HAAGA-HELIA
ammattikorkeakoulu

Toimintakäsikirja-alusta IMS-järjestelmässä

Hanna-Mari Kinnunen

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2011

Sisällys

Keskeiset käsitteet

Apache Tomcat	Avoimen lähdekoodin palvelin, jolla voi ajaa JavaServlettejä ja JSP-sivuja.
CKEditor	Suosittu avoimen lähdekoodin JavaScript-pohjainen html-editori.
EFQM-malli	Oman toiminnan kehittämis- ja arviointityökalu organisaatioille. Sen on kehittänyt EFQM-järjestö (European Foundation for Quality Management).
Flying Saucer (XHTML renderer)	Avoimen lähdekoodin ohjelmakirjasto, jonka avulla pystyy muuntamaan XHTML:n, XML:n ja CSS:n pdf-formaattiin. Se käyttää muuntamiseen iText-kirjastoa.
Framework	Tavallista ohjelmakirjastoa laajempi ohjelmistokehys, jonka päälle voi rakentaa kokonaisen ohjelman tai sen osan.
Hibernate	Suosittu Object-Relational-Mapping ohjelmistokehys, jonka avulla tehostetaan tietokantojen käyttöä Javan avulla.
ISO 9001	Joukko laadunhallintajärjestelmän vaatimuksiin liittyviä standardeja.
iText	Javan ohjelmakirjasto, jonka avulla pystyy luomaan pdf-dokumentteja.

JavaEE	Java Enterprise Edition. Ohjelmistokehitysalusta, jonka avulla voi luoda ja ajaa monikerroksisia web-sovelluksia.
JavaServlet	Java-luokka, joka käyttää hyväkseen Java Servlet API –ohjelmakirjastoa. Se pystyy käsittelemään http-pyyntöjä.
JavaScript	Komentosarjakieli, jota käytetään web-ympäristössä.
JSP	Java Server Pages. Web-sivu, jolle voi tehdä Javalla sisältöä. Sisältö muokataan dynaamisesti palvelinpäässä ennen sivun tuomista käyttöliittymään.
JSTL	Java Server Page Standard Tag Library. Tagikirjasto, joka laajentaa JSP-sivujen käyttöä lisäämällä joukon tageja, joiden avulla voi muokata käyttöliittymän logiikkaa käyttämättä Javaa.
JTidy	Javan ohjelmakirjasto, jonka avulla voi muuttaa esim. html-sivun xhtml:ksi.
JQuery	JavaScript-kirjasto, jonka tarkoitus on tehostaa web-sivun koodia.
MVC	Model-view-controller. Sovelluskehityksen arkkitehtuurimalli.
MySQL	Suosittu vapaan lähdekoodin tietokantaohjelmisto.

PDF	Portable Document Format. Käyttöjärjestelmäriippumaton tiedostomuoto, jota käytetään sähköiseen julkaisemiseen, tulostamiseen ja painamiseen.
Microsoft SQL Server	Microsoftin kaupallinen tietokantaohjelmisto.
Standardi	Jonkin tahon esittämä suositus siitä, miten jokin asia tulisi tehdä.
SQL	Structured Query Language. Standardoitu kyselykieli, jonka avulla voi käsitellä relaatiotietokantaa.
Toimintajärjestelmä	Organisaation johtamis- tai ohjausjärjestelmä. Kutsutaan myös laatujärjestelmäksi.
Toimintajärjestelmäalusta	Sähköinen ohjelmisto, jonne pystyy kokoamaan kaikki organisaation toimintajärjestelmän osat.
Toimintakäsikirja	Toimintajärjestelmän yhteenvetokuvaus, joka toimii ovenavauksena koko järjestelmään. Kutsutaan myös laatukäsikirjaksi.
XHTML	Html:stä kehitetty kieli, joka täyttää xml:n muotovaatimukset.

1 Johdanto

Toimintakäsikirja-alusta IMS-järjestelmässä –projekti tehtiin HAAGA-HELIA Ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman opinnäytetyönä 26.10.2010-4.3.2011 välisenä aikana. Projektin toimeksiantaja oli helsinkiläinen IMS Business Solutions Oy. Se on laadunhallintaan erikoistunut noin 20 hengen yritys. IMS tarjoaa asiakasyrityksille asiantuntijapalveluita laadun-, prosessien ja tiedonhallintaan. Yritys on kehittänyt oman kokonaisvaltaisen ohjelmistoratkaisun, IMS toimintajärjestelmäalustan(jatkossa IMS-järjestelmä) laadunhallinnan tueksi. Sen avulla asiakasyritys voi luoda ja ylläpitää toimintajärjestelmää verkossa ja esittää kaiken toimintajärjestelmässä tarvittavan tiedon samassa käyttöliittymässä. Alusta on käytössä useilla kunnallisilla ja yksityisillä tahoilla.

