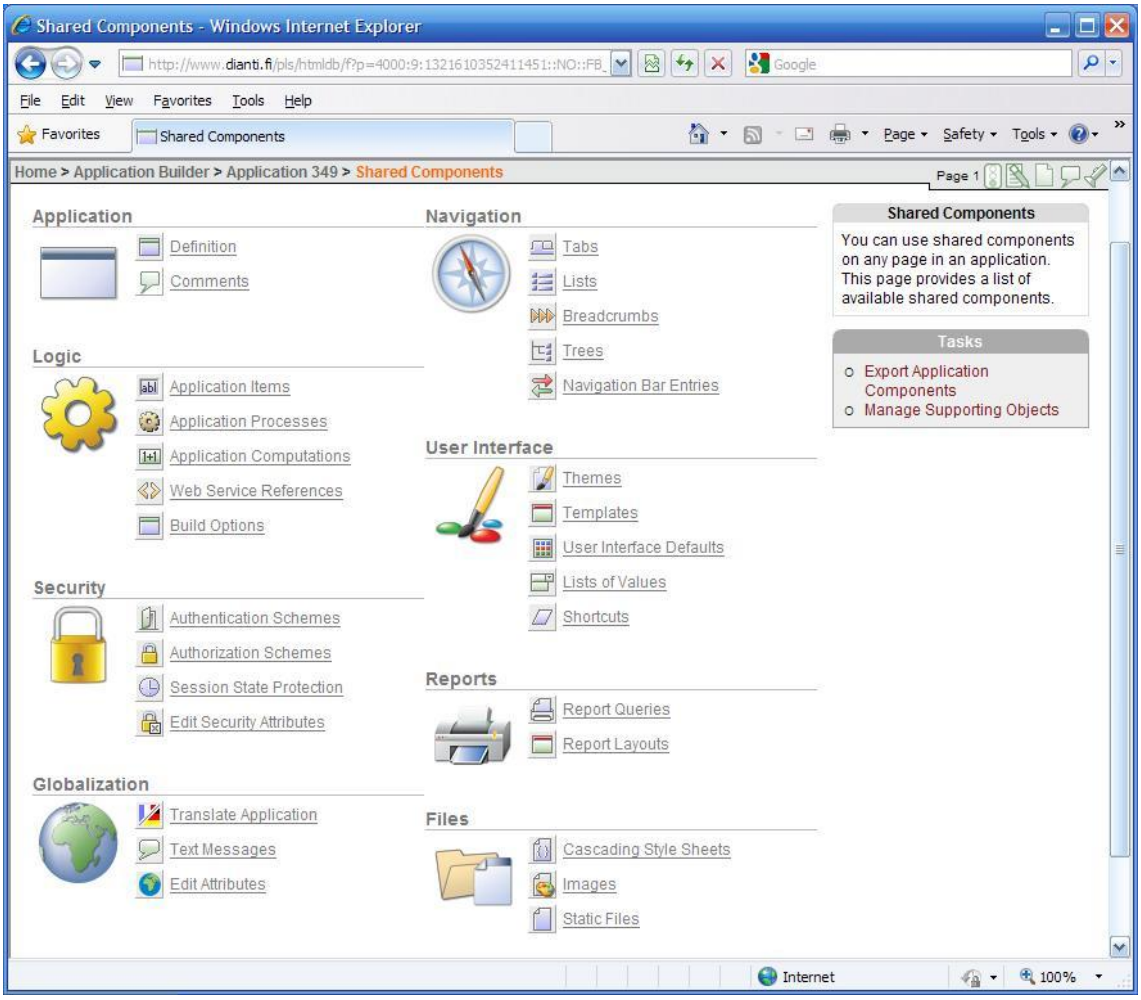
Sovelluksen yhteisessä käytössä olevien jaettujen komponenttien (Shared Components) osuus näkyy kuvan 1 oikeassa osiossa. Sovellukselle voidaan määritellä minkälaisessa muodossa välilehdet (Tabs) näytetään. Valmis

arvolista (List of Values) voidaan määritellä tarvittaessa esim. alasvetovalikon käytettäväksi. Sovellukselle voidaan määrittää hierarkkinen navigointi- tai sivupolku (Breadcrumbs), sekä haluttaessa voidaan määrittää linkkilista (Lists) navigointia varten. Ulkoasumalleja (Templates) voidaan määritellä eri osille, esimerkiksi painikkeille ja sivun osioille erikseen. Sovelluksen sivujen käyttäjänäkymän ulkoasun yhtenäisyyden vuoksi määritellään sivujen teema (Theme), johon halutut ulkoasumallit kootaan kokoelmaksi. Turvallisuusmäärityksiin (Security) sisältyvät mm. määritykset kenelle sivu näytetään. Navigointiosioon (Navigation Bar) voidaan tallentaa linkkipainikkeita ja ne näkyvät sivulla valitun ulkoasumallin mukaisesti.

Jokaisella osioiden otsikoiden alla olevalla nimikkeellä, on oma sekvenssinumero, joka määrää suoritus- tai esiintymisjärjestystä. Osioissa olevat nimikkeet voidaan myös ehdollistaa käytettäviksi, näytettäviksi tai suoritettaviksi vain joissakin tietyissä tapauksissa. Nimikkeisiin viitataan niiden nimen perusteella, mikä on myös nimikkeen tunniste eli ID. Nimi sisältää tiedon sovelluksen sivun sivunumerosta ja mikäli nimike liittyy tietokantakenttään, kentän nimi käy ilmi nimestä. Esim. sovelluksen sivulla 38 olevaan task_pk-kenttään viittaava nimikkeen nimi on P38_TASK_PK.

4.2 Sovelluksen yhteisesti käytettävät komponentit Apexissa

Lisää yhteisiä komponentteja löytää määrittelysivun oikean yläosan ratas-painiketta painamalla. Kuvassa 2 Logic-otsikon alla olevat Application Items ja Application Processes ovat komponentteja, joita tarvitaan haettaessa tietoa tietokannasta AJAXia (Asynchronous JavaScript and XML) käyttäen. AJAX on tekniikka, jolla dataa voidaan vaihtaa sivun ja serverin välillä ilman, että sivua ladataan uudelleen (w3schools 2010). Tämä tekee sovelluksesta nopeamman ja käyttäjäystävällisemmän.



Kuva 2. Näkymä Shared Components -sivusta.

Oraclen Application Express on mielestäni erittäin käytännöllinen ja monipuolinen sovelluskehitin, josta löytää paljon tietoa Internetistä. Kehittimen tiimoilta on kosolti aiheeseen liittyviä opettavia blogi-kirjoituksia ja keskustelupalstoja, joita olen käyttänyt ja joiden avulla olen oppinut paljon hyödyllisiä tietoja ja taitoja.

Mikäli kehitettävänä olisi ollut itsenäisesti toimiva järjestelmä, eikä osa jo olemassa olevaa suurempaa kokonaisuutta, olisi toteuttamisessa käytettäviä välineitä pitänyt vertailla ja etsiä tarkoitukseen sopivin vaihtoehto. Ohjelmointikielen ja tietokannan valintaan vaikuttavat tekijän omat tiedot ja

taidot sekä asiakkaan asettamat vaatimukset. Ohjelmointikielen ja kehitysvälineen tuntemus helpottaa suunnittelutyötä ja nopeuttaa toteuttamista. Ennalta on hyvä voida arvioida, mitä asiakkaan järjestelmältä vaatimia ominaisuuksia ja toimitoja on mahdollista toteuttaa kohtuullisen ajan ja kustannusten puitteissa. Myös saatavilla oleva tuki, esimerkiksi verkossa toimivat tukipalvelut ovat ohjelmistokehitystyössä tärkeitä.

5 TOIMEKSIANTOJÄRJESTELMÄN MÄÄRITTELY

Toimeksiantojärjestelmän kehitysprosessi alkoi Dianti Oy:n toimitusjohtajan koettua tarpeelliseksi kirjata ylös tehtävät työt, ikään kuin työjonoksi, jotta niiden tekemistä oli helppo seurata. Hänen omaan käyttöönsä tekemä ratkaisu olikin alkusysäys ja eräänlainen protomalli Diantin oppilaitosasiakkaalle suuntaamalle toimeksiantojärjestelmälle, jonka määrittelyn ja toteutuksen sain tehtäväkseni.

Vaatimusmäärittelyssä punnitaan projektin tarpeellisuus ja toteuttamiskelpoisuus sekä kartoitetaan asiakasvaatimukset ja tavoitteet (Haikala & Märijärvi 2002, 78). Vaatimusmäärittelyn tulee olla mahdollisimman tarkka ja yksiselitteinen, jotta sen perusteella voidaan toteuttaa asiakkaan toiveet ja odotukset täyttävä järjestelmä, mikä on laadukkaan järjestelmän määritelmä (Haikala & Märijärvi 2002, 48). Toiminnallisten vaatimusten määrittelyssä listataan asiakkaan järjestelmältä odottamat toiminnot, eli miten järjestelmää halutaan käyttää. Ei-toiminnallisten vaatimusten määrittelyssä asetetaan rajat, joissa järjestelmä täyttää toiminnalliset vaatimukset. (Pohjonen 2002, 28.)

Ensimmäinen vaihe oman työni aloittamisessa oli määritellä toimeksiantojärjestelmän toiminnalliset vaatimukset. Keskustelimme Dianti Oy:n toimitusjohtajan kanssa, mihin tarkoitukseen järjestelmä tehdään, mitä järjestelmältä odotetaan ja minkälaista tietoa sen avulla halutaan kerätä ja raportoida.

5.1 Toiminnalliset vaatimukset

Toimeksiantojärjestelmän tarkoituksena on helpottaa markkinointitoimen ja toimenpiteiden seurauksena syntyvien ja toteutettavien töiden kartoittamista. Toimenpiteellä tarkoitetaan oppilaitoksen ja sen asiakkaan välistä

vuorovaikutusta. Toimeksiantoja tulee voida lisätä myös ilman, että ne liittyvät millään lailla markkinointitoimiin tai toimenpiteisiin. Toimeksianto voi liittyä asiakasyritykseen.

