Projektisuunnitelma

NJC2

Helsinki 1.2.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Eero Anttila
Olli Jokinen
Jesse Liukkonen
Jani Markkanen
Jere Salonen
Jouni Tuominen

Asiakas

Olli Lahti

Johtoryhmä

Juha Taina

Kotisivu

`http://www.cs.helsinki.fi/group/njc2`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	1.2.2004	Ensimmäinen versio

Sisältö

1 Projektin kuvaus **2**

2 Organisaatio **2**

3 Toimintasuunnitelma **2**

3.1 Projektin aikataulu 2

3.2 Tarkistuspisteet 3

3.3 Kriittinen polku 3

4 Projektin koon arviointi **3**

4.1 Ohjelmiston kokoarvio LOC-menetelmällä 3

4.2 Ohjelmiston kokoarvio FP-menetelmällä 4

5 Riskien hallinta **6**

5.1 Projektiryhmään kohdistuvat riskit 6

5.2 Projektin hallintaan ja hallintoon liittyvät riskit 6

5.3 Tekniikkaan liittyvät riskit 7

5.4 Asiakkaaseen liittyvät riskit 7

6 Menetelmät ja työkalut **7**

7 Laadunvalvonta **8**

7.1 Dokumenttien laatu 8

7.2 Ohjelmiston laatu 8

Liite 1

1 Projektin kuvaus

Projektiryhmän tarkoituksena on suunnitella ja toteuttaa Nordic Journal of Computing -lehden julkaisutoimintaa helpottava ohjelmisto. Tarjottujen artikkelien toimituksellista käsittelyä ja sidosryhmien välistä kommunikointia halutaan saada nykyistä automatisoidummaksi.

Järjestelmän tarkoituksena on hoitaa lähetetyn aineiston hallinto ja kirjanpito sekä toimituksen ja asiantuntijoiden välinen kommunikaatio; varsinaista julkaisuvälinettä ei toteuteta. Käyttäjinä toimivat aineiston tuottajat (eli artikkelien kirjoittajat), lehden toimitus sekä joukko asiantuntijoita. Artikkeleita voi tarjota kuka tahansa.

Webbisivujen kautta lehdelle tarjotut artikkelit ja niiden tiivistelmät rekisteröidään saapuneiksi, minkä jälkeen toimitus valitsee muutaman sopivaa alaa edustavan asiantuntijan ja lähettää artikkelin eteenpäin odottamaan asiantuntijalausuntoa. Hyväksyttyään toimeksiannon asiantuntija saa artikkelin arvioitavakseen. Päätös artikkelin julkaisusta tai palauttamisesta korjattavaksi tehdään lausuntojen perusteella NJC:n toimituksessa.

Järjestelmä tarjoaa artikkelien kirjoittajille mahdollisuuden seurata omaa artikkeliaan koskevan käsittelyprosessin etenemisen seuraamiseen. Toimitus näkee käynnissä olevista prosesseista kokonaiskuvan ja saa niistä tarpeellista tietoa. Asiantuntijat näkevät arvioitavaksi tarjotut artikkelit. Järjestelmän tarkoitus on myös parantaa asiantuntijoiden reagoimista tarjottuihin artikkeleihin sovittujen tarkastusaikojen sisällä.

Kirjoittajat eivät pysty vakoilemaan toistensa aktiviteettia järjestelmän kautta, ja asiantuntijat pysyvät kirjoittajille anonyymeina.

2 Organisaatio

Projektin asiakkaana on Olli Lahti Helsingin yliopiston tietojenkäsittelytieteen laitokselta. Ohjaajana toimii Juha Gustafsson. Ryhmän jäseniä ovat projektin ohjelmistopäälliköt Olli Jokinen ja Jouni Tuominen, ohjelmistoarkkitehdit Eero Anttila ja Jani Markkanen, laatupäällikkö Jesse Liukkonen sekä projektipäällikkö Jere Salonen.

3 Toimintasuunnitelma

3.1 Projektin aikataulu

Projektin aikataulusta on laadittu Gantt-kaavio (Liite 1) ja sen mukaan pyritään pysymään aikataulussa. Prosessimallina käytetään vesiputousmallia. Projekti pyritään saamaan valmiiksi ylläpitodokumentteja myöten 13.5.2004. Projektiryhmä kokoontuu vähintään kaksi kertaa viikossa: maanantaisin kello 16-18 sekä torstaisin kello 10-12. Luokkana on B436. Jos jollekin ryhmäläisistä tulee poikkeuksellinen este päästä kokoukseen, pyritään sitä siirtämään kaikille sopivaan ajankohtaan.