Tässä projektissa lähdettiin suunnittelemaan uutta toimintakäsikirja-alustaa IMS-järjestelmään. Nykyinen, tuotantokäytössä oleva versio on hyvin vanhaa koodia, eikä sen ylläpitäminen ja jatkokehittäminen ole enää järkevää. Lisäksi alustan ulkoasu on vanhahtava ja kaipaa kipeästi uutta ilmettä. Uuden toimintakäsikirja-alustan määrityksessä lähdettiin liikkeelle standardeista ja edellisen version olemassa olevista toiminnallisuuksista. Niiden lisäksi uuteen versioon on suunniteltu hyödyllisiä toiminnallisuuksia asiakkaiden sekä yrityksen laadunhallinnan asiantuntijoiden toivomusten pohjalta.

Toimintakäsikirja IMS-järjestelmässä –projektin tavoitteena oli tuottaa mahdollisimman kattava määrityskuvasto, suunnittelukuvasto ja tärkeimmät perustoiminnallisuudet sisältävä demo uudesta toimintakäsikirja-alustasta. Projekti aloitti uuden, entistä helppokäyttöisemmän ja monipuolisemman toimintakäsikirja-alustan kehitystyön. Ratkaisun toiminnalliset vaatimukset on kuvattu tarkemmin luvussa 4.2.

Laadunvarmistusmenettelynä käytettiin hyväksymistestausta. Sen suorittivat kaksi IMS Business Solutions Oy:n asiakasrajapinnassa työskentelevää henkilöä. Projektin aikana syntynyt demo ei sellaisenaan sovi tuotantoon, mutta sitä voidaan käyttää jatkokehitystyön pohjana. Määritystyössä on pyritty ottamaan huomioon mahdollisimman laajasti osioon suunnitellut toiminnallisuudet. Työn painopisteinä

olivat erityisesti määrittely ja demon toteutus. Demon toteutuksessa etsittiin sopivia ratkaisuja alustan haastavimpiin ongelmiin. Näitä olivat muun muassa käsikirjan lukujen muokkaaminen ja kokonaisen käsikirjan pdf-tulostus.

Projektin ohjelmointikieleksi valittiin Java EE, koska IMS-järjestelmä on toteutettu sillä. Arkkitehtuuri on rakennettu MVC-mallin mukaisesti. Järjestelmä käyttää hyväksi JavaServlet-tekniikkaa ja käyttöliittymä on toteutettu JSP-sivujen avulla. Käyttöliittymässä käytetään myös JSTL-tekniikkaa, JavaScriptia ja JQuery-kirjastoa. Kantakäsittely on hoidettu Hibernaten avulla ja sovelluksen tietokannanhallintajärjestelmäksi sopii MySQL tai MSSQL Server 2000/2005/2008.

IMS-järjestelmän käyttäjät luovat toimintakäsikirjan suoraan selaimessa, joten hyvän tekstieditorin valinta oli ensiarvoisen tärkeää. Editoriksi valikoitui JavaScript-pohjainen avoimella lähdekoodilla toteutettu CKEditor. Se on monipuolinen, ja sen toiminnallisuuksia on mahdollista laajentaa myös itse tehdyillä lisäosilla. CKEditor tuottaa html -koodia. Koska CKEditorin tuottama html-koodi ei ole välttämättä validia, tarvittiin työkalu html:n muuttamiseksi validiksi xhtml:ksi. Siihen valittiin JTidy-niminen Java-kirjasto. Xhtml:n tulostaminen pdf-formaattiin on työlästä, koska järjestelmän käyttäjä on voinut tuottaa järjestelmään lähes mitä tahansa sisältöä. Toimintakäsikirja-alustan vanhassa toteutuksessa pdf-tulostus oli toteutettu käsin iText-nimisen Java-kirjaston avulla. Tähän haluttiin löytää parempi ratkaisu, ja lyhyen selvityksen jälkeen löytyi lupaava avoimen lähdekoodin Java-kirjasto xhtml renderer (Flying Saucer). Se vähentää huomattavasti tarvittavan lähdekoodin määrää ja tarjoaa uusia vaihtoehtoja pdf-tulostuksen parantamiseen.

Projekti onnistui sille asetetuissa tavoitteissa erinomaisesti. Sen tuloksena syntyi demo, jonka avulla pystyy luomaan ja muokkaamaan sisältöä toimintakäsikirjaan. Demon kehityksen aikana on ratkaistu muutamia haastavimpia toimintakäsikirja-alustan kehitykseen liittyviä kysymyksiä. Projektin sivutuotteena valmistui myös yleistettävissä oleva työkalu, jonka avulla minkä tahansa html-sivun saa muunnettua pdf-muotoon räätälöityjen ylä- ja alatunnisteiden kanssa. Demon toiminta on ollut luotettavaa, ja siitä saatu palaute on ollut positiivista. Tämän projektin tuloksia tullaan käyttämään toimintakäsikirja-alustan jatkokehityksen pohjana.

2 Toimintakäsikirja

2.1 Laatujärjestelmä ja standardit

Laatujärjestelmä (toimintajärjestelmä) on eräänlainen yläjärjestelmä, jossa kuvataan organisaation yhteiset pelisäännöt ja parhaat menettelytavat. Siinä kuvataan myös, kuinka erilaisissa poikkeustilanteissa tulee toimia. (Elamo 2010, 12-13.) Laatujärjestelmä on väline, jolla varmistetaan että tuote täyttää sille asetetut vaatimukset (Biaudet 1995, 22.)