Toimeksianto voidaan osoittaa yksittäisen henkilön, opintoalan tai tiimin tehtäväksi. Henkilö, jolle toimeksianto osoitetaan tehtäväksi, kuuluu aina johonkin opintoalaan ja vastaavasti opintoala kuuluu johonkin tiimiin.

Toimeksiantoja pitää voida priorisoida ja järjestää tekijäkohtaisesti työn kiireellisyyden mukaan. Sovimme Dianti Oy:n toimitusjohtajan kanssa, että priorisointi jaetaan kolmeen eri kategoriaan seuraavasti:

A. kiireellisiin, heti työn alle otettaviin tehtäviin
B. melko kiireellisiin tehtäviin
C. ei niin kiireellisiin tehtäviin.

Näiden kategorioiden sisällä tärkeysjärjestys ilmaistaan numeroilla yhdestä alkaen.

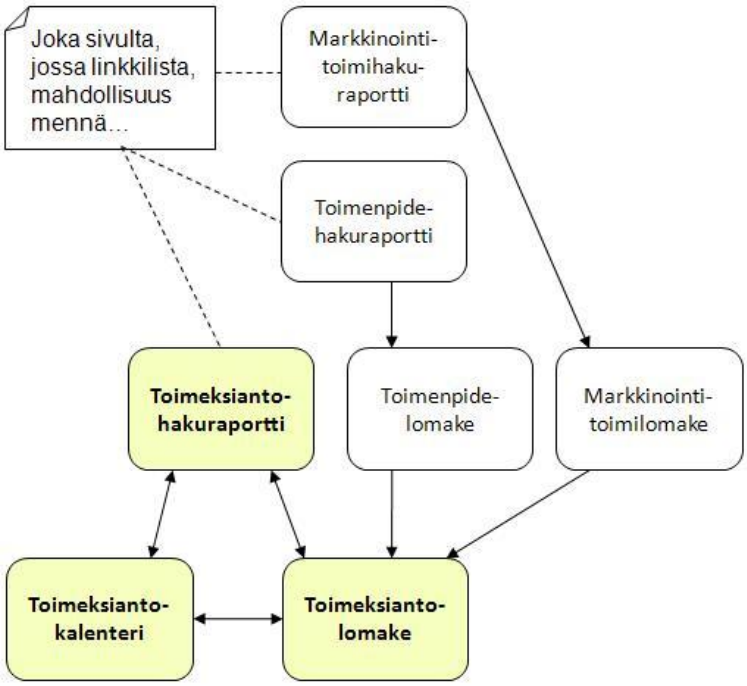
Tallennettaessa uusi toimeksianto toimeksiannon osoitetulle tekijälle tai opintoalan/tiimin vetäjälle, järjestelmä lähettää ilmoituksen uudesta toimeksiannosta sähköpostitse. Vastaavasti kun toimeksianto on tallennettu tehdyksi, järjestelmä lähettää sähköpostiviestin toimeksiannon antajalle, että toimeksianto on tehty.

Järjestelmän avulla voidaan seurata tehtävien laatua, määrää ja toteutumista henkilö-, opintoala- ja tiimitasolla. Järjestelmässä voidaan tarkastella toimeksiantoja myös havainnollisessa kalenterinäkymässä. Tallennettu

toimeksianto voidaan jakaa uusiksi toimeksiannoiksi, jolloin toimeksiannoille tulee hierarkkinen rakenne.

5.2 Navigointi

Toimeksiantojärjestelmässä käyttäjä tulee Diantin asiakkaalle tarjoaman Focusa-oppilaitosohjelmiston periaatteiden mukaisesti ensin interaktiiviselle raporttisivulle, josta hän voi valita haluamansa toimeksiannon tiedot avattavaksi ja muokattavaksi lomakkeelle. Hän voi avata raporttisivulta tyhjän lomakkeen uuden toimeksiannon lisäämiseksi. Käyttäjä voi avata toimeksiantokalenterisivun raporttisivulta, jolloin käyttäjä valitsee kalenterisivulta, kenelle osoitettuja toimeksiantoja hän haluaa tarkastella kalenterinäkymässä. Käyttäjä voi avata toimeksiantokalenterisivun myös lomakkeelta, jolloin avataan kalenterinäkymä, jossa on henkilön/opintoalan/tiimin toimeksiannot, jolle lomakkeella avoinna oleva toimeksianto on osoitettu.

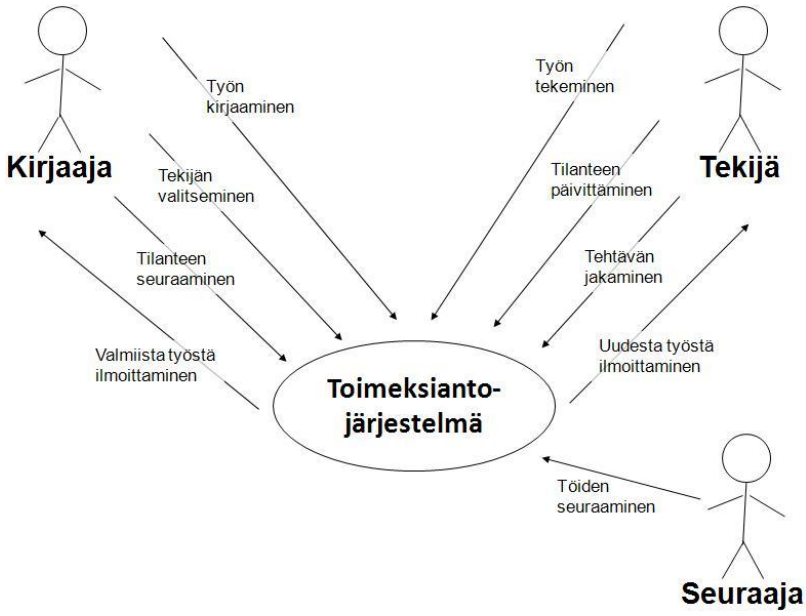


Kuvio 8. Navigointi asiakkuus-osiossa toimeksiantojärjestelmän osalta.

Kuten kuviosta 8 voi nähdä, toimeksiantojärjestelmän raporttisivulle voi navigoida kaikilta asiakkuus-osion sivuilta, joilta on linkki toimeksiantoihin. Toimeksiantojärjestelmän sisällä navigointi tapahtuu edellä kuvatulla tavalla. Asiakkuus-osion markkinointitoimi- ja toimenpidelomakkeelta voidaan lisätä asiaan liittyvä toimeksianto, jolloin markkinointitoimen tai toimenpiteen yksilöivä tunniste siirretään muutoin tyhjälle toimeksiantolomakkeelle.

5.3 Käyttäjien roolit

Käyttäjä voi edustaa toimeksiantojärjestelmässä kirjaajan, tekijän tai seuraajan roolia kuvion 9 mukaisesti. Kirjaajalla tarkoitan henkilöä, joka lisää toimeksiannon, eli työn jonkun tehtäväksi. Tekijä-roolin omaava henkilö on saanut tehtäväkseen työn tekemisen. Tekijä voi olla myös opintoalan tai tiimin, jolle toimeksianto on osoitettu, vetäjä. Hän päivittää toimeksiannon tilaa, mahdollisesti jakaa toimeksiannon hierarkkisesti edelleen useiksi tehtäviksi ja kirjaa sen aikanaan valmistuneeksi. Seuraaja on henkilö, joka käy selaamassa toimeksiantotietoja.



Kuvio 9. Käyttäjien roolit toimeksiantojärjestelmässä.

Kaikilla käyttäjillä on mahdollisuus toimia kussakin roolissa. Sovimme Dianti Oy:n toimitusjohtajan kanssa, että toimeksiantojärjestelmässä käyttöoikeuksia ei rajoiteta roolien mukaan.

5.4 Toimeksiantojärjestelmätehtävän rajaaminen

Tehtävänäni oli toteuttaa Diantin Focusa-oppilaitosohjelmiston asiakkuus-osion osaksi toimeksiantojärjestelmä, joka koostui raportti-, lomake- ja kalenterisivusta. Järjestelmän käyttäjiä varten kirjoitin sivu- ja kenttäkohtaiset käyttöohjeet.

