Testaussuunnitelma

myva

Helsinki 24.8.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (9 op)

Projektiryhmä

Jaana Diakite
Heikki Hämäläinen
Teemu Hynönen
Lasse Nordgren
Sampsa Somerma
Petri Vuorio

Ryhmän ohjaaja

Jari Suominen

Asiakas

Rasmus Nybergh, Creo Consulting

Johtoryhmä

Kimmo Simola

Kotisivu

`http://www.cs.helsinki.fi/group/myva`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	20.5.2007	Ensimmäinen versio
2.0	24.8.2007	Lopullinen versio

Sisältö

1 Johdanto

Myyntihenkilöiden valmennuksen hallintajärjestelmä (myva) on Helsingin yliopiston tietojenkäsittelytieteen laitoksen kesän 2007 ohjelmistotuotantoprojekti. Järjestelmän tilaajana on Creo Consulting. Creo Consulting auttaa yrityksiä myymään paremmin avustamalla uusia yrityksiä myyntistrategian ja tehokkaiden myynnin käytäntöjen luomisessa sekä pidempään toimineita yrityksiä myynnin uudelleensynnyttämisessä. Lisäksi Creo Consulting tarjoaa myyntihenkilöille valmennuksen, jossa opetellaan parempia myynnin käytäntöjä erityisesti uusasiakashankinnassa. Kokemuksen mukaan parhaiten valmennuksen opit menevät perille, mikäli valmennettavat saavat harjoitella oman työnsä ohessa. Tarkoituksena on tuottaa järjestelmä, joka toimii valmennusta tukevana ympäristönä siten, että valmennettavat voivat tehdä työtä samanaikaisesti oikeiden asiakkaiden parissa.

Myyntihenkilöiden valmennuksen hallintajärjestelmän tarkoituksena on sisältää asiakasrekisteri ja rekisterin ylläpito sekä valmennettavan oppimistavoitteet ja muut tiedot. Ohjelmiston tuella valmennettava toteuttaa oppimansa uudet taidot ja toimenpiteet taltioituvat järjestelmään. Valmennettavan työskentelyä ja edistymistä voidaan järjestelmän avulla seurata ja ohjelmisto tarjoaa alkeellisen tavan tukea valmennettavaa vinkkien ja kannustusviestien avulla.

Tämä testausuunnitelmadokumentti toteutettavalle myyntihenkilöiden valmennuksen hallintajärjestelmälle (jäljempänä järjestelmä) perustuu myva projektin vaatimusmäärittely- ja suunnitteludokumentteihin. Dokumentti on tarkoitettu projektiryhmän testaussuunnitelmaksi järjestelmän toteutukseksessa ja järjestelmän varsinaisessa testauksessa. Projekti toteutetaan kesän 2007 aikana myvan projektisuunnitelman mukaisesti. Testaus on jaettu kolmeen vaiheeseen: yksikkötestaus, joka suoritetaan koodauksen aikana, integrointitestaus sekä järjestelmätestaus.

Järjestelmän eri osien toimivuuden validointi löytyy toteutusdokumentista.

Tämän dokumentin toisessa luvussa määritellään dokumentissa käytettävät termit. Kolmannessa luvussa kuvataan yksikkötestaus, neljännessä integrointitestaus ja viidennessä järjestelmätestaus. Kuudes luku käsittelee testausaikataulua.

- Ohjelmointikielenä käytetään PHP:tä ja SimpleTest-testityökalua (http://simpletest.org/)

2 Sanasto

Kappaleessa listataan dokumentissa käytetyt termit ja niiden selitteet.

ACID Määritelmä ominaisuuksista, jotka tietokantatransaktioiden tulee täyttää, jotta tietokannan eheys ja luotettavuus voidaan taata. Nämä on kuvattu tarkemmin asiakirjassa ISO/IEC 10026-1:1992 Section 4.

AJAX Asynchronous JavaScript and XML on ohjelmointitekniikka, jonka avulla voidaan luoda interaktiivisia verkkosivuja. Asynkronisuus tarkoittaa sitä, että verkkosivua voidaan päivittää osa kerrallaan vaatimatta koko sivun uudelleenlataamista. JavaScript viittaa ohjelmointikieleen, jolla AJAX-komponentit yleensä toteutetaan. XML on tiedonkuvauskieli, jota käytetään viestiliikenteessä palvelimen ja selaimen välillä.

CakePHP Nopean sovelluskehityksen (RAD) kehysohjelmisto PHP-kielelle.

CSS Cascading Styling Sheets on erityisesti verkkosivuille kehitetty tyyliohjeiden kuvaustapa. CSS:n avulla verkkosivun ulkoasu (fontit, värit jne.) ja taitto voidaan erottaa varsinaisesta tekstisisällöstä.

CSV Comma-separated values. Tallennustyyppi taulukkomuotoiselle datalle.

