Testaussuunnitelma

Anno3

Helsinki 7.5.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Jukka Huhta
Juho Iso-Markku
Jarno Laitinen
Timo Myyryläinen
Roger Sandström
Miro Wikgren

Asiakas

Sami Palhomaa

Johtoryhmä

Juha Taina
Jaakko Saaristo

Kotisivu

`http://www.cs.helsinki.fi/group/anno3/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	21.1.2007	Ensimmäinen versio
1.1	11.2.2007	Lisätty vakioselostuksia eri testivaiheista ja vastuualueista.
1.2	14.3.2007	Testityökalut ja -prosessit kirjattu. Järjestelmätestitapauksia lisätty.
1.3	2.4.2007	Lisätty työkalujen käyttöohjeet ja esimerkkitestitapaus.
1.4	10.4.2007	Ehdotus lopulliseksi versioksi.
1.5	17.4.2007	Lopullinen versio.
1.6	20.4.2007	Lopullinen ja jäädytetty versio.
2.0	6.5.2007	Testiraportti liitetty, jäädytetty uudelleen.

Sisältö

1 Johdanto

Anno3-ohjelmistotuotantoprojektin tarkoitus on tuottaa annotointityökalu, jota käytetään www-oppimisympäristö Moodlen kanssa. Annotointi tarkoittaa merkintöjen tekemistä ja kommenttien lisäämistä www-sivuihin ja muihin Moodlen dokumenttiformaatteihin muokkaamatta itse varsinaisen dokumentin sisältöä. Työkalua voidaan käyttää yhteisöllisen prosessikirjoittamisen ja palautteen antamisen apuvälineenä, jolla merkinnät voidaan tehdä suoraan kontekstiin eli oikeaan paikkaan dokumentissa.

Anno3 jatkaa aiemman, Mooan-ohjelmistoprojektin työtä, joka jäi osittain keskeneräiseksi ja puutteelliseksi. Tehtävät muutokset painottuvat järjestelmän toiminnallisuuden korjaamiseen, parantamiseen ja lisäämiseen.

Tämä testaussuunnitelmadokumentti määrittelee projektissa käytettävän testausprosessin, käytettävät menetelmät, testauksen kattavuuden ja testauksen raportoinnin. Testaus jaetaan kolmeen vaiheeseen: yksikkötestaus, integrointitestaus ja järjestelmätestaus.

Erillisiä asiakkaan kanssa määriteltyjä hyväksymistestitapauksia ei tehdä, vaan järjestelmätestauksen testitapaukset laaditaan hyväksymistestauksen tapaan, koska kyseessä on harjoitusprojekti.

2 Sanasto

Anno t. anno Järjestelmän nimi. Käytetään esimerkiksi ohjelmakoodissa tunnisteena.

Anno3 Anno-järjestelmän tuottavan ohjelmistotuotantoprojektiryhmän nimi.

Annotea, Annotea-palvelin Keskustelun tallennusalustana toimiva HTTP:lla käytettävä ulkoinen järjestelmä.

annotointi (v.) Luokitellun merkinnän lisääminen.

annotointi (s.) Dokumenttiin maalaamalla tehty merkintä, jonka yhteydessä on merkintään liittyviä kommentteja.

dokumentti HTML-, XML, teksti- tai Wiki-sivu Moodle-oppimisalustalla.

järjestelmä Ellei kontekstista muuta ilmene: projektin puitteissa toteutettava kommentointityökalu kokonaisuutena (Anno). Tähän ei lueta Moodlea eikä Annotea-palvelinta.

keskustelu Merkintään liittyvät viestit, sekä kommentti että vastaukset.

kommentti Annotointiin liittyvä yksi kommentti.

kommenttinäkymä Näkymä, jossa voi lukea keskusteluja.

kommenttityyppi-ikkuna Käyttöliittymän osa, jossa valitaan, onko kyseessä oikeinkirjoitus- vai sisältökommentti.

käsittelijämoduuli Palvelimella suoritettava järjestelmän varsinainen toimintalogiikka. Toteutettu PHP:lla. Engl. handler module.

käyttäjämoduuli Selaimella suoritettava osa järjestelmää, joka on toteutettu JavaScriptillä ja tyylimäärittelyin. Engl. user module.

merkintä Korostettu kohta tekstissä, vrt. annotointi

merkintäkategoria Merkinnän luokka, esim. oikeinkirjoitus- tai sisältömerkintä.

muokkausnäkymä Näkymä, josta voi lukea ja kirjoittaa kommentteja sekä vastauksia.

opasteikkuna Pieni ikkuna dokumentin ylälaidassa, joka kertoo annotointiominaisuudesta ja mahdollistaa sen päälle- ja poiskytkemisen.

päällekkäisyysalue Alue tekstissä, jonka sisällä on voimassa sama joukko päällekkäisiä merkintöjä. Mikäli merkintä loppuu tai alkaa, päällekkäisyysalue vaihtuu.

tiedosto Käsittelijä- tai käyttäjämoduulin osa, jossa on joku toiminnallinen kokonaisuus.

vastaus Vastaus kommenttiin, ja sikäli myös kommentti.

väli-ikkuna Käyttöliittymän elementti, jossa valitaan joku päällekkäisistä annotoinneista tarkasteltavaksi.