3.2 Tarkistuspisteet

Tärkeimmät tarkistuspisteet:

- Vaatimusmäärittely 19.2. (katselmointi 23.2.)
- Suunnittelu 25.3.
- Toteutus: 22.4.
- Integrointi- ja järjestelmätestaus 6.5.
- Käyttöohje, ylläpitodokumentti ja loppuraportti 13.5.

Jäädytykset ovat aina viikon päästä dl:stä. Tarkistuspisteitä on projektisuunnitelman, vaatimusanalyysin, suunnittelun, koodin valmistumisen, testauksen ja loppuraportin yhteydessä. Tarkistuspiste on vaiheen päättyminen, eli eri asia kuin seurantakokous.

3.3 Kriittinen polku

Polkuja on toistaiseksi vain yksi, joten kriittinen polku on se sama.

4 Projektin koon arviointi

Ohjelmiston koko päädyttiin arvioimaan kahdella toisistaan riippumattomalla arviointimenetelmällä: LOC- ja FP-menetelmällä. LOC-menetelmällä ohjelmiston koko arvioidaan ohjelmiston sisäisten ominaisuuksien perusteella, kun taas FP-menetelmällä arviointi tapahtuu ulkoisten ominaisuuksien perusteella. Tästä syystä voidaan olettaa arvioiden eroavan toisistaan huomattavastikin.

4.1 Ohjelmiston kokoarvio LOC-menetelmällä

LOC-menetelmällä arvioidaan ohjelmiston koodirivien määrää. Ohjelmisto jaetaan karkeasti pienempiin komponentteihin tarkemman rivimäärän arvioinnin mahdollistamiseksi. Arviot perustuvat suurelta osin ohjelmistotuotantoprojekti-kurssin vanhojen projektien kokoarvioihin.

- Käyttöliittymät
  - Artikkelien kirjoittajat - 200 riviä
  - Toimitus - 300 riviä
  - Asiantuntijat - 200 riviä

- Artikkelien vastaanotto - 500 riviä
- Artikkelien hallinta - 900 riviä
- Asiantuntijoiden hallinta - 400 riviä
- Asiantuntijoiden valinta - 200 riviä
- Tietokannan hallinta - 300 riviä

Ohjelmiston kokoarvio on siis yhteensä 3000 riviä koodia. Ohjelmistotuotantoprojektissa voidaan olettaa jokaisen opiskelijan tuottavan 450-700 riviä koodia, mikä projektiryhmämme kohdalla tarkoittaisi 2700-4200 koodirivin kokoista ohjelmaa. LOC-menetelmän alustava arvio osuu sopivasti juuri tälle välille.

4.2 Ohjelmiston kokoarvio FP-menetelmällä

FP-menetelmällä arvioidaan ohjelmiston toiminnallisuuden määrää. Järjestelmän ulkoiset ominaisuudet kerryttävät toimintopisteitä. Ominaisuudet luokitellaan seuraaviin ryhmiin: syötteet, tulosteet, kyselyt, tiedostot ja ulkoisten liittymät. Näiden ominaisuuksien lukumäärät lasketaan ja luokitellaan toteutuksen vaativuuden mukaan ryhmiin: helppo, normaali ja vaikea. Toimintopistekertoimet (*k) määrittyvät ominaisuuden ryhmän ja luokituksen mukaan.

	helppoja	*k	normaaleja	*k	vaikeita	*k	pisteet
syötteitä	4	3	5	4		6	32
tulosteita	3	4	5	5		7	37
kyselyjä	2	3	4	4		6	22
tiedostoja	2	7	1	10		15	24
liittymiä		5	2	7		10	14
Σ							129

Näin laskettujen peruspisteiden lisäksi määritetään kompleksisuustekijät, jotka koostuvat 14:sta kysymyksestä. Kysymyksiin vastataan seuraavilla vaihtoehdoilla:

- 0p = Ei koskaan
- 1p = Harvoin
- 2p = Toisinaan
- 3p = Keskimääräisesti
- 4p = Merkittävästi
- 5p = Oleellisesti.