GDD Goal-Derived Design. Helsingin yliopiston tietojenkäsittelytieteen laitoksen sekä Interacta Designin kehittämä käyttöliittymäsuunnitteluprosessi, jossa käyttöliittymä suunnitellaan käyttäjän työtehtävien suorituspolkujen optimaalisuutta ajatellen.

HTML Hypertext Markup Language on verkkosivujen kuvauskieli. HTML-määritelmiä on useita, kuten XHTML, joka on XML-syntaksin mukaista HTML-kieltä.

Inoa Fonectan tarjoama palvelu, jolla saa tarkempaa tietoa yrityksestä, esim. yhteystiedot jne.

JavaScript JavaScript on pääasiassa verkkoympäristössä käytettävä komentosarjakieli. JavaScriptin tärkein sovellus on mahdollisuus lisätä verkkosivuille dynaamista toiminnallisuutta.

Järjestelmä Järjestelmän tilaajan tilaaman tuotteen yleisnimitys.

Järjestelmän tilaaja Creo Consulting.

Kampanja Kokonaisuus, joka sisältää tavoitteet, kampanjaan kuuluvat myyjät ja tavoitteiden kohteena olevat kohdeasiakkaat.

Kampanjan tavoite Tavoite, johon myyjä pyrkii jokaisen kohdeasiakkaan kohdalla, esim. päästä sopimukseen tapaamisesta.

Kampanjan tilaaja Yritys, joka ostaa kampanjan palveluna.

Kampanjan vastuuhenkilö Käytetään vaatimusmäärittelydokumentin luvussa 4.2 Tavoitepohjaiset käyttötapaukset yleiskäsitteenä henkilölle, joka vastaa kampanjoiden hallinnollisista asioista. Muissa luvuissa vastuuhenkilön rooli jakaantuu kampanjan ylläpitäjän ja pääkäyttäjän rooleihin.

Kampanjan ylläpitäjä Myyjä, joka luo ja ylläpitää kampanjoita.

Kohdeasiakas Kampanjan puitteissa yhteydenototojen kohteena oleva yritys.

Käyttäjä Yläkäsite, sisältää sekä kampanjan ylläpitäjän, myyjän ja pääkäyttäjän.

Käyttäjäasiakas Yritys joka käyttää järjestelmää vuokralaisena järjestelmää palvelimillaan ylläpitävältä taholta.

Luokittelutieto Myyjän kohdeasiakkaalle antama, yhteydenoton perusteella selviävä kategorisointitieto.

Lähiosuma Haun tuloksena palautettava tieto, joka on täsmälleen tai muistuttaa läheisesti hakuparametria.

Majakka Sähköpostiviestiin upotettava viittaus näkymättömään kuvatiedostoon, joka ladataan järjestelmästä viestiä avattaessa. Latauskutsun perusteella järjestelmä saa tiedon, että vastaanottaja on avannut viestin. Monet sähköpostiohjelmat eivät kuitenkaan lataa kuvia ulkoisista lähteistä.

MVC Model-View-Controller. Ohjelmistosuunnittelussa käytetty arkkitehtuurimalli, jossa tietosisältö, sitä esittävä käyttöliittymä sekä edellisten hallinta on erotettu toisistaan.

MySQL Tietokannan hallintajärjestelmä.

Myyjä Työntekijä, joka kampanjassa on yhteydessä kohdeasiakkaisiin.

Myyntiargumentti Väite, jota voi käyttää avuksi keskustelussa kohdeasiakkaan kanssa.

Myyntisuppilo Myyjän työtä helpottamaan tarkoitettu työkalu. Myyntisuppilon suuaukko sisältää tehdyt soitot. Tämän jälkeen se rajautuu niihin, jotka on tavoitettu sekä niihin, joiden kanssa on päästy tavoitteeseen.

PHP Useilla eri alustoilla toimiva ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristöissä dynaamisten web-sivujen luonnissa.

Profilointitiedot Tietoja myyjien henkilökohtaista ominaisuuksista, esim. luurikammoisuus.

Pääkäyttäjä Ylemmän tason käyttäjä, joka voi luoda myyjiä sekä määrittää näitä kampanjan ylläpitäjiksi. Ei kuitenkaan ole myyjä eikä kampanjan ylläpitäjä.

RAD Rapid Application Development. Ohjelmistojen kehitysmalli, joka tähtää ohjelmiston nopeaan ja helppoon valmistamiseen.

UML Unified Modeling Language. Mm. ohjelmistojen rakenteen kuvaamiseen käytetty graafinen mallinnuskieli.

URL Uniform Resource Locator on merkkijono, jolla osoitetaan www-sivun sijainti verkossa.

Viesti Kohdeasiakkaille lähettävä markkinointisähköpostiviesti.