3 Testauksen lähtökohdat ja ohjeet

3.1 Testauksen rajaus

Anno-järjestelmä toimii yhteistyössä Moodlen ja kahden tietokannan kanssa. Järjestelmä ei aiheuta muutoksia Moodleen (järjestelmän kutsua lukuunottamatta) eikä ulkoisena komponenttina toimivaan Annotea-palvelimeen, joten näitä ei testata. Tarkempi kuvaus järjestelmän arkkitehtuurista käy ilmi suunnitteludokumentin arkkitehtuurikuvauksesta.

Tämän projektin puitteissa ei testata jo toimivaksi havaittuja toiminteita olemassaolevassa Anno-järjestelmän ohjelmakoodissa, vaan ainoastaan muutettuja ja lisättyjä osuuksia.

3.2 Käytetyt ohjelmointikielet ja erikoispiirteet

Anno-järjestelmän koodi on tehty PHP-kielellä, joka muokkaa Moodlen käyttäjälle lähettämien HTML-sivujen sisältöä esimerkiksi lisäten niihin selaimella suoritettavaa JavaScript-koodia. Anno-tietokannan tietokantaoperaatiot tehdään SQL-kielellä.

Yksikkötestaus kohdistuu enimmäkseen PHP-kielellä toteutettuun palvelinkoodiin. Järjestelmätestaus tehdään enimmäkseen käsin testitapauksen ohjeiden mukaisesti.

3.3 Suunnittelun ja testauksen ajoitus ja työnjako

Projektissa sovelletaan suunnittelu-, valmistus- ja evaluointivaiheiden iterointia erityisesti yksikkötestauksessa. Toteutusvaihe ja yksikkötestaus integroidaan siten, että PHP-koodin kirjoittamisen yhteydessä kirjoitetaan myös tarvittavat yksikkötestitapaukset.

Ohjelmoija on ensisijaisesti vastuussa testitapauksista, mutta testausvastaava auttaa tarvittaessa testitapausten laatimisessa. Toteutusvaihe on valmis kunkin funktion tai toiminteen osalta vasta, kun myös siihen liittyvät testitapaukset on laadittu ja ajettu läpi onnistuneesti.

Suunnittelun toteutus ja yksikkötestausvaihe kulkevat limittäin projektissa, tarkoituksena on ajaa yksikkötestitapauksia sitä mukaa kuin PHP-koodi katsotaan valmiiksi. Ensimmäisessä testivaiheessa havaitaan usein koodivirheitä, ja niiden korjaamisen jälkeen testit tulee toistaa.

3.4 Testauksen tarkkailu ja muu hallinto

Testausvastaava raportoi testaukseen käytettäviä resursseja ja aikatauluja. Saamiensa tietojen perusteella hän kirjoittaa testiraportin.

Yksikkötestauksen aikana kerätään kattavuustietoa. Jos millään testitapausten yhdistelmällä ei saada yksikkötestauksen kattavuusvaatimuksia toteutettua kyseisen funktion osalta, selvittää koodaaja yhdessä testausvastaavan kanssa onko koodissa ylimääräisiä haaroja tai muita syitä miksi osaa koodista ei suoriteta. Testausvastaava myös tarkkailee testaustavoitteiden täyttymistä sekä muiden määrättyjen kohtien toteutumista. Mikäli testaus ei vastaa suunniteltua, keskustellaan ryhmän kokouksissa epäkohtien syistä ja vaadituista toimenpiteistä. Ryhmän jäsenten tulee ilmoittaa testausvastaavalle eri testausvaiheiden läpimenosta ja koodikattavuudesta.

3.5 Työkalut

Yksikkötestauksessa PHP-koodi testataan PHPUnit 3.0 -työkalulla, (http://www.phpunit.de). Järjestelmätestaus tehdään Firefox- ja Internet Explorer -selaimilla, joiden versiot käyvät ilmi järjestelmävaatimuksista.

Yksikkötestauksessa koodin suorituksen seuranta tapahtuu PHPUnit-työkalun avulla, järjestelmätestauksessa testitapauksen suoritus arvioidaan manuaalisesti.

4 Yksikkötestaus

4.1 Lähestymistapa

Ohjelmakoodi yksikkötestataan osana suunnitteluvaiheessa tapahtuvaa koodausta. Valmiiksi saatu funktio testataan erikseen muista funktioista, jos mahdollista. Jos funktiota ei ole mahdollista eristää muista funktioista, pyritään löytämään minimijoukko funktioita

joita testitapaus vaatii. Yllämainittu vaatimus toteutetaan ns. tyngillä (stub), joilla saadaan simuloitua oikeaa toimintaympäristöä hallitusti.

