Verso-sovellusprojekti

Projektiraportti

Tero Hänninen
Juho Nieminen
Marko Peltola
Heikki Salo

Versio 0.5.0
Julkinen
27.6.2010

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: 56

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: aikataulu, Git, Gitorious, ketterä ohjelmistokehitys käytänteet, projektin läpivienti, projektiorganisaatio, projektiraportti, resurssit, riskit, sovellusprojekti, tavoitteiden toteutuminen, tehtävät, Trac, työnjako, versiohallinta

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.	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.	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. Tehty pieniä Santasen ehdottamia korjauksia. Lisätty jäsenten kokemuksia kuvaava luku.	HS
0.3.1	28.5.2010	Tehty pieniä korjauksia.	HS
0.3.2	31.5.2010	Tehty pieniä Santasen ehdottamia sisällöllisiä korjauksia kaikkiin lukuihin ja muokattu dokumentin rakennetta niiden perusteella.	HS
0.3.3	31.5.2010	Päivitetty avainsanoja, termejä sekä poistettu versiohistoriasta ilmaisu “Lähetetty tarkistettavaksi”. Lisätty Tero Hännisen ja Juho Niemisen kokemukset projektista.	HS
0.4.0	6.6.2010	Kirjoitettu riskien toteutumista kuvaava luku. Täydennetty tavoitteita kuvaavaa lukua.	HS

Versio	Päivämäärä	Muutokset	Tekijät
0.4.1	14.6.2010	Täydennetty vaatimuksien toteutumista kuvaavaa lukua. Korjattu pieniä virheitä ja tehty pieniä parannuksia Santasen palautteen perusteella.	HS
0.4.2	21.6.2010	Lisätty uudet Gantt-kuvat projektin vaiheista j päivitetty termejä, aikataulua sekä prosessimal kuvaavia lukuja. Laajennettu Teron kokemuksia projektista.	HS
0.5.0	22.6.2010	Pieniä korjauksia.	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/.

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 [25]. Sovelluksen rakenne ja toiminta kuvataan sovellusraportissa [18]. Projektisuunnitelma [17] 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 5 esitellään projektiorganisaatio ja muita projektin käytössä olevia resursseja. Luvussa 6 kuvataan Verso-projektin yhteistyötä Gitorious-sovellukseen liittyvän yhteisön kanssa. Luku 7 käsittelee projektin yleisiä käytänteitä. Luvussa 8 esitellään erityisesti sovellukseen liittyvä laadunvarmistus. Luku 9 esittelee projektissa käytetyt versiohallintakäytänteet. Luvussa 10 esitellään projektissa käytetyn Trac-järjestelmän käytänteet. Luvussa 11 tarkastellaan tehtävien ja työmäärän jakautumista projektin jäsenten kesken. Luvussa 12 kuvataan Verso-projektissa käytetty prosessimalli. Luku 13 kuvaa riskien toteutumista. Luvussa 14 ryhmän jäsenet kuvaavat kokemuksiaan projektista.

2 Termejä

Luvussa kuvataan dokumentissa esiintyviä termejä aihealueittain.

2.1 Projektin läpivientiin liittyviä termejä

Luvussa kuvataan projektin läpivientiin liittyviä termejä. Projektissa käytettiin Trac-projektinhallintaohjelmista, jonka käytöstä lisää luvussa 10.

Ketterä ohjelmistokehitys tarkoittaa ohjelmiston kehitystä menetelmillä, jotka pyrkivät ensisijaisesti tuottamaan halutunlaisen ohjelmiston reagoimalla mahdollisiin muutoksiin projektin kuluessa.

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.

2.2 Git-versiohallintaohjelmiston termejä

Luvussa kuvataan projektissa sekä kehittämiseen että sovelluksen yhteydessä käytettyyn Git-versiohallintaohjelmistoon liittyviä termejä. Versiohallinnan käytöstä lisää luvussa 10.

Branch on versiohistorian haara, committien tietty jatkumo.

Cherry pick on tietyn commitin poimimista haarasta toiseen.

Commit on muutos, joka sisältää tiedot muuttuneista tiedostoista, kuvauksen, lähettäjän ja hashin, joka yksilöi commitin.

Git	on hajautetun versiohallinnan ohjelmisto.
Hash	on SHA1-tiiviste, jota Git-ohjelmisto käyttää esimerkiksi yksilöimään committeja.
Merge request	on pyyntö liittää pyytäjän tekemät ja toimittamat muutokset jonkun toisen ylläpitämään tietovarastoon.
Repository	on versioitu tietovarasto; dokumentissa yleensä Git-ohjelmistolla ylläpidetty.