Y-tunnus Suomen viranomaisten yritykselle myöntämä tunnus, joka yksilöi yrityksen.

Yhteydenotto Myyjä ottaa kohdeasiakkaaseen yhteyttä soittamalla tai lähettämällä viestin.

Yhteyshenkilö Kohdeasiakkaan työntekijä, johon myyjä on suoraan yhteydessä.

XML Extended Markup Language on yleiskäyttöinen kuvauskieli, jossa tietosisällön lisäksi voidaan kuvata tiedon merkitystä. Se on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin.

3 Yksikkötestaus

Yksikkötestauksessa testataan järjestelmän pienimmät kokonaisuudet. Testauksen suorittaa kokonaisuuden ohjelmoija. Testauksen apuna käytämme SimpleTest-testaustyökalua. SimpleTestillä kirjoitetut testit toimivat samalla yksikkötestauksen dokumentaationa.

3.1 Lähestymistapa

Kaikki ryhmän jäsenet suorittavat yksikkötestausta tuottamalleen koodille. Jokainen on siis itse vastuussa koodinsa toimivuudesta ja yksiköiden riittävästä testauksesta järkevillä syötteiden arvoalueilla sekä myös siitä, että kaikki yksikön mahdolliset tilat tulee testatuksi.

3.2 Toiminnallisuuden testaustapa

Koodaajat suorittavat toiminnallisuutta testaavia testejä koodaamilleen yksiköille ja testaavat, että kaikki yksikön tarjoamat palvelut toimivat virheettömästi. Kukin koodaaja testaa koodinsa yksikkö kerrallaan. Kun yksikkö on valmis, koodaaja suorittaa yksikön syötteille arvoalueanalyysin ja testaa, että yksikkö toimii täsmälleen niin kuin sen pitääkin poikkeustilanteet mukaanlukien.

3.3 Valittu kattavuus

Testauksen tulee kattaa kaikki yksikön mahdolliset toiminnot ja tilat. Lausekattavuuden tulee olla 100 %.

3.4 Testisyötteiden valintatapa

Testisyötteet valitaan arvoalueanalyysin perusteella. Arvoalueanalyysi tuottaa osa-arvoalueita, joiden reunoilta valitaan testauksessa käytettävät arvot.

3.5 Hyväksymiskriteerit

Koknaisuus katsotaan hyväksyttävästi yksikkötestatuksi kun testi on suoritettu täsmälleen edellä määriteltyjen sääntöjen mukaisesti, eikä virheitä löydy.

4 Integrointitestaus

Integrointitestauksessa testataan yksiköiden rajapintoja, eli yksiköiden rajapintojen kautta toisilleen tarjoamia palveluja ja niiden oikeellisuutta ja toimivuutta.

4.1 Lähestymistapa

Integrointitestauksessa integoitavien yksiköiden rajapinnat testataan kun yksikkötestaus on suoritettu. Käytämme bottom-up strategiaa, eli ohjelma rakennetaan siten, että ensin luodaan tietokanta ja siitä yksikkö kerrallaan edetään ylöspäin. Kun yksiköt ovat valmiit, testaan integrointitestauksessa niiden yhteistyön toimivuus rajapintojen avulla.

4.2 Rajapintojen testaustapa

Integrointitestauksessa testataan yksiköiden toisilleen tarjoamat palvelut. Kaikille palveluille tehdään arvoalueanalyysi sopivien testisöytteiden löytämiseksi. Näitä testisyötteitä annetaan kutsuvan palvelun kautta testattavalle rajapinnalle, eli testataan kahden integroitavan yksikön välistä yhteistyötä.

4.3 Valittu kattavuus

Kaikki testattavien yksiköiden toisilleen tarjoamat palvelut tulee testata arvoalueanalyysillä saaduilla testisyötteillä.

4.4 Testisyötteiden valintatapa

Testisyötteet saadaan jokaiselle palvelulle suoritetun arvoalueanalyysin avulla.

4.5 Hyväksymiskriteerit

Kahden yksikön integrointitestaus voidaan hyväksyä kun yksiköiden väliset rajapinnat ja poikkeustilanteet on testattu. Koko järjestelmän integrointitestaus on valmis kun kaikki yksiköt on integroitu yhteen ja ne on testattu edellä mainittujen vaatimusten mukaisesti.

5 Järjestelmätestaus

Järjestelmätestaus suoritettaan viimeisenä testausvaiheena, kun järjestelmää voidaan testata yhtenä kokonaisuutena.

5.1 Lähestymistapa

Järjestelmätestaus tehdään käyttöliittymän kautta kokonaiselle järjestelmälle, eikä toteutukseen enää puututa.