Testitapaukset nimetään seuraavin kriteerein:

- alkaa sanalla test
- funktion nimi
- järjestysnumero.

Esimerkkinä test_anno_is_comment_locked_1, joka alkaa sanalla test, seuraavana on funktion nimi, ja viimeisenä osana järjestysnumero 1.xi Alaviiva erottaa testitapauksen nimen osat toisistaan. Jokainen funktio aloittaa järjestysnumerot alusta. Järjestysnumeroiden tulisi soveltuvin osin seurata hyväksymiskriteereissä mainittua testikriteereiden järjestystä, mutta vain soveltuvin osin. Mikään tietty järjestysnumero ei tarkoita aina samanlaista testitapausta testattavasta funktiosta toiseen.

4.2 Testattavat kohdat

Ainoastaan Anno-järjestelmän koodi yksikkötestataan, ei Moodle- ja Annotea-palvelinten koodia. Tarkoitus on testata ainoastaan uudet tai olennaisesti muutetut toiminnallisuudet. Edelliseltä ryhmältä periytyneiden toteutusten oletetaan toimivan dokumentaation mukaisesti. Varsinaisia testitapauksia ei kirjata yksikkötestauksen osalta tähän dokumenttiin niiden suuren lukumäärän vuoksi. Testitapaukset kirjoitetaan suoraan suoritettavana koodina käytettävän työkalun ohjeistuksen mukaisesti, ja kommentoitu testikoodi toimii dokumenttina.

4.2.1 Testattavat modulit ja funktiot

Testattavat PHP-moduulit (PHP-tiedostot) ja niiden sisällä olevat funktiot on määritelty suunnitteludokumentissa.

4.3 Hyväksymiskriteerit

Testitapaukset valitaan ekvivalenssiluokituksen avulla. Ekvivalenssiluokitus tarkoittaa funktion kutsuparametrien arvoalueiden luokittelua. Arvoalueet käsitellään eri tavoin riippuen siitä, onko kyseessä lukuarvo vai merkkijono.

Lukuarvojen ekvivalenssiluokat ovat seuraavat:

1. NULL
2. alle minimiarvon
3. täsmälleen minimiarvo

4. minimiarvon ja maksimiarvon väliltä
5. täsmälleen maksimiarvo
6. yli maksimiarvon
7. muu kuin lukuarvo (esim. merkkijono)

Vastaavasti merkkijonojen ekvivalenssiluokat ovat:

1. NULL
2. tyhjä merkkijono
3. oikea, toimiva arvo
4. väärä, mutta oikeanmuotoinen arvo
5. liian pitkä arvo, testataan vain erityistapauksissa
6. vääränmuotoinen arvo

Merkkijonon testitapaus ”liian pitkä arvo” ajetaan vain, jos testattava funktio käyttää kiinteän mittaista tietorakennetta merkkijonon käsittelyssä. Normaalisti PHP käyttää merkkijonojen käsittelyssä dynaamisia tietorakenteita, eikä tässä projektissa testata PHP-kielen merkkijonojen maksimipituuden käsittelyä.

Kaikkia mahdollisia useamman kutsuparametrin eri ekvivalenssiluokkien yhdistelmiä ei testata vaan testaus kohdistetaan koodaajan ja testausvastaavan näkemyksen mukaan olennaisimpiin testitapauksiin. Yksikkötestaus katsotaan läpimenneeksi koko tuotteen osalta jos yksikään testitapaus ei epäonnistu ja testit kattavat yhteensä vähintään 90 prosenttia koodiriveistä.

5 Integrointitestaus

5.1 Lähestymistapa

Integrointitestauksessa varmistetaan ohjelmistokomponenttien yhteensopivuus ja yhteistoiminta. Tässä projektissa integrointitestausta ei tehdä erikseen, koska yksikkötestauksen testitapaukset testaavat jo funktioiden yhteistoimintaa. Projektin koodissa olevat toiminnallisuudet ovat niin laajoja, ettei niiden testausta voida tehdä täysin puhdasoppisena yksikkötestauksena. Tästä syystä katsotaan yksikkötestauksen kattavan integrointitestauksen tehtävät.

5.2 Testattavat kohdat

Integrointitestausta ei tehdä erikseen, vaan yksikkötestauksen testitapausvalinnat kattavat myös integrointitestauksen.

5.3 Hyväksymiskriteerit

Integrointitestaus katsotaan läpimenneeksi koko tuotteen osalta, jos yksikkötestaus on mennyt läpi.