Hajautettu versiohallinta

2.3 Sovellukseen liittyviä termejä

Gitorious	on WWW-sovellus Git-tietovarastojen selaamiseen ja hallintaan.
Hajautettu versiohallinta	tarkoittaa ilman pakollista keskustietovarastoa tehtävää versiohallintaa.
Julkistaminen	tarkoittaa lähdekoodin asettamista julkisesti saataville.
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.

Wiki	on verkkosivukokonaisuus, jonka tavoitteena on helppo muokattavuus ja sivujen välinen vuorovaikutteisuus.

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 [7] 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

Luvussa kuvataan projektin tavoitteiden toteutumista ja tuloksia. Projektissa tuotetulle sovellukselle asetetuista tavoitteista ryhmä ehti toteuttaa kaikki pakollisiksi määritellyt vaatimukset sekä lähes kaikki tärkeiksi nähdyt vaatimukset. Myös projektille asetetut oppimistavoitteet toteutuivat.

4.1 Sovellukselle asetetut tavoitteet

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.

Prioriteetti	**Yhteensä**	**Toteutetut**	**Toteuttamatta**
Mandatory	11	11	0
Important	6	4	2
Useful	13	5	8
Left out	2	1	1

Taulukko 4.1: Toteutettujen ja toteuttamattomien tikettien lukumäärät prioriteeteittain.

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 Gitorious-sovelluksessa helpotettiin muokkaamalla olemassaolevia toimintoja. Lisäksi sovellus kytkettiin 19.4.2010 käyttämään kirjautumisessa Jyväskylän yliopiston Korppi-opintotietojärjestelmän käyttäjätunnuksia, mikä poisti kokonaan käytön aloituksessa vaaditun erillisen rekisteröitymisen.

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 [18].

Käyttöliittymän osalta Gitorioukseen ei tehty merkittäviä rakenteellisia muutoksia, mutta projektin aikana käyttöliittymiin tehtiin useita projektin aikana ilmenneisiin parannusehdotuksiin perustuvia käytettävyyskorjauksia. Uusien ominaisuuksien myötä Gitorioukseen lisättiin myös muutama täysin uusi käyttöliittymä.

Gitorious tarjosi sovellukselle asetetuista vaatimuksesta seuraavat:

- tietovaraston luonti ja hallinta WWW-käyttöliittymän kautta
- käyttöoikeuksien hallitseminen WWW-käyttöliittymän kautta
- tietovarastojen selaaminen ja niistä hakeminen
- sivuston yleisen aktiviteetin esittäminen
- tietovarastokohtaisen kuvauksen asettaminen.

Projektiryhmä toteutti sovellukseen uusina toimintoina tai muokkasi huomattavasti Gitoriuksen osalta seuraavia:

- projektien ja tietovaraston näkyvyyksien asteittainen rajoittaminen
- Korppi-opintotietojärjestelmän käyttäjätunnuksillä kirjautuminen
- ohjattu uuden tietovaraston luonti
- selausnäkymä tietovarastoille
- tietovaraston metatietojen tallentaminen tietovarastoon

- tietovaraston päivittäminen tiedoston tai paketin antamalla
- tietovaraston luonti ja hallinta WWW-käyttöliittymän kautta
- yleisiä käytettävyysparannuksia
- sovelluksen ajossa olevan version tunnisteen näyttäminen käyttäjille
- tietovaraston asettaminen peilaamaan ulkoista SVN- tai tiedostopaketti- tietovarastoa
- lisenssien asettaminen tietovastokohtaisesti projektikohtaisuuden sijaan
- projektin wikin käyttämisen mahdollistaminen projektin muiden tietovarastojen tapaan.

Tilaajan kanssa sovitiin jatkokehitykseen tavoitteina olevista ominaisuuksista:

- autentikoituminen Kerberosta käyttäen tietovarastojen päivittämisen yhteydessä
- selaimen kautta tapahtuvan päivittämisen mahdollistaminen ilman käyttäjälle lisättyä SSH-avainta
- tietovaraston yksittäisen version julkistamisen versiohallinnan kautta tallennetuista versiosta
- *merge requestien* hyväksymisen mahdollistaminen WWW-käyttöliittymällä
- tiedostojen muokkauksen mahdollistaminen WWW-käyttöliittymällä
- tietovaraston päivittäminen sähköpostilla
- lähdekooditiedostojen kommentointimahdollisuus
- lähdekoodin sertifikointi
- tietovarastokohtaiset tunnisteetkategoriat.