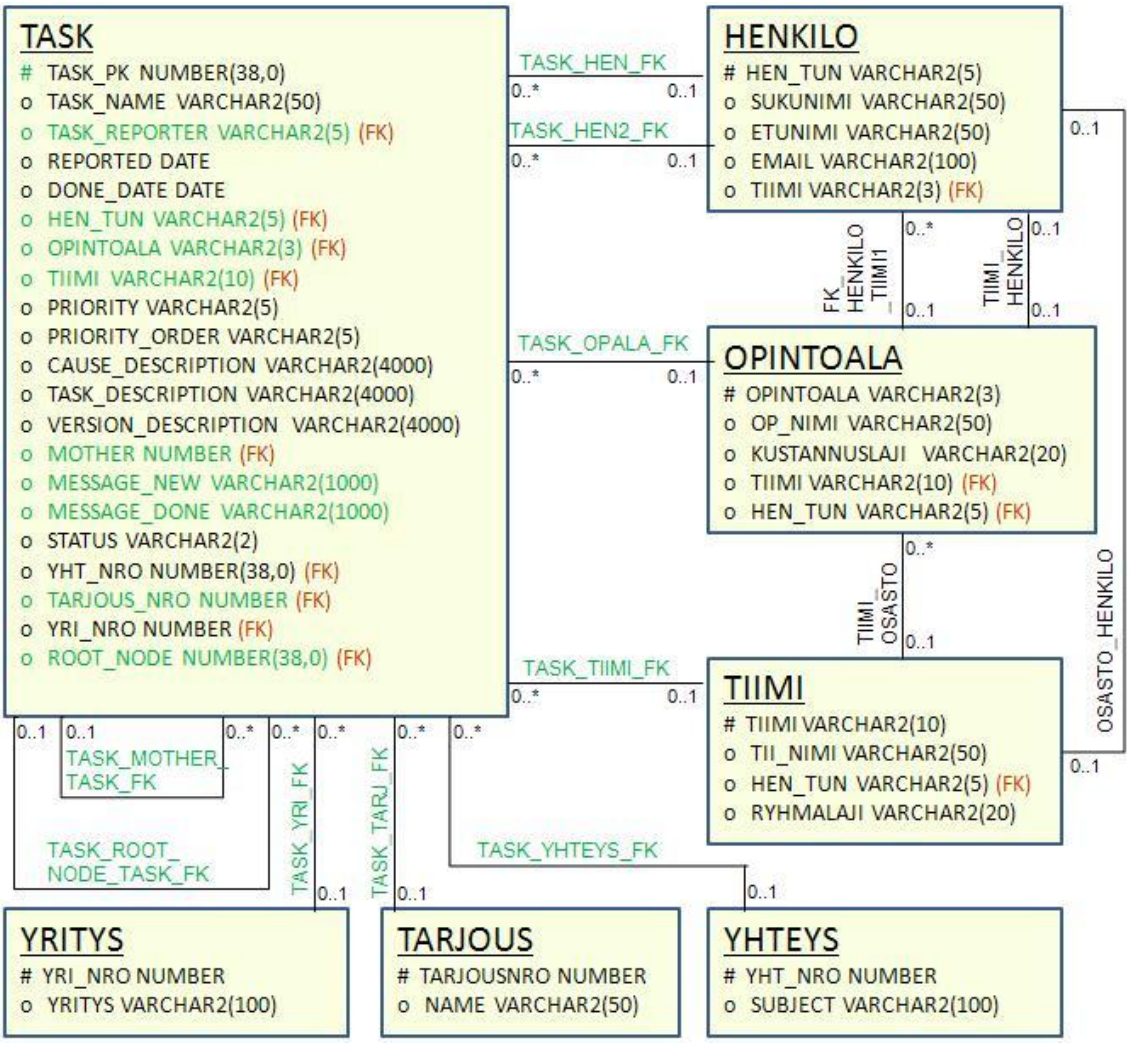
Dianti Oy:n henkilökunta huolehtii järjestelmän käyttöönotosta ja liittämisestä asiakkuus-osioon. Tehtävääni ei siis kuulunut markkinointitoimi- ja toimenpidesivuille tarvittavien painikkeiden tekeminen, liittäminen sivuhierarkiaan, eli sivupolkujen (Breadcrumb) tekeminen, eikä toimeksiantosivujen sijoittaminen välilehdille (Tabs).

6 TOIMEKSIANTOJÄRJESTELMÄN SUUNNITTELU

Suunnitteluvaiheessa kuvataan, miten määritelty järjestelmä toteutetaan (Haikala & Märijärvi 2002, 81).

6.1 Tietokanta

Koska Diantin valmistamien Focusa-sovellusten pohjalla oli jo laaja relaatiotietokanta, käytin toimeksiantojärjestelmän avulla tallennetavan ja näytettävän tiedon tallentamiseen jo olemassa olevia tauluja. Määrittelin toimeksiantojärjestelmän vaatimusmäärittelyn perusteella käytettävät tietokantakentät ja tarvittavien tietokantataulujen väliset yhteydet, jotka on kuvattu kuviossa 10. Tekemästäni toimeksiantojärjestelmästä tiedot tallennetaan task-tauluun. Taulujen attribuuteista, eli sarakkeista, on kuvioon kirjattu vain ne, joita toimeksiantojärjestelmä käyttää. Kaikissa tauluissa on lisäksi sarakkeet, joista näkee tietueen tallentajan tiedot ja muutoshistorian. Tekemäni rakenteelliset lisäykset on merkitty kuvioon vihreällä värillä.



Kuvio 10. Graafinen esitys toimeksiantojärjestemän käyttämästä tietokannasta.

Taulujen nimet on kuviossa 10 alleviivattu, sarakkeiden nimet ja tietotyypit on lueteltuina taulun nimen alapuolella. Sarakenimen edessä oleva o-merkki tarkoittaa, että tietuerivin sarakkeeseen annettava arvo ei ole vaadittava. Tauluissa ainoa vaadittava arvo on pääavaimelle, joka on merkitty #-merkillä, annettava arvo. Sarakenimen ja tietotyypin jäljessä sulkuihin on merkitty merkkijonon maksimipituus tai numeroiden kokonaislukumäärä, joista desimaalien osuus pilkulla erotettuna. Mikäli sarakerivin lopussa on (FK)-merkintä, se tarkoittaa, että kyseinen sarakekenttä on taulun vierasavain, eli viittaa toisen taulun pääavaimeen. Taulujen väliset yhteydet, eli viiteavainten ja pääavainten väliset yhteydet, on piirretty ja nimetty taulujen välille.

6.2 Käyttöliittymä ja tietojen käsittely

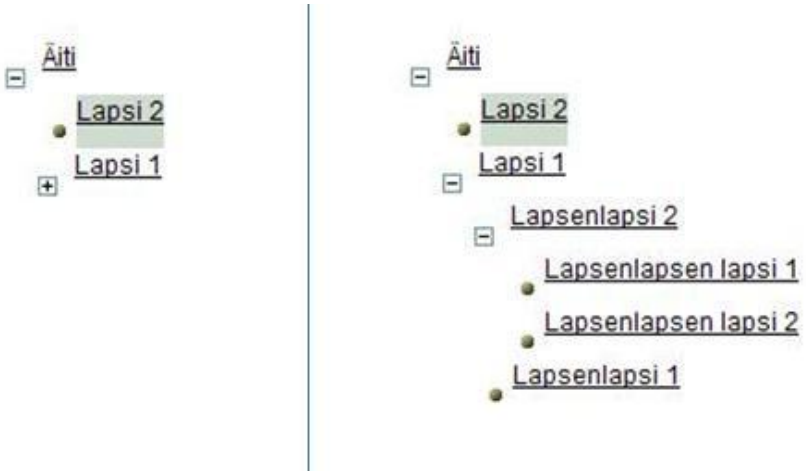
Tutustuin ennalta Diantin asiakkaille tarjoamaan Focusa-oppilaitosohjelmistoon ja yrityksen käytäntöihin Dianti Oy:ssä suorittamani työharjoittelujakson aikana. Suunnittelu- ja toteutusvaihetta helpotti huomattavasti, kun tiesin käytettävän sovelluskehitysvälineen, Oraclen Application Expressin ominaisuuksista ja toiminnasta sekä Diantissa käytetyistä työskentelymenetelmistä ja heidän Focusa-oppilaitosohjelmistossa käyttämistään sivurakenteista ja navigointilogiikasta.

Piirsin tietovuokaaviot havainnoimaan käyttäjän ja järjestelmän välistä vuorovaikutusta tilanteissa, joissa käyttäjä lisää uuden toimeksiannon (liite 1), päivittää toimeksiannon tietoja (liite 2) ja poistaa toimeksiannon (liite 3). Toteutin käyttöliittymän ja tietojen käsittelyyn liittyvän yksityiskohtaisemman suunnittelun, toteutuksen ja testauksen iteroiden sykleissä yksi tietovuokaavioiden toimintokokonaisuus kerrallaan, jotka on kerrottu liitteessä 4. Koin menetelmän mielekkääksi, sillä iteroiden työ eteni koko ajan kohti valmista järjestelmää pala kerrallaan.

7 TOIMEKSIANTOJÄRJESTELMÄN TOTEUTUS

Tietokantamuutosten jälkeen loin edellä kuvaamani task-taulun sarakkeiden mukaiset kentät sisältävän lomakkeen. Lähdin muokkaamaan lomakkeen kenttiä sellaisiksi, että niiden avulla pystyi tekemään toimintoja vaatimusmäärittelyn mukaan tekemieni vuokaavioiden osoittamalla tavalla. Kuten edellisessä luvussa mainitsin, vuokaaviot, jotka ovat liitteenä (liitteet 1, 2 ja 3), kuvaavat käyttäjän ja järjestelmän välistä vuorovaikutusta ja järjestelmän toimintoja eri tilanteissa. Loin myös alustavan raportin, jotta voin todeta, että lomakkeella tekemäni tallennus-, päivitys- ja poistokokeilut toimivat. Lopuksi, kun lomakesivu alkoi olla valmis, loin kalenterisivun, tarkensin raportissa näytettävät kentät Diantin toimitusjohtajan toiveiden mukaisiksi ja kirjoitin kenttä- ja sivukohtaiset käyttöohjeet.

Toteutusvaiheen aikana kävin Diantissa kahdesti, jolloin kävimme läpi senhetkistä tilannetta, ja sain tarkennuksia määrittelyyn liittyen mm. raportissa näytettäviin kenttiin, kalenterissa näytettävien toimeksiantojen ulkoasuun ja tietojen muokattavuuteen lomakkeella eri tilanteissa. Sain myös opastusta Diantin Focusa-oppilaitosohjelmistossa käytettävän hierarkkisen puurakenteen tekemiseen, jonka toteutin lomakesivulle. Kuvassa 3 on esitetty lomakkeella oleva hierarkkinen puurakenne. Kuvassa vasemmalla näkyy tilanne, kun Lapsi 2 –niminen toimeksianto on valittu lomakkeelle. Kuvassa oikealla on esitetty edelleen sama toimeksiantopuu, mutta Lapsi 1 –nimisen toimeksiannon hierarkkinen rakenne on avattu.



Kuva 3. Toimeksiantojen hierarkiaa esittävä puumalli.