6 Järjestelmätestaus

6.1 Lähestymistapa

Testitapausten valinta perustuu käyttäjävaatimuksiin. Testitapaukset on kirjoitettu siten, että niiden avulla voi yksiselitteisesti määritellä täyttääkö järjestelmä annetun vaatimuksen vai ei, eli voiko määritelty operaatiota tehdä järjestelmällä vai ei. Testitapauksessa käsitellään yleensä vain yksi tapa suorittaa operaatio. Jos saman operaation voi tehdä useammalla tavalla, testataan se kuitenkin vain tässä dokumentissa kuvatulla tavalla.

6.2 Testattavat kohdat

6.2.1 Testitapaus: Kommenttien merkistö

Testaa vaatimuksen: käyttäjävaatimus 4.1

1. Maalaa hiirellä ilman merkintää oleva teksti.
2. Valitse pop-up-ikkunasta Sisältö.
3. Kirjoita kommentiksi ääöåÄÖÅ, tai muu skandinaavisia kirjaimia sisältävä teksti, ja valitse Julkaise.
4. Tarkista onko avautuvassa kommentti-ikkunassa näkyvissä samat skandinaaviset kirjaimet kun kohdistin viedään maalatun tekstin päälle.

Hyväksyntä: skandinaavisten kirjainten näkyminen kommenteissa.

6.2.2 Testitapaus: Merkinnän tekeminen, ilman päällekkäisyyttä.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.

1. Maalaa hiirellä ilman merkintää oleva teksti, esimerkiksi kaksi peräkkäistä sanaa.
2. Valitse pop-up-ikkunasta Sisältö.
3. Kirjoita kommentiksi Testi1, ja valitse Julkaise.
4. Tarkista onko värillä merkitty alue tekstissä alkuperäisen maalauksen mukainen.

5. Tarkista avautuuko kommentti näkyviin pop-up-ikkunaan näkyviin kun kohdistin viedään maalatun tekstin päälle.

Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.

6.2.3 Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä suurempi kuin vanha.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.

1. Maalaa hiirellä valmiiksi merkityn alueen ympäriltä oleva teksti, esimerkiksi yksi lause.
2. Valitse pop-up-ikkunasta Sisältö.
3. Kirjoita kommentiksi Testi2, ja valitse Julkaise.
4. Tarkista onko värillä merkitty alue tekstissä alkuperäisen maalauksen mukainen.
5. Tarkista avautuuko kommentti näkyviin pop-up-ikkunaan näkyviin kun kohdistin viedään maalatun tekstin päälle.

Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.

6.2.4 Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä pienempi kuin vanha.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.

1. Maalaa hiirellä valmiiksi merkityn alueen sisällä oleva teksti, esimerkiksi yksi sana.
2. Valitse pop-up-ikkunasta Sisältö.
3. Kirjoita kommentiksi Testi3, ja valitse Julkaise.
4. Tarkista onko värillä merkitty alue tekstissä alkuperäisen maalauksen mukainen.
5. Tarkista avautuuko kommentti näkyviin pop-up-ikkunaan näkyviin kun kohdistin viedään maalatun tekstin päälle.

Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.

6.2.5 Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä limittäin vanhan kanssa.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.

1. Maalaa hiirellä valmiiksi merkityn alueen kanssa limittäin oleva teksti, esimerkiksi sanan puolivälistä seuraavaan välilyöntiin.
2. Valitse pop-up-ikkunasta Sisältö.
3. Kirjoita kommentiksi Testi4, ja valitse Julkaise.
4. Tarkista onko värillä merkitty alue tekstissä alkuperäisen maalauksen mukainen.
5. Tarkista avautuuko kommentti näkyviin pop-up-ikkunaan näkyviin kun kohdistin viedään maalatun tekstin päälle.

Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.

6.2.6 Kommentin poistaminen, ylläpitäjä.

Testaa vaatimuksen: käyttäjävaatimus 4.7.

1. Kirjaudu sisään ylläpitäjänä.
2. Avaa kommenttinäkymä ja valitse Poista.
3. Avaa kommenttinäkymä uudestaan, ja tarkista poistuiko valittu kommentti.

Hyväksyntä: kommentin poistaminen onnistuu.

6.2.7 Kommentin poistaminen, oma kommentti, määritetyn ajan sisällä.

Testaa vaatimuksen: käyttäjävaatimus 4.6.

1. Luo kommentti kuten testitapauksessa X.
2. Avaa kommenttinäkymä ja valitse Poista.
3. Avaa kommenttinäkymä ja tarkista poistuiko valittu kommentti.

Hyväksyntä: kommentin poistuminen onnistuu.

6.2.8 Kommentin poistaminen, oma kommentti, määritetyn ajan jälkeen.

Testaa vaatimuksen: käyttäjävaatimus 4.6.

1. Luo kommentti kuten testitapauksessa X.
2. Odota määritetty aika (30 min).
3. Avaa kommenttinäkymä ja valitse Poista.
4. Avaa kommenttinäkymä ja tarkista poistuiko valittu kommentti.