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 [25]. Trac-tiketit toimivat mahdollisena pohjana jatkokehitykselle. Projektiryhmästä Marko ja Tero jäivät Verso-projetin loputtua jatkokehittämään sovellusta.

4.3 Oppimistavoitteet

Sovellusprojekti on tietotekniikan syventävien opintojen kurssi, jossa neljästä opiskelijasta muodostettu ryhmä toteuttaa tilaajalle ohjelmiston tietotekniikan 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.

Oheiskurssien opetustapahtumissa Verso-ryhmä sai koulutusta projekti- ja ryhmätyöskentelystä, projektin johtamisesta ja hallinnasta, tekijänoikeuksista, sopimuksista ja käytettävyydestä.

Projektiviestintä IT-alalla -oheiskurssiin kuului sekä kirjoitus- että puheviestintää. Kirjoitusviestinnässä projektiryhmä paneutui virallisten dokumenttien kieli- ja ulkoasuvaatimuksiin. Puheviestinnässä projektiryhmä harjoitteli kommunikointitaitoja ryhmän keskinäistä viestintää sekä esitystilanteita varten. Esitystilanteita varten harjoiteltiin pitämällä kaksi väliesittelyä, joissa projektiryhmä harjoitteli projektin loppuesittelyä varten.

Oheiskurssin opetussisältöön kuuluu myös katsaus sovelluksien käytettävyyteen. Meeri Mäntylä piti projektiryhmälle käytettävyyspäivän, jossa käytiin läpi yleisiä käytettävyyteen liittyviä seikkoja sekä harjoiteltiin käytettävyystestauksen pitämistä. Lopuksi Mäntylä koekäytti sovellusta ja kertoi huomioitaan, jotka ovat kirjattu muistioon [12].

4.4 Projektin kuluessa opituista tekniikoista

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.

4.5 Projektiin liittyvät dokumentit

Sovellustietovarasto sisältää YouSource-prototyypin lähdekoodin sekä Git-versiohallinnan.

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.
Järjestelmätestaussuunnitelma	määrittelee sovelluksen toimivaksi toteamiseen vaaditut kohdat.
Järjestelmätestausraportit	kuvaavat kehitetyn sovelluksen järjestelmätestauksen tulokset.
Esikäyttäjien hyödyntämissuunnitelma	kuvaa, miten esikäyttäjiä käytetään projektissa sekä kuvaa käytettävyystestauksien läpivientiä.
Käytettävyysmuistiot	sisältävät kahden käytettävyystestauksen muistiot.
Projektiraportti	kuvaa projektin läpiviennin ja asetettujen tavoitteiden saavuttamista.
Projektisuunnitelma	kuvaa projektin tavoitteita, tehtäviä, aikataulua, yleisiä käytäntöjä ja riskien hallintaa.
Sovellusraportti	kuvaa toteutetun sovelluksen osat ja toiminnot sekä jatkokehitysideat.

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.

Edellä mainittujen dokumenttien lisäksi ryhmä laati seuraavat tulokset:

Itsearvioinnit sisältävät ryhmän jäsenten arvioinnit omasta panoksesta, onnistumisesta ja oppimisesta.

KorppiLDAPin käyttökuvaus kuvaa, miten projektissa tuotettu sovellus käyttää KorppiLDAPia.

Lisenssisopimus sisältää jäsenten suostumuksen projektissa tuotetun ohjelmakoodin sijoittamisesta AGPLv3-lisenssin alaisuuteen.

Lähdekoodi sisältää tuotetun lähdekoodin kommentteineen.

Palaverien dokumentit sisältävät palaverien esityslistat ja pöytäkirjat.

Palavereiden esityskalvot sisältävät palavereissa käsitellyt asiat tilakatsauksineen 4. palaverista alkaen.

Sähköpostiarkistot sisältävät kaikki projektin sähköpostilistalla käydyt keskustelut ja esimerkiksi päivittäiset tilakatsaukset.

