Verso-sovellusprojekti

Projektiraportti

Tero Hänninen
Juho Nieminen
Marko Peltola
Heikki Salo

Versio 0.3.0
Julkinen
28.5.2010

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Hyväksyjä	**Päivämäärä**	**Allekirjoitus**	**Nimenselvennys**
Projektipäällikkö	__.__.2010		
Tilaaja	__.__.2010		
Ohjaaja	__.__.2010		

Tietoa dokumentista

Tekijät:

- Tero Hänninen (TH) tejohann@jyu.fi 0400-240468
- Juho Nieminen (JN) juho.nieminen@jyu.fi 050-3831825
- Marko Peltola (MP) marko.peltola@jyu.fi 041-4498622
- Heikki Salo (HS) heikki.ao.salo@iki.fi 050-3397894

Dokumentin nimi: Verso-projekti, Projektiraportti
Sivumäärä: 48

Tiivistelmä: Verso-projekti toteutti Jyväskylän yliopiston tietotekniikan laitokselle lähdekoodien julkistamisjärjestelmän. Dokumentti kuvaa projektin taustaa, resursseja ja läpivientiä. Dokumentissa tarkastellaan myös suunniteltujen tavoitteiden, käytänteiden, aikataulujen ja riskien toteutumista.
Avainsanat: Sovellusprojekti, projektiraportti, tavoitteiden toteutuminen, työnjako, resurssit, aikataulu, projektiorganisaatio, versiohallinta, projektin läpivienti, tehtävät, käytänteet, riskit, ketterä ohjelmistokehitys

Muutoshistoria

Versio	Päivämäärä	Muutokset	Tekijät
0.0.1	21.4.2010	Ensimmäiseen luonnokseen dokumentista kirjoitettiin taustaa ja tavoitteiden toteutumista kuvaavat luvut.	HS
0.0.2	23.4.2010	Kuvattu oppimistavoitteiden ja tulosten toteutumista.	HS
0.0.3	29.4.2010	Korjattu termejä.	HS
0.0.4	4.5.2010	Kuvattu organisaatio ja resurssit sekä tehtävien jakautumista.	HS
0.1.0	5.5.2010	Korjattu lähdeviitteet. Lähetetty tarkistettavaksi.	HS
0.1.1	6.5.2010	Lisätty kuvia tekijöiden ajankäytöstä.	HS
0.1.2	19.5.2010	Kirjoitettu alustavat luvut käytänteistä, Trac-käytänteistä ja prosessimallista. Pieniä Santasen ehdottamia korjauksia.	HS
0.1.3	24.5.2010	Muokattu lukujen järjestystä luontevammaksi. Lisätty luku yhteistyöstä Gitorious-yhteisön kanssa. Pieniä Santasen ehdottamia korjauksia.	HS
0.2.0	25.5.2010	Lisätty laadunvarmistusta käsittelevä luku. Parannettu taustaa kuvaavaa lukua ja muokattu rakennetta luontevammaksi. Lähetetty tarkistettavaksi.	HS
0.2.1	27.5.2010	Lisätty versiohallinnan käyttöä kuvaava luku. Täydennetty Trac-käytänteitä kuvaavaa lukua.	HS
0.3.0	28.5.2010	Muokattu rakennetta. Pieniä Santasen ehdottamia korjauksia. Lisätty projektiryhmäläisten kokemuksia kuvaava luku. Lähetetty tarkistettavaksi.	HS

Tietoa projektista

Verso-projekti toteutti Jyväskylän yliopiston tietotekniikan laitokselle lähdekoodien julkistamisjärjestelmän.

Tekijät:

- Tero Hänninen (TH) tejohann@jyu.fi 0400-240468
- Juho Nieminen (JN) juho.nieminen@jyu.fi 050-3831825
- Marko Peltola (MP) marko.peltola@jyu.fi 041-4498622
- Heikki Salo (HS) heikki.ao.salo@iki.fi 050-3397894

Tilaaja:

- Paavo Nieminen paavo.j.nieminen@jyu.fi 040-5768507
- Tapani Tarvainen tt@it.jyu.fi 050-3130446
- Ville Tirronen ville.e.t.tirronen@jyu.fi 014-2604987
- Tero Tuovinen tero.tuovinen@jyu.fi 050-4413685

Ohjaajat:

- Antti-Juhani Kaijanaho antkaij@jyu.fi 014-2602766
- Jukka-Pekka Santanen santanen@mit.jyu.fi 014-2602756

Yhteystiedot:

- Sähköpostilistat verso@korppi.jyu.fi ja yousource-users.group@korppi.jyu.fi
- Sähköpostiarkistot https://korppi.jyu.fi/kotka/servlet/list-archive/verso/ ja https://korppi.jyu.fi/kotka/servlet/list-archive/yousource-users.group/
- Työhuone AgC 222.2, puh. 014-2604963.

Sisältö

1 Johdanto

Verso-projekti oli Jyväskylän yliopiston tietotekniikan laitoksella keväällä 2010 toteutettu sovellusprojekti. Projekti määritteli, suunnitteli ja toteutti Jyväskylän yliopiston tietotekniikan laitokselle prototyypin lähdekoodien julkistamisjärjestelmästä.

Jyväskylän yliopistolla ei ole käytössä yhtenäistä tietojärjestelmää lähdekoodien jakamista tai julkistamistamista varten. Jokainen lähdekoodia tuottava tutkimusryhmä on joutunut kehittämään omat käytänteensä lähdekoodin kanssa toimimiseen.

Lähdekoodien julkistamista varten Verso-projekti kehitti WWW-sovelluksen, jolla käyttäjät voivat luoda tietovarastoja ja julkistaa lähdekoodeja. Tietovarastoja voi ylläpitää käyttäen hajautetun versiohallinnan työkaluja.

Toteutetun sovelluksen vaatimuksia on ylläpidetty Trac-järjestelmässä [11]. Sovelluksen rakenne ja toiminta kuvataan sovellusraportissa [14]. Projektisuunnitelma [13] määritteli osittain projektin läpiviennin käytänteet, resurssit ja aikataulun. Dokumentissa arvioidaan projektisuunnitelman toteutumista.

Luvussa 2 kuvataan projektiin liittyviä termejä. Luvussa 3 tarkastellaan kehitetyn sovelluksen ja projektin taustaa. Luku 4 tarkastelee projektin ja projektissa tuotettuun sovellukseen liittyvien tavoitteiden toteutumista. Luvussa 6 kuvataan Verso-projektin yhteistyötä Gitorious-sovellukseen liittyvän yhteisön kanssa. Luvussa 5 esitellään projektiorganisaatio ja muita projektin käytössä olevia resursseja. Luku 7 käsittelee projektin toteutuneita käytänteitä. Luku **??** käsittelee tehtäviä, työmääriä ja vastuualueita. Luvussa 11 tarkastellaan tehtävien ja työmäärän jakautumista projektin jäsenten kesken. Luvussa 12 kuvataan Verso-projektissa käytetty prosessimalli. luku 13 riskien toteutumista. Luvussa 14 ryhmän jäsenet kuvaavat kokemuksiaan projektista.

2 Aihealueen termit

Dokumentissa esiintyvät aihealueen termit ovat seuraavat:

Git	on hajautetun versiohallinnan ohjelmisto.
Gitorious	on WWW-sovellus Git-tietovarastojen selaamiseen ja hallintaan.
Hajautettu versiohallinta	tarkoittaa ilman keskustietovarastoa tehtävää versiohallintaa.
Julkistaminen	tarkoittaa lähdekoodin asettamista julkisesti saataville.
Ketterä ohjelmistokehitys	tarkoittaa ohjelmiston kehitystä menetelmillä, jotka pyrkivät ensisijaisesti tuottamaan halutunlaisen ohjelmiston reagoimalla mahdollisiin muutoksiin projektin kuluessa.
Lokaali tietovarasto	on paikallinen, vain käyttäjän tietokoneella sijaitseva tietovarasto.
Metatiedostot	ovat tietovaraston sisältöä kuvailevaa lisätietoa, jotka on tallennettu tietovaraston varsinaisesta sisällöstä erotettuihin tiedostoihin.
Palveluprosessi	on tietyn palvelun tarjoava, taustalla suoritettava ohjelma (engl. *daemon*).
Projekti	on YouSource-järjestelmässä käyttäjän luoma sisältökokonaisuus, jolla voi olla monia tietovarastoja.
Tietovarasto	(engl. *repository*) on versioitu lähdekoodin tallennuspaikka.
Tiketti	(engl. *ticket*) on yksittäinen ominaisuusvaatimus tai muu työnkuvaus.
Trac	on projektinhallintaohjelmisto ja tikettijärjestelmä, jolla voi ylläpitää projektin vaatimusmäärittelyä.

Vaihe (engl. *phase*) on lyhyt ajanjakso projektin toteutuksessa.

3 Taustaa

Jyväskylän yliopiston alaisuudessa toimii monenlaisia tutkimusryhmiä, joissa tuotetaan lähdekoodia. Yliopistolla ei ole yhtenäisiä toimintatapoja tai järjestelmiä lähdekoodin säilyttämiseen, mistä seuraa monenlaisia ongelmia.

Hajanaisesti tallennettujen lähdekoodien takia tieto lähdekoodista ei leviä välttämättä edes kehityksen aikana eri ryhmien välillä, mikä johtaa mahdollisesti päällekkäisen työn tekemiseen. Työntekijän lähtiessä talosta hänen yksin tuottamansa lähdekoodi käytännössä menetetään, kuten tilaajan alustavassa aihekuvauksessa [6] kerrotaan.

Osa tutkimusryhmistä ei käytä ollenkaan versiohallintaa, ja osa vain paikallisesti. Ilman yhteisesti saatavissa olevaa lähdekoodin tallennuspaikkaa itse lähdekoodin jakaminen eri tutkijoiden kesken vaatii jatkuvaa soveltamista, ja kätevin ratkaisu voi lopulta olla lähdekoodin version siirtäminen muistitikulla tai sähköpostilla [3]. Lähdekoodin jakamisen ollessa työlästä, on riskinä, ettei lähdekoodi leviä edes saman tutkimusryhmän sisällä.

Käytännön seikkojen lisäksi yhtenäisen tallennuspaikan puuttuessa lähdekoodien käyttöön liittyy haasteita. Käytäntöjen puuttuessa lähdekoodin yhteyteen ei aina tallenneta tietoa sen lisenssistä. Niinpä tutkimusryhmien osaamista on vaikea markkinoida jakamalla lähdekoodia, kun ei ole tietoa, onko lähdekoodin lisenssi yhteensopiva edes sisäisesti käytettäväksi.

Projektin tilanneella Jyväskylän yliopiston tietotekniikan laitoksella oli ennen Verso-projektia demokäytössä Redmine-sovellus, jonka avulla oli kerätty tietoa järjestelmään liittyvistä tarpeista. Redmine ei kuitenkaan soveltunut tilaajalle, minkä johdosta muodostui tarve uuden prototyypin toimittavalle sovellusprojektille.

4 Tavoitteiden toteutuminen

Verso-projekti tutki projektin alussa erilaisia alustavaihtoehtoja [12] ja valitsi lähtökohdaksi sovelluksen kehittämiselle olemassa Gitorious-sovelluksen. Projektin alussa välttämättömiksi määritellyt vaatimukset ja lähes kaikki tärkeät toteutettiin. Myös monia pienemmälle prioriteetille määriteltyjä vaatimuksia toteutettiin.

4.1 Tavoitteet sovelluksen osalta

Tilaajana toimivan Jyväskylän yliopiston tietotekniikan laitoksen tavoitteena oli kehittää lähdekoodien julkistamisjärjestelmä, jonka tulisi mahdollistaa koodin julkistaminen maailmalle, mutta toisaalta myös levittää tietoisuutta kehitetyistä ohjelmista oman yliopiston sisällä. Kehitettävän sovelluksen tavoitteena oli myös mahdollistaa yksityisen tilan luominen, jotta järjestelmää voi käyttää kehittämiseen ilman tarvetta julkaista kaikkea. Projektin ja sovelluksen tavoitteena oli myös yleisesti kannustaa aloittamaan versiohallinnan käyttöä.