On olemassa paljon erilaisia standardeja, jotka määrittelevät erilaisten laatujärjestelmien kriteerejä. Biaudetin (1995, 3) mukaan yksi tunnetuimmista laadun kehittämisen työkaluista on ISO 9000 standardisarja. Siihen kuuluu 5 erilaista standardia

Laatujärjestelmän tarkoitus on menettelyjen ja ohjeiden laatiminen ja soveltaminen tarpeen mukaan (Biaudet 1995, 22). Laadunvarmistamisen tavoitteena on, että tuote on asiakkaan vaatimusten mukainen (Voutilainen, Ritola & Moisio 2001, 12). Tärkeimpiä laadunhallinnan periaatteita ovat asiakaslähtöisyys, johtajuus, ihmisten osallistuminen, prosessimainen lähestymistapa, järjestelmäkeskeinen johtamistapa, jatkuva parantaminen, tosiasioihin perustuva päätöksenteko ja yhteistyökumppanuus toimittajien kanssa (Voutilainen ym. 2001, 16).

2.2 Laatukäsikirja

Laatukäsikirjan lähtökohtana ovat aina organisaation omat tarpeet. Parhaimmillaan se antaa hyvän kokonaiskuvan koko organisaation toiminnasta kokonaisuutena. Sitä voi käyttää myös apuvälineenä perehdytyksessä ja työn suorittamisessa. Ulkoasun tulee olla selkeä ja jäsennelty. Käsikirjan on hyvä pysyä mahdollisimman yleisellä tasolla, jolloin sitä tarvitsee päivittää melko harvoin. (Lecklin 2002, 32-35.)

Esimerkiksi ISO 9001 –standardi vaatii yritystä ylläpitämään laatukäsikirjaa. Se myös asettaa laatukäsikirjalle tiettyjä vaatimuksia. Käsikirjan tulee:

- kertoa organisaation visiosta, arvoista, tavoitteista ja käytännöistä
- kertoa, miten toimintajärjestelmä on suunniteltu
- kertoa prosessien välisistä yhteyksistä
- kuvata yrityksen roolit
- auttaa koulutuksessa
- olla työkalu parannusmahdollisuuksien arviointiin
- olla keino havainnollistaa yhteensopivuus standardien kanssa. (Hoyle 2003, 122.)

Hyvä laatukäsikirja myös kertoo laatujärjestelmän laajuuden mukaan lukien yksityiskohdat ja reunaehdot. Se dokumentoi menetelmät laadunhallintajärjestelmälle ja kuvaa prosessien väliset vuorovaikutukset. Laatukäsikirjan tulee sisältää ainakin johdanto, yleiskatsaus liiketoimintaan, liiketoiminnan prosessit ja toimintaa kuvaavat matriisit. (Hoyle 2003, 123.)

Termejä laatukäsikirja ja toimintakäsikirja käytetään paljon sekaisin. Niiden voidaan kuitenkin tulkita tarkoittavan samaa asiaa. Nykyään laatukäsikirjoista käytetään pääasiassa nimitystä toimintakäsikirja.

3 Toteutusalusta ja tekniikat

Projektin ohjelmointiympäristönä käytettiin Java EE:tä. Ympäristön tavoitteena on tarjota kehittäjille mahdollisimman laaja ohjelmointirajapinta ja samalla lyhentää kehitysaikaa, vähentää sovelluksen monimutkaisuutta ja parantaa sen suorituskykyä. Java EE on kehitetty monien tahojen yhteistyönä. Tämän ansiosta alusta on vakaa ja sen yhteensopivuus muihin alustoihin on laaja. (Oracle 2010, Chapter 1.)

3.1 MVC-malli

MVC on arkkitehtuurimalli. Sen avulla sovellus jaetaan kolmeen toisistaan riippumattomaan osaan. Malli (model) sisältää sovelluksen liiketoimintalogiikan. Se voi palvella useita erilaisia näkymiä. Näkymä (view) on loppukäyttäjälle näytettävä käyttöliittymäkomponentti. Ohjain (controller) on se komponentti, joka vasta käyttäjän syötteeseen. Se muuntaa käyttäjän pyynnön mallille ymmärrettävään muotoon, ja määrittelee, kuinka käyttöliittymä reagoi tapahtumiin. (Allamaraju ym. 2001, 714-715.)

Yksi MVC-mallin vahvuuksista on luokkien uudelleenkäytettävyys. Sovellus joustaa myös paremmin muutoksen edessä, kun sisältö on jaettu loogisiin kerroksiin. Mallia noudattamalla työ on helpompi jakaa osiin, jolloin saman toiminnallisuuden parissa voi työskennellä useita henkilöitä yhtä aikaa (esimerkiksi graafinen suunnittelija voi suunnitella käyttöliittymän ulkoasua samalla kun ohjelmoija suunnittelee liiketoimintalogiikkaa). Mallia sovellettaessa on tavoitteena myös se, ettei samaa ohjelmakoodia tarvitsisi kirjoittaa useaan kertaan. (Allamaraju ym. 2001, 715; Huhtala 2010, 10-11.)