Hyväksyntä: kommentin poistuminen ei onnistu.

6.2.9 Merkintöjen tulostaminen.

Testaa vaatimuksen: käyttäjävaatimus 4.10.

1. Avaa dokumentti.
2. Valitse Tulosta.

Hyväksyntä: tulostus sisältää kaikki selaimessa näkyvät kommentit, ja merkinnät on numeroitu käyttäjävaatimuksen mukaisesti.

6.2.10 Merkinnät ovat versiokohtaisia

Testaa vaatimuksen: käyttäjävaatimus 4.11.
Tämän vaatimuksen suunnittelu on kesken, ja myös testiohjeistus on tekemättä.

6.2.11 Kommenttien muokkaus

Testaa vaatimuksen: käyttäjävaatimus 4.12. Tämän vaatimuksen suunnittelu on kesken, ja myös testiohjeistus on tekemättä. Allaoleva ohjeistus on vain ehdotus.

1. Luo kommentti kuten testitapauksessa 5.2.2.
2. Avaa kommenttinäkymä ja kirjoita kommenttikenttään Testi. Valitse Julkaise.
3. Avaa kommenttinäkymä ja tarkista näkyykö äsken kirjoitettu kommentti.
4. Valitse Muokkaa, ja kirjoita tilalle Uusi testi. Valitse Julkaise.
5. Avaa kommenttinäkymä ja tarkista näkyykö muokattu kommentti.

6.3 Hyväksymiskriteerit

Järjestelmätestaus katsotaan läpimenneeksi jos kaikki testitapaukset menevät läpi.

7 Muu testaus

Tässä projektissa ei tehdä muuta testausta kuin yksikkö- ja järjestelmätestaus.

8 Testausaikataulu

Yksikkötestaukselle ei ole erillistä aikataulua, koska koodi yksikkötestataan ohjelmoinnin yhteydessä.

Järjestelmätestaus alkaa 17.4.2007 ja päättyy 4.5.2007.

9 Raportointi

Havaituista virheistä ilmoitetaan testausvastaavalle, joka ylläpitää testausraporttia.

10 Virheiden korjaus ja regressiotestaus

Yksikkötestauksessa havaitut virheet korjataan välittömästi koodiin.

Järjestelmätestauksessa havaitut virheet ilmoitetaan ensin testausvastaavalle, joka neuvottelee ryhmän kanssa uusintatestien (regressiotestien) ajankohdasta ja suorituksesta. Jos virhe on liian vaikea korjata tai havaitaan liian myöhään, kirjataan se luovutusdokumenttiin.

Uusintatestit tehdään erillisenä prosessina jotta ohjelmavirheiden korjausten seurannaisvaikutus saadaan selville.

Liite 1. PHPUnit-ohjeita

Tämä on suppea opastus PHPUnit-työkaluun ja sen käyttöön tässä projektissa. Lisäohjeet työkalun kotisivuilta, osoitteesta <http://www.phpunit.de/>. Samalla nimellä on kaksikin eri työkalua, mutta tässä tarkoitetaan mainitulta sivulta löytyvää.

Testitapaukset kirjoitetaan erilliseen php-tiedostoon, joka ajetaan phpunit-komennolla. Tämän ajon tuloksena on listaus läpimenneistä testitapauksista ja ”hylätyistä” testitapauksista.

Testitapausta varten on tiedettävä kolme asiaa: testattavan funktion ominaisuudet, kutsuarvo, ja paluuarvo. Samaa funktiota joudutaan testaamaan useammalla testitapauksella, jotta saadaan testattua eri kutsuparametrien arvoilla. Jokainen testitapaus testaa vain yhden asian, testitapauksen läpimenon tai ”hylkäämisen” merkitys on oltava yksiselitteinen. Testitapauksen läpimeno päätetään vertaamalla funktion paluuarvoa odotettuun tulokseen. Jos tulos ei täsmää on testitapauksen tulos ”hylätty”.