Tärkeimmät tilaajan tavoitteet kohdistuivat käytön helppoon aloittamiseen ja yksityisen tilan mahdollistamiseen kehittämistä varten. Projektin kuluessa järjestelmälle syntyi paljon pienemmälle prioriteetille määriteltyjä vaatimuksia, kuten esimerkiksi tiedoston muokkaaminen WWW-selaimella sekä lähdekooditiedostosta keskustelemisen mahdollistaminen.

Suurin osa tärkeimmistä vaatimuksista oli selvillä projektin alusta asti. Esimerkiksi alustaksi valittu Gitorious kuitenkin aiheutti projektin edetessä pakolliseksi määriteltyjä muutostarpeita. Yksi tällainen muutos oli tietovarastojen selaaminen, koska Gitorious mahdollisti selaamisen vain projekteille, jonka alaisuudessa tietovarastot Gitoriouksessa ovat.

Käytön aloittamisen helpottamiseksi muun muassa tietovaraston luontia Gitoriousovelluksessa helpotettiin muokkaamalla olemassaolevia toimintoja. Lisäksi sovellus kytkettiin 19.4.2010 käyttämään kirjautumisessa Jyväskylän yliopiston Korppiopintotietojärjestelmän käyttäjätunnuksia, mikä poisti kokonaan käytön aloituksessa vaaditun erillisen rekisteröitymisen.

Käyttöliittymän osalta Gitorioukseen ei tehty merkittäviä muutoksia, vaan vain vaatimuksiin liittyvät muutokset ja lisäykset sekä korjauksia projektin aikana ilmennei-

siin parannusehdotuksiin sovelluksen käytettävyydessä. Sovellukselle logon persoonallisen ja persoonallisen ulkoasun kehitti graafinen suunnittelija Auri Kaihlavirta, joka määritti sivustolla käytettävät värit sekä valmisti sovellukselle logokuvan.

Toteutettu järjestelmä kattaa tärkeimmät siihen kohdistuneet vaatimukset. Projektin tuloksena on kehityskelpoinen järjestelmä, jota voi jo sellaisenaan käyttää lähdekoodien julkistamiseen ja työkaluna ohjelmoinnin aikana. Sovelluksen toteuttaminen kuvataan tarkemmin sovellusraportissa [14].

4.2 Sovelluksen jatkokehitys

Projektissa tuotettava YouSource-sovellus oli luonteeltaan prototyyppi, jonka kehittämiseksi kerättiin palautetta myös projektiorganisaation ulkopuolisilta esikäyttäjiltä. Projektin kuluessa oli selvää, ettei kaikkia ideoituja toiminnallisuuksia tulla toteuttamaan Verso-projektin aikana.

Verso-projekti toteutti YouSource-järjestelmään kaikki pakollisiksi määritellyt ja lähes kaikki tärkeiksi määritellyt ominaisuudet. Suurin osa poisjääneistä ominaisuuksista karsittiin pieneksi määritellyn prioriteetin takia, mutta esimerkiksi Kerberos-autentikoinnin toteuttaminen nähtiin liian suuritöiseksi tehtäväksi projektin loppuun.

Toteutetut ja toteuttamattomat ominaisuudet sekä ominaisuuksiksi hyväksymättömät ideat on kirjattu Trac-järjestelmään [11].

4.3 Oppimistavoitteet

Sovellusprojekti on tietotekniikan syventävien opintojen kurssi, jossa neljästä opiskelijasta muodostettu ryhmä toteuttaa tilaajalle ohjelmiston tietotekkniikan laitoksen ohjauksessa. Kurssissa tutustutaan projektiluontoiseen toimintatapaan, ja sen tärkeimpiin tavoitteisiin kuuluu antaa ryhmän jäsenille kokemusta ryhmätyöstä, projektin läpiviennistä sekä erilaisista käytänteistä kirjallisessa ja suullisessa viestinnässä. Sovellusprojektin ohessa ryhmän jäsenet suorittavat oheiskurssin, jossa he saavat yleistä koulutusta ohjelmistokehitykseen liittyvistä aiheista.

Oheiskurssin opetustapahtumissa Verso-ryhmä sai koulutusta projekti- ja ryhmätyöskentelystä, projektin johtamisesta ja hallinnasta, tekijänoikeuksista, sopimuksista ja käytettävyydestä. Oheiskurssiin kuului myös kaksi väliesittelyä, joissa projektiryhmä harjoitteli projektin esittämistä loppuesittelyä varten.

Verso-projektin jäsenet saivat Gitorious-sovellusta jatkokehittäessään kokemusta Ruby-ohjelmointikielestä, Ruby on Rails -ohjelmistokehyksestä sekä WWW-sovelluksen ylläpitämisestä projektin aikana. Tietokantana toimi MySQL 5.0 -tietokannan hallintajärjestelmä, ja projektin aikana ryhmäläiset oppivat muun muassa SQL-kieltä.

Projektiryhmä oppi projektin aikana käyttämään hajautettua versiohallintaa käyttämällä Git-versioohjelmistoa sekä sovelluksen kehittämiseen että projektin dokumenttien hallintaan. Ryhmä oppi myös avoimen lähdekoodin ohjelmistoprojekteihin liittyviä versiohallintakäytänteitä. Lisäksi oman sovelluksen käyttämiselle ja esikäyttöön tarjoamiselle projektin aikana edellytys oli, että ryhmä oppi kehittämään keskeneräisiä ominaisuuksia omissa haaroissaan versiohallinnassa, jotta päähaara on jatkuvasti otettavissa käyttöön.

Projektin vaatimuksia, kehitysideoita sekä löydettyjä virheitä ylläpidettiin projektin ajan Trac-projektinhallintasovelluksessa. Projektin aikana ryhmä oppi perustaidot yleisen projektinhallintatyökalun käytöstä.

Projektin aikana ryhmän jäsenet oppivat käyttämään Linux-työpöytäympäristöä sekä esikäyttöä varten olevaa Linux-virtuaalipalvelinympäristöä. Projektiryhmä oppi ylläpitämään virtuaalipalvelinta sovellukseen liittyvien palveluiden osalta itse. Ryhmä oppi myös toimimaan tietotekniikan laitoksen ATK-käytänteiden kanssa halutessaan muutoksia työpöytäkoneille esimerkiksi asennettavien sovellusten osalta.

4.4 Projektiin liittyvät dokumentit

Sovellustietovarasto	sisältää YouSource-prototyypin lähdekoodin sekä Git-versiohallinna
Projektitietovarasto	sisältää projektiin liittyvät dokumentit ja tiedostot sekä niiden Git-versiohallinnan.

Projektiryhmä toteutti lisäksi seuraavat julkiset suunnitelmat ja raportit:

Asennusohje	sisältää sovelluksen asennusohjeet.

Ajankäyttöraportti	sisältää ryhmän jäsenten kirjaamat työtunnit.
Esittelymateriaali	sisältävät väli- ja loppuesittelyn materiaalit.
Esityslistat	sisältävät projektin palavereissa käsiteltävät asiat.
Käytettävyysmuistiot	sisältävät kahden käytettävyystestauksen muistiot.
Projektiraportti	kuvaa projektin läpiviennin ja asetettujen tavoitteiden saavuttamista.
Projektisuunnitelma	kuvaa projektin tavoitteita, tehtäviä, aikataulua, yleisiä käytäntöjä ja riskien hallintaa.
Pöytäkirjat	sisältää projektipalavereissa käsitellyt, asiat, päätökset ja tehtävät.
Sovellusraportti	kuvaa toteutetun sovelluksen osat ja toiminnot sekä jatkokehitysideat.

Edellä mainittujen dokumenttien lisäksi ryhmä laatii seuraavat tulokset:

Itsearvioinnit	sisältävät ryhmän jäsenten arvioinnit omasta panoksesta, onnistumisesta ja oppimisesta.
Palaverien dokumentit	sisältävät palaverien esityslistat ja pöytäkirjat.
Sähköpostiarkistot	sisältävät kaikki projektin sähköpostilistalla käydyt keskustelut.
Lähdekoodi	sisältää tuotetun lähdekoodin kommentteineen.

4.5 Oheisdokumentit

Projektille perustettiin kaksi sähköpostilistaa, joiden viestit arkistoitiin ja projektin lopuksi arkiston viestit liitettiin projektikansioon. Toinen sähköpostilista oli tarkoitettu koko projektiorganisaation väliseen viestintään ja toinen esikäyttäjien tiedottamista ja viestintää varten.

FIXME: tulevaisuudessa Projektin aikana ryhmä piti kaksi väliesittelyä ja loppuesittelyn projektistaan ja kehitetystä järjestelmästä. Näiden esitysten **esitysmateriaalit** ja **esityksien arvioinnit** liitettiin projektikansioon.

FIXME: tulevaisuudessa Projektin lopuksi kukin ryhmän jäsen kirjoitti henkilökohtaisen arvion ryhmän työstä sekä omasta panoksestaan projektissa. Ryhmän jäsenten **itsearvio** liitettiin projektikansioon ja CD:lle.

4.6 Tulosten koostaminen

FIXME: tulevaisuudessa Sovellukseen liittyvät tulokset koostettiin CD-levylle, joka toimitettiin tilaajalle.

5 Organisaatio ja resurssit

Luvussa esitellään projektiorganisaatioon kuuluvat henkilöt, ryhmän käytössä olevat tilat, laitteet ja ohjelmistot sekä jäsenille järjestettävät perehdytykset.

5.1 Projektiorganisaatio

Verso-projektiryhmään kuului neljä Jyväskylän yliopiston tietotekniikan opiskelijaa. Tero Hänninen on 3. vuoden ja Marko Peltola 7. vuoden tietotekniikan opiskelija. Juho Nieminen ja Heikki Salo ovat 4. vuoden tietotekniikan opiskelijoita.

Tilaajana toimineen tietotekniikan laitoksen edustajina toimivat Paavo Nieminen, Tapani Tarvainen, Ville Tirronen ja Tero Tuovinen, joista Tirrosella oli määräysvalta tilaajaa koskevissa asioissa. Ryhmän vastaavana ohjaajana toimi Jukka-Pekka Santanen ja teknisenä ohjaajana toimi Antti-Juhani Kaijanaho.

Projektiryhmän työkoneiden ohjelmistojen ja ylläpidon hoiti Jyväskylän yliopiston ATK-lähituki. Projektiryhmän käytössä olleen virtuaalipalvelimen ylläpidosta vastasi Jyväskylän opiopiston tietohallintokeskuksen sovelluspalvelut, pääasiassa Harri Tuomi.

Lisäksi sovellusprojektikurssin aikana projektiryhmää auttoi kirjoitusviestinnän opettaja Leena Peltomaa sekä puheviestinnän opettaja Minna Koljonen.

5.2 Projektin tilat ja laitteet

Projektiryhmän käytössä ollut huone AgC222.2 sijaitsi Agoran C-siivessä toisessa kerroksessa sovellusprojektien tiloissa. Huoneessa oli ryhmän käytössä neljä tietokonetta, joista kolmessa oli käyttöjärjestelmänä Linux Fedora Core 12 ja yhdessä Windows XP SP3 ryhmän jäsenten työtuntien kirjaamista varten.

Huoneen Linux-tietokoneisiin oli projektia varten asennettu MySQL 5.0 -tietokantaohjelmisto, Apache-WWW-palvelin, Ruby-ohjelmatulkki sekä Ruby on Rails -sovelluskirjasto. Projektiryhmällä on lisäksi verkon kautta käyttöoikeus myös erilliseen testipalvelimeen versotest, jonka käyttöjärjestelmänä oli Red Hat Enterprise Linux 5.0.