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. Opiskelijoista Heikillä oli kokemusta WWW-sovelluksien kehittämisestä sekä ohjelmistoprojektissa toimimisesta Korppi-kehityksessä toimimisesta. Kellään ryhmäläisistä ei ennen projektia ollut kokemusta projektin läpiviennistä eikä suuresta osasta kehitysympäristöä. Koko ryhmällä oli vahvuutena hyvät sosiaaliset taidot sekä erinomaiset valmiudet omaksua uusia työkaluja ja tekniikoita.

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, pääasiassa Santeri Lapinmäki. Projektiryhmän käytössä olleen virtuaalipalvelimen ylläpidosta vastasi Jyväskylän opiopiston tietohallintokeskuksen sovelluspalvelut, pääasiassa Harri Tuomi.

Lisäksi projektin aikana suoritetun puhe- ja kirjoitusviestinnän kurssin aikana projektiryhmää auttoi kirjoitusviestinnän opettaja Leena Peltomaa sekä puheviestinnän opettaja Minna Koljonen.

Graafinen suunnittelija Auri Kaihlavirta valmisti sovellukselle logokuvan ja määritti värit, joiden avulla sovellukselle saatiin persoonallinen ilme.

Projektissa tuotettua sovellusta tarjottiin myös esikäyttöön, ja projekti teki näin ollen kevään aikana yhteistyötä parinkymmenen esikäyttäjän kanssa.

5.2 Projektin tilat ja laitteet

Projektiryhmän käytössä ollut huone AgC222.2 sijaitsi Agoran C-siivessä toisessa kerroksessa sovellusprojektien tiloissa. Huone tarjosi kevään ajan työskentelytilan neljälle projektilaisille sisältäen työasemat, työpöydät, tuolit, naulakon, kirjahyllyn sekä kaksi tussitaulua.

Projektiryhmän käytössä oli lisäksi sovellusprojektien kannettava tietokone, videoprojektori ja sanelin sekä projektitilojen monitoimilaite. Projejektiryhmän käytössä oli myös kokoustila AgC222.6, jossa myös pidettiin kaikki projektipalaverit.

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.

Huoneessa oli ryhmän käytössä neljä tietokonetta, joista kolmessa oli käyttöjärjestelmänä Linux Fedora Core 12 ja yhdessä Windows XP SP3 ryhmän jäsenten työtuntien kirjaamista varten.

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.it.jyu.fi*, jonka käyttöjärjestelmänä oli Red Hat Enterprise Linux 5.0.

Projektiryhmä ei käyttänyt ohjelmointiin raskaita ohjelmointityökaluja- tai ympäristöjä. Ohjelmointi sekä dokumentointi suoritettiin pääasiassa käyttäen Notepad++-ohjelmaa (Windows XP -työasema) sekä Gedit- ja Vim-ohjelmia (Fedora Core -työasemat).

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 versiohallintaa 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 [10] 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. Lähetetyt korjaukset sisälsivät korjaukset käyttäjäprofiiliin liittyvän avatar-kuvan poistoon, automaattisesti täydentyvien tekstikenttien korjukset sekä salasanojen suodattamisen sovelluksen poikkeusten yhteydessä lähetyistä virheviesteistä.

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 [24].

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.fi`.

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

Projektin aikana projektiryhmä lähetti päivittäin tilakatsauksen kuluneesta päivästä. Katsaus sisälsi henkilöittäin koosteen tehdyistä asioista, vastaantulleista ongelmista sekä seuraavaksi toteutettavista asioista. Päivittäisen katsauksen lähetti projektipäällikkö tai hänen estyessään projektipäällikön nimeämä ryhmäläinen.

Projektin ajan Trac-järjestelmän muutoksista ja lisätyistä tiketeistä ohjautui projektin sähköpostilistalle tiedote. Tracin käyttöä projektissa on kuvattu tarkemmin luvussa **??**.

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

Toteutusvaiheiden ajan palavereiden tärkein sisältö oli edellisen toteutusvaiheen tuloksien hyväksyminen ja seuraavassa vaiheessa toteutettavien toiminnallisuuksien valitseminen ja kiinnittäminen.

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 [11] Ruby on Rails -ohjelmistokehyksen ohjelmointityyliä sekä Gitorious- ohjelmiston ohjelmointikäytänteitä [11]. 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
```

Kehityksessä käytetyt käytänteet ovat tarkemmin kuvattu sovellusraportissa **??**.

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. Dokumenttien versionumerointi toteutui suunnitellusti.

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
```

7.7 Tulosten koostaminen

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

8 Sovellukseen liittyvä 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 isää 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" [22].