Esimerkkinä testitapaus, joka testaa anno_add_comment()-funktiota.

```
<?php

$path = "/home/tkt_ann3/public_html/moodle3/anno/handler_module";
set_include_path(get_include_path() . PATH_SEPARATOR . $path);

require_once "Anno_Stub.php";
require_once "PHPUnit/Framework.php";
require_once "anno_annotea_lib.php";

class AnnoteaLibTest extends PHPUnit_Framework_TestCase {

  // Testcase for function anno_is_comment_locked($commentid).
  // Testcase 1  - a NULL value.
  // A negative testcase, should return true.
  public function test_anno_is_comment_locked_1() {
    $commentid = NULL;
    $expected_result = true;
    $result = anno_is_comment_locked($commentid);
    $this->assertEquals($expected_result, $result);
  }

  // Testcase for function anno_is_comment_locked($commentid).
  // Testcase 2  - an empty string.
  // A negative testcase, should return true.
  public function test_anno_is_comment_locked_2() {
    $commentid = "";
    $expected_result = true;
```

```
  $result = anno_is_comment_locked($commentid);
  $this->assertEquals($expected_result, $result);
}

// Testcase for function anno_is_comment_locked($commentid).
// Testcase 3  - a correct value.
// A positive testcase, this comment should be locked.
public function test_anno_is_comment_locked_3() {
  $commentid = "0000000030";
  $expected_result = true;
  $result = anno_is_comment_locked($commentid);
  $this->assertEquals($expected_result, $result);
}

// Testcase for function anno_is_comment_locked($commentid).
// Testcase 4  - an incorrect value, but correct format.
// A non-existent commentid, should return true.
public function test_anno_is_comment_locked_4() {
  $commentid = "00000FFFFF";
  $expected_result = true;
  $result = anno_is_comment_locked($commentid);
  $this->assertEquals($expected_result, $result);
}

// Testcase for function anno_is_comment_locked($commentid).
// Testcase 5  - a correct value, but locked.
// A positive testcase, this comment should not be locked.
// This testcase requires special attention.
// The comment needs to be created by this user just before
// running this testcase.
public function test_anno_is_comment_locked_5() {
  $commentid = "0000000041";
  $expected_result = false;
  $result = anno_is_comment_locked($commentid);
  $this->assertEquals($expected_result, $result);
}

// Testcase for function anno_is_comment_locked($commentid).
// Testcase 6  - value too long.
// A negative testcase, should return true.
public function test_anno_is_comment_locked_6() {
  $commentid = "000000003000000FFFFF";
  $expected_result = true;
  $result = anno_is_comment_locked($commentid);
  $this->assertEquals($expected_result, $result);
```

```
  }

  // Testcase for function anno_is_comment_locked($commentid).
  // Testcase 7  - value in wrong format.
  // A negative testcase, should return true.
  public function test_anno_is_comment_locked_7() {
    $commentid = "XXXX000030";
    $expected_result = true;
    $result = anno_is_comment_locked($commentid);
    $this->assertEquals($expected_result, $result);
  }

  // Testcase for function anno_is_comment_locked($commentid).
  // Testcase 8  - incorrect value type (integer).
  // A negative testcase, should return true.
  public function test_anno_is_comment_locked_8() {
    $commentid = 30;
    $expected_result = true;
    $result = anno_is_comment_locked($commentid);
    $this->assertEquals($expected_result, $result);
  }
}
?>
```

Ennen varsinaisia testifunktioita ovat testausympäristön vaatimat path- ja set_include_path-määrittelyt. Nämä määrittelyt voidaan kopioida php-tiedostosta toiseen. HUOM! Jos Moodlen www-osoite ja hakemistopolku muuttuvat, esimerkiksi moodle3:sta moodle4:ään on sama muutos korjattava näihin määrittelyihin.

Luokkamäärittelyn, class-operaatio, on oltava samalla nimellä kuin tiedostonimenkin. Eli AnnoteaLibTest.php-tiedosto sisältää AnnoteaLibTest-luokan. Jokaista testattavaa php-tiedostoa kohden on yksi Test-tiedosto, ja tämä Test-tiedosto sisältää kaikki testattavan php-tiedoston sisältämien funktioiden testaamiseen tarvittavat testitapaukset.

Jokainen testitapaus on oma funktionsa, joka voi tosin sisältää useamman assertEquals-lausekkeen. Testiraportin tulostuksen perustana on testifunktion ja testitapauksen yhtäläisyys. Tässäkin esimerkissä yhden php-funktion testaaminen vaatii kahdeksan testifunktiota ja -tapausta.

Edellämainittujen osien lisäksi testitiedostossa on require_once-lausekkeita, joiden avulla saadaan mukaan varsinainen testattava php-tiedosto. Erityisen huomioitavaa on ”Anno_Stub.php”:n mukaanottaminen. Anno_Stub.php on ns. tynkätiedosto, joka sisältää testifunktioiden tarvitsemat muuttujamäärittely. Tynkätiedoston avulla voidaan testitapaukset ajaa yksikkötestauksessa minimaalisessa ympäristössä, mahdollisimman hyvin eristettynä muista funktioista.

Testitapaukset ajetaan komennolla: ./phpunit ”Test-tiedosto” Test-tiedoston nimestä jätetään pois .php, eli käytännössä viitataan kyseisen tiedoston sisältämän luokan nimeen.

Testit on ajettava handler_module/tests-alihakemistosta. Tähän hakemistoon on myös luotava symbolinen linkki phpunit-ohjelmalle.

Testausvastaava auttaa tarvittaessa edellämainittujen esivaatimusten täyttämisessä.