1. Onko järjestelmä vikasietoinen? Tarvitaanko luotettavaa tietojen varmistus- ja palautusmenettelyä? = 4p
2. Tarvitaanko tietoliikenneominaisuuksia? = 4p
3. Onko hajautettua prosessinhallintaa? = 0p
4. Onko suorituskyky kriittinen elementti? = 3p
5. Käytetäänkö järjestelmää olemassaolevassa raskaassa käytössä olevassa koneympäristössä? = 4p
6. Tarvitaanko interaktiivista tietojen syöttöä? = 3p
7. Täytyykö interaktiivinen tietojen syöttö synkronoida usealle näytölle tai operaatiolle? = 2p
8. Päivitetäänkö tiedostoja interaktiivisesti? = 1p
9. Ovatko syötteet, tulosteet, tiedostot tai kyselyt monimutkaisia? = 3p
10. Onko ohjelman koodi monimutkaista? = 3p
11. Onko koodi tarkoitettu uudelleenkäytettäväksi? = 2p
12. Ovatko ohjelmiston muunnokset ja asennus mukana suunnitelmassa? = 2p
13. Onko ohjelmisto suunniteltu toimivaksi useina asennuksina eri organisaatioissa? = 0p
14. Onko sovellus suunniteltava käyttäjäystävälliseksi? = 5p

FP-arvo määritetään kaavalla

$$FP = N * (0.65 + 0.01 * \Sigma(F_i)),$$

missä N on peruspisteet ja $\Sigma(F_i)$ kompleksisuustekijöiden pisteiden summa. Eli

$$FP = 129 * (0.65 + 0.01 * 36) = 130.$$

Koska ohjelmisto toteutaan pääasiassa Javalla, voidaan koodirivien määrä laskea arvioimalla, että 53 riviä koodia vastaa yhtä toimintopistettä.

$$LOC = LOC/FP * FP = 53 * 130 = 6890.$$

Kuten huomaamme, FP-menetelmän kokoarvio eroaa suuresti LOC-menetelmän vastaavaasta arviosta, kuten jo aikaisemmin oletettiinkin. Ohjelmiston kokoarviona koodimääränä ilmaisten voidaan siis pitää näitä molempia menetelmiä soveltaen väliä 3000...6890.

5 Riskien hallinta

5.1 Projektiryhmään kohdistuvat riskit

Riski	Todennäköisyys	Vastatoimet	Vakavuus
Vaatimusmäärittely tai suunnittelu epäonnistuu	mahdollinen	Varataan suunnitteluun riittävästi aikaa ja varmistetaan dokumenttien laadukkuus ennen sen jäädyttämistä.	vakava
Joku ryhmän jäsenistä jättää projektin kesken	epätodennäköinen	Psyykataan ihmisiä pysymään kurssilla. Jäsenet ilmoittavat myös, jos aikataulu ei sovi hänelle.	vakava
Ryhmän jäsen sairastuu	todennäköinen	Ryhmän jäsen ilmoittaa sairastumisestaan mahdollisimman aikaisin, jolloin ryhmä voi jakaa hänelle määritettyjä tehtäviä uudestaan.	siedettävä
Ryhmätyöskentelyn epäonnistuminen	epätodennäköinen	Pyritään selvittämään mahdolliset erimielisyydet ajoissa. Lisäksi panostetaan avoimuuteen ja säännölliseen kommunikointiin ryhmässä.	vakava

5.2 Projektin hallintaan ja hallintoon liittyvät riskit

Riski	Todennäköisyys	Vastatoimet	Vakavuus
Aika loppuu kesken	mahdollinen	Suunnitelaan projektin ajankäyttö hyvin. Tarvittaessa jatketaan projektia suunniteltua viimeistä päivämäärää pidempään.	tuhoisa
Ohjaajan poissaolo kokouksesta	epätodennäköinen	Yritetään soittaa ohjaajalle tai pidetään kokous ilman ohjaajaa.	vähäinen

5.3 Tekniikkaan liittyvät riskit

Riski	Todennäköisyys	Vastatoimet	Vakavuus
Valittu toteutuskieli/-ympäristö ei ole kaikille tuttu	mahdollinen	Valitaan toteutuskieleksi ja -ympäristöksi, kaikille tuttu tai vaihtoehtoisesti sellainen, joka kaikkien on helppo omaksua.	siedettävä
Ongelmia laitteistossa tai palvelinohjelmistossa	epätodennäköinen	Ei varsinaisia vastatoimia, luotetaan tktl:n ylläpitoon ja siihen, että ongelmat saadaan korjattua. Hätätapauksessa voidaan harkita ohjelmiston siirtämistä toiseen järjestelmään.	vakava
Tiedostoja tai tiedostot katoavat TKTL:n palvelimelta	epätodennäköinen	Pidetään varmuuskopioita esimerkiksi ryhmän jäsenten omilla koneilla.	vakava