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 Järjestelmätestaukset

Tero Hänninen laati YouSource-sovelluksen järjestelmätestaussuunnitelman [19], 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.

Ensimmäisessä sovelluksen RC1-version järjestelmätestauksessa [20] löydettiin kaksi projektissa tuotettuun näkyvyyksien säätelyyn liittyvää virhettä sekä SVN-peilaukseen liittyvä virhe. Näkyvyysvirheet olivat korjattuna sovelluksen RC2-version järjestelmätestauksessa [21], jossa SVN-peilaukseen liittyvä ongelma jätettiin jatkokehityksessä ratkaistavaksi.

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 [12].

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. Ensimmäisestä [13] ja toisesta [14] käytettävyystestauksesta laaditut muistiot löytyvät 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 koodinkatselmoinneissa 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. Ryhmä valitsi katselmoinneissa läpikäytävät ohjelmaosiot niiden tärkeyden ja valmiusasteen perusteella.

Sekä ensimmäisestä [2] että toisesta [4] katselmoinnista on muistiot, jotka löytyvät projektikansiosta liitteenä. Kaijanaho lähetti lisäksi 2. katselmoinnin jälkeen huomionsa lähdekoodista [23] projektin sähköpostilistalle.

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 committeja**. 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 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 haarojenkäyttöstrategia **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 muun muassa versiohistoriaa jälkeenpäin 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 [17] 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 teki 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.

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 [6] päätettiin, että erillisen dokumentin sijaan vaatimukset kirjataan Trac-järjestelmään. Tracissa vaatimuksista tehtyjä tikettejä on ylläpitänyt koko projektiorganisaatio.

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.

11.3 Vastuualueet ohjelmoinnin osalta

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**

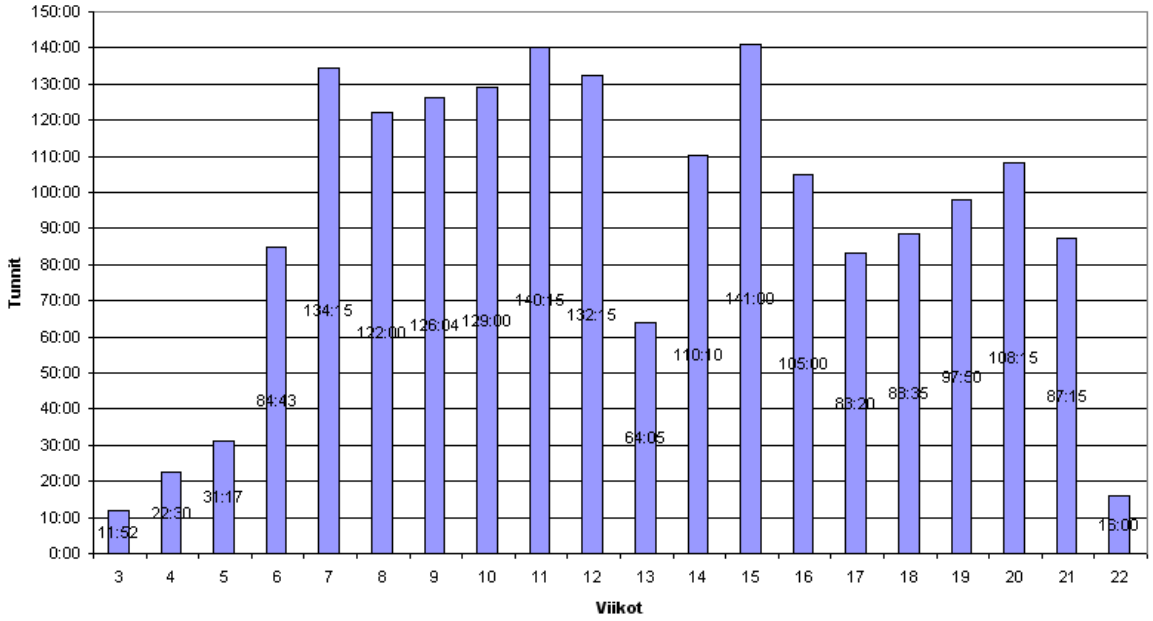
Kuva 11.1: Jäsenten työmääräarvio

11.4 Työtunnit ja tehtäväjako

Kesken.