3.2 JavaServletit ja Java Server Pages (JSP)

Servletti on Java-luokka, joka on alun perin luotu generoimaan HTML-koodia erilaisista lähteistä. Koska se on varsin kömpelö tapa suurien sovellusten rakentamiseen ja vaikea ylläpitää, Sun kehitti JSP-sivut. Niiden tarkoitus on sama kuin Microsoftin ASP-tiedostoilla. JSP-sivun avulla käyttöliittymän rakentaminen tehdään yhdessä keskitetyssä paikassa. JSP ja Servletit ovat kaksi eri tapaa toteuttaa sama päämäärä. JSP-

sivujen sisältö käännetään ajon aikana sitä vastaavaksi Servletiksi, ja ajetaan vasta sitten. (MassLight 2000-2001.)

JSP koostuu HTML- ja XML-sisällöstä. JSP-sivut ovat melko yksinkertaisia rakentaa, mutta niiden etuna on mahdollisuus käyttää Javaa ja Java Server –ohjelmakirjastoja dynaamisen sisällön luomiseen. Ne käyttävät hyödykseen JavaBeaneja, joiden avulla sivulla on helppo näyttää dynaamista sisältöä. (Webber 2000.)

Servlettejä ja JSP-näkymiä voi hyödyntää MVC-mallin mukaisessa sovellusarkkitehtuurissa. Molemmat ovat erittäin suosittuja ja paljon käytettyjä tekniikoita.

3.3 Hibernate

Suurin osa erilaisista sovelluksista käyttää muistiin tallennettua dataa. Historian kuluessa näiden ohjelmien kirjoittamiseen on käytetty monia erilaisia ohjelmointikieliä ja vähintään yhtä monia tapoja tallentaa tietoa muistiin. Ajan myötä ohjelmointikielet ovat muuttuneet tukemaan hierarkkisuutta ja olioajattelua yhä enemmän. Tietojen tallennuksessa taas suosituimmaksi tavaksi on jäänyt relaatiotietokanta. Sovelluskehittäjät ovat joutuneet kehittämään erilaisia ratkaisuja, joiden avulla nämä kaksi eri tyyppistä ympäristöä saadaan yhdistettyä toisiinsa. Hibernate on kehitetty helpottamaan tätä ohjelmointityötä. Se on siis ohjelmistokehys, joka yhdistää relaatiotietokannan ja Java-luokat toisiinsa. (Iverson 2004, 1.)

Hibernaten tavoite on vapauttaa kehittäjä mekaanisesta tiedon säilytykseen liittyvästä ohjelmointityöstä. Hibernate käyttää tavallisia Java-luokkia ja tietokantatauluja kuvaavia XML-tiedostoja. Niiden avulla se tarjoaa suoraan olioajattelun mukaisen näkemyksen tietokannan rakenteesta. Hibernateen on luotu oma kyselykieli, HQL (Hibernate Query Language), joka helpottaa tietokantakyselyitä. (Iverson 2004, 2.)

3.4 IMS-järjestelmä

IMS-järjestelmä on ohjelmisto, jonka avulla asiakasyritys voi luoda ja hallita omaa kokonaisvaltaista toimintajärjestelmää (laatujärjestelmää). Ohjelmisto pyrkii tarjoamaan

käyttäjille monipuolisia työkaluja erilaisiin dokumentointitarpeisiin. IMS-järjestelmässä on viisi pääosiota: prosessit, käsikirjat, dokumentit, mittarit ja raportit.

Prosessit

Prosessiosiota käytetään organisaation prosessien kuvaamiseen. Jokaisella kuvatulla prosessilla on kolme tietosivua: yhteenvetosivu, prosessikaavio ja vaiheiden kuvaukset. Yhteenvetosivulla esitetään prosessin perustiedot. Prosessikaavio kuvaa prosessin kulun kaaviona ja vaiheiden kuvaukset –sivulla kuvataan prosessin vaiheet ja aliprosessit tarkasteltavan prosessin näkökulmasta.
Prosesseja muokataan vastaamaan kehittyviä liiketoimintaprosesseja. Järjestelmä säilyttää myös vanhat versiot, mutta tavallisille katsojakäyttäjille näytetään ainoastaan viimeisin versio. (Lakso 2010, 3)

Käsikirjat

Käsikirjat-osiossa sijaitsee yrityksen toimintakäsikirja. Sinne yritys kuvaa oman toimintansa ja periaatteensa. Toimintakäsikirjan mahdollisesta sisällöstä löytyy lisää luvusta 2.2.

Dokumentit

Dokumenttiosio sisältää sekä versioitavia asiakirjoja, että ns. tallenteita. Versioitavia asiakirjoja ovat esimerkiksi organisaatiossa itse laaditut ja ulkopuolelta tulleet viralliset asiakirjat ja toimintaohjeet. Ohjelmisto tukee asiakirjojen käsittelyä versiointi-, tarkastus- ja hyväksymiskäytännöillä. Lisäksi dokumenteille voi asettaa voimassaoloajan, jonka jälkeen ne siirtyvät automaattisesti arkistoon.