5.3 Ohjelmointi- ja dokumentointityökalut

Ryhmän käytössä oleviin Fedora Core -tietokoneisiin oli ATK-lähituen toimesta asennettu palvelinohjelmistoina Apache, Git, MySQL sekä Ruby ja Ruby on Rails -ohjelmakirjastot, jotka mahdollistivat Gitorious-sovelluksen kehittämisen. Windows XP -tietokoneeeseen oli ATK-tuelta pyydetty TortoiseGit- ohjelmisto versiohallintaa ja Microsoft Office 2003 ajankäytön merkitsemistä varten.

Työmäärien kirjaamiseen käytettiin Petri Heinosen kehittämää Excel-pohjaista ajankäytönseurantasovellusta.

Projektiorganisaation käytössä oli myös Trac-sovellus, jota käytettiin sovelluksen vaatimusten ja projektin tehtävien hallintaan. Tracin käytössä noudatettuja käytänteitä on tarkasteltu luvussa 10.

5.4 Luennot ja perehdytykset

Ohjelmointikielenä oli Ruby ja alustana Ruby on Rails, jotka eivät olleet ryhmän jäsenille ennestään tuttuja. Ryhmä tutki kirjallisuutta verkosta ja tilasi Santasen kautta käyttöönsä Agile Web Development with Rails (Ruby, Thomas, Heinemeier Hansson) -kirjan. Lisäksi Tarvainen lainasi projektin käyttöön Ruby on Rails Bible (Fisher) -kirjan.

Projektissa vaadittiin hyvää tuntemusta Git-versiohallinnasta, joka ei ollut ryhmäläisille entuudestaan tuttu. Tekninen ohjaaja Antti-Juhani Kaijanaho toimi asiantuntijana versiohallintaan liittyvissä kysymyksissä ja järjesti ryhmälle ongelmalähtöistä perehdytystä.

Projektissa toteutettun prototyypin ohjelmointiin tarvittiin tuntemusta Rubystä, Ruby on Railsista sekä sovelluksen pohjana toimivan Gitoriouksen toiminnasta. Projektiryhmä perehtyi Rubyyn ja Ruby on Railsiin omatoimisesti tutkimalla Internetissä saatavilla olevia ilmaisia oppaita. Gitoriouksen asentamiseen ja toimintaan projektiryhmä tutustui lähdekoodeihin perehtymällä sekä kysymällä apua Gitoriouksen kehitysyhteisöltä sähköpostilla ja IRC-kanavalla.

Jukka-Pekka Santanen piti luennon projektin hallinnasta ja läpiviennistä. Luennolla käsiteltiin myös ryhmän jäsenten keskinäistä viestintää.

Oheiskurssilla käytäviin asioihin kuuluvat

- kokous- ja neuvottelukäytänteet
- esittely ja esiintyminen
- kirjoitusviestintä
- projektin johtaminen ja hallinta
- käytettävyyden luennot ja ryhmätyöt
- tekijänoikeus ja sopimukset
- versiohallinta sekä
- kaksi väliesittelyä.

6 Yhteistyö Gitorious-yhteisön kanssa

Luvussa kuvataan projektin aikana toteutunutta yhteistyötä Gitorious-yhteisön kanssa. Projektiryhmä kommunikoi projektin aikana Gitorious-yhteisön kanssa pääasiassa Freenode-verkossa #gitorious-IRC-kanavan sekä Gitorious-sähköpostilistan [16] avulla.

6.1 Ongelmien ratkaiseminen

Gitorious-sovellus sisältää monia ohjelmakokonaisuuksia, ja sen asentaminen sisältää monia vaiheita. Myös sovellukseen liittyvien virhetilanteiden selvittäminen on välillä haastavaa. Gitorious- sovellukseen liittyvissä ongelmatilanteissa projektiryhmä on ensin yrittänyt itse ratkaista ongelman lähdekoodia tai Interetistä löytynyttä dokumentaatiota tutkimalla. Hankalissa tapauksissa projektiryhmä on kysynyt apua #gitorious-kanavalta, josta on usein saatu tarvittava apu.

Asennus- ja ohjelmointipulmissa kysymykset vähentyivät projektin alun jälkeen projektiryhmän oppiessa sovelluksesta ja sen ympäristöstä lisää. Projektiryhmä on kevään kuluessa myös neuvonut #gitorious-kanavalla muita.

6.2 Tulosten julkistaminen yhteisölle

FIXME: tähän raporttimuodossa asiat, kun versiohallintapulma on ratkennut.

6.3 Yhteisöltä saatu palaute

Verso-projekti on lähettänyt Gitorious-alustaan muokkauspyyntöinä (engl. *merge request*) bugikorjauksia, joista 24.5.2010 mennessä varsinaiseen Gitorious-sovellukseen oli Gitoriouksen pääkehittäjän Johan Sørensenin toimesta hyväksytty kolme.

Projektilaiset ovat lisäksi lähettäneet sähköpostitse huomioita Gitorious-sovelluksen parantamiseksi. Esimerkki onnistuneesta vuorovaikutuksesta oli Gitoriouksen mainline-version ulkoasun vaihduttua 18.5.2010, kun projektilaiset esittivät sekä IRC-kanavalle

että postilistalle parannusehdotuksia siihen. Ulkoasun kehittänyt Ole Martin Kristiansen kiitti ehdotuksista ja kertoi toteuttaneensa yhden ja kirjanneen loput kehitettäväksi [17].

FIXME: palautteesta lisää kun projektiraportti 1.0, tilanne elää. Tällä hetkellä palautetta lähinnä "cool", "just what I need", "nice", "I'll have a look with that".

7 Käytänteet

Luvussa kuvataan projektissa käytettyjen käytänteiden toteutumista. Käytänteet pyrkivät ennenkaikkea varmistamaan laatua kaikilta osin. Sovellukseen keskittyvä laadunvarmistus on kuvattu myöhemmin luvussa 8. Käytänteisiin kuuluvat myös versiohallintaan liittyvät käytänteet, jotka ovat laajuutensa takia kuvattu omassa luvussaan 9.

7.1 Tiedotus

Projektiryhmän tiedotuksesta vastasi käsillä olevasta asiasta vastaava projektiryhmän jäsen tai projektipäällikkö.

Projektiryhmän jäsenten, tilaajan ja ohjaajien välinen tiedotus hoidettiin sähköpostilla käyttämällä sähköpostilistaa `verso@korppi.jyu.fi`. YouSource-sovelluksen esikäyttäjiä varten perustettiin oma säköpostilista `yousource-users.group@korppi.jyu.`

Projektiryhmän jäsenten välinen tiedotus hoidettiin sähköpostia ja IRC-kanavaa käyttämällä. Jäsenillä oli muiden sähköpostilistojen lisäksi oma sähköpostilista keskinäiseen viestinvaihtoon.

7.2 Palaverit

Projektipalavereja pidettiin 1–2 viikon välein. Projektipalavereihin osallistuivat ryhmän jäsenet, ohjaajat ja tilaajan edustajat. Ensimmäisessä palaverissa [3] sovittiin palaverin olevan päätösvaltainen, kun projektiryhmästä ja tilaajista oli paikalla vähintään yksi edustaja sekä ohjaajista vastaava ohjaaja Santanen. Tilaajalla oli myös mahdollisuus tarvittaessa nimittää ulkopuolinen edustaja palaveriin. Palaverit olivat laillisia, kun palaverikutsu ja esityslista olivat lähetetty vähintään vuorokautta ennen palaveria.

Palaverit avasi edellisen palaverin puheenjohtaja, minkä jälkeen projektiryhmä ehdotti keskuudestaan puheenjohtajaa ja sihteeriä. Projetiryhmä kierrätti puheenjohtajan ja sihteerin rooleja siten, että kukin toimi projektin aikana vähintään kahdesti puheenjohtajana ja sihteerinä. Palavereissa käytiin läpi kokouskutsun mukana lähe-

tetty esityslista, johon kokoukseen osallistujilla oli mahdollisuus esittää tarvittaessa lisäyksiä esityslistaa hyväksyttäessä.

Edellisen palaverin pöytäkirja oli lähetetty sähköpostilistalle tarksitettavaksi etukäteen, joten palavereissa voitiin käydä läpi edellinen palaveri vain päätöksien osalta. Edellisen palaverin pöytäkirja voitiin hyväksyä tai se voitiin hyväksyä muutoksin. 4. palaverista lähtien palavereiden runkona oli projektiryhmän laatima kalvoesitys, joka heijastettiin videoprojektorilla nähtäville.

Palavereiden jälkeen sihteeri laati palaverista pöytäkirjan, joka oli ensin projektiryhmän hyväksyttävänä, minkä jälkeen se lähetettiin projektiroganisaatiolle sekä kirjoitusviestinnän opettaja Leena Peltomaalle tarkistettavaksi.

7.3 Lähdekoodin käytänteet

Ohjelmoidessa noudatettiin ensimmäisen palaverin [3] mukaisesti Ruby-kielen [9] Ruby on Rails -ohjelmistokehyksen ohjelmointityyliä sekä Gitorious- ohjelmiston ohjelmointikäytänteitä [9]. Ohjelmakomponentit pyrittiin pitämään mahdollisimman yhteensopivana Gitorious-pääprojektin kanssa. Tavoitteena oli tuottaa mahdollisimman monesta Gitorioukseen projektia varten tehdystä muokkauksesta muokkauspyyntö (engl. *merge request*) Gitorious-pääprojektiin.

Lähdekoodiesimerkki:

```
class Repository < ActiveRecord::Base
  include ActiveMessaging::MessageSender
  ...

  def can_view?(a_user)
    if REPO_VIEWABLE_EVERYONE
      return true
    else
      return viewer?(a_user)
    end
  end
  ...

end
```

7.4 Dokumentoinnin käytänteet

Sovellukseen liittyvät dokumentit kirjoitettiin englanniksi ja projektiin liittyvät dokumentit suomeksi. Englanninkielisissä dokumenteissa käytettiin amerikanenglantia.

7.5 Julkistettujen dokumenttitiedostojen versionumerointi

Dokumenteissa käytettiin kolmitasoista versionumerointia ensimmäisen palaverin [3] mukaisesti. Ensimmäinen ryhmän jäsenien ulkopuolelle julkistettu versio oli 0.1. Ryhmän sisäiseen käyttöön tarkoitettuja versionumerot kasvatettiin 0.0.1:llä, ohjaajalle, kirjoitusviestinnän opettajalle ja tilaajalle versionumerot kasvatettiin 0.1:llä. Dokumentin ensimmäinen hyväksytty versio oli 1.0.0.

7.6 Dokumenttien hakemistorakenne

Dokumenttitietovaraston hakemistorakenne oli projetin WWW-sivuilla sekä CD-levyillä seuraavanlainen:

```
.
|-- application
|   |-- application_report
|   |-- pics
|   |-- requirement_specification
|   |-- solutions
|   |-- system_test_plan
|   `-- wishlist
`-- project
    |-- esittelyt
    |   |-- loppuesittely
    |   |-- valiesittely1
    |   `-- valiesittely2
    |-- katselmoinnit
    |-- kaytettavyystestaukset
    |-- palaveriesitykset
    |   |-- palaveri4
    |   |-- palaveri5
    |   |-- palaveri6
    |   |-- palaveri7
    |   |-- palaveri8
    |   `-- palaveri9
    |-- palaverit
    |-- projektiraportti
    `-- projektisuunnitelma
```

8 Laadunvarmistus

Luvussa keskitytään sovelluksen laadun varmistamiseen ja laadunvarmistuksen toteutumiseen.