5.4 Asiakkaaseen liittyvät riskit

Riski	Todennäköisyys	Vastatoimet	Vakavuus
Asiakkaan kaikkia vaatimuksia ei ehditä toteuttaa	erittäin todennäköinen	Vaatimusmäärittelyä tehtäessä annetaan jokaiselle toteutettavalle vaatimukselle prioriteetti ja aletaan karsia pieniprioriteettisia ominaisuuksia, jos näyttää siltä, ettei kaikkea ehditä toteuttaa.	siedettävä
Asiakas on tyytymätön lopputulokseen	mahdollinen	Yritetään tehdä niin täydellisen hyvä ohjelma, ettei asiakkaalla ole mitään valittamista.	siedettävä

6 Menetelmät ja työkalut

Ohjelmistoa tullaan pyörittämään interaktiivisena web-sovelluksena, joten se vaatii Internetiin kytketyn tietokoneen, jossa on asennettuna web-palvelinohjelmisto. Myös palvelinohjelmiston tarvittavat Java-komponentit tulee olla asennettuna, jotta sillä voidaan ajaa JSP-sivuja.

Ohjelmaa tullaan ajamaan tietojenkäsittelytieteen laitoksen alkokrunni-palvelimella (db.cs.helsinki.fi), johon on asennettu tarvittavat palvelin- ym ohjelmistot. Laitteiston ylläpidon hoitaa tietojenkäsittelytieteen laitoksen ylläpito, joten projektin ei tarvitse osallistua siihen.

Ohjelmisto tullaan toteuttamaan Java-ympäristössä. Ydin toteutetaan normaaleina Java-

luokkina ja asiakkaalle näkyvät dynaamiset HTML-sivut Java Server Pages (JSP)-tekniikalla. Ohjelmiston toteuttamisessa käytetään ilmaista Eclipse-sovelluskehitintä ja yksikkötestauksessa JUnit-testausohjelmaa. Tietokantaohjelmistona tullaan käyttämään Postgresiä, joka löytyy valmiiksi asennettuna alkorunni-palvelimelta.

7 Laadunvalvonta

Tämän projektin laatua valvotaan sekä dokumenttien että itse ohjelmakoodin osalta. Laadunvalvonnan tavoitteena on saavuttaa sellainen valmis ohjelmisto, joka on dokumentoitujen vaatimusten mukainen ja toimii virheettömästi.

7.1 Dokumenttien laatu

Ennen kunkin dokumentin hyväksymistä ja jäädyttämistä käymme dokumentin läpi erillisessä dokumentoinnin katselmuksessa. Virallisissa katselmuksissa on ryhmän jäsenten lisäksi läsnä myös ohjaaja ja ainakin toinen asiakkaista. Katselmoitava dokumentti on osapuolten nähtävillä hyvissä ajoin. Katselmuksessa kenellä tahansa on oikeus puuttua dokumentin sisältöön ja tehdä siihen korjausehdotuksia.

Korjausehdotukset voivat olla sellaisia että uutta katselmusta ei tarvita korjauksen jälkeen tai sellaisia että ne vaativat uuden katselmuksen. Dokumentti on valmis kun molemmat osapuolet ovat sitä mieltä että korjattavaa ei enää ole. Tämän jälkeen dokumentti "jäädytetään"jonka jälkeen siihen ei saa enää tehdä muutoksia.

7.2 Ohjelmiston laatu

Ohjelmiston laadun varmistamiseksi testaamme JUnit-yksikkötestauskehystä käyttäen ohjelman eri komponentit heti yksittäisen komponentin valmistuttua. Jokaisessa komponentissa pyritään arvoaluetestaukseen, toisin sanoen valitaan yksi testitapaus arvoalueen sisältä ja lisäksi kaksi testitapausta jokaisen arvoalueen rajan molemmin puolin. Testattujen komponenttien yhteistoiminta testataan integrointitestauksena sitä mukaan kun kokonaisuuksia saadaan kasaan.

Ohjelmiston ja vaatimusmäärittelyn vastaavuus testataan, jotta saadaan selville ovatko asiakkaan vaatimukset tulleet täytetyiksi ja mitä on mahdollisesti jäänyt toteuttamatta. Vastaavuus vaatimusmäärittelyyn kuten kaikki muutkin testaukset kirjataan luonnollisesti ylös. Lopuksi testataan koko järjestelmä esimerkkikäyttötapausten avulla.

Tietokantaoperaatioiden tarkastamiseksi teemme mahdollisimman aikaisessa vaiheessa kattavan esimerkkitietokannan, jolla tarkastetaan kaikki ohjelmistossa käytettävät tietokantaoperaatiot.