Versioimattomiksi asiakirjoiksi luokitellaan kaikki muut (kertaluontoiset) tiedostot, jotka järjestelmään lisätään. Sellaisia dokumentteja ovat esimerkiksi pöytäkirjat ja erilaiset kertaluontoiset tiedostot.

Mittarit

Mittarit-osion avulla organisaatio voi mitata omia prosessejaan ja toimintaansa. Mallina voi käyttää esimerkiksi tasapainoisen mittariston (BSC, Balanced Scorecard)

periaatteita. Mitattavia asioita voi olla esimerkiksi toimitusten ja virheiden lukumäärät, rahavirrat ja asiakastyytyväisyys. (Lakso 2010, 4.)

Raportit

Raportit-osio on sähköinen palaute- ja lomakejärjestelmä. Sen avulla voidaan hallita esimerkiksi reklamaatiot, sisäiset auditoinnit ja katselmukset, korjaavat ja ehkäisevät toimenpiteet, aloitteet ja palautteet. (Lakso 2010, 4)

4 Vaatimukset toimintakäsikirja-alustalle

4.1 Käytettävyys

IMS-järjestelmän tavoitteena on olla helppokäyttöinen työkalu laatujärjestelmän luomiseen. Järjestelmä sisältää monipuolisesti paljon eri toiminnallisuuksia. Helppokäyttöisyys onkin ehdottomasti tärkein vaatimus järjestelmän kehittämisessä.

Käytettävyys on tuotteen laatuominaisuus, jolla kuvataan, kuinka helppoa ja tehokasta sen käyttäminen on. Käytettävyys on abstrakti ja monisyinen ominaisuus, jonka arviointi vaatii sen purkamista pienemmiksi osakokonaisuuksiksi. ISO 9241–11 – standardi määrittelee käytettävyyden seuraavasti: ”käytettävyys on mittari, jolla mitataan tuotteen käytön tuottavuutta, tehokkuutta ja miellyttävyyttä”. Nämä ominaisuudet arvioidaan aina suhteessa käyttäjiin sekä työhön ja käyttöympäristöön, joille ja joihin tuote on tarkoitettu.(Sinkkonen 2004.) Käytettävyyttä voi mitata erilaisilla käytettävyystesteillä.

Jakob Nielsen on luonut käytettävyyden mittaamiseksi listan tärkeimmistä heuristiikoista, eli ”nyrkkisäännöistä”. Niitä voi käyttää heuristisen analyysin pohjalla, tai esimerkiksi muistilistana sovelluksen käyttöliittymän suunnitteluvaiheessa. Nielsenin 10 heuristiikkaa ovat:

1. Käyttäjän tulee aina tietää sijaintinsa ja muut mahdollisuudet liikkua sivustolla. Navigointiin liittyvän tiedon tulee saavuttaa käyttäjä kohtuullisessa ajassa.

2. Sivuston tulee olla intuitiivinen sekä käyttää tuttuja termejä ja käsitteitä.

3. Käyttäjällä tulee olla tunne hallinnasta ja valinnanvapaudesta. Palaamisen pääsivulle tai edelliselle sivulle tulee olla helppoa ja nopeaa.

4. Järjestelmän tulee olla johdonmukainen. Sivustolla tulee käyttää yksiselitteisiä sanoja ja käsitteitä. Toimintojen tulee olla standardien mukaisia.

5. Virheiden tekemisen tulee olla hankalaa. Järjestelmän huolellinen suunnittelu minimoi virhetilanteet. Käyttäjää tulee informoida tulevasta, ennen kuin hän suorittaa toiminnon.

6. Eri vaihtoehtojen tulee olla tunnistettavia, jotta käyttäjän ei tarvitse muistaa ulkoa monia asioita. Järjestelmän käyttämistä koskevien toimintaohjeiden tulee olla selkeästi näkyvillä.

7. Järjestelmän tulee olla joustava ja tehokas: aloittelijalle helppo ja asiantuntijalle tehokas. Käyttäjällä tulisi olla mahdollisuus muokata sivuston käytettävyyttä hänelle sopivaksi.

8. Ulkoasun pitää tuoda esiin ainoastaan olennainen informaatio. Liian runsas sisältö voi hukuttaa olennaisen asian, jota käyttäjä on ollut etsimässä.

9. Käyttäjän tulee saada selkokieliset ohjeet virheen laadusta sekä sen korjaamisesta.

10. Jos käyttöohjeita tarvitaan, niiden tulee olla saatavilla ja helppokäyttöiset. Ohjeiden tulee olla myös tiiviitä ja käytännönläheisiä. (Nielsen 1994; Ollikainen 2008.)

4.2 Ratkaisun toiminnalliset vaatimukset

Toiminnalliset vaatimukset ratkaisulle on esitetty vaatimusmäärityksdokumentissa (liite 2). Kaikki vaatimukset tuli huomioida määritysvaiheessa. Vaatimukset priorisoitiin yrityksen toimesta, ja toteutusvaiheeseen otettiin kaikki tärkeimmiksi priorisoidut toiminnallisuudet. Niiden lisäksi demoon toteutettiin pdf-tulostus, koska työssä haluttiin löytää siihen uusia, entistä toimivampia ratkaisuja.