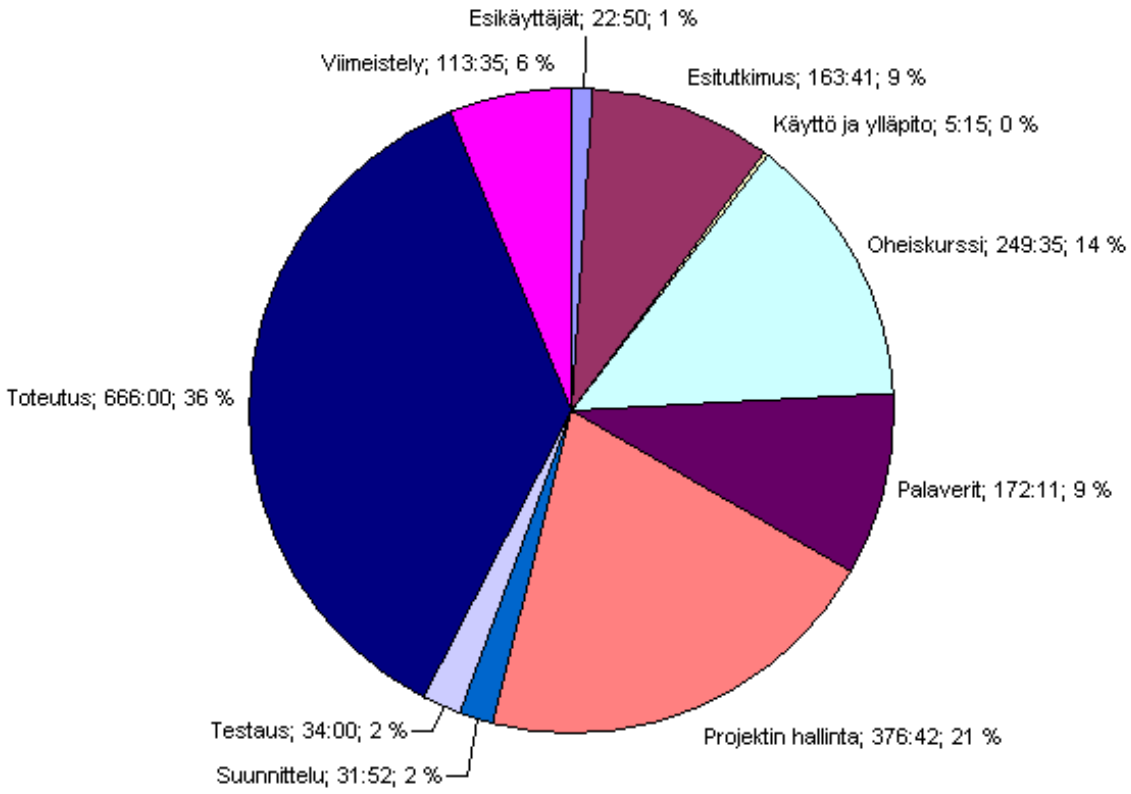
11.5 Ryhmän ajankäyttö tehtäväkokonaisuuksittain

Ryhmä käytti omisti projektille paljon tunteja, ja kukin neljästä projektilaisesti teki kehitysvaiheiden aikana parhaimmillaan yli 30 tunnin keskimääräistä viikkoa.