Testitapauksen ajamisesta syntyy tuloste, joka kertoo mitkä testitapaukset menivät läpi ja mitkä epäonnistuivat (hylättiin). Alla esimerkki AnnoteaLibTest-luokan ajosta komennolla ./phpunit AnnoteaLibTest.

```
tkt_ann3@alkokrunni:tests$ ./phpunit AnnoteaLibTest
X-Powered-By: PHP/5.2.1
Content-type: text/html; charset=UTF-8

PHPUnit 3.0.5 by Sebastian Bergmann.

....F...

Time: 00:01

There was 1 failure:

1) test_anno_is_comment_locked_5(AnnoteaLibTest)
Failed asserting that <boolean:true> is equal to <boolean:false>.
/home/tkt_ann3/[...]/handler_module/tests/AnnoteaLibTest.php:62

FAILURES!
Tests: 8, Failures: 1.
```

Rivi "....F..." kuvaa yksittäisten testitapausten onnistumista. Jokaista testitapausta kohden on yksi merkki; piste merkitsee onnistumista ja F epäonnistumista. Lisäksi jokaisesta epäonnistuneesta testitapauksesta tulee tarkempi selostus.

Liite 2. Testiraportti

Yleistä

Testaus jaettiin kahteen vaiheeseen: yksikkötestaus ja järjestelmätestaus. Erillisiä asiakkaan kanssa määriteltyjä hyväksymistestitapauksia ei tehty.

Projektissa oli tarkoitus iteroida suunnittelu-, valmistus- ja evaluointivaiheita yksikkötestauksessa. Valitettavasti projektin aika loppui kesken, ja varsinainen dokumentoitu yksikkötestaus tehtiin lopulliselle koodille. Toteutusvaiheen ja yksikkötestauksen oikeaoppinen integrointi jäi tekemättä aikapulan takia.

Ohjelmoija oli ensisijaisesti vastuussa testitapauksista, mutta vielä toteutusvaiheessa jouduttiin muuttamaan suunnitelmia joka vaikeutti yksikkötestitapausten kirjoittamista. Aivan projektin loppuvaiheessakin lisättiin uusia funktioita. Suurin syy aikataulun pettämiseen oli vaatimus Internet Explorer -tuesta; toteutukseen varatut resurssit joutuivat selvittämään Internet Explorerin dokumentoimattomia ominaisuuksia ja siihen meni arvioitua enemmän aikaa.

Yksikkötestauksen koodikattavuusraportit ajettiin erikseen, ja lopulliset prosenttiluvut laskettiin käsin. Projektissa testattiin vain ryhmän oma koodi joka muodosti osan kaikesta järjestelmään liittyvästä koodista. Käytetty yksikkötestaustyökalu PHPUnit laskee koodikattavuuden koko PHP-tiedostolle, eikä per funktio kuten tässä projektissa raportoidaan.

Yksikkötestaus katsotaan läpimenneeksi koko tuotteen osalta jos yksikään testitapaus ei epäonnistu ja testit kattavat yhteensä vähintään 90 prosenttia koodiriveistä. Tähän lopputulokseen ei päästy; yksikkötestaukseen ei käytetty tarpeeksi aikaa.

Muu testaus

Hyväksymistestausta ei tehty.

Testausaikataulu

Anno-järjestelmää yksikkötestattiin projektin loppupuolella, 30.4.–6.5.2007. Järjestelmätestaus tehtiin projektin lopussa, 4.–6.5.2007.

Yksikkötestaus

Tähän tulee lyhyt maininta kustakin ajetusta testitiedostosta, esim. AnnoteaLibTest.php, ja lisäksi raportti testien läpimenosta ja koodikattavuudesta. Jokaista testitiedostoa kohden

on oma väliotsikkonsa.

AnnoteaLibTest.php

Testattava php-tiedosto: anno_annotea_lib.php.
Testaa funktiot: anno_is_comment_locked, anno_has_replies.
Testitapaukset (läpi/hylätty): 14/1, hylätty test_anno_has_replies_5(AnnoteaLibTest).
Koodikattavuus testatuissa funktioissa: anno_is_comment_locked(91%), anno_has_replies(83%).

HandleRequestTest.php

Testaa PHP-tiedoston: anno_handle_request.php.
Testaa funktiot: utf2html, anno_create_comment_views, anno_create_multiview_pane, anno_add_comment_array.
Testitapaukset (läpi/hylätty): 14/0.
Koodikattavuus testatuissa funktioissa: utf2html(60%), anno_create_comment_views(60%), anno_create_multiview_pane(100%), anno_add_comment_array(93%).

XPointerLibTest.php

Testattava php-tiedosto: anno_x_pointer_lib.php.
Testaa funktiot: anno_do_add_rangeto, anno_do_overlapping_comment anno_do_stringrange.
Testitapaukset (läpi/hylätty): 6/0.
Koodikattavuus testatuissa funktioissa: anno_do_add_rangeto(75%), anno_do_stringrange(70%), anno_do_overlapping_comment(14%).