Projektiin ei osin sen prototyyppiluonteen vuoksi ja osin kesken jääneen projektisuunnitelman vuoksi suunniteltu kattavaa testausta. Tilaaja vaati projektin alusta lähtien esikäyttäjien käyttämistä, mutta toisaalta esimerkiksi järjestelmätestaus suunniteltiin täysin projektisuunnitelman kehittämisen keskeyttämisen jälkeen.

8.1 Integraatiotestaus

Sovelluksessa käytettiin versiohallintaa siten, että keskeneräinen koodi pidettiin erillään varsinaisesta päähaaran (engl. *master branch*) ohjelmakoodista, kunnes kehittäjä totesi sen olevan valmis esikäytettäväksi. Versiohallinnan käytänteistä lisää kappaleessa 9.

Käytännössä toteuttaessaan ominaisuutta sekä sen toteuttamisen jälkeen kehittäjä varmisti toteuttamansa ominaisuuden toimimisen omalla kehityspalvelimellaan osana sovellusta ennen sen siirtämistä osaksi varsinaista sovelluskoodia. Kehittäjä testasi kehittämänsä toiminnon lisäksi ne paikat, joihin arveli ominaisuuden aiheuttaman muutoksen vaikuttavan. Ominaisuutta käyttöönottaessa ei tehty raskasta järjestelmätestausta, vaan kehittäjän oman mielipiteen jälkeen koodi siirtyi projektiorganisaation ja esikäyttäjien käyttöön.

8.2 Automaattiset poikkeustiedotteet

Gitorious-sovellus mahdollisti automaattisten poikkeustilanteiden raportoinnin. Käytännössä aina, kun esikäytössä olleessa sovelluksessa tapahtui poikkeus, siitä lähti sähköpostiviesti koko projektiryhmälle. Näin sovellukseen jäämien virheiden löytäminen ei ulkopuolistenkaan esikäyttäjien puolesta nojannu ilmoitusahkeruuteen, vaan virhetilanteesta ja siihen johtaneesta sivupyynnöstä saatiin tieto aina.

Poikkeuksen tapahduttua ryhmä kommunikoi keskenään, miten tilanteeseen oli tarpeen reagoida. Sovellus tuottaa lisäksi viestejä kymmenkuntaan eri lokitiedostoon, josta oli usein löydettävissä lisää tietoa virhetilanteen toistamiseksi kehitystä varten.

8.3 Esikäyttäjien hyödyntäminen

Projektin aikana sovellusta päästettiin käyttämään parikymmentä esikäyttäjää, joilta pyydettiin palautetta sovelluksen ominaisuuksista ja käytettävyydestä. Esikäyttäjien hyödyntämisestä laadittiin erillinen dokumentti "Suunnitelma esikäyttäjien hyödyntämisestä" [2].

Tilaajan halusi ottaa projektiorganisaation ulkopuolisia jäseniä esikäyttäjiksi mahdollisimman aikaisessa vaiheessa. Esikäytön toteuttamisesta laaditun suunnitelman valmistuminen kuitenkin venyi siihen esitettyjen parannusehdotusten vuoksi usean projektipalaverin ajan. Yli kuukauden projektiryhmän käytön alkamisesta, suunnitelma hyväksyttiin ja YouSource-sovellus tarjottiin esikäytettäväksi 30.3.2010.

Palvelimen `versotest.it.jyu.fi` verkkokonfiguraatio salli pääsyn vain tiettyihin Jyväskylän yliopiston tietotekniikan laitoksen projekti- sekä henkilökuntatiloihin. Tämä esti käytännössä täysin opiskelijoiden hyödyntämisen esikäytössä. Sovellus avattiin koko Jyväskylän yliopiston verkkoon 13.4.2010, projektiryhmän oman käytön alkamisesta noin 7 viikkoa myöhemmin. Avaus mahdollisti viimein opiskelijoiden sekä kotoaan esikäyttävien tutkijoiden käytön.

Projektiorganisaation ulkopuolinen esikäyttö jäi Verso-projektin osalta vähäiseksi. Sovellukseen kirjautui projektin aikana noin 30 projektiorganisaation ulkopuolista käyttäjää, joista vain harva loi sovellukseen omia tietovarastoja. Myös esikäyttäjien viestintään varattu sähköpostilista `yousource-users.group@korppi.jyu.fi` jäi lähes täysin yksipuoleiseksi projektiryhmän tiedotuskanavaksi.

On hankala sanoa, olisivatko aiempi verkosta pääsyn helpottaminen ja 19.4.2010 julkistetun Korppi-tunnuksilla tapahtuvan kirjautumisen julkistaminen aiemmin vaikuttaneet ratkaisevasti esikäyttöön. `versotest.it.jyu.fi`-palvelimelle ei edelleenkään pääse Jyväskylän yliopiston verkon ulkopuolelta, joten varsinaista omien lähdekoodien julkistamista esikäyttäjä ei voi vielä tehdä. Näin ollen voi olla, että houkutus käyttää voisi kasvaa palvelun päästessä täysin julkiseksi.

8.4 Järjestelmätestaus

Tero Hänninen laati YouSource-sovelluksen järjestelmätestaussuunnitelman [15], joka määrittelee järjestelmän toimintojen hyväksymiskriteerit kohdittain.

YouSource-sovelluksen version `verso-project-rc1` -järjestelmätestauksessa saatiin kiinni ainakin yksi virhe, joka korjattiin sovellukseen ennen sen lopullista versiota.

FIXME: lisää järjestelmätestausraportit jahka valmistuvat

8.5 Käytettävyyspäivä

Meeri Mäntylä piti projektille käytettävyysluennon 29.3.2010. Luennolla käsiteltiin sovelluksen käytettävyyteen liittyviä kriteerejä ja toteutettiin ryhmätyönä käytettävyystestaus, mikä toimi hyvänä harjoituksena myöhemmin toteutetuille käytettävyystestauksille.

Luennon lisäksi Mäntylä kävi läpi YouSource-sovellusta ja nosti esiin havaitsemiaan seikkoja. Selkeitä käytettävyyspuutteita kirjattiin tiketeiksi, minkä lisäksi käytettävyyspäivästä tehtiin kokonaisuudessaan muistio. Muistio löytyy liitteenä projektin dokumentaatiosta.

8.6 Käytettävyystestaukset

Verso-projekti suoritti YouSource-sovelluksen käytön aloittamiseen liittyvistä toimista kaksi käytettävyystestausta, joissa käytettiin tilaajan nimeämiä testihenkilöitä. Käytettävyystestauksissa esille noussee asiat kirjattiin muistioiksi ja tiketeiksi, minkä lisäksi niiden yleiset huomiot käytiin läpi projektipalaverissa. Käytettävyystestauksista laaditut muistiot löytyvät liitteinä projektin dokumentaatiosta.

8.7 Koodinkatselmoinnit

Sovellusprojekti-kurssiin kuuluu luonnostaan sovelluksen koodinkatselmointeja, joita pidettiin kaksi.

Projekti asetti vieraan kielen Rubyn ja sovelluskehyksensä Ruby on Railsin osalta haasteita sekä projektiryhmälle että tekniselle ohjaajalle, koska molemmille osapuolille nämä olivat entuudestaan vieraita. Projektin tekninen ohjaaja Antti-Juhani Kaijanaho oli projektia varten tutustunut Rubyyn ja Ruby on Railsiin ja pystyi koodin-

katselmoinneissa nostamaan huomioissaan myös monia kielen ja sovelluskehyksen käytänteisiin liittyviä seikkoja.

Koska projekti käytti järjestelmän pohjana valmista Gitorious-sovellusta, niin sanotusti kaikkea projektin tuottamaa lähdekoodia ei katselmoinneissa voitu käydä läpi.

8.8 Pariohjelmointi

FIXME: add juttua pariohjelmoinnista.

9 Versiohallinnan käyttö

Luvussa kuvataan suunniteltua ja toteutunutta versiohallinnan käyttöä projektissa. Versiohallinta on ollut monella tavalla tärkeä osa Verso-projektin kehitystyötä sekä oppimiskokemusta.

Kesken jääneessä projektisuunnitelmassa kuvattiin projektissa käytetyt versiohallintakäytänteet, jotka vastasivat hyvin toteutunutta. Kuitenkin, kuten projektiryhmä myöhemmin huomasi tarjotessaan ominaisuuksia muille kehittäjille, suunnitelma olisi voinut olla monella tavalla parempi.

Toteutuneet tavat käyttää versiohallintaa toimivat kuitenkin Verso-projektin ajan hienosti. Koska versiohallinnan käyttö oli valtaosalle projektiryhmästä vierasta, on todettava, että oppimisen kannalta Verso-projekti oli erittäin hyödyllinen. Myös havaittujen ongelmien korjaaminen on mahdollista ja pakottaa opettelemaan edelleen uusia menetelmiä.

9.1 Suunnitellut käytänteet

Sovellukseen liittyvä lähdekoodi tallennettiin **Git-versiohallintaan**. Sovellusta kehitettiin omaan tietovarastoonsa ja projektin dokumentteja, kuten muistioita, palaverien pöytäkirjoja, väliesittelymateriaaleja omaan tietovarastoonsa. Väliaikaishaaroja (engl. *topic branch*) käytettiin sekä projektitietovarastossa että etenkin sovelluksen tietovarastossa. Väliaikaishaara sisälsi esimerkiksi tiettyyn ominaisuuteen tai asiakokonaisuuteen (kuten tiettyyn pöytäkirjaan) liittyvät päivitykset.

Sovelluksen kehityksen kannalta **haarojen käyttäminen** oli tärkeää, koska sovelluksen päähaaraara oli yhtä aikaa julkaisuhaara esikäyttöä varten. Epävakaa ohjelmakoodi oli pidettävä omissa haaroissaan sen koekäyttöön valmistumiseen asti. Kun sovellukseen liittyvä ominaisuus valmistui, sen haara yhdistettiin (engl. *merge*) päähaaraan (engl. *master branch*) ja ominaisuuden historian sisältävä haara jätetään versiohallintaan. Ominaisuuksien tarjoamisen Gitorious-sovellukselle ajateltiin olevan helpompaa, kun projektissa tuotettavien ominaisuuksien haarat olisivat lähdekoodeja tarjotessa tallessa ja erillään muusta ohjelmakoodista.

Projektin ajan käytetty **haarojenkäyttöstrategia** (engl. *branching strategy*) toimi projektin oman kehityksen tarpeisiin hienosti. Vasta toukokuussa tehdessään yhdistä-

mispyyntöjä (engl. *merge request*) projektiryhmä huomasi puutteita haarojenkäyttöstrategiassaan, jotka tekivät muokkauspyyntöjen lähettämisestä alkuperäiseen sovellukseen haastavaa.

9.2 Havaitut puutteet

Vaikka käytetty menetelmä suurempien ominaisuuksien pitämisestä erillään päähaarasta toimi projektin ajan, projektiryhmä huomasi projektin päättyessä, että haarojen kanssa olisi pitänyt olla huomattavasti tarkempi.

YouSource-sovelluksen **kehitys tehtiin Gitorious-sovelluksen kantaversion** (engl. *mainline*) **päähaaran päälle** lisäten siihen omia muutoksiaan ja yhdistäen (engl. *merge*) siihen kantaversion päähaarassa tapahtuneet muutokset. Verso-projektin kehittämät ominaisuudet kehitettiin tämän yhdistetyn päähaaran päälle.

Kuitenkin, ominaisuuksien tarjoamista varten olisi ollut parempi pohjata kehitys **täysin koskemattomaan Gitorious-sovelluksen päähaaraan**, jonka päälle muutokset olisi pohjattu. Näin omat muutokset, ominaisuudet ja korjaukset, olisivat olleet suoraan yhteensopivia alkuperäisen Gitorious-kantaversion ja monien sen kloonien kanssa.

YouSource-sovelluksen logo, omat tekstit ja pienet käyttöliittymämuutokset olisi lisäksi pitänyt pitää muiden ominaisuuksien tavoin haarassa, jota pidetään yhteensopivana Gitorious-kantaversion päähaaran kanssa.