Kuva 11.2: Ryhmän ajankäyttö viikoittain

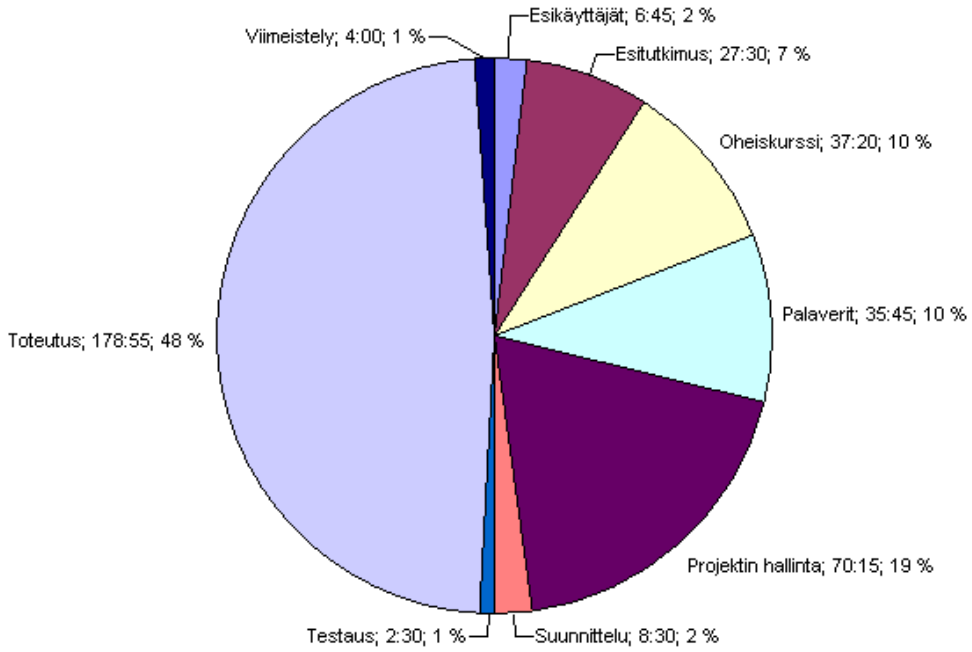
Viikottaisen ajankäytön kuvassa 11.2 näkyy selvästi ensimmäisen viikolla 6 pidetin palaverin jälkeen alkanut perehtyminen ja alustavaihtoehtojen kartoittaminen. Viikoilla 13 ja 14 näkyy selvästi Juhon sairastumisen ja pääsiäispäivien aiheuttama vaje tunneissa. Myös ryhmän kesätöiden alkaminen näkyy selvästi romahduksena tunneissa viikolla 22.



Kuva 11.3: Ryhmän ajankäyttö vaiheittain

11.6 Juho Niemisen ajankäyttö tehtäväkokonaisuuksittain

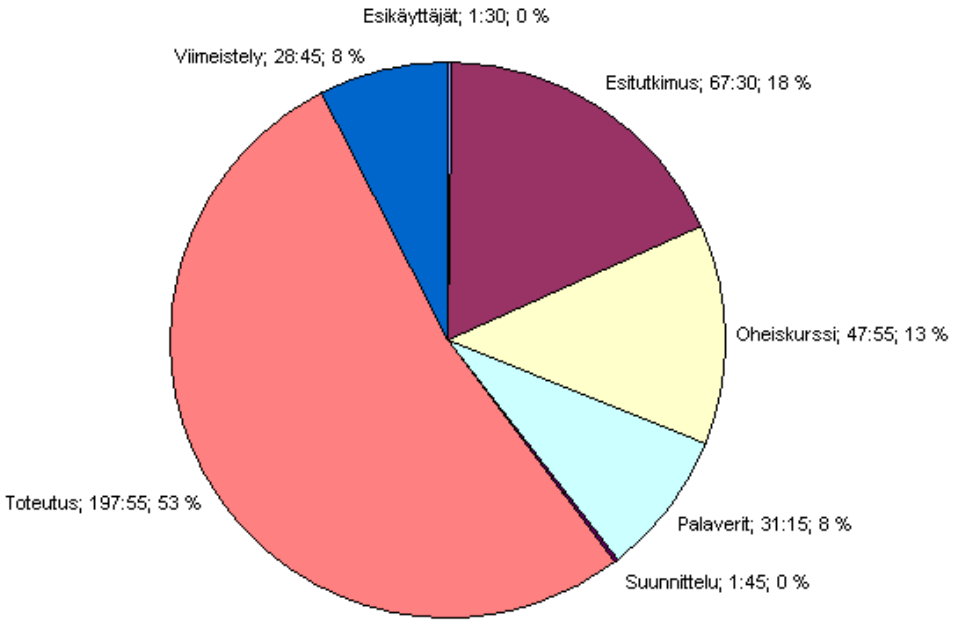
Juho oli Verso-projektin varaprojektipäällikkö, mikä näkyy hänen Teroa ja Markoa suurempana projektinhallintaan käytettynä aikana. Juhon vastuulla oli projektin alussa laatia vaatimusmäärittelydokumentti, jonka sisältö myöhemmin siirrettiin Trac-järjestelmään. FIXME: uudet tekstit kun tulee lopulliset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