Järjestelmätestaus

Tässä on listattu vain testatut käyttäjävaatimukset.

Testitapaus: Kommenttien merkistö

Testaa vaatimuksen: käyttäjävaatimus 4.1
Hyväksyntä: skandinaavisten kirjainten näkyminen kommenteissa.
Testitulos: hyväksytty.
Tarkennus: Firefox – ei huomautettavaa.
Internet Explorer – ei huomautettavaa.

Testitapaus: Merkinnän tekeminen, ilman päällekkäisyyttä.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.
Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.
Testitulos: hyväksytty.
Tarkennus: Firefox – ei huomautettavaa.
Internet Explorer – selainikkuna ei siirry automaattisesti muokkausnäkymään, vaan ikkuna pitää käsin rullata alas. Julkaise-painikkeen painamisen jälkeen ei selainikkuna päivity automaattisesti, vaan selaimelle pitää antaa ”refresh”-komento.

Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä suurempi kuin vanha.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.
Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.
Testitulos: hyväksytty.
Tarkennus: Firefox ja Internet Explorer – värien käyttö jo luettujen merkintöjen tunnistamiseksi ei toimi kunnolla päällekkäisissä merkinnöissä. Jos kaikki päällekkäiset merkinnät on luettu, käyttää järjestelmä samaa väriä erottelemaan merkinnät toisistaan, joka ei toimi.

Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä pienempi kuin vanha.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.
Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.
Testitulos: hyväksytty.
Tarkennus: Firefox ja Internet Explorer – värien käsittelyssä on sama ongelma kuin edellisessä testitapauksessa. Internet Explorer vaatii näkymän päivittämisen käsin (refresh).

Merkinnän tekeminen, päällekkäinen merkintä, uusi merkintä limittäin vanhan kanssa.

Testaa vaatimuksen: käyttäjävaatimus 4.2, 4.3 ja 4.4.
Hyväksyntä: merkinnän teko, kommentointi, merkinnän paikallaan pysyminen ja kommentin lukeminen.
Testitulos: hyväksytty.
Tarkennus: Firefox ja Internet Explorer – värien käsittelyssä on sama ongelma kuin edellisessä testitapauksessa. Internet Explorer vaatii näkymän päivittämisen käsin (refresh).

Kommentin poistaminen, ylläpitäjä.

Testaa vaatimuksen: käyttäjävaatimus 4.7.
Hyväksyntä: kommentin poistaminen onnistuu.
Testitulos: hyväksytty.
Tarkennus: Testattu kahdella eri tavalla:
1) poistettu merkintään tehty ainoa kommentti.
2) poistettu keskustelupuun ensimmäinen kommentti.
Internet Explorer vaatii näkymän päivittämisen käsin (refresh).

Kommentin poistaminen, oma kommentti, määritetyn ajan sisällä.

Testaa vaatimuksen: käyttäjävaatimus 4.6.
Hyväksyntä: kommentin poistuminen onnistuu.
Testitulos: hyväksytty.
Tarkennus: Määritetty aika testiympäristössä on 30 minuuttia. Internet Explorer vaatii näkymän päivittämisen käsin (refresh).

Kommentin poistaminen, oma kommentti, määritetyn ajan jälkeen.

Testaa vaatimuksen: käyttäjävaatimus 4.6.
Hyväksyntä: kommentin poistuminen ei onnistu.
Testitulos: hyväksytty.
Tarkennus: Määritetty aika testiympäristössä on 30 minuuttia.

Merkintöjen tulostaminen.

Testaa vaatimuksen: käyttäjävaatimus 4.10.
Hyväksyntä: tulostus sisältää kaikki selaimessa näkyvät kommentit, ja merkinnät on numeroitu käyttäjävaatimuksen mukaisesti.
Testitulos: Ei testattu, vaatimusta ei ole toteutettu.

Merkintöjen tulostaminen.

Testaa vaatimuksen: käyttäjävaatimus 4.10. Hyväksyntä: tulostus sisältää kaikki selaimessa näkyvät kommentit, ja merkinnät on numeroitu käyttäjävaatimuksen mukaisesti.
Testitulos: Ei testattu, vaatimusta ei ole toteutettu.

Merkinnät ovat versiokohtaisia

Testaa vaatimuksen: käyttäjävaatimus 4.11.
Tämän vaatimuksen suunnittelu on kesken, ja myös testiohjeistus on tekemättä.

Testitulos: Ei testattu, vaatimusta ei ole toteutettu.

Kommenttien muokkaus

Testaa vaatimuksen: käyttäjävaatimus 4.12. Tämän vaatimuksen suunnittelu on kesken, ja myös testiohjeistus on tekemättä.
Testitulos: Ei testattu, vaatimusta ei ole toteutettu.