Järjestelmän suunnittelu, toteutus ja testaus edeten toiminnallinen kokonaisuus kerrallaan, jotka on lueteltu liitteessä 4, oli tekemisen kannalta mielekästä, sillä työ eteni loogisesti ja järjestelmän toiminta oli varmennettavissa testaamalla koko ajan. Testaamalla ilmenneet virheet oli heti korjattavissa ennen seuraavaa kierrosta.

Käytin toteutuksessa paljon JavaScriptiä, etenkin syötteiden oikeellisuuden tarkistuksissa sekä käyttäessäni Ajaxia. Kuten Apex-osuudessa mainitsin, Ajax on menetelmä, jossa käyttöliittymä kommunikoi tietokannan kanssa päivittämättä sivua palvelimelta (w3schools 2010).

7.1 Toimeksiannon tekijäkohtainen prioriteetti ja prioriteetin järjestys

Järjestelmän haastavin vaihe toteuttaa oli toimeksiannon tekijäkohtaisen prioriteetin ja prioriteetin järjestyksen päivittäminen (liitteen 4, tehtävä 4). Haastavan tehtävästä teki se, että järjestyksen päivittämisessä pitää ottaa huomioon monia eri tilanteita.

Käyttäjän lisätessä uutta toimeksiantoa, hän voi sijoittaa uuden tehtävän haluamaansa prioriteettiin, haluamalleen järjestyspaikalle. Mikäli tekijällä on ennestään käyttäjän valitsemalla prioriteetilla tehtäviä, pitää kyseisen prioriteetin tehtävien järjestykset päivittää, mikäli tehtävää ei lisätä järjestyslistan viimeiselle sijalle.

Käyttäjän päivittäessä toimeksiannon prioriteettia tai järjestystä, pitää tekijän prioriteetin järjestykset päivittää sekä vanhan että uuden prioriteetin osalta. Kun toimeksianto tallennetaan tehdyksi, sen prioriteetti ja prioriteetin järjestys poistetaan, jolloin tekijän prioriteetin järjestykset päivitetään kyseisen prioriteetin osalta.

Poistettaessa toimeksianto, pitää huomioida mahdollisuus, että sillä on hierarkiassa alempana olevia toimeksiantoja, jotka poistetaan äititoimeksiannon mukana. Näillä äititoimeksiannon mukana poistettavilla toimeksiannoilla voi olla eri tekijöitä. Ennen poistoa pitää etsiä siis kaikki hierarkiassa alempana olevat toimeksiannot ja käydä ne läpi alimmasta alkaen niin, että päivitetään toimeksiannon tekijän prioriteettikohtaiset järjestysnumerot ennen poistamista, ja edetään seuraavaan sillä hetkellä hierarkiassa alimpana olevaan toimeksiantoon ja niin edelleen.

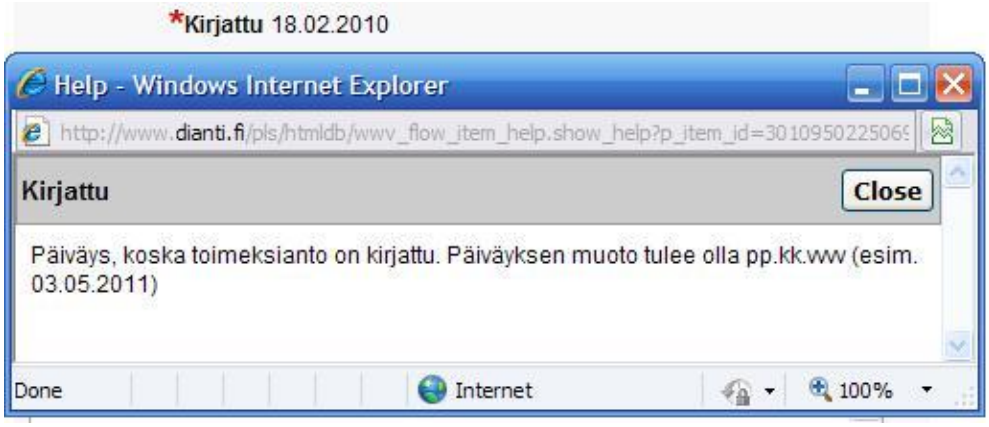
Suunnitellessani miten toteutan prioriteetin järjestyksen antamisen ja päivittämisen, tein useita kokeiluja. Kokeilin aluksi tehdä herätteen (trigger), joka huolehtisi numeroinnin päivityksestä, mutta kävi ilmi, että se ei onnistu. Kun heräte laukaistaan esimerkiksi ennen päivitystä, herätteen käynnistämä proseduuri ei voi toimia, sillä proseduurissa pitää päivittää tietueita, jolloin heräte laukaistaan uudelleen kesken proseduurin suorittamista ja saadaan virheilmoitus.

Seuraavassa vaiheessa kokeilin käynnistää proseduurin uuden tietueen lisäämisen, päivityksen tai poistamisen jälkeen JavaScriptin avulla ja se toimikin, kun selaimena oli Internet Explorer (v. 8.0). Kun testasin toimintaa Mozilla Firefoxilla (v. 3.5.5) ja Operalla (v. 9.64), JavaScript-koodin suoritus loppui tallentamiseen, eikä järjestyksen päivitystä suoritettu lainkaan. Tästä syystä järjestyksen päivitys käynnistetään JavaScriptin avulla jo ennen tietueen lisäämistä, päivittämistä tai poistamista.

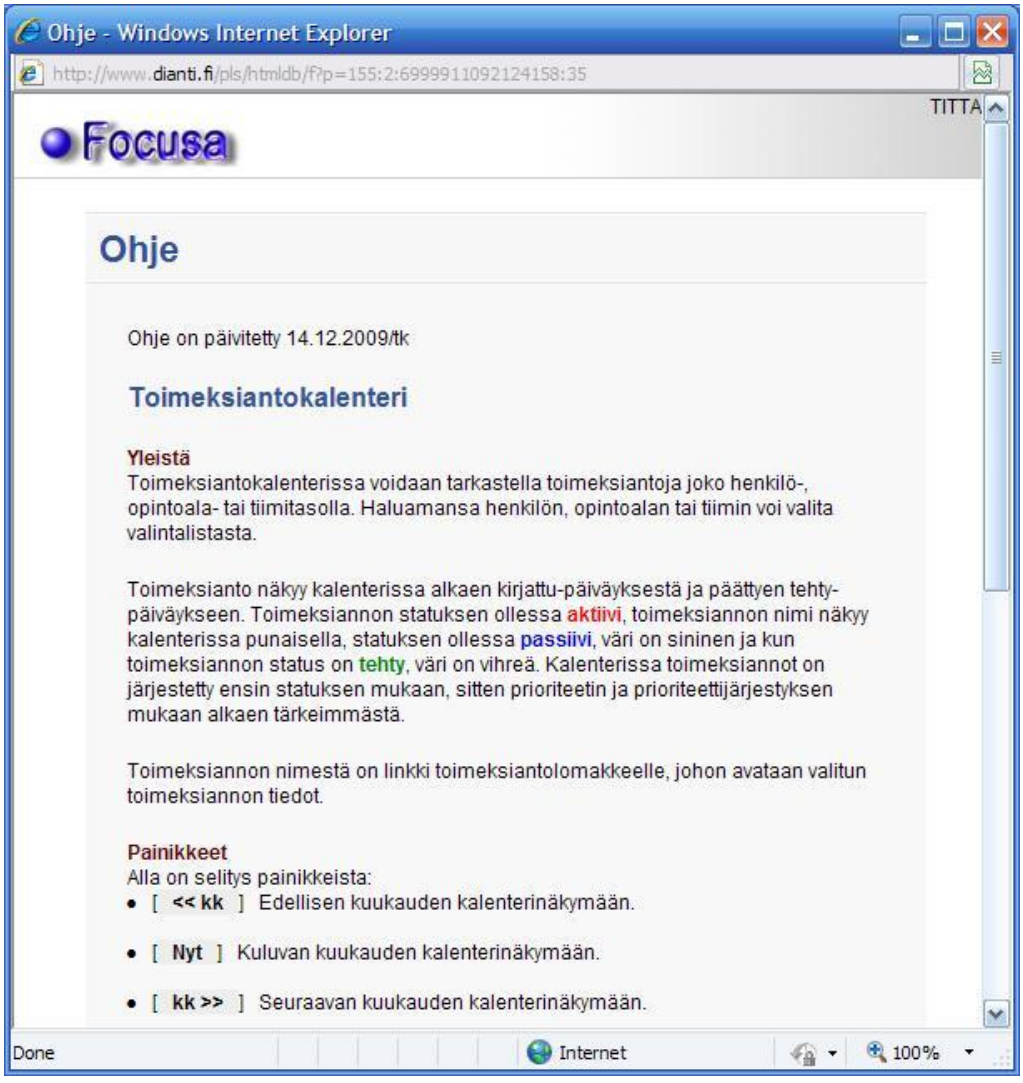
7.2 Sivujen viimeistely ja käyttöohjeiden laatiminen

Raportti-, lomake- ja kalenterisivujen ollessa toiminnallisesti valmiit, tarkistin kenttien otsikot ja tein tarvittavat muutokset. Pyrin tekemään järjestelmästä käyttäjäystävällisen ja Diantin asiakkaille tarjoaman Focusa-oppilaitosohjelmiston periaatteiden mukaisesti toimivan kokonaisuuden, jossa käytettävät termit ovat asiakasoppilaitoksen käyttämiä. Näillä toimilla pyrin tekemään järjestelmästä loogisen ja helppokäyttöisen.