Demoon toteutettiin uuden toimintakäsikirjan luominen, poistaminen ja valitseminen sekä uuden luvun luominen, poistaminen, valitseminen ja muokkaaminen. Lisäksi toteutettiin toimintakäsikirjan html- ja pdf-tulostukset. Lukuja pystyy luomaan toistensa alaluvuiksi, eli vaatimus hierarkkisuuden mahdollisestamisesta toteutettiin myös.

5 Valitut työtavat

5.1 Projektin ohjaus

Projekti tehtiin IMS Business Solutions Oy:n toimeksiannosta ja työ toteutettiin yrityksen laitteistolla ja tiloissa. Projektin etenemistä valvoi ohjausryhmä, joka koostui opinnäytetyön tekijästä, IMS Business Solutions Oy:n edustajasta ja HAAGA-HELIAn edustajasta. Opinnäytetyöprojekti aloitettiin aihe-ehdotuksen jättämisellä 26.10.2010. Virallisesti projekti alkoi aloituskokouksessa 9.11.2010. Projekti päätettiin päätöskokouksella, joka pidettiin 4.3.2011.

5.2 Valitut menetelmät

Projektissa noudatettiin HAAGA-HELIAn opinnäytetyöohjeita ja vakiintunutta projektinhallintakäytäntöä, tarvittaessa hieman soveltaen.

Projektissa keskityttiin olemassa olevan IMS-järjestelmän jatkokehitykseen. Siksi muualla sovelluksessa käytetyistä tekniikoista tai arkkitehtuuriratkaisuista ei voinut juuri poiketa. IMS-järjestelmä pyörii Apache Tomcat –palvelimella. Tietokantapalvelimena käytetään joko MySQL- tai Microsoft SQL Server –tietokantaa. Ohjelmointikielenä käytetään JavaEE:tä.

Ohjelmiston rakenne on tehty noudattaen MVC-mallia. Tietokantarajapintana käytettiin Hibernate-sovelluskehystä. Http- pyynnöt käsittelee Servlet, joka ottaa sen vastaan, tekee tarvittavat kyselyt ja palauttaa vastauksena yleensä JSP-sivun. JSP-sivujen dynaamista sisältöä varten käytetään JavaBean-olioita. JSP-sivuilla käytetään myös JSTL-kirjastoa, JavaScriptiä sekä JQuerya. Dynaamisen sisällön luontiin jouduttiin paikoin käyttämään myös Javaa, koska JSTL ei tarjonnut kaikkia tarvittavia työkaluja.

5.3 Projektin kulku

Projekti koostui yhteensä neljästä eri vaiheesta. Tarkempi kuvaus projektin ajoituksesta löytyy projektisuunnitelmasta ja liitteestä 1 (loppuraportti).

Projektin ensimmäinen vaihe oli projektin käynnistys, jonka aikana mietittiin sopivaa työn rajausta ja toteutettiin projektisuunnitelma. Ensimmäinen vaihe päättyi projektin aloituskokoukseen, jonka jälkeen projekti virallisesti aloitettiin.

Toinen vaihe oli määritysvaihe, joka sisälsi toimintakäsikirja-alustan vaatimusmäärityksen yhteenvedon ja varsinaisen määrityksen. Vaiheen tuloksena syntyi määrityskuvasto, joka katselmoitiin asiakkaan toimesta. Vaiheen lopuksi pidettiin ohjauskokous, jossa määrityskuvasto hyväksyttiin ja projekti eteni suunnittelu- ja toteutusvaiheeseen.

Määritysvaiheen jälkeen vuorossa oli kolmas vaihe, jonka aikana tehtiin suunnittelukuvasto ja toteutettiin demo uudesta toimintakäsikirja-alustasta. Demo sisälsi IMS:n tärkeimmäksi priorisoimat käyttötapaukset. Sekä määrityskuvasto, että demo katselmoitiin projektin päätöskokouksessa.

Neljäs vaihe, projektin päättäminen, alkoi hieman ennen kolmannen vaiheen loppumista. Sen aikana syntyi projektin loppuraportti. Tässä vaiheessa kirjoitettiin valmiiksi myös tämä opinnäytetyöraportti ja viimeisteltiin projektin muu dokumentaatio. Viimeisen vaiheen jälkeen ohjausryhmä kokoontui viimeistä kertaa. Kokous totesi projektin päättyneeksi.

5.4 Resurssien käyttö

Tarkempi selostus resurssien käytöstä löytyy Loppuraportista (liite 1). Yleisesti ottaen voidaan kuitenkin todeta, että projekti pysyi aikataulussaan, eikä mainittavia poikkeamia ilmennyt.

6 Tulokset

6.1 Projektin tavoitteet ja saavutetut tulokset

Projektin tärkeimpänä tavoitteena oli tehdä vankka pohjatyö toimintakäsikirja-alustan jatkokehitykselle. Tämä tavoite saavutettiin erinomaisesti. Projektin alussa laadittu määrityskuvasto otti hyvin laajasti huomioon alustaan suunnitellut uudet ominaisuudet. Määritys täytti sille asetetut vaatimukset, ja se tulee olemaan erittäin hyvä pohja alustan jatkokehitystyössä.