Lisäksi kevään alussa **yhdistämispyyntöjen tekemiseen ei perehdytty käytännössä**. Sen tehdessä olisi selvinnyt, että itse asiassa jokainen yhdistämispyynnön arvoinen muokkaus tai korjaus kannattaisi pitää omassa haarassaan. Näin ollen Verso-projektin aikana tarvittujen haarojen määrä olisi kasvanut merkittävästi nykyisestä noin kymmenistä moniin kymmeniin. Sopivalla nimeämiskäytännöllä tämä ei kuitenkaan olisi ollut ongelma. Haarojen tekeminen ei varsinaisesti ole yleinen vaatimus, mutta Gitorious-sovelluksessa yhdistämispyyntöjen tekeminen olisi siten helpointa, koska yhdistämispyyntötoiminto ei mahdollista vain tiettyjen muokkauksien (engl. *commit*) poimimista (engl. *cherry pick*).

Yleinen ongelma versiohallinnan käytössä ovat **toisistaan riippuvat ominaisuudet**, joiden toteuttamisen kanssa tilanne on samankaltainen kuin niiden työajankirjaukseenkin liittyen. Ohjelmoidessa toteutettu toiminnallisuus on mahdollisesti vaatinut jonkin ongelman korjaamista tai toisen toiminnallisuuden kehittämistä ennen työn

alla olleen toiminnallisuuden toteuttamista. Kun työn alla ollut toiminnallisuus valmistuu, siihen liittyvät muokkaukset pitäisi lähettää versiohallintaan kuin toimintoa varten kehitetyt toiminnallisuudet olisi kehitetty ensin ja lähettää työn alla ollut toiminnallisuus vasta sitten.

Selkeiden muutoksien lähettämisestä on kuitenkin joissain tilanteissa työlästä pitää kiinni. Riskinä on, että versiohistoriaan päätyy **useita toimintoja sisältäviä muutoksia**. Työn alla ollut ominaisuus on mahdollisesti vaatinut korjauksia muissa haaroissa kehitettyihin toimintoihin, ja muutoksien lähettäminen oikein vaatisi niiden synkronointia, mikä voi olla työlästä. Kuitenkin, muutoksien jakamiseen käytetty työ niitä lähettäessä on kevyempi kuin muutoksien jakaminen myöhemmin.

Muokkausviesteihin (engl. *commit message*) olisi ollut hyödyllistä kirjoittaa viestin alkuun työn alla olleen Trac-tiketin numero. Tämä olisi ensinnäkin auttanut YouSource-tietovaraston jälkikäteen tehtyä muokkausta, josta lisää kappaleessa 9.4. Lisäksi käytänne tikettinumeron lisäämisestä olisi osaltaan auttanut varmistamaan, että muokattavasta asiasta on tehty Trac-tiketti. Tämä käytänne ei kuitenkaan tullut projektiryhmälle ennen toukokuuta mieleen, eikä projektihallintasovelluksen käyttämiseen sovellusprojekteissa ole valmiita käytänteitä, koska näin sovellusten käyttö projekteissa on toistaiseksi vapaaehtoista ja harvinaista.

Ryhmä teki myös suunnitelmaan nähden muutaman **selkeän virheen**. Esimerkiksi kun tietyn toiminnon kehityshaaran kehittämistä varten tarvittiin jokin tuore muutos päähaarasta, joka ei siis automaattisesti näy kehityshaarassa, muutos otettiin kehityshaaraan yhdistämällä siihen päähaaraan tehdyt muutokset. Tarve olisi pitänyt ehdottomasti ratkaista siirtämällä kehityshaara alkamaan uudesta alkupisteesta (engl. *rebase*), jotta kehityshaaraan ei päätyisi ylimääräisiä muutoksia.

9.3 Soveltuvampi haarojenkäyttöstrategia

Idea soveltuvasta haarajenkäyttöstrategiasta syntyi toukokuussa. Kuten edellä kuvattiin, Gitorious-kantasovellus olisi pitänyt pitää erillään kaikesta omasta kehityksestä ja kehittää toiminnallisuudet ja korjaukset sen päälle. Haaroja tulee mahdollisesti paljon, mutta sopivilla nimeämiskäytänteillä ne pysyvät hallittavissa. Vaikkapa nimeämällä ne `fix/` ja `feature/` -alkuisesti ja mahdollisesti lisäämällä niihin Trac-tiketin numeron pääsisi nimeämiskäytänteissä jo pitkälle.

Omat sovelluksen personoinnit (engl. *customization*) olisi pitänyt pitää ominaisuuk-

sien tavoin omassa haarassaan.

Tärkeintä olisi **koota erillisistä haaroista koostuva oma sovellus omaan haaraansa** käyttöä varten. Nykyisen "YouSource-tietovaraston" sijaan olisi parempi olla YouSource vain yhtenä haarana Gitorious-kantasovellukseen pohjautuvassa tietovarastossa. Haara voi yhtä hyvin olla palvelimen nimi, etenkin jos personointeja halutaan jatkossa erilaisia.

Myös edellä kuvattu haarojenkäyttöstrategiassa **asettaa haasteita**. Kehityksen tapahtuessa jatkuvasti ylläpidettävissa haaroissa, niitä pitää toisinaan esimerkiksi synkronoida alkavaksi uudemmasta pisteestä Gitorious-kantasovelluksen päähaarasta. Tämän tekeminen vaatii sovitut käytänteet kehittäjien keskuudessa, sillä hallitsematon kehityshaaran alkupisteen siirtäminen voi johtaa vaikeuksiin muiden kehittäjien synkronoidessa tietovarastoaan.

9.4 Tietovaraston rakenteen jatkokehitys

Koska YouSource-sovelluksen kannalta on hyödyllistä tarjota ominaisuuksista yhdistämispyyntöjä, ominaisuuksien "perkaamista" Gitorious-kantasovelluksen päähaaran päälle rakentuvaksi kannattaa jatkaa. Verso-projektin aikana sitä ehdittiin neljän korjauksen osalta jo tekemään, joiden pohjalta tehdyistä muutospyynnöistä kolme hyväksyttiin projektin aikana osaksi Gitorious-kantasovellusta.

Haaroja muokatessa on tärkeää, että **muokkauksien kuvauksissa** kerrotaan selkeästi, mitä toiminnallisuutta muokkaus koskee. Projektiryhmä on projektin aikana valvonut omatoimisesti muokkausviestien sisältöä ja kuvaavuutta, mikä auttaa muokatessa suuresti.

10 Trac-käytänteet

Projektissa käytettiin vaatimusmäärittelyn hallintaan Trac-ohjelmistoa. Koska erillistä vikojenseurantaohjelmistoa (engl. *bugtracker*) ei käytetty, projektiryhmän ja tilaajan välisen tehtävänhallinnan lisäksi Traciin kirjattiin myös käyttäjien kautta tulevat vikailmoitukset ja toiveet.

Trac-sovellusta käytettiin pääosin projektisuunnitelman [13] kuvaamalla tavalla, mutta Trac osoittautui myös joiltain osin puutteelliseksi. Projektin aikana havaitut puutteet olivat pääasiassa käytettävyysongelmia.

10.1 Käyttöoikeudet

Verso-projektin Tracin selaamiseen ei vaadittu kirjautumista, mutta muokkaaminen oli mahdollista vain luvussa 5.1 mainituille projektiorganisaation jäsenille.

Avoin pääsy antoi mahdollisuuden selata tehtäviä myös esikäyttäjille ja muille projektista kiinnostuneille. Projektiorganisaation ulkopuolisen ilmoittajan kanssa yhteyttä pitäessä voitiin viitata suoraan Tracin tiketteihin.

10.2 Kirjoitusasu ja otsikot

Tracia ei käytetty koko Verso-projektin tehtävien hallintaan, vaan ainoastaan siinä tuotettavan YouSource-sovelluksen ja sen esikäyttöä varten pystytetyn asennuksen tehtävien osalta. Pääsääntönä oli, että mikäli tehtävä tai idea liittyy sovellukseen, sen sai kirjata Traciin. Kuten muissakin sovellukseen liittyvissä dokumenteissa, Tracissa käytettiin kirjauskielenä englantia.

Tiketin **tiivistelmäksi** (engl. *summary*) tiketin lisääjä kirjasi mahdollisimman yksiselitteisen kuvauksen, koska tiivistelmä toimi tiketin otsikkona. Selkeän otsikon avulla tiketti löytyi tikettilistoista ja sähköpostikansioista helpommin.

Tiketin **kuvausta** (engl. *summary*) käytettiin selvittämään tarkemmin tiketissä käsiteltävä asia. Kuvaukseen pystyi kirjaamaan esimerkiksi tikettiin liittyviä kehitysratkaisuja tai sovellukseen lisättäviä tekstejä.

Tiketin luominen oli samalla pienimuotoinen keskustelunavaus tiketissä kuvattuun tehtävään liittyen. Tiketin **kommenteissa** (engl. *comment*) oli mahdollista kysyä tarkennuksia tehtävään liittyen, minkä perusteella sen lisääjä pystyi esimerkiksi muokkaamaan tiketin kuvausta tarkemmaksi.

Tikettien **otsikot eivät olleet muuttumattomia**. Mikäli tiketin otsikko ei kuvannut siinä käsiteltyä asiaa kyllin selkeästi, kaikilla projektiorganisaatioon kuuluvilla oli mahdollisuus muokata sitä paremmaksi. Tikettien vanha otsikko jäi tikettiin talteen, joten vaatimuuksiin liittyvällä informaatiolla ei ollut mahdollisuutta hukkua tässä yhteydessä.

Otsikkoja muokatessa on vaarana, ettei otsikko enää muokkauksen jälkeen kuvannut tilaajan hyväksymää asiaa. Tämä riski ei kuitenkaan ollut projektissa todellinen, koska tiketit toimivat pääosin muistin tukena palavereissa käsitellyistä asioista, ja ne tarkastettiin ennen toteutusvaiheiden aloittamista palavereissa.

10.3 Sähköposti ja tiketit

Trac-tiketteihin merkittiin cc-kenttään eli muutosilmoitusten vastaanottajaksi projektiorganisaation sähköpostilista. Lisäksi, jos ilmoitus tulee projektiorganisaation ulkopuolelta (esimerkiksi esikäyttäjiltä) jakelulistaan lisättiin ilmoittajan sähköpostiosoite, jotta ilmoittaja pysyi ajantasalla tehtävään liittyvistä muutoksista.

Muutosilmoitusten lähettäminen projektilistalle nähtiin tarpeelliseksi, jotta tilaaja, projektiryhmä ja ohjaajat saavat automaattisesti tiedon muutoksista. On vaikea sanoa, olivatko sähköposti-ilmoitukset lopulta hyödyllisiä, sillä muutosilmoitus lähti kaikista muutoksista. Tikettien muutoksien tekeminen oli pitkälle käsityötä parempien toimintojen puuttuessa ja kustakin muutoksesta lähti erikseen ilmoitus, vaikkei se tiketin sisältöön vaikuttanutkaan. Jos projektin päättyessä haluaisi esimerkiksi vaihtaa versionumeron kaikkiin tiketteihin, jonka siis joutuisi tekemään käsin kaikille tiketeille erikseen, lähtisi tästä toimenpiteestä lähes **150 sähköpostia ilman viestinnällistä hyötyä**.

10.4 Tikettityypit

Projektisuunnitelmassa **??** kuvatut tikettityypit osoittautuivat käyttökelpoisiksi, ja projekti toteutui pitkälti niitä käyttäen. Trac-sovellus asetti kuitenkin haasteita tikettien keskinäisten suhteiden ylläpitoon.