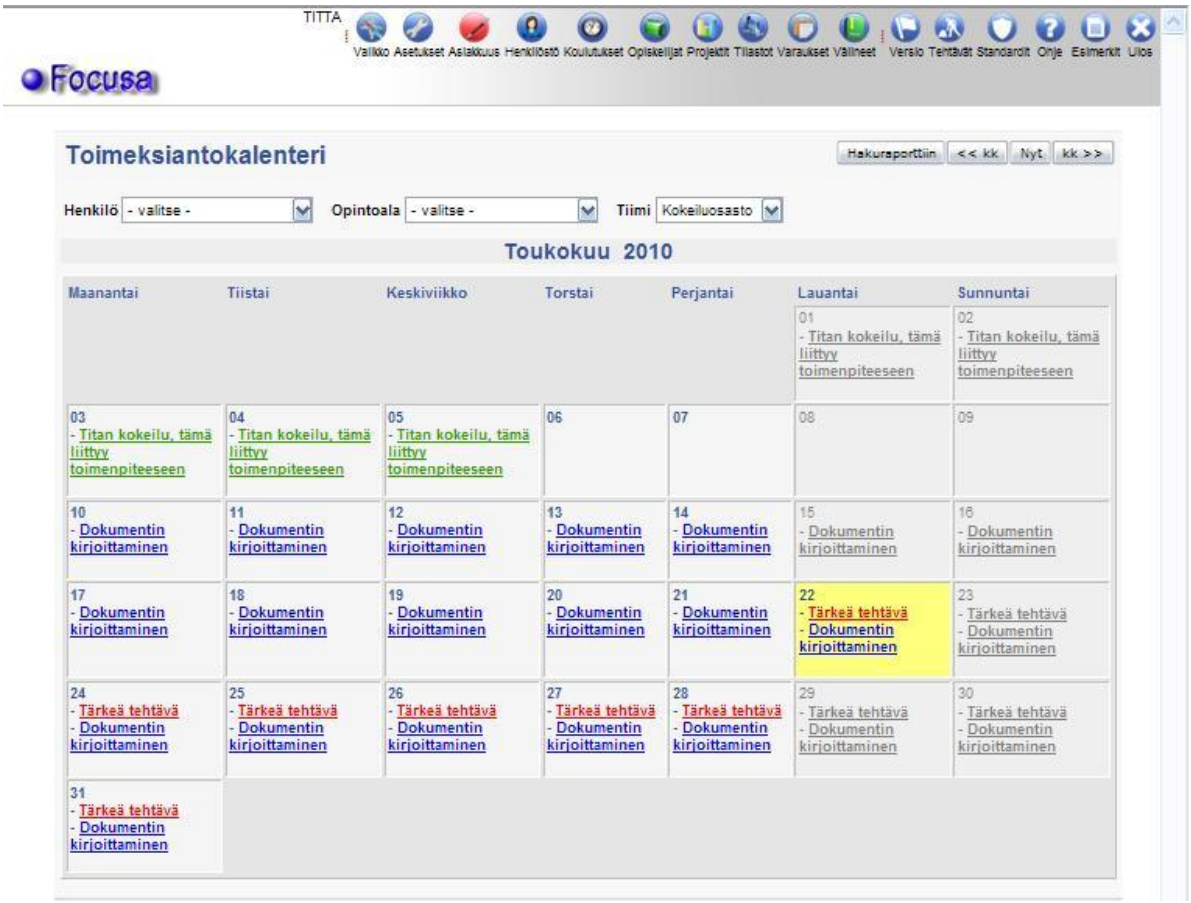
Viimeiseksi kirjoitin kenttä- ja sivukohtaiset käyttöohjeet. Pyrin kirjoittamaan ohjeista riittävän yksityiskohtaiset ja selkeät, jolloin kokematonkin käyttäjä pystyy turhautumatta käyttämään toimeksiantojärjestelmää. Kenttäotsikot sisältävät linkin, jota hiiren painiketta painamalla käyttäjälle avataan kenttäohje. Kuvassa 4 on Kirjattu-kentän kenttäohje. Järjestelmän sivujen, kuten kalenterisivun, joka on kuvassa 6, yläreunassa olevan Ohje-painikkeen valitsemalla käyttäjälle avataan sivukohtainen ohje. Kuvassa 5 on kalenterisivun sivuohje.



Kuva 4. Kirjattu-kentän kenttäohje.



Kuva 5. Kalenterisivun sivuohje.



Kuva 6. Näkymä kalenterisivusta.

8 TOIMEKSIANTOJÄRJESTELMÄN TESTAUS

Ohjelmiston testaamisen tavoitteena on löytää ohjelmasta mahdollisimman paljon virheitä. Testaamisen avulla voidaan myös arvioida, noudattaako toteutus määrittelyä. (Katara 2009, 22.) Riippuen siitä, missä vaiheessa ohjelmiston kehitys on, testaukseen voidaan käyttää lasilaatikko- ja mustalaatikkotyyppistä testausta (Katara 2009, 52).

Testasin järjestelmän toimivuutta koko toteutusvaiheen ajan. Lisätessäni uuden toimintokokonaisuuden (liite 4) mukaisen toiminnon, testasin sen toiminnan kaikissa senhetkisissä tilanteissa. Korjasin testaamalla löytämäni virheet niiden löydyttyä, minkä takia jouduin välillä muokkaamaan jo aiemmin toimiviksi havaitsemiani ratkaisuja.

8.1 Lasilaatikkotestaus, eli rakenteellinen testaus

Lasilaatikkotestausta voidaan sanoa myös sisäiseksi testaukseksi, eli sen avulla testataan ohjelmiston sisäistä toimintaa (Pohjonen 2002, 36). Lasilaatikkotestauksessa testataan esimerkiksi ohjelmakoodin suorituspolkujen ja silmukoiden toimintaa, virhetilanteiden käsittelyä ja rajapintojen toimivuutta. Rajapintojen testaaminen tarkoittaa pääasiassa funktioiden parametrien ja paluuarvojen testaamista. Ohjelmakoodin suorituspolkujen kattavassa testaamisessa pyritään antamaan muuttujille ja parametreille sellaisia arvoja, että todetaan kaikki koodin ehtojen mukaiset suoritusvaihtoehdot. Mahdollisimman monta ohjelmakoodin kriittistä suorituspolkua pitäisi pyrkiä testaamaan. (Katara 2009, 57-67.)

Ohjelmoidessani toimeksiantojärjestelmää, jonka toteutin osaksi Diantin Focusa-oppilaitosohjelmistoa, testasin koodin toimivuutta JavaScriptin avulla. Tarkkailin suorituspolkuja, missä kohdassa ohjelman suoritus oli menossa ja

mitkä arvot parametreilla ja muuttujilla kulloinkin oli. Erityisesti kiinnitin huomiota ehtolauseiden raja-arvoihin, funktioiden parametriarvoihin ja siihen, että funktiot suoritettiin oikein. Kun ohjelman suoritus eteni testauksen mukaan halutulla tavalla, piti huomioida kaikki tilanteet, jossa käyttäjä voi tehdä muutoksia arvoihin, joita testatun ohjelmakoodin suorittamisessa oli käytetty. Tällaisten tilanteiden varalta piti miettiä, missä kaikissa tilanteissa ohjelman suoritus pitää laukaista uudelleen, ettei käyttäjälle anneta väärää/vanhaa informaatiota tai anneta käyttäjän tehdä virheellisiä valintoja. Pyrin testaamaan kaikki tällaiset tilanteet ja korjaamaan havaitsemani virheet.

Testauksen kohteena oli myös käyttäjän antamien syötteiden oikeellisuuden tarkistus. Mikäli käyttäjä yrittää tallentaa lomakkeelta vääränmuotoista tietoa nähden tietokantakentän määritystä, järjestelmä joutuu virhetilanteeseen ja antaa automaattisesti virheilmoituksen. Järjestelmän virhetilanteissa automaattisesti antamat ilmoitukset saattavat olla käyttäjälle hämmentäviä, eivätkä kerro selkeästi, mikä virheen on aiheuttanut. Jotta virheilmoituksesta sai käyttäjälle ymmärrettävän, tein syötteiden oikeellisuuden tarkistukset ja käyttäjälle virheistä annettavat ilmoitukset ennen tallennusvaihetta pääosin JavaScriptiä käyttäen.

Testattavia käyttäjän antamien syötteiden tarkistuksia oli toimeksiantolomakkeelle annettujen päiväysten tarkistukset, että päiväys oli paikkansa pitävä ja syötetty oikeassa muodossa. Pakollisten kenttien arvojen sekä tekstikenttien ja –alueiden syötteiden pituudet tuli olla sallituissa rajoissa. Tarkistusten ja virheilmoitusten antamisen testaamisen tein syöttämällä kenttiin arvoja, jotka olivat sallittujen raja-arvojen sisä- ja ulkopuolella. Kuvassa 7 on esimerkki käyttäjälle annettavasta virheilmoituksesta tallennuksen yhteydessä, kun tekstikenttään on annettu liian pitkä syöte.

Kuva 7. Esimerkki käyttäjälle annettavasta virheilmoituksesta.

8.2 Mustalaatikkotestaus, eli toiminnallinen testaus

Mustalaatikkotestauksella todetaan, toimiiko ohjelma vaatimusten mukaisesti välittämättä siitä, miten se on kooditasolla toteutettu. Testattava ohjelma nähdään siis mustana laatikkona. Testitapaukset voidaan määritellä jo aikaisessa vaiheessa vaatimusmäärittelyn perusteella. (Katara 2009, 163.) Mustalaatikkotestaus, jota voidaan kutsua myös ulkoiseksi testaukseksi, ei välttämättä kata kaikkea toteutettua ohjelmaa, eikä sen avulla aina löydetä virheen syitä, vaan syyt selvitetään lasilaatikkotestauksella (Pohjonen 2002, 36).

Testitapaukset ovat etukäteen määriteltäviä ohjelman suoritussuunnitelmia virheiden etsimiseksi. Niitä kannattaa suunnitella niin, että ne kattavat monia tutkittavia tilanteita. Ajettaessa ohjelma testitapauksen mukaan, tulokset arvioidaan odotettuihin tuloksiin nähden ja päätetään, onko testi läpäisty hyväksytysti. Testitapauksen yksilöivät tiedot, sen tyyppi, tarkoitus ja esiehdot tulee mainita testitapauksen kirjaamisessa. Myös annettavat syötteet, odotetut tulokset, jälkiehdot ja ajohistoria tulee kirjata. (Katara 2009, 39-4.)