Tärkeimmiksi priorisoitujen ominaisuuksien toteutusta varten tehtiin suunnittelukuvasto. Se on hieman suppea, mutta täytti sille asetetut vaatimukset. Sovelluksen arkkitehtuurikäytännöt ja monet ulkoasuun liittyvät yksityiskohdat halutaan pitää mahdollisimman yhtenäisinä, mikä asetti suunnittelulle omat rajoitteensa. Kaikki tärkeimmiksi priorisoidut ominaisuudet saatiin toteutettua ja ne läpäisivät hyväksymistestauksen. Lisäksi toteutettiin vielä toimintakäsikirjan pdf-tulostus. Se oli toteutettu vanhassa toimintakäsikirja-alustassa kömpelösti, eikä sen toiminta ole riittävän luotettavaa. Tähän haluttiin ehdottomasti löytää kevyempi ratkaisu. Kyseinen Java-kirjasto on vaikuttanut erittäin lupaavalta, ja sen avulla on jo nyt ratkaistu useita pdf-tulostukseen liittyviä haasteita.

Käytettävyyden parantamiseksi käytettiin hyväksi Nielsenin 10 heuristiikan listaa. Demolle ei ole tehty heuristista analyysia, mutta heuristiikkoja käytettiin muistilistan tapaan tuotekehityksen aikana.

6.2 Oppimiskokemukset

Olen työskennellyt IMS Business Solutions Oy:ssä sovelluskehittäjänä vuoden 2010 alusta, joten opinnäytetyön tekeminen yritykselle oli luonnollinen vaihtoehto. Sain alusta asti vaikuttaa työn aiheen valintaan ja sopivaan rajaukseen.

Työ eteni tasaisesti koko projektin ajan. Projekti kokosi mukavasti yhteen aiemmin oppimaani, ja itsenäinen ongelmanratkaisutaitoni kehittyi entisestään. Selvisin vastaan

tulleista haasteista pääosin itsenäisellä tiedonhaulla. Sain myös tarvittaessa tukea yrityksen muilta työntekijöiltä mieltä askarruttavissa kysymyksissä. Suurin haaste projektin aikana oli saada muunnettua html-dokumentti ensin xhtml-muotoon ja siitä pdf:ksi. Kyseinen prosessi ei vaatinut varsinaisesti paljon kirjoitettua lähdekoodia, mutta se vaati paljon selvitystyötä ja hiomista ennen kuin toiminto onnistui. Jatkossa pdf-tulostuksen parantaminen tulee olemaan helppoa, koska käytetyt tekniikat ovat minulle nyt tuttuja. Pääsin työn aikana tutustumaan muutamiin minulle uusiin ohjelmakirjastoihin. Lisäksi JavaScript-pohjaisen tekstieditorin rakenne ja muokkaaminen tulivat projektin aikana minulle erityisen tutuksi.

7 Johtopäätökset ja jatkotoimenpiteet

Tämän projektin aikana toteutettu demo ei ole valmis asiakaskäyttöön. Siitä puuttuu kokonaan osa toiminnallisuuksista, jotka valmiiseen versioon halutaan. Projektin aikana eteen tuli myös kysymyksiä siitä, kuinka osa toiminnallisuuksista olisi lopulta järkevintä toteuttaa. Demoa toteutettaessa tavoitteena oli, että jätetään edelleen ovi auki erilaisille vaihtoehdoille. Yksi pohdinnan aihe oli esimerkiksi toimintakäsikirjan kokonaisversiointi. Tietokanta on luotu siten, että toiminto on mahdollinen, mutta käyttöliittymän rakenne on vielä ratkaisematta.

Tämä projekti tarjoaa erinomaisen pohjan toimintakäsikirja-alustan jatkokehitykselle. Demon lähdekoodin päälle voi suunnitella ja toteuttaa lisäominaisuuksia, joista suurin osa on jo käsitelty tämän projektin tuottamassa määrityskuvastossa. Suunnitteluvaiheen alussa on syytä järjestää kokous, jossa käydään läpi toteutettavat ominaisuudet asiakasrajapinnan kanssa. Asiakasrajapinnassa työskentelevät henkilöt saavat jatkuvasti asiakkailta palautetta ja kehitysehdotuksia, ja tuotetta kannattaa lähteä kehittämään heiltä saadun palautteen pohjalta.

Demoon jo toteutettuja ominaisuuksia tulee myös kehittää pidemmälle ennen tuotantoversion julkaisemista. Toimintakäsikirjaan tulee voida linkittää esimerkiksi prosesseja. Lisäksi kuvia tulee voida lisätä järjestelmän sisältä. Näitä ominaisuuksia ei ole vielä toteutettu. Erittäin toimivaksi todettua ohjelmakirjastoa (Flying Saucer), joka tekee xhtml:stä pdf-dokumentteja, kannattaa hyödyntää jatkossa enemmän. Opinnäytetyön sivutuotteena valmistui erillinen Java-luokka, joka tekee kyseisen muunnoksen. Sitä voisi melko helposti käyttää hyväksi myös IMS-toimintajärjestelmäalustan muissa osissa. Asiakasyritykset haluavat dokumentteihin esimerkiksi oman logonsa. Mahdollisuus muokata ylä- ja alatunnistetta parantaisivat pdf-tulosteen arvostusta entisestään. Sisällysluettelon lisääminen pdf-tulosteeseen tulee myös mahdollistaa.