5.2 Vaatimusten testaustapa

Järjestelmätestauksen tarkoitus on tarkistaa, että kaikki vaatimusmäärittelyssä toteutettavaksi määritellyt järjestelmän toiminnot ja palvelut saadaan katettua järjestelmän toimintojen avulla. Kukin ohjelman palvelu testataan arvoalueanalyysin avulla valituilla syötteillä ja testauksen apuna käytetään laajennettuja käyttötapauksia ja niistä tehtyjä päätöstauluja.

5.3 Valittu kattavuus

Järjestelmätestauksen tulee kattaa kaikki toiminnalliset vaatimukset, jotka on valittu toteutettavaksi. Myös järjestelmän kaikki mahdolliset tilat tulee testata.

5.4 Testisyötteiden valintatapa

Testisyötteet valitaan toiminnallisten vaatimusten arvoalueanalyysin avulla saatujen reuna-arvojen perusteella.

5.5 Hyväksymiskriteerit

Järjestelmätestaus voidaan hyväksyä kun kaikki ohjelman tarjoamat palvelut ja toiminnot on testattu ja hyväksytty. Tämän jälkeen ohjelman testaus on suoritettu.

6 Testausaikataulu

6.1 Tarkistuspisteet

Yksikkötestaus tehdään koodin kirjoittamisen ohessa. Yksikkötestaus siis alkaa yhtäaikaa toteutusvaiheen kanssa ja loppuu, kun viimeinenkin yksikkö on koodattu ja testattu. Yksikkötestaus on täten riippuvainen toteutusaikataulusta joten viivästykset toteutuksessa aiheuttavat myös testausaikataulun venymisen.

Integrointitestaus suoritetaan, kun yksikkötestaus on valmis. Varsinaisia tarkistuspisteitä ei ole, vaan jokainen raportoi omasta edistymisestään työn edetessä.

Järjestelmätestaus aloitetaan kun integrointitestaus on valmis ja ohjelma on toimintakuntoinen.

7 Testautuksen toteutuminen

Järjestelmän yksikkötestaus tapahtui suunnitellulla tavalla. Järjestelmän malliluokkien metodeille kirjoitettiin SimpleTest-yksikkötestiluokat. Testauksessa osoittautui jonkin verran ongelmalliseksi se, että testien läpimeno vaati valmiin testidatan syöttämisen. Tietokantaan tämän jälkeen tehdyt muutokset aiheuttivat sen, että testit eivät enää menneet läpi. Osa testeistä oli tehty niin, että tätä ongelmaa ei esiintynyt. Yksikkötesteissä ei täyteen lausekattavuuteen päästy.

Järjestelmän kontrollereita, näkymiä sekä JavaScript-komentosarjoja ei yksikkötestattu lainkaan. Erityisesti kontrollerien näkökulmasta tätä voidaan pitää merkittävänä puutteena, sillä CakePHP:n RAD-lähestymistavassa raja mallin ja kontrollerin välillä on hieman häilyvä. Järjestelmässä havaittiin koodin jäädytyksen jälkeen joitakin kriittisiä bugeja, jotka olisivat luultavasti paljastuneet yksikkötestauksen kautta. Kontrollerien testauksessa on kuitenkin otettava huomioon, että CakePHP:n kontrollereita on erittäin vaikea integroida osaksi yksikkötestausta. Käytännön ongelmaksi muodostui mm. se, kuinka kontrolleriluokka on mahdollista instansioida muun kuin ohjelmistokehyksen kautta. Jos aika olisi riittänyt, olisi ohjelmistoa pitänyt uudelleenkirjoittaa niin, että mahdollisimman suuri osa toimintalogiikasta olisi siirretty malleihin helpottamaan testausta.

Järjestelmän integraatiotestausta ei toteutettu suunnitelman mukaisesti.

Johtuen todella kireästä aikataulusta ja järjestelmän osien myöhäisestä valmistumisesta, täydellistä järjestelmätestausta ei suoritettu. Tähän voidaan pitää merkittävänä osasyynä sitä, että testausta ei tehty kattavasti koodaamisen ohella. Järjestelmätestaus olisi ollut mahdollista aloittaa jo aikaisemmassa vaiheessa toteutusvaihetta, toisin kuin suunnitelmassa oli kirjattu. Järjestelmätestausta tehtiin laajemmassa mielessä vasta projektin aivan viime vaiheissa, jolloin koodi oli jo jäädytetty. Tästä tilanteesta johtuen koodia jouduttiin muuttamaan vielä lopullisen jäädytyksen jälkeen. Testauksessa havaitut ei-kriittiset bugit on kirjattu osaksi toteutusdokumenttia.

Jälkianalyysina voidaan todeta, että projektissa olisi pitänyt jäädyttää koodi uusien ominaisuuksien osalta vähintään viikkoa aikaisemmin, jotta testaukselle ja bugien korjaamiselle olisi jäänyt enemmän aikaa. Nykyistä versiota ei voi pitää riittävän hyvin testattuna.