Testattuani Diantin asiakkaille tarjoaman Focusa-oppilaitosohjelmiston osaksi tekemäni toimeksiantojärjestelmän toimintaa toteutuksen kanssa käsi kädessä, tein toiminnallista testausta rinnakkain rakenteellisen testauksen kanssa. Mikäli jokin toiminto ei vastannut odotuksia, testasin koodin suoritusta tulostamalla muuttujien, parametrien ja funktioiden palauttamia arvoja JavaScriptin avulla, kuten lasilaatikkotestaus-kohdassa kerroin.

Saatuani toteutettua järjestelmän, testasin sitä suunnittelemieni testitapausten mukaisesti, josta esimerkki kuvassa 8. Kuvan testitapauksessa testataan prioriteetin järjestysten oikea päivittyminen tilanteessa, jossa lisätään uusi toimeksianto työn tekijälle testitapauksen mukaisesti.

Kuvan 9 yläosan raporttinäkymässä näkyy tekijällä jo ennestään olevat tallennetut työt, joiden prioriteetit ja prioriteetin järjestykset näkyvät Prioriteetti-sarakkeessa. Kuvan 9 yläosa kuvaa tilannetta ennen testitapauksen suorittamista. Kuvan 9 alaosassa on tilanne testin jälkeen, jolloin uusi kuvassa 8 olevan testitapauksen mukaisesti järjestyssijalle kolme tallennettu tehtävä näkyy raportissa sijalla kolme ja aiemmin järjestyssijalla kolme ja siitä eteenpäin olevat on päivitetty järjestyksessä ylöspäin. Testituloksen perusteella voi nähdä, että järjestelmä toimi odotusten mukaisesti. Testitapauksen perusteella ei voi ottaa kantaa ohjelmakoodin suoritukseen, vain siihen, että lopputulos oli odotusten mukainen.

- Identiteetti: TT05
-Tyyppi: toiminnallisuustesti
-Tarkoitus: testataan toimiiko prioriteetin järjestyksen päivittäminen oikein uuden toimeksiannon lisäämisen yhteydessä, toiminto 4
-Esiehdot: täytetään uusi toimeksianto tekijälle ja prioriteetille, jolla on entuudestaan useita tehtäviä
-Syötteet: lisätään toimeksianto sijalle 3
-Odotetut tulokset: Uuden toimeksiannon järjestys on tallennuksen jälkeen 3. Aiemmin sijoilla 1 ja 2 olleet ovat ennallaan, mutta aiemmin sijalla 3 ollut tehtävä on sijalla 4 ja muut saman tekijän ja prioriteetin toimeksiannot ovat siirtyneet yhdellä sijalla eteenpäin
-Jälkiehdot: -
-Ajohistoria:
-Aikaleima: 29.5.2010
-Testin tulos: toimii odotetusti
-Versio: selaimet Internet Explorer v.8.0, Mozilla Firefox v.3.5.5, Opera v.9.64
-Testaaja: Titta Korhonen

Kuva 8. Esimerkki testitapauksesta.

Toimeksiannon nimi	Tiimi	Opintoala	Henkilö	Status	Prioriteetti
testausta ensimmäinen	KOK	KOK	-	Passiivi	A - 1
testausta toinen	KOK	KOK	-	Passiivi	A - 2
testausta kolmas	KOK	KOK	-	Passiivi	A - 3
testausta neljäs	KOK	KOK	-	Passiivi	A - 4
testausta viides	KOK	KOK	-	Passiivi	A - 5

Toimeksiannon nimi	Tiimi	Opintoala	Henkilö	Status	Prioriteetti
testausta ensimmäinen	KOK	KOK	-	Passiivi	A - 1
testausta toinen	KOK	KOK	-	Passiivi	A - 2
testi - tämä lisätään kolmanneksi	KOK	KOK	-	Passiivi	A - 3
testausta kolmas	KOK	KOK	-	Passiivi	A - 4
testausta neljäs	KOK	KOK	-	Passiivi	A - 5
testausta viides	KOK	KOK	-	Passiivi	A - 6

Kuva 9. Tilanne ennen ja jälkeen testitapauksen suoritusta.

Ennalta suunniteltujen testitapausten lisäksi testasin järjestelmää lisäämällä järjestelmään paljon tietueita lomaketta käyttäen ja navigoimalla linkkien ja painikkeiden avulla sivujen välillä. Annoin lomakkeen täyttämisen yhteydessä tahallani ja tahattomasti virheellisiä syötteitä, joista sain virheilmoitukset, ja joiden mukaan tein tarvittavat korjaukset. Tein lisäyksiä myös hierarkkisesti,

jolloin pystyin toteamaan lomakkeella olevan hierarkkisen puurakenteen (kuvassa 3 ja 4) oikean toiminnan. Testasin toimintaa myös päivitys- ja poistotilanteissa, joissa toimeksiannon tekijäkohtaisen prioriteetin järjestyksen virheetön päivittyminen aiheutti eniten virheitä ja työllisti ehdottomasti eniten.

Testasin järjestelmää kolmella eri selaimella: Internet Explorerilla (v. 8.0), Mozilla Firefoxilla (v. 3.5.5) ja Operalla (v. 9.64). Kuten aiemmin mainitsin, selaimet toimivat joissakin tilanteissa erilailla. Onnistuin kuitenkin tekemään ratkaisut, joissa järjestelmä toimii mainituilla selaimilla yhdenmukaisesti

9 OMAN TYÖN ARVIOINTI

Tämän opinnäytetyön tavoitteena oli määritellä ja toteuttaa toimeksiantojärjestelmä osaksi Dianti Oy:n Focusa-oppilaitosohjelmiston asiakkuus-osiota. Toteutin työn Dianti Oy:n käyttämin välinein olemassa olevaan relaatiotietokantaan. Työ alkoi vaatimusten määrittelyllä ja eteni sen jälkeen suunnittelu-, toteutus- ja testausvaiheita iteroiden, kunnes järjestelmä vastasi vaatimusmäärittelyä. Diantin henkilökunta huolehtii järjestelmän käyttöönotosta ja liittämisestä asiakkuus-osioon.

Iteratiivinen ja inkrementaalinen työskentelytapa tuntui mielekkäältä ja motivoivalta, sillä edistystä tapahtui koko toteutusvaiheen ajan. Menetelmä on myös joustava tilanteissa, joissa vaatimusmäärittelyyn tehdään lisäyksiä tai muutoksia prosessin aikana.

Valmis toimeksiantojärjestelmä vastaa määriteltyjä vaatimuksia. Käytin järjestelmän toteutuksessa sovitusti Diantin oppilaitosasiakkaan käyttämiä termejä ja Diantin asiakkaalle tarjoaman Focusa-oppilaitosohjelmiston ulkoasuun ja navigointilogiikkaan liittyviä periaatteita. Varmistin järjestelmän sulavan ja käyttäjäystävällisen toiminnan käyttämällä Ajax-menetelmää. Omalta osaltani olen tyytyväinen tekemääni työhön ja Dianti Oy:n toimitusjohtajalta saamani palaute on ollut positiivista. Koska toimeksiantojärjestelmä ei vielä ole oppilaitoksen käytössä, käytettävyyspalautetta ei vielä ole.

Koska Dianti Oy:ssä ei ole käytössä dokumentointimalleja, noudatin työvaiheiden dokumentointia tehdessäni eri lähteistä löytämiäni ohjeita, jotta dokumentaatio olisi riittävää tasoa. Tämä onkin osa-alue, josta opin paljon tätä työtä tehdessäni. Dokumentointi on myös alue, johon Dianti Oy:ssä voisi

jatkossa kiinnittää enemmän huomiota. Opin paljon ohjelmistokehitystyöstä yleisellä tasolla etsiessäni perusteluja, miksi tein työn määrätyllä tavalla. Jotkin koulussa opetetut asiat, jotka ovat olleet mielessäni irrallisina, löysivät paikkansa laajemmassa kokonaisuudessa.

Sekä toimeksiantojärjestelmän kehitysprosessi että opinnäytetyön kirjoitusprosessi oli minulle haastava, mielekäs ja opettava. Dianti Oy:n toimitusjohtajan luottava ja kannustava suhtautuminen työskentelyyni ja Turun ammattikorkeakoulusta saamani hyvät opit mahdollistivat työn hyvän onnistumisen.

10 LÄHTEET

Bayer, J. & Muthig, D. 2006. A View-Based Approach fo Improving Software Documentation Practices in Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06). IEEE/IET Electronic Library (IEL)

Fowler, M. & Scott, K. 2002. UML.1., Jyväskylä: Docendo Finland Oy

Haikala, I. & Märijärvi, J. 2002. Ohjelmistotuotanto. 8., Helsinki: Satku

Katara, M. 2009. Ohjelmistojen testaus – tekniikat, työkalut ja prosessit. Viitattu 24.5.2010 http://www.cs.tut.fi/~testaus/s2009/OHJ-3060_2009-net.pdf.

Kroll, P. & Kruchten, P. 2004. The Rational Unified Process Made Easy: a Practitioner's Guide to the RUP. 4., Boston: Pearson Education, Inc.

Kruchten, P. 2004. The Rational Unified Process: an introduction. 3., Boston: Pearson Education, Inc.

