Testaussuunnitelma

Koskelo

Helsinki 16.12.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Tom Bertell
Johan Brunberg
Lauri Liuhto
Eeva Nevalainen
Harri Tuomikoski

Asiakas

Teemu Kerola

Johtoryhmä

Juha Taina
Turjo Tuohiniemi

Kotisivu

`http://www.cs.Helsinki.FI/group/koskelo/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	19.10.2004	Ensimmäinen versio
1.0	7.11.2004	Valmis dokumentti

Sisältö

1 Johdanto

Tämä on Koskelo-projektin testaussuunnitelma. Dokumentin tavoitteena on selventää ryhmäläisille mitä projektin testausvaiheessa on tarkoitus tehdä ja miten siihen voidaan valmistautua jo toteutusvaiheen aikana.

1.1 Testausstrategia

Ohjelman testauksessa noudatetaan testauksen V-mallia, joka oletetaan ryhmäläisille tutuksi Ohjelmistotuotanto-kurssilta. Ideana on aloittaa yksikkötestauksella pienistä ohjelmiston osista kuten metodeista ja luokista. Tämän jälkeen siirrytään testaamaan komponenttien yhteistoimintaa. Kolmannessa vaiheessa testataan osajärjestelmiä kokonaisuuksina. Neljännessä vaiheessa testataan koko järjestelmän toimivuutta, jonka jälkeen se luovutetaan asiakkaalle testattavaksi. Asiakas testaa ohjelmistoa verraten sitä antamiinsa vaatimuksiin.

Yksikkötestaus ja integrointitestaus pyritään automatisoimaan mahdollisimman hyvin. Järjestelmätestauksessa jouduttaneen hyödyntämään loppukäyttäjiä tutkimaan onko ohjelman looginen toiminta oikein. Loppukäyttäjinä projektin testausvaiheessa toimivat ensisijaisesti projektilaiset, mutta mahdollisuuksien rajoissa hyödynnetään lähisukulaisia, tietojenkäsittelytieteen opiskelijoita ja muita vapaaehtoisia. Kun ohjelmistoa testaa joku muu kuin ohjelmistoprojektin jäsen, on paikalla kirjaamassa testaajan kommentteja joku projektilainen.

1.2 Testauksen tarkkailu

Projektin testausvastaava Eeva Nevalainen tarkkailee testauksen etenemistä ja ylläpitää kirjanpitoa testien läpäisemisestä. Mikäli testauksen aikataulu, kattavuus tai jokin muu asia alkaa luisua raiteiltaan keskustellaan ryhmän kesken tarvittavista toimenpiteistä. Testauksen lopuksi testausvastaava kirjoittaa testausraportin.

2 Testauksen vaiheet

Ryhmän testauksessa pyritään noudattamaan testauksen V-mallin eri vaiheita. V-mallia sovelletaan projektiin lineaarisesti varsinaisen toteutusvaiheen jälkeen. Testitapauksia suunnitellaan toteutusvaiheen aikana suunnittelun kuitenkin painottuen toteutuksen loppuvaiheeseen.

2.1 Yksikkötestaus

Yksikkötestauksessa hyödynnettään JUnit-ohjelmistoa [GB04]. Kaikki metodit testataan mahdollisimman täydellisesti ja testitapaukset kirjoitetaan siten, että metodin kaikissa haaroissa käydään.

Kaikkia ehtolauseita pyritään testaamaan testitapauksilla siten, että niiden ehto täyttyy ja ei täyty, sekä näiden yhdistelmiä jos metodissa on useampi ehtolause. Silmukoille testataan syötteitä, joiden seurauksena silmukkaa ei suoriteta tai silmukan pituus olisi yhtä pidempi kuin maksimi. Ylipitkiä silmukoita ei välttämättä kuitenkaan ole mahdollista testata, jos ohjelmiston kaikissa silmukoissa pituutena käytetään syötteen pituutta.

2.2 Integraatiotestaus

Integraatiotestauksessa testataan Koskelo-komponenttien toimintaa keskenään ja erityisesti testataan niiden toimivuutta eAssarin ja Titokoneen kanssa. Integraatiotestauksessa pyritään huomioimaan kunkin komponentin kohdalla, että se kutsuu oikeita osia ulkopuolisista järjestelmistä, sillä käytännössä kaikki järjestelmän sisäiset viestit kulkevat eAssarin kautta eivätkä suoraan Koskelon toteuttamilta komponenteilta toisille.

2.3 Järjestelmätestaus

Järjestelmätestauksessa tutkitaan Koskelon tuottaman ohjelmiston toimivuutta kokonaisuutena ja erityisesti huomiodaan toiminnassa vaatimusmäärittelyssä eristettyjen vaatimusten täyttyminen. Järjestelmätestauksessa pyritään kustakin vaatimuksesta kirjaamaan täyttyykö vaatimus tuotetussa ohjelmistossa tai jos vaatimus ei täyty, miten vaatimuksen täyttymättömyys käy ilmi. Myöskin kirjataan ylös, jos jonkin vaatimuksen täyttymistä ei ole mahdollista havaita tai testata.

2.4 Hyväksymistestaus

Hyväksymistestauksen suorittaa asiakas. Jos hyväksymistestauksessa ilmenee suuria virheitä, niiden korjaamisesta keskustellaan asiakkaan ja ryhmän kesken, mutta pienet toiminnalliset virheet pyritään korjaamaan.