Tikettityyppi	Omistaja	Kuvaus
Feature	Tilaaja	Sovellukseen hyväksytty ominaisuus.
Chore	Tilaaja	Projektiryhmällä teetetty muu tehtävä.
Wish	Kuka tahansa	Kehitysidea tai muu toive.
Bug	Kuka tahansa	Sovelluksesta löytynyt vika.
task	Projektiryhmä	Pieni kehitystehtävä.

Taulukko 10.1: Trac-tikettien tyypit.

Taulukossa 10.1 on kuvattu erilaiset Tracissa käytetyt tikettityypit. Taulukon termillä omistaja tarkoitetaan sitä tahoa, jonka kommunikointivälineeksi tikettityyppi oli tarkoitettu.

Verso-projektin Traciin oli järjestetty lukuoikeus kaikille, mutta tietojen muokkaamiseen olivat oikeutettuja vain projektiorganisaatioon kuuluvat käyttäjät. Kehitysideoita ja bugeja tuli kuitenkin myös projektiorganisaation ulkopuolelta. Omistajalla "Kuka tahansa" tarkoitetaan kenen tahansa lähettämää huomiota, jonka vastaanottaja projektiorganisaatiossa tekee siitä tiketin.

10.5 Prioriteetit

Projektisuunnitelmassa **??** kuvatut tikettien prioriteetit olivat kaikki käytössä, ja projekti toteutui ilman suuria muutoksia niihin.

Prioriteetit (engl. *priorityt*) ovat tarkoitettu käytettäväksi Feature-tyyppisten tikettien kanssa, muiden tikettityyppien kanssa niitä käytettiin suuntaa antavana lisätietona. Prioriteetti ei suoraan liity ominaisuuden kehitetyksi tulemiseen Verso-projektissa, vaan niitä hyödynnettiin, kun toteutusvaiheisiin valittiin ja kiinnitettiin tehtäviä.

Prioriteetti	Kuvaus
Mandatory	Pakolliseksi nähtävä, toteutetaan ensimmäisten joukossa
Important	Tärkeä, ei pakollinen, toteutetaan ajan salliessa
Useful	Hyödyllinen, ei käytön kannalta tärkeä, toteutetaan ajan salliessa
Left out	Projektin osalta tarpeettomaksi nähty
–	Toistaiseksi priorisoimaton ominaisuus.

Taulukko 10.2: Trac-tikettien prioriteetit selityksineen.

10.6 Kestojen arviointi

Tikettien kestoja arvoitiin ensimmäisen kolmen kehitysvaiheen ajan vain ryppäittäin valittaessa kehitystehtäviä seuraaviksi kahdeksi viikoksi. Arviot pitivät paikkansa kohtuullisesti; yleensä vaiheelle valittuja tehtäviä jäi tekemättä yhden tai kahden kehityspävän edestä.

Koska neljäs kehitysvaihe oli viimeinen, jolle otettiin uusia kehitystehtäviä, projektiryhmä arvioi jäljelläolevien potentiaalisesti tilaajaa tärkeiksi katsomien tikettien kestot tunteina Traciin. Tämä osoittautui hyödylliseksi, kun neljännen vaiheen aloittavassa projektipalaverissa valittiin, mitkä ominaisuudet kehitetään ja mitkä rajataan jatkokehitykseen.

Projektiryhmän neljännelle vaiheelle arvoimat tikettien kestot pitivät keskimäärin paikkansa hyvin: kun KorppiLDAP:in toteuttaminen ja näkyvyyksien muokkaaminen venyivät pitkiksi, muut valitut tehtävät valmistuvat nopeammin.

11 Tehtävien jakautuminen

Luvussa tarkastellaan toteutunutta projektin vastuualueiden ja tehtävien jakautumista ja verrataan sitä suunniteltuun. Dokumentoinnin ja ohjelmoinnin vastuualueisiin ei tullut muutoksia projektin alussa suunniteltuun.

11.1 Vastuualueet dokumentoinnin osalta

Dokumenttien vastuualueita ei vaatimusmäärittelyä ja projektidokumentointia lukuunottamatta suunniteltu projektin alussa. Projektipäällikkönä Salo aloitti projektisuunnitelman kirjoittamisen ja Nieminen vaatimusmäärittelyn kirjoittamisen. Projektin neljännessä palaverissa [5] päätettiin, että erillisen dokumentin sijaan vaatimukset kirjataan Trac-järjestelmään. Tracissa vaatimuksista tehtyjä tikettejä on ylläpitänyt koko projektiorganisaatio.

Projektisuunnitelma jäädytettiin 12.4.2010, koska suunnitelman tekemisen ei enää katsottu edistävän projektin läpiviemistä. Projektisuunnitelmassa oli valmiina Trac- ja muiden käytänteiden kuvaus, projektin tavoitteet tuloksien ja oppimistavoitteiden osalta sekä kuvaus projektissa käytettävästä prosessimallista.

Projektisuunnitelmasta puuttui projektissa tuotettavan sovelluksen kuvaaminen, ja siinä oli vasta alustavasti kuvattu riskejä sekä tuntien jakautumista.

Tulos	Kieli	Vastuuhenkilö	Tarkastettavaksi	Hyväksytty
Projektisuunnitelma	suomi	HS		
Vaatimusmäärittely	englanti	JN		
Projektiraportti	suomi			
Sovellusraportti	englanti			

Taulukko 11.1: Dokumenttien kielet ja vastuualueet.

11.2 Toteutunut työmäärä

Kesken.

Vaiheet ja tehtävät		Tekijä				
Vaihe	Tehtävä	Juho	Marko	Tero	Heikki	Yhteensä
Esikäyttäjät	käytettävyystestaus	2:00	2:00	2:00		6:00
	haastattelu	1:30	1:30	1:30		4:30
	esikäyttötiedotus				8:00	8:00
	esikäyttöviestintä	6:00	6:00	6:00	14:00	32:00
	suunnittelu				10:00	10:00
Esikäyttäjät Total		*9:30*	*9:30*	*9:30*	*32:00*	*60:30*
Esitutkimus	tutustuminen	30:00	50:00	50:00	10:00	140:00
Esitutkimus Total		*30:00*	*50:00*	*50:00*	*10:00*	*140:00*
Oheiskurssi	esittelyt	15:00	15:00	15:00	15:00	60:00
	katselmoinnit	5:00	5:00	5:00	5:00	20:00
	koulutus	30:00	30:00	30:00	30:00	120:00
Oheiskurssi Total		*50:00*	*50:00*	*50:00*	*50:00*	*200:00*
Palaverit	raportointi	20:00	20:00	20:00	20:00	80:00
	seuranta ja hallinta	15:00	15:00	15:00	15:00	60:00
Palaverit Total		*35:00*	*35:00*	*35:00*	*35:00*	*140:00*
Projektin hallinta	muut tehtävät				5:00	5:00
	suunnittelu	25:00			100:00	125:00
	raportointi				60:00	60:00
	tiedotus	2:00	2:00	2:00	30:00	36:00
Projektin hallinta Total		*27:00*	*2:00*	*2:00*	*195:00*	*226:00*
Suunnittelu	suunnittelu	20:00	20:00	20:00	30:00	90:00
Suunnittelu Total		*20:00*	*20:00*	*20:00*	*30:00*	*90:00*
Toteutus	käyttöliittymä	40:00	40:00	40:00		120:00
	metatiedostot	20:00	20:00	70:00		110:00
	oikeudet ja näkyvyydet	50:00	80:00	30:00		160:00
	päivitystavat	30:00	30:00	40:00		100:00
	tukitehtävät	10:00	20:00	10:00	50:00	90:00
Toteutus Total		*150:00*	*190:00*	*190:00*	*50:00*	*580:00*
Viimeistely	sovellusraportti	50:00	5:00	5:00	5:00	65:00
	lähdekoodin siistiminen	10:00	10:00	10:00		30:00
	merge requestit	10:00	20:00	20:00		50:00
Viimeistely Total		*70:00*	*35:00*	*35:00*	*5:00*	*145:00*
Yhteensä		391:30	391:30	391:30	407:00	1581:30

Kuva 11.1: Jäsenten työmääräarvio

11.3 Vastuualueet ohjelmoinnin osalta

Kesken.

11.4 Työtunnit ja tehtäväjako

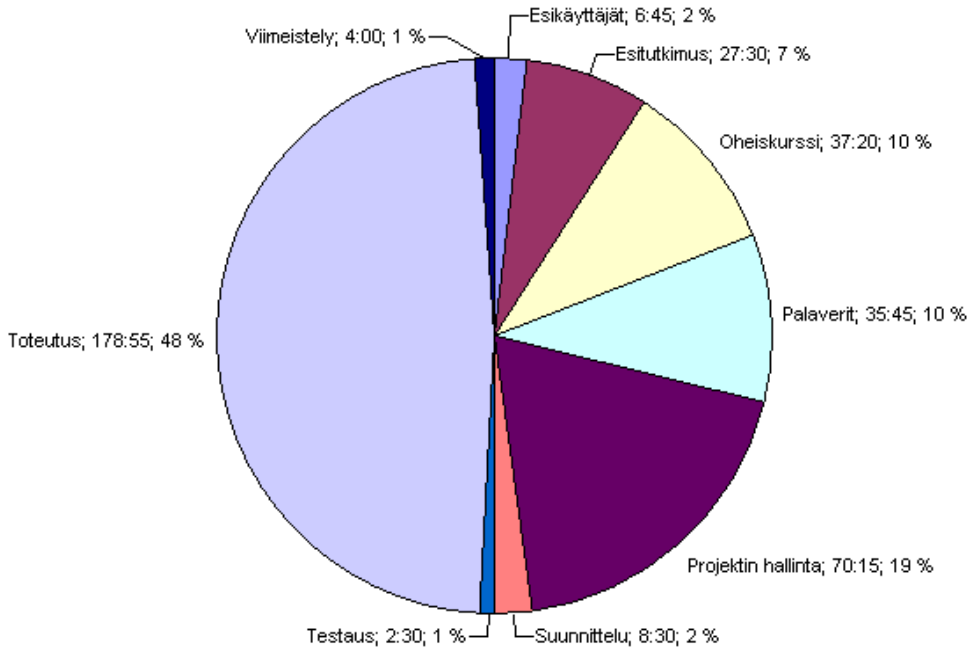
Kesken.

11.5 Ryhmän ajankäyttö tehtäväkokonaisuuksittain

Ryhmä käytti suhteellisesti selkeiten eniten aikaa toteutukseen.

11.6 Juho Niemisen ajankäyttö

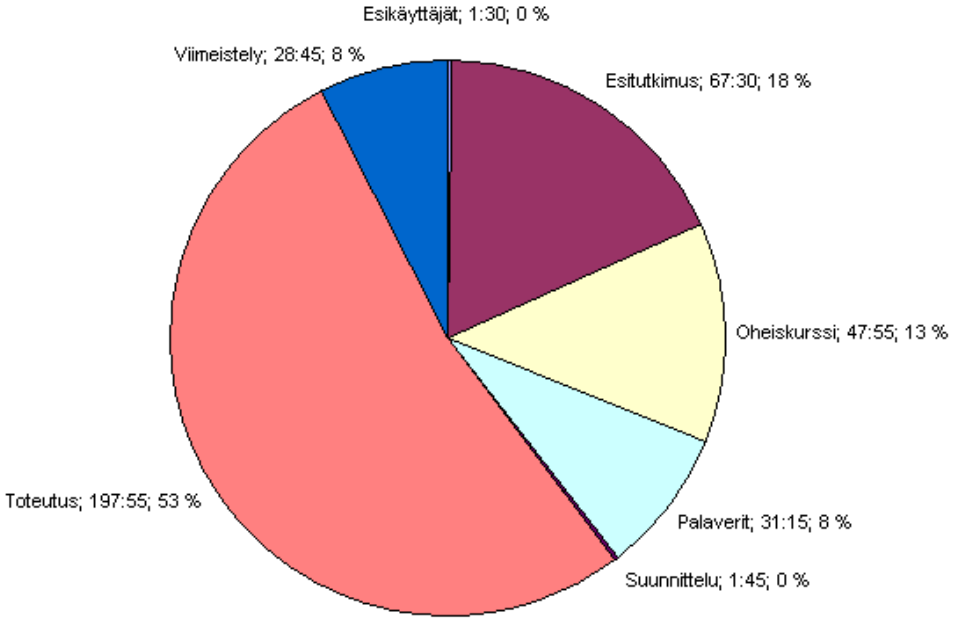
Juho oli Verso-projektin varaprojektipäällikkö, mikä näkyy hänen Teroa ja Markoa suurempana projektinhallintaan käytettynä aikana. Juhon vastuulla oli projektin alussa laatia vaatimusmäärittelydokumentti, jonka sisältö myöhemmin siirrettiin Trac-järjestelmään. FIXME: uudet tekstit kun tulee lopulliset kuvat.