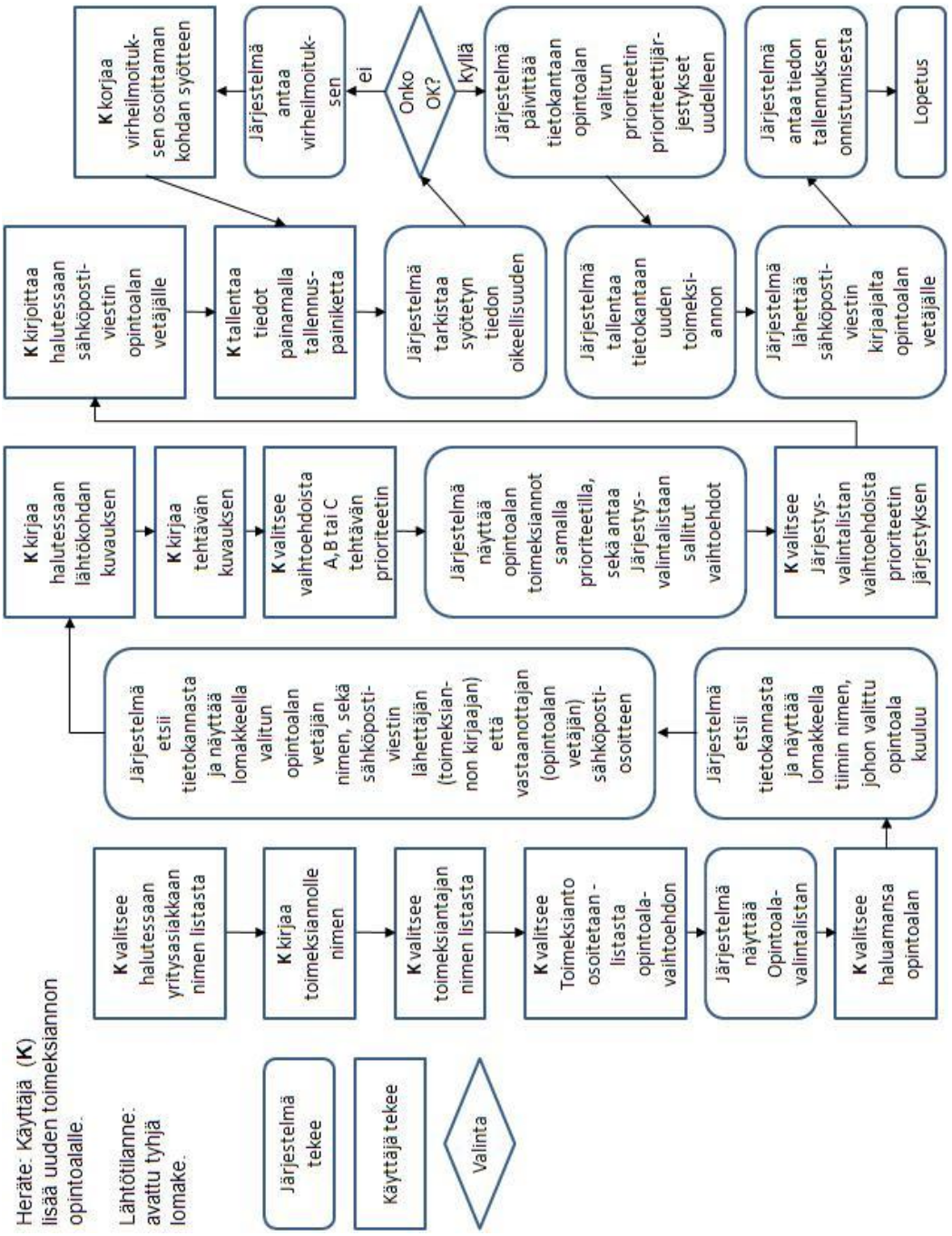
Manifesto for Agile Software Development. Viitattu 15.3.2010 http://agilemanifesto.org/.

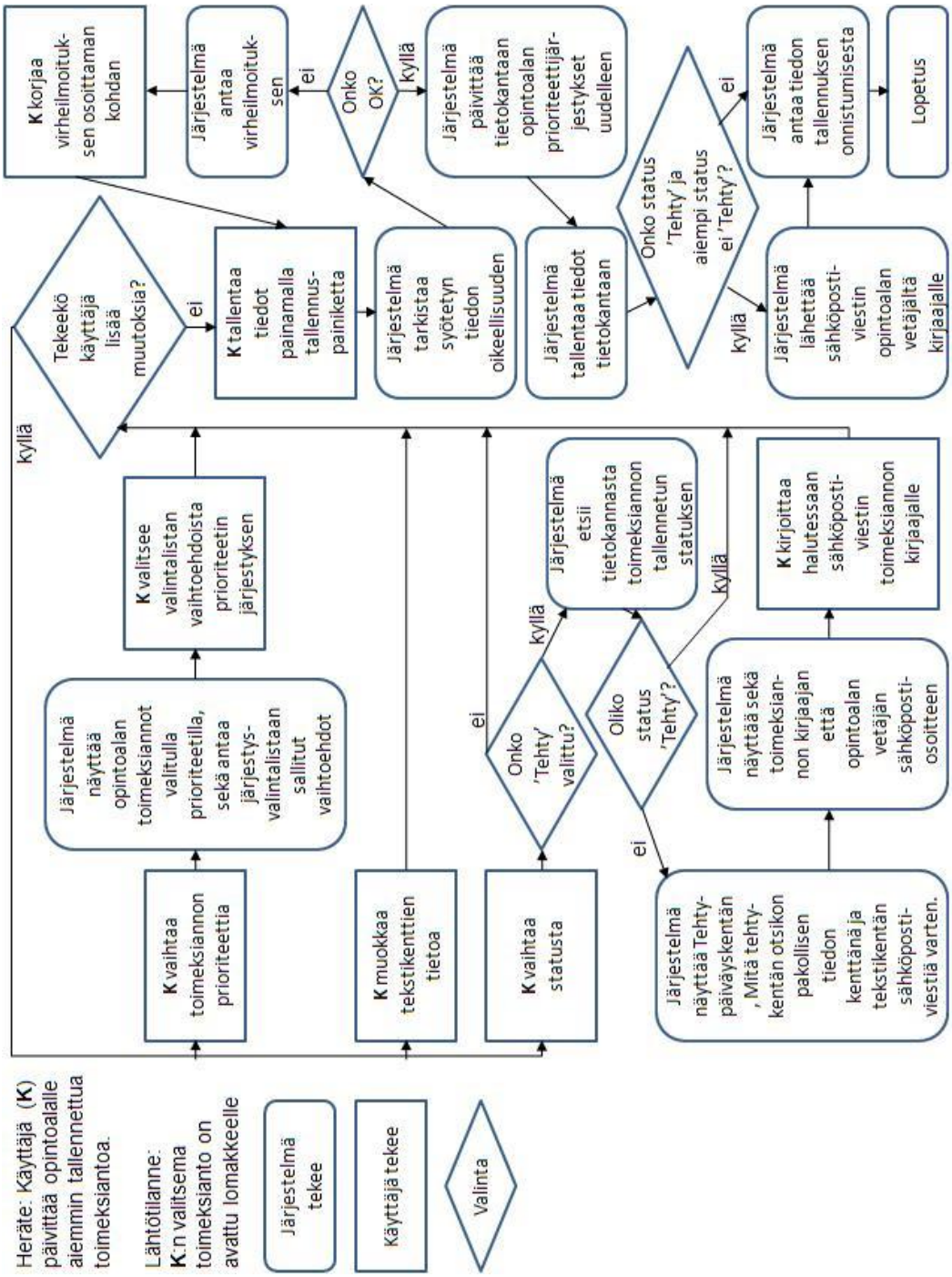
Oracle. What is Oracle Application Express? Viitattu 19.1.2010 http://www.oracle.com/technology/products/database/application_express/html/what_is_apex.html.

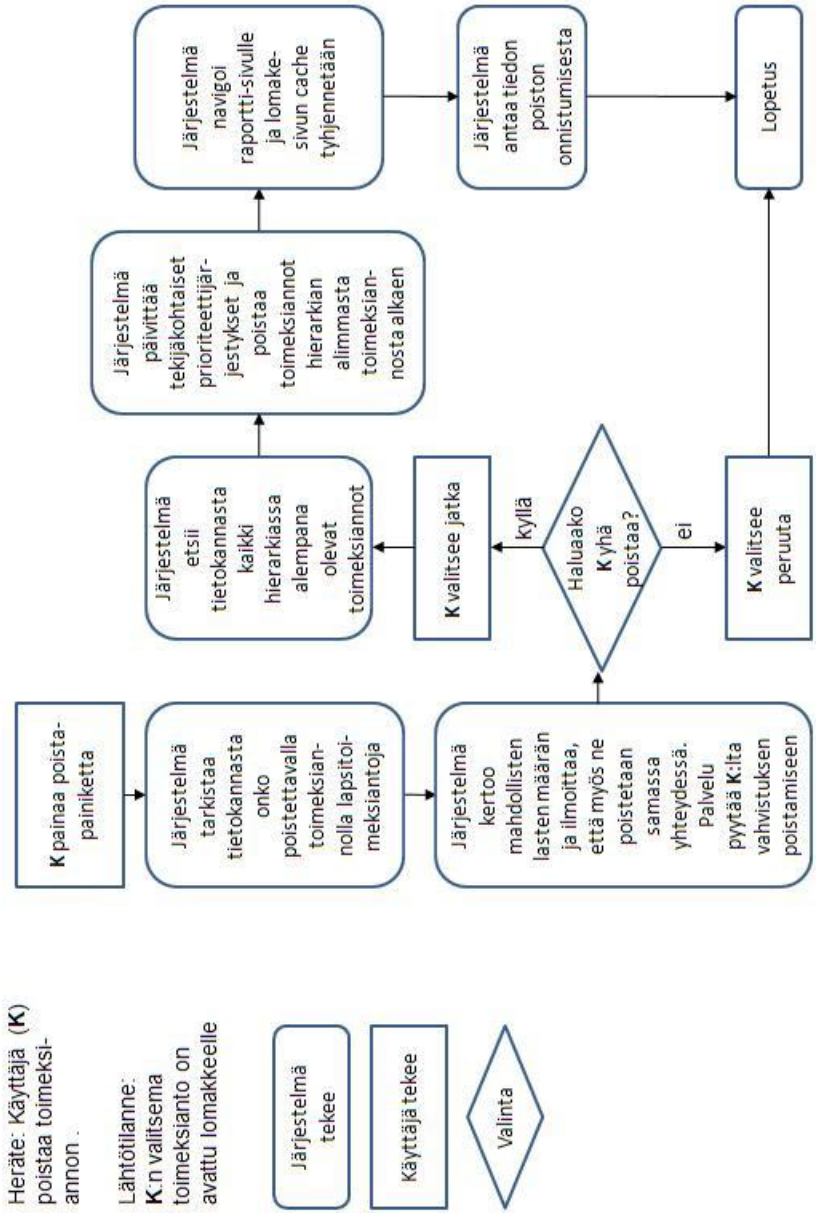
Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. 1., Jyväskylä: Docendo Finland Oy

Sinkkonen, I.; Nuutila E. & Törmä, S. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanama Oy

w3schools. Ajax Introduction. Viitattu 27.1.2010 www.w3schools.com/Ajax/ajax_intro.asp.