Ennen tätä projektia yhteen IMS-järjestelmään pystyi luomaan ainoastaan yhden toimintakäsikirjan. Uusi alusta ei aseta rajoituksia määrälle, joten alustan nimeksi

muutettiin projektin aikana pelkästään ”käsikirjat”. Tämän ominaisuuden tarjoamia mahdollisuuksia kannattaa myös pohtia tarkemmin. Onko yrityksillä tarvetta tehdä järjestelmään myös muita käsikirjoja, kuin toimintakäsikirjoja? Ja jos on, niin tuoko alustan uusi käyttötarkoitus mukanaan myös uusia tarpeita järjestelmälle? Mahdolliset lisävaatimukset tulee kartoittaa mahdollisimman nopeasti, jotta ne voidaan huomioida osana jatkokehitysprojektia.

Demon testaus suoritettiin hyväksymistestauksen muodossa, ja sen suorittivat kaksi yrityksen asiakasrajapinnassa työskentelevää henkilöä. Demoon ei toteutettu automaatiotestausta, koska sitä ei nähty tarpeelliseksi. Tilanne on täysin päinvastainen tuotantokäyttöön menevällä sovelluksella. Tästä syystä jatkokehitys kannattaa ehdottomasti aloittaa automaatiotestauksen suunnittelusta. Tärkeimmille metodeille on syytä kirjoittaa yksikkötestit, ja liittää projekti jatkuvan integraation Hudson – projektiksi. Tällöin integraatiotestit tehdään automaattisesti määrätyin aikavälein.

Lähteet

Allamaraju, S., Buest, C., Davies, J., Jewell, T., Johnson, R., Longshaw, A., Nagappan, R., Dr. Sarang, P. G., Toussaint, A., Tyagi, S., Watson, G., Wilcox, M., Williamson, A. & O'Connor, D. 2001: Java Server Programming J2EE 1.3 Edition, Wrox Press Ltd. Birmingham, Iso-Britannia.

Biaudet, R., Virtanen V. 1995. ISO 9000 - perusta toiminnan kehittämiselle. Metalliteollisuuden Kustannus Oy. Helsinki.

Elamo, T. 2010. Opinnäytetyö. Laatujärjestelmän rakentaminen rakennustoimisto R.Laiho Oy:lle. Luettavissa: http://urn.fi/URN:NBN:fi:amk-201005209945.

Hoyle, D. 2003. ISO 9000:2000 An A-Z Guide. Butterworth Heinemann. Iso-Britannia.

Huhtala, P. 2010. Opinnäytetyö. Www-sovelluskehitys. Luettavissa: http://urn.fi/URN:NBN:fi:amk-201005037608

Iverson, W. 2004. Hibernate: A J2EE™ Developer's Guide. Pearson Education. Yhdysvallat.

Lakso, J. 2010. Diplomityö. Toimintajärjestelmäalustan hakemistorakenteen hallinnan uudistaminen.

Lecklin, O. 2002. Laatu yrityksen menestystekijänä. 4.p. Gummerus. Jyväskylä.

MassLight. 2000-2001. Developing for the J2EE Tomcat Platform. Luettavissa: http://j2ee.masslight.com. Luettu: 20.2.2011.

Nielsen, J. 1994. Heuristic evaluation. In Nielsen, J. & Mack, R.L. (Eds.). Usability Inspectio Methods. New York: John Wiley & Sons. Luettavissa: http://www.useit.com/papers/heuristic/heuristic_list.html. Luettu 20.2.2011.

Ollikainen, P. 2008. Opinnäytetyö. Käytettävyystutkimus Sue-lehden internetsivustosta Jacob Nielsenin heuristisen evaluoinnin kautta.
Luettavissa:
https://publications.theseus.fi/bitstream/handle/10024/1301/kaytetta.pdf

Oracle. 2010. The Java EE 6 Tutorial. Luettavissa:
http://download.oracle.com/javaee/6/tutorial/doc/bnaaw.html. Luettu: 20.2.2011.

Sinkkonen, I. 2004. Käyttöliittymä ja käytettävyys. Luettavissa:
http://www.adage.fi/blogi/2004/kayttoliittymat-ja-kaytettavyys . Luettu: 22.2.2011.

Voutilainen, P., Ritola O., Moisio J. 2001. IMS-johtamisjärjestelmä. Edita Oyj. Helsinki.

Webber, M. 2000. Building Java Server Pages. Luettavissa:
http://www.webdevelopersjournal.com/articles/jsp_build.html. Luettu 20.2.2011.

Liitteet

Liite 1 – Loppuraportti

Salaiset liitteet

Liite 2 – Vaatimusmääritysdokumentti (salainen)

Liite 3 – Määrityskuvasto (salainen)

Liite4 – Suunnittelukuvasto (salainen)