Kuva 11.2: Juhon ajankäyttö tehtäväkokonaisuuksittain

11.7 Tero Hännisen ajankäyttö

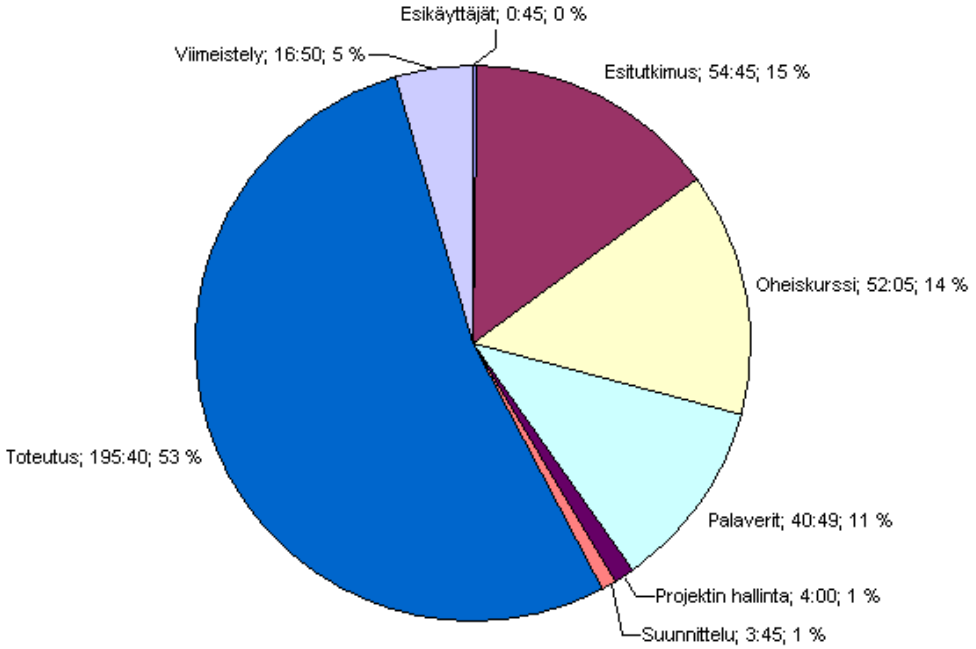
Teron ajasta kolme neljännestä kului esitutkimukseen, sovellusta kehittämiseen ja sen viimeistelyyn. FIXME: uudet tekstit kun tulee lopulliset kuvat.



Kuva 11.3: Teron ajankäyttö tehtäväkokonaisuuksittain

11.8 Marko Peltolan ajankäyttö

Markon ajasta kolme neljännestä kului esitutkimukseen, sovellusta kehittämiseen ja sen viimeistelyyn. FIXME: uudet tekstit kun tulee lopulliset kuvat.

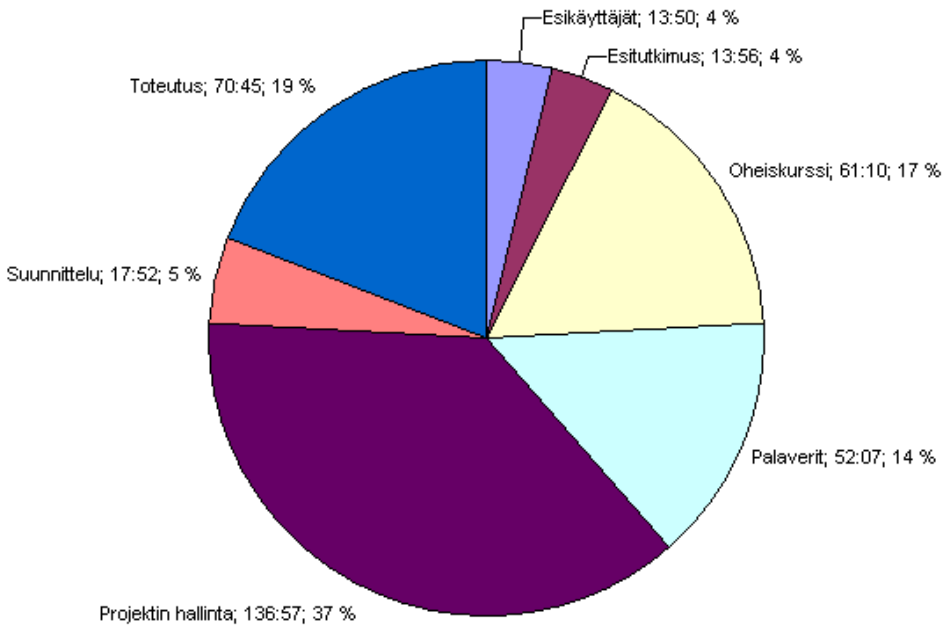


Kuva 11.4: Markon ajankäyttö tehtäväkokonaisuuksittain

11.9 Heikki Salon ajankäyttö

Heikki oli Verso-projektin projektipäällikkö, mikä näkyy suurena ajankäyttönä projektinhallintaan. Heikin ajasta noin viidennes on kulunut toteutukseen, joka on sisältänyt LDAP-autentikoinnin kehittämistä ja palvelimen asentamista. Koska projektisuunnitelma

FIXME: uudet tekstit kun tulee lopulliset kuvat.



Kuva 11.5: Heikin ajankäyttö tehtäväkokonaisuuksittain

12 Prosessimalli ja aikataulu

Projekti toteutettiin kevätlukukaudella 2010. Sovelluksen oli määrä olla valmis esiteltäväksi julkisesti toukokouun puolessa välissä. Projektin uusien ominaisuuksien ohjelmoinnin oli määrä olla valmis neljännen vaiheen päättyessä 23.4.2010.

12.1 Prosessimalli

Projekti vietiin läpi käyttäen ketterää ohjelmistokehitystä. Ohjelmistokehitys jaettiin kahden viikon mittaisiin vaiheisiin, joihin kiinnitetään käytettävissä oleviin resursseihin nähden järkevä määrä tehtäviä. Kiinnittämättömien Trac-tikettien, palaverikeskutelujen ja muun yhteydenpidon perusteella projektiryhmä valmisti tilaajalle ehdotuksen seuraavan vaiheen sisällöstä ja sen tehtävät hyväksyttiin projektipalavereissa.

12.2 Sovelluksen ominaisuuksien hyväksyminen

FIXME: raporttimuotoon Kun ominaisuus eli vaatimusmäärittelyssä Trac-tikettityyppi Feature valmistui, se hyväksytetään tilaajalla. Hyväksyttäminen tehdään tavallisesti siten, että projektiryhmä järjestää ominaisuuden nähtäville ja ilmoittaa siitä projektin sähköpostilistalla. Jos tilaaja ei hyväksy ominaisuutta, projektiryhmä korjaa ominaisuuteen liittyvät puutteet.

FIXME: raporttimuotoon Sovelluksen osalta projekti on hyväksytty viimeiseen 5. iteraatioon kiinnitettyjen Feature, Chore ja Bug -tyyppisten tehtävien tultua hyväksytyksi. Viimeiseen iteraatioon ei oteta Feature-tyyppisiä tehtäviä, ellei niiden kehitystä ole jo aloitettu.

12.3 Jatkuva integraatiotestaus

Sovellukseen tuotettavan lähdekoodin laatua testattiin jatkuvasti. Kun uutta ominaisuutta kehitettiin, sen lähdekoodi sijoitettiin omaan haaraansa (engl. *branch*) sovelluksen lähdekoodin tietovarastossa, kuten luvussa Versiohallinnan käytänteet 9

kuvataan. Kehityksen alla olevaa ohjelmakoodia testattiin pelkästään kehitystyöasemilla sen kehityksen ajan. Ominaisuuden ohjelmoinnin valmistuttua ja sen koodin vakauduttua se yhdistettiin sovelluksen versiohallinnan päähaaraan ja esiteltiin tilaajalle virtuaalipalvelimella. Epävakaata ohjelmakoodia voitiin tarvittaessa esitellä tilaajalle myös kehitystyöasemilta.

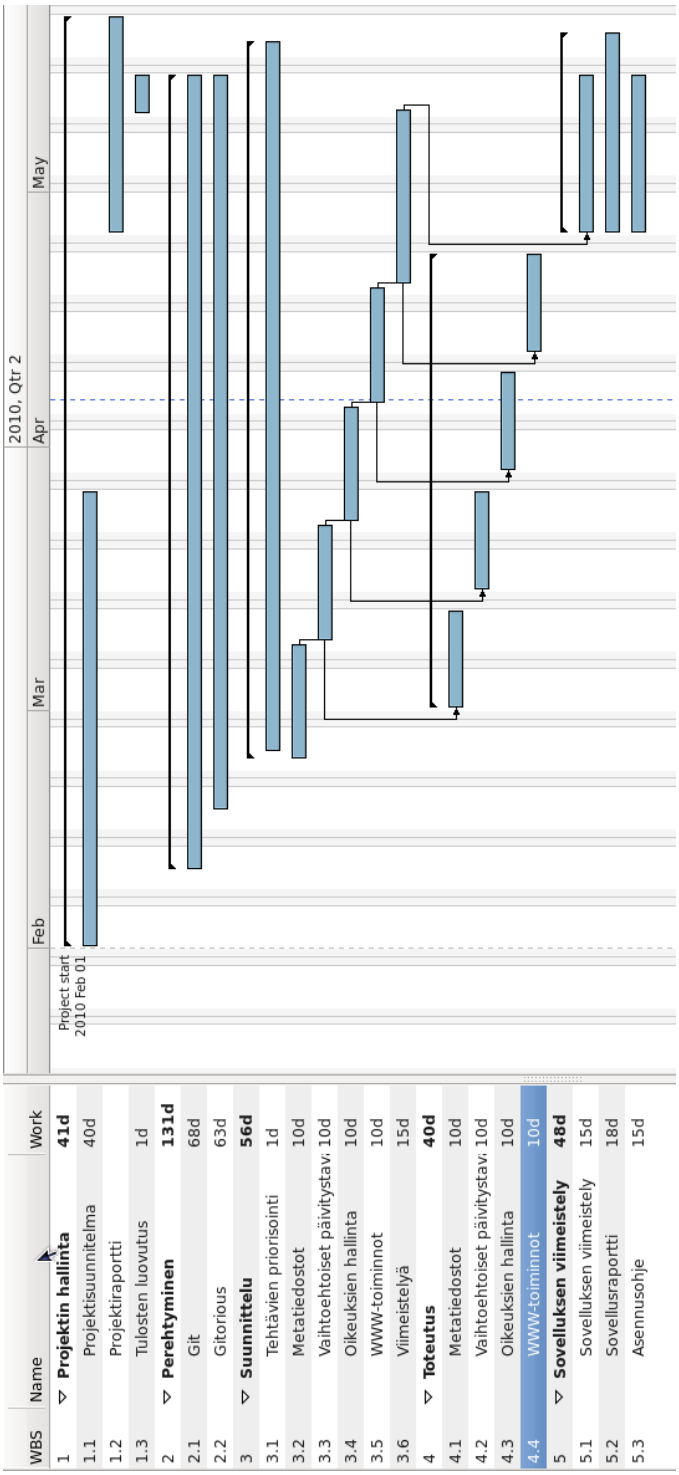
Kun ominaisuus yhdistettiin päähaaraan, se tuli osaksi projektiorganisaation ja esikäyttäjien käytössä olevaa sovellusta. Uusin vakaana pidetty ohjelmakoodi oli siis projektin ajan jatkuvassa integraatiotestauksessa. FIXME: raportin osalta tynkä

12.4 Aikataulu

Projektin tehtävien aikataulutus on esitetty kuvan 12.1 Gantt-kaaviossa.