Heräte: Käyttäjä (K) lisää uuden toimeksiannon opintoalalle.
Lähtötilanne: avattu tyhjä lomake.
Järjestelmä tekee
Käyttäjä tekee
Valinta
K valitsee halutessaan yritysasiakkaan nimen listasta
K kirjaa toimeksiannolle nimen
K valitsee toimeksiantajan nimen listasta
K valitsee Toimeksianto osoitetaan - listasta opintoala-vaihtoehdon
Järjestelmä näyttää Opintoala-valintalistan
K valitsee haluamansa opintoalan
Järjestelmä etsii tietokannasta ja näyttää lomakkeella tiimin nimen, johon valittu opintoala kuuluu
Järjestelmä etsii tietokannasta ja näyttää lomakkeella valitun opintoalan vetäjän nimen, sekä sähköposti-viestin lähettäjän (toimeksian-non kirjaajan) että vastaanottajan (opintoalan vetäjän) sähköposti-osoitteen
K kirjaa halutessaan lähtökohdan kuvauksen
K kirjaa tehtävän kuvauksen
K valitsee vaihtoehdoista A, B tai C tehtävän prioriteetin
Järjestelmä näyttää opintoalan toimeksiannot samalla prioriteetilla, sekä antaa Järjestelmä-valintalistaan sallitut vaihtoehdot
K valitsee Järjestelmä-valintalistan vaihtoehdoista prioriteetin järjestyksen
K kirjoittaa halutessaan sähköposti-viestin opintoalan vetäjälle
K tallentaa tiedot painamalla tallennus-painiketta
Järjestelmä tarkistaa syötetyn tiedon oikeellisuuden
Järjestelmä tallentaa tietokantaan uuden toimeksi-annon
Järjestelmä lähettää sähköposti-viestin kirjaajalta opintoalan vetäjälle
K korjaa virheilmoituk-sen osoittaman kohdan syötteen
Järjestelmä antaa virheilmoituk-sen
Onko OK?
ei
kyllä
Järjestelmä päivittää tietokantaan opintoalan valitun prioriteetin prioriteettijär-jestykset uudelleen
Järjestelmä antaa tiedon tallennuksen onnistumisesta
Lopetus


Heräte: Käyttäjä (K) päivittää opintoalalle aiemmin tallennettua toimeksiantoa.
Lähtötilanne: K:n valitsema toimeksianto on avattu lomakkeelle
Järjestelmä tekee
Käyttäjä tekee
Valinta
K korjaa virheilmoituksen osoittaman kohdan
Järjestelmä antaa virheilmoituksen
ei
Onko OK?
kyllä
Järjestelmä päivittää tietokantaan opintoalan prioriteettijärjestykset uudelleen
Järjestelmä antaa tiedon tallennuksen onnistumisesta
Lopetus
Tekeekö käyttäjä lisää muutoksia?
ei
K tallentaa tiedot painamalla tallennuspainiketta
Järjestelmä tarkistaa syötetyn tiedon oikeellisuuden
Järjestelmä tallentaa tiedot tietokantaan
Onko status 'Tehty' ja aiempi status ei 'Tehty'?
ei
kyllä
Järjestelmä lähettää sähköpostiviestin opintoalan vetäjältä kirjaajalle
kyllä
K valitsee valintalistan vaihtoehdoista prioriteetin järjestyksen
Järjestelmä näyttää opintoalan toimeksiannot valitulla prioriteetilla, sekä antaa järjestys-valintalistaan sallitut vaihtoehdot
K vaihtaa toimeksiannon prioriteettia
K muokkaa tekstikenttien tietoa
K vaihtaa statusta
ei
Onko 'Tehty' valittu?
kyllä
Järjestelmä etsii tietokannasta toimeksiannon tallennetun statuksen
Oliko status 'Tehty'?
kyllä
K kirjoittaa halutessaan sähköposti-viestin toimeksiannon kirjaajalle
Järjestelmä näyttää sekä toimeksiannon kirjaajan että opintoalan vetäjän sähköpostiosoitteen
ei
Järjestelmä näyttää Tehty-päiväyskentän, Mitä tehty-kentän otsikon pakollisen tiedon kenttänä ja tekstikentän sähköpostiviestiä varten.


Heräte: Käyttäjä (K) poistaa toimeksiannon .
Lähtötilanne: K:n valitsema toimeksianto on avattu lomakkeelle
Järjestelmä tekee
Käyttäjä tekee
Valinta
K painaa poista-painiketta
Järjestelmä tarkistaa tietokannasta onko poistettavalla toimeksian-nolla lapsitoi-meksiantoja
Järjestelmä kertoo mahdollisten lasten määrän ja ilmoittaa, että myös ne poistetaan samassa yhteydessä. Palvelu pyytää K:lta vahvistuksen poistamiseen
Haluaako K yhä poistaa?
kyllä
ei
K valitsee jatka
K valitsee peruuta
Järjestelmä etsii tietokannasta kaikki hierarkiassa alempana olevat toimeksiannot
Järjestelmä päivittää tekijäkohtaiset prioriteettijär-jestykset ja poistaa toimeksiannot hierarkian alimmasta toimeksian-nosta alkaen
Järjestelmä navigoi raportti-sivulle ja lomake-sivun cache tyhjennetään
Järjestelmä antaa tiedon poiston onnistumisesta
Lopetus

	Toiminto	**Kuvaus**
1	Tehtävän osoittaminen kohderyhmälle ja kohteen valitseminen (lomakkeella)	Toimeksianto voidaan osoittaa henkilölle,opintoalalle tai tiimille. Käyttäjän pitää ensin valita henkilö/opintoala/tiimi –ryhmä, jonka jälkeen hän voi valita tarkan kohteen
2	Valitun tekijän ja prioriteetin jo tallennettujen tehtävien näyttäminen (lomakkeella)	Näytetään käyttäjälle tekijän ja valitun prioriteetin tehtävät: järjestys, status ja toimeksiannon nimi. Tarkoituksena on helpottaa näkemään tekijän toimeksiantojen määrä ja statukset esim. uutta toimeksiantoa lisättäessä.
3	prioriteetin järjestys – valintalistan arvojen tulostaminen (lomakkeella)	Tehtävät järjestetään tekijä- ja prioriteettikohtaisesti tärkeysjärjestykeen. Tärkeintä tehtävää kuvaa numero yksi. Valintalistassa valittavina olevia arvoja voi olla: - uutta toimeksiantoa lisättäessä: 1 - valitun tekijän ja prioriteetin jo tallennettujen tehtävien lukumäärä + 1 - muokattaessa toimeksiantoa: 1 - osoitetun tekijän ja valitun prioriteetin jo tallennettujen toimeksiantojen lukumäärä.
4	prioriteetin ja prioriteetin järjestyksen lisääminen, muokkaus tai poisto (lomakkeella)	Tekijän tallennettujen toimeksiantojen prioriteettien järjestysten päivittäminen, kun lisätään uusi, muokataan tai poistetaan jo tallennettu toimeksianto. Tehtävän prioriteettia voidaan vaihtaa, kuten myös prioriteetin järjestystä. Poistettaessa sijalla 1 oleva tehtävä, pitää saman tekijän saman prioriteetin tehtävien järjestykset päivittää niin, että tehtävä nro 2. tulee nro 1 jne…

5	sähköpostin lähettäminen tekijälle uuden tehtävän lisäämisen yhteydessä sekä kirjaajalle tehdyn tehtävän tallentamisen yhteydessä (lomakkeella)	Kun tallennetaan uusi toimeksianto, tehtävän osoitetulle tekijälle tai opintoalan/tiimin vetäjälle lähetetään järjestelmästä sähköpostiviesti. Kun toimeksianto tallennetaan tehdyksi, järjestelmä lähettää sähköpostiviestin toimeksiannon kirjaajalle.
6	Toimeksiantojen hierarkiaa kuvaavan puurakenteen lisääminen lomakesivulle	Lomakesivulle lisätään toimeksiantojen hierarkiaa kuvaava puurakenne. Puu toteutetaan Focusa-oppilaitosohjelmistossa olevien puurakenteiden mukaiseksi.
7	Kalenterisivun toteuttaminen	Toimeksiantokalenterissa näytetään valitun henkilön, opintoalan tai tiimin toimeksiannot kuukausinäkymässä. Kalenterisolussa näytetään aktiivi-toimeksiannot punaisella, passiivit sinisellä ja tehdyt vihreällä. Näytettävästä toimeksiannon nimestä on linkki lomakesivulle, johon avataan valittaessa kyseisen toimeksiannon tiedot.
8	Interaktiivisen raportin kenttien valitseminen	Interaktiivisen raportin näytettävät kentät valitaan Diantin toiveiden mukaan. Valittu: toimeksiannon nimi, tiimi, opintoala, henkilö, status, prioriteetti, ilmoitettu, äititoimeksianto
9	Kenttä- ja sivuohjeiden kirjoittaminen	Käyttäjää varten kirjoitetaan tietokenttiä ja koko sivua koskevat ohjeet.