Hyväksymistestauksen voidaan todeta olevan valmis, kun kaikki havaitut virheet on joko korjattu, tai niiden korjaamatta jättäminen on sovittu asiakkaan kanssa.

3 Testausvälineet

3.1 JUnit ja HttpUnit

Ohjelman osat, jotka on mahdollista yksikkötestata testataan JUnitilla [GB04]. JUnit on vapaan lähdekoodin testausympäristö, jossa on helppo kirjoittaa ja ajaa toistettavia testejä. JUnit tarjoaa muun muassa mahdollisuuden jakaa dataa eri testitapausten kesken sekä käyttöliittymän testien ajamiseen.

HttpUnitia hyödynnetään servlettien ja jsp-sivujen testauksessa. HttpUnit on erityisesti tällaisten komponenttien testaamista varten kehitetty apuväline.

3.2 Loppukäyttäjä

Tehtävien oikeellisuuden analysointia jouduttaneen testaamaan loppukäyttäjällä. Tätä varten muodostetaan etukäteen lopputuloksia, joiden lopputulos tunnettaan, että testien tulokset voidaan kirjata heti, eikä kirjaukseen jää epämääräisyyksiä sen suhteen, mitä tapahtui kunkin testin kohdalla.

Analysoinnin päättelyluonteen vuoksi sitä ei voida kokonaan testata mekaanisesti, sillä on yhtä todennäköistä kirjoittaa virheellinen testi testaamaan ohjelman päättelyä kuin päättelyn toimia virheellisesti.

4 Testitapaukset

4.1 Komponentit

Ohjelman kaikkia komponentteja testataan mahdollisimman automatisoidusti. Kaikilta osin testaaminen automatisoidusti ei ole mahdollista. Vaikka automatisoitu testaus ei ole mahdollista, komponenteille pyritään kehittämään mahdollisimman kattavia testitapauksia.

Analysoinnin testauksessa on huolehdittava tarkasti testitapausten noudattamisesta ja kaikkien tilanteiden yksityiskohtaisesta dokumentoinnista ennen ja jälkeen testin suorituksen. Kaikki odottamattomat tilanteet on kirjattava testin suorituksen aikana. Näin voidaan vielä jälkikäteen tutkia oliko havaittu virhe testissä vai ohjelmistossa ilman turhaa ajanhukkaa.

4.2 Testitapaukset

Testitapauksista kirjataan kohdekomponentti, syötteet, odotettu lopputulos ja testin suoritusmenetelmä. Testitapauksen suorituksen jälkeen kirjataan testin lopputulos. Jos suorituksessa on tapahtunut virhe, niin dokumenttiin kirjataan mahdollisesti tapahtuneen virheen korjauksen jälkeen onnistunut suoritus.

4.3 Järjestelmätestauksen testitapaukset

4.3.1 Staattisen tehtävän laatiminen

Kokeillaan luoda uusi tehtävä. Uuteen tehtävän syötetään tehtävänanto, mallira-tkaisu, tarkastuskriteerit ja palaute.

Kokeillaan:

- Tehtävää, jonka kriteeri on syntaksiltaan oikein
- Tehtävää, jonka yksi kriteeri on syntaksiltaan väärin
- Tehtävää, jossa ei ole kriteerejä
- Tehtävää, jossa on useampia kriteerejä
- Tehtävää, jossa verrataan malliohjelman suorituksen jälkeiseen lopputilaan
- Tehtävää, jossa verrataan virheellisen malliohjelman suorituksen jälkeiseen lopputilaan (malliohjelman, jota ei saada suoritettua titokoneella)

4.3.2 Dynaamisen tehtävän laatiminen

Kokeillaan kaikkia tapauksia, jotka on esitetty staattisen tehtävän luomisen yhteydessä.

Lisäksi kokeillaan

- Jokaista yksittäistä dynaamista osaa
- Virheellisesti määriteltyä dynaamista osaa
- Määrittelemättä jätettyä dynaamista osaa

4.3.3 Staattisen tehtävän muokkaaminen

Kokeillaan muokata jo olemassa olevaa tehtävää.

Kokeillaan:

- Muuttaa tehtävän yksi kriteeri virheelliseksi
- Poistaa tehtävän kriteeri
- Poistaa tehtävän kriteerit
- Muuttaa vertailu staattisesta lopputilasta simuloituun ja toisinpäin
- Lisätä kriteeri
- Lisätä virheellinen kriteeri

4.3.4 Dynaamisen tehtävän muokkaaminen

Testataan kuten staattisen tehtävän muokkaamista, mutta lisäksi kokeillaan:

- Lisätä dynaaminen osa
- Muuttaa dynaamista osaa
- Tehdä virheellinen dynaaminen osa
- Tehdä duplikaatti dynaamisesta osasta

4.3.5 Tehtävän ratkaiseminen

Kokeillaan ratkaista tehtävää.

Kokeillaan:

- Antaa oikea ratkaisu
- Antaa tyhjä ratkaisu
- Antaa virheellinen ratkaisu
- Antaa virheellinen TTK-91 ohjelma
- Antaa melkein oikea ratkaisu

- Kokeillaan ratkaista tehtävää jossa on kaikenlaisia kriteereitä määritelty sekä ratkaisulla joka täyttää kaikki kriteerit että ei täytä yhtään.
- Kokeillaan vastaavaa tehtävää myös ratkaisulla joka on osin oikea.

Lähteet

GB04 Gamma, E. ja Beck, K., Junit testing framework, 2004. `http://junit.org/`. [20.10.2004]