12.5 Tarkistuspisteet ja versiot

Tarkistuspiste (engl. *milestoneilla*) viitataan projektin kehitysprosessin vaiheisiin. Vaiheiden aikataulut on kuvattu luvussa 12.4.



Kuva 12.1: Gantt-kaavio tehtävien aikataulutuksesta

13 Riskit ja niiden seuranta

FIXME: Kehitä luku raporttimuotoon

Luvussa kuvataan projektissa tiedostetut riskit sekä kuvataan toimia niiden ennakoimiseksi ja ehkäisemiseksi.

13.1 Riskien todennäköisyydet ja haitat

Riskien todennäköisyydet ja niistä seuraavat haittavaikutukset on esitetty taulukossa 13.1. Todennäköisyyttä ja haittavaikutusta arvioidaan asteikolla pieni, keskinkertainen ja suuri.

Riski	**Todennäköisyys**	**Haittavaikutus**
Muutostarve kehityskoneissa	suuri	keskinkertainen
Tavoitteiden rajaus	keskinkertainen	suuri
Valitun alustan ongelmat	keskinkertainen	suuri
Poissaolot	keskinkertainen	pieni
Testipalvelimen ongelmat	keskinkertainen	keskinkertainen
Motivaation puute	pieni	keskinkertainen

Taulukko 13.1: Arvioidut riskit, niiden todennäköisyys ja haittavaikutus.

13.2 Muutostarve kehityskoneissa

Projektiryhmän käytössä oleviin kehityskoneisiin on asennettu ryhmän toistaiseksi tarvitsemat kehitystyökalut ja sovelluskirjastot. Projektin edetessä kehitetään kuitenkin toisistaan hyvin poikkeavia kokonaisuuksia, joihin liittyy mahdollisesti tarve uusille asennuksille tai kehitysympäristön asetusten muutoksille. Projektiryhmä ei itse pysty toteuttamaan tarvittavia muutoksia, joten projektiryhmä on riippuvainen yliopiston lähituen palvelusta. Riskinä on, että jos vastaantullut ongelma on haastava ja vaatii selvittämistä yhdessä lähituen kanssa, sen ratkaisuun kuluu kauan. Riskiä hallitaan ennakoimalla muutostarpeita ja pyytämällä tarvittavat muutokset etukäteen.

13.3 Tavoitteiden rajaus

Tähän juttua siitä, että vaikka periaatteessa Gitorioukselle näyttäisi mahdolliselta toteuttaa kaavaillut ominaisuudet, sen käytön estyminen tekee projektista todella suuritöisen. Lisäksi, vaikka pohjaksi saataisiin toimiva alusta, ohjelmointitekniikat ovat uusia, ja vaatimusten toteuttamiseen saattaa liittyä ennalta tuntemattomia rajoitteita.

13.4 Valitun alustan ongelmat

Alustaksi valittu Gitorious valittiin, koska sen käyttöönoton estämiseen ei ollut tiedossa syytä ja se vaikutti lupaavimmaksi tavoitteiden saavuttamisen kannalta. On mahdollista, että ilmaantuu syy, joka estää sen käytön toistaiseksi. Esimerkiksi Gitorious-projektissa vielä toteutumattoman tietoturvariskin ilmeneminen, joka estää sen käytön yhdessä yliopiston autentikointimenetelmien kanssa. Tämä riski koskee kaikkea toimimista ulkopuolisten ohjelmaosien kanssa ja tätä voi hallita vain seuraamalla Gitorious-kehittäjälistalta ilmoitettuja vikoja ja niiden ratkaisuja.

13.5 Poissaolot

Ryhmän jäsenille voi tulla suunniteltuja poissaoloja, kuten matkat, ja yllättäviä, kuten sairastuminen. Jos ryhmän jäsen jää yllättäen pitkäksi aikaa pois, muu ryhmä joutuu ratkaisemaan tehtävät ilman poisjääneen tietämystä. Tämän riski on pieni, koska suuri osa ohjelmoinnista tehdään pariohjelmointina ja tietämys leviää sitä kautta hyvin.

13.6 Testipalvelimen ongelmat

Testipalvelimen ongelmilla tarkoitetaan odottamattomia tilanteita, joissa esikäytössä oleva sovellus toimii tuntemattomasta syystä väärin.

Projektissa tuotettava sovellus sisältää paljon erilaisia komponentteja, joiden yhteistoiminnasta sovellus on riippuvainen. Jos sovelluksen jokin osa vikaantuu ja alkaa

toimia väärin, tilanteen selvittäminen on mahdollisesti haastavaa ja aikaavievää, mikä myöhästyttää kehitystyötä.

Riskin hallitsemiseksi projektiryhmä harjoittelee erillisellä testipalvelimella erilaisten vikatilanteiden ratkaisua. Mahdollisia vikatilanteita pyritään myös ennakoimaan ja niissä toimimiseen määritellään selkeät käytänteet.

13.7 Motivaation puute

Tähän juttua siitä, että jos motivaatio laskee eikä intoa sovelluksen tekemiseen ole, alkaa sovelluksen hommat kasaantua kohti toukokuuta, mistä ei seuraa mitään hyvää.

14 Projektiryhmäläisten kokemuksia

Luvussa kuvataan projekilaisten kokemuksia Verso-projektista. Projekti onnistui laajuuteensa nähden hienosti. Projekti sisälsi ryhmän kannalta monia uusia asioita: uudet tekniset työkalut, tilaajan kanssa toimiminen, projektikäytänteet, ATK-tuen ja sovelluspalveluiden kanssa toimiminen ja niin edelleen.

FIXME: Tämä kappale ei ole valmis, ennen kuin kaikki ovat tähän kommentoineet.

14.1 Mitä tekisimme toisin?

Suurin yksittäinen toive toisin tehtävästä liittyy versiohallintaan ja siinä projektin loppupuolella vastaantulleisiin ongelmiin, jotka kuvattiin parannusehdotuksineen luvussa 9.

Projektin vaatimusmäärittelyyn olisi projektin alussa voinut myös keskittyä enemmän. Ajatus pysyvien ja versiottomien (engl. *static*) tiedostojen jakamisesta muotoutui vasta 8. projektipalaverissa, vaikka se liittyi olennaisesti ajatukseen lähdekoodien julkistamisjärjestelmästä ja siitä, mitä järjestelmällä oikeastaan halutaan julkistaa.

Kolmas muutostoive liittyy dokumentteihin. Esimerkiksi projektisuunnitelma olisi ehdottomasti ensin pitänyt luonnostella erillisiin pieniin, helposti hallittaviin osiinsa eikä yrittää tehdä siitä heti ladottua dokumenttia. Myös täysin keskeneräisiin dokumentteihin liittyvät sisältöä koskemattomat kieliasu- tai tyylihuomiot olisi pitänyt osata sivuuttaa täysin.

FIXME: kesken.

14.2 Tero Hännisen kokemuksia

FIXME: kesken.

14.3 Juho Niemisen kokemuksia

FIXME: kesken.

14.4 Marko Peltolan kokemuksia

FIXME: kesken.

14.5 Heikki Salon kokemuksia

FIXME: kesken.

15 Yhteenveto

Verso-projekti toteuttaa Jyväskylän yliopiston tietotekniikan laitokselle lähdekoodien julkistamisjärjestelmän prototyypin. Prototyyppi kehitetään Ruby-kielellä Ruby on Railsia käyttävän Gitorious-sovelluksen päälle.

Projektin aikana ryhmä saa kokemusta työskentelystä ohjelmistoprojektissa, useista eri työkaluista sekä projektin hallinnasta.

Projekti alkoi 3.2.2010 ja se päättyy 22.5.2010 mennessä. Projektin projektin läpiviennissä noudatetaan iteratiivista prosessimallia ja käytetään 5 iteraatiota.

16 Lähteet

[1] Petri Heinonen, Ajankäytönseurantasovellus, saatavissa Excel-muodossa <URL: http://appro.mit.jyu.fi/tools/ajankaytto/ajankaytonseuranta.xls>, Jyväskylän yliopisto, informaatioteknologian tiedekunta.

[2] Heikki Salo, "Suunnitelma esikäyttäjien hyödyntämisestä" <URL: http://sovellusprojektit.it.jyu.fi/verso/src/docs/project/suunnitelma_esikayttajien_hyodyntamisesta.txt>.

[3] Juho Nieminen, "1. palaverin pöytäkirja", saatavissa raakatekstimuodossa <URL: http://sovellusprojektit.it.jyu.fi/verso/src/docs/project/palaverit/poytakirja_1_palaveri.txt>, 17.2.2010.

[4] Juho Nieminen, "2. palaverin pöytäkirja" <URL: http://sovellusprojektit.it.jyu.fi/verso/src/docs/project/palaverit/poytakirja_2_palaveri.txt>, 25.2.2010.

[5] Marko Peltola, "4. palaverin pöytäkirja" <URL: http://sovellusprojektit.it.jyu.fi/verso/src/docs/project/palaverit/poytakirja_4_palaveri.txt>, 12.3.2010.

[6] Ville Tirronen, "Alustavaa lukemistoa projektiin liittyen".

[7] "About Gitorious", saatavissa HTML-muodossa <URL: http://gitorious.org/about>.

[8] Wikipedia, "Gitorious", saatavissa XHTML-muodossa <URL: http://en.wikipedia.org/wiki/Gitorious>.

[9] Gitorious, "Gitorious Coding Style Guide", saatavissa HTML-muodossa <URL: http://gitorious.org/gitorious/pages/CodingStyleGuide>.

[10] "Ruby Style Guide", saatavissa raakatekstimuodossa <URL: http://github.com/chneukirchen/styleguide/raw/master/RUBY-STYLE>.

[11] Verso-projekti, “Verso-projektin tehtävähallinta”, saatavissa HTML-muodossa <URL: https://trac.cc.jyu.fi/projects/verso/>.

[12] Tero Hänninen, “Vertailu alustoista”, saatavissa raakatekstimuodossa <URL: http://sovellusprojektit.it.jyu.fi/verso /src/docs/project/vertailu_alustoista.txt>.

[13] Heikki Salo, “Projektisuunnitelma”, saatavissa PDF-muodossa <URL: http://sovellusprojektit.it.jyu.fi/verso /src/docs/project/projektisuunnitelma /Verso_projektisuunnitelma_0.6.0.pdf>.

[14] Juho Nieminen, “Sovellusraportti”, saatavissa PDF-muodossa <URL: http://sovellusprojektit.it.jyu.fi/verso /src/docs/application/application_report /application_report_0.0.5.pdf>.

[15] Tero Hänninen, “System test plan”, saatavissa raakatekstimuodossa <URL: http://sovellusprojektit.it.jyu.fi/verso /src/docs/application/system_test_plan /system_test_plan_21st_May.txt>.

[16] Gitorious, “Gitorious”, saatavissa HTML-muodossa <URL: http://groups.google.com/group/gitorious/>, viitattu 24.5.2010.

[17] Gitorious, “Usability suggestions to dashboard vs. profile, activity feed and projects”, saatavissa HTML-muodossa <URL: http://groups.google.com/group/gitorious /browse_thread/thread/a6a888ff10b770ef>, viitattu 24.5.2010.