Kuva 11.4: Juhon ajankäyttö tehtäväkokonaisuuksittain

11.7 Tero Hännisen ajankäyttö tehtäväkokonaisuuksittain

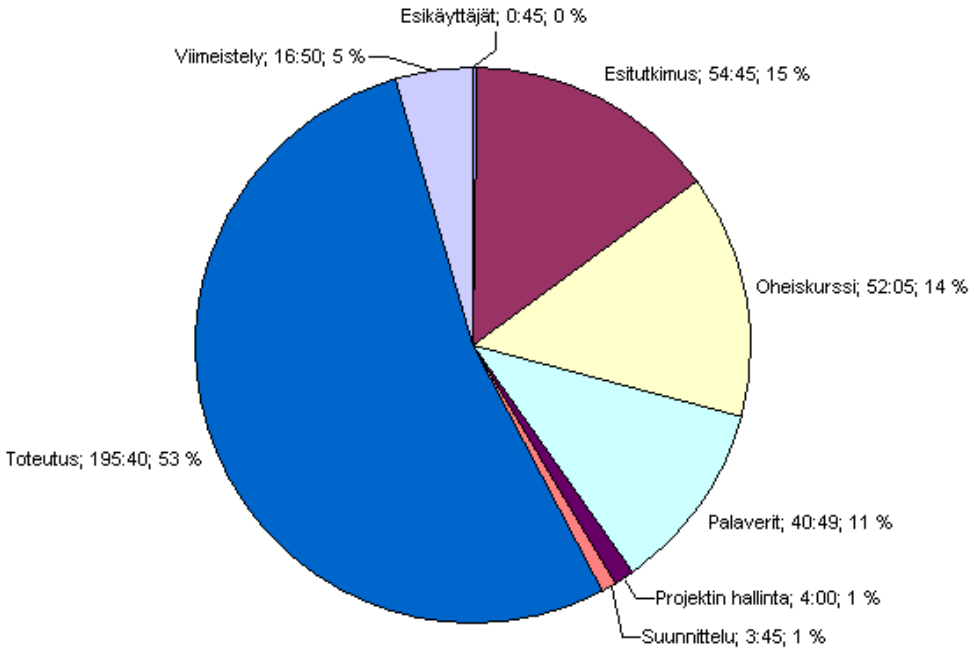
Teron ajasta kolme neljännestä kului esitutkimukseen, sovellusta kehittämiseen ja sen viimeistelyyn. FIXME: uudet tekstit kun tulee lopulliset kuvat.



Kuva 11.5: Teron ajankäyttö tehtäväkokonaisuuksittain

11.8 Marko Peltolan ajankäyttö tehtäväkokonaisuuksittain

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

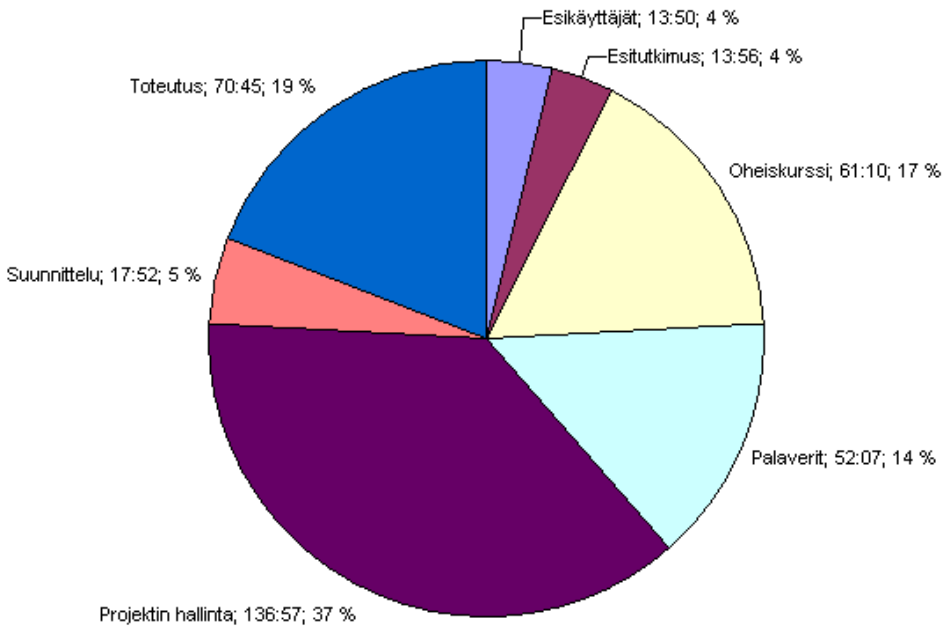


Kuva 11.6: Markon ajankäyttö tehtäväkokonaisuuksittain

11.9 Heikki Salon ajankäyttö tehtäväkokonaisuuksittain

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 11.7: Heikin ajankäyttö tehtäväkokonaisuuksittain

12 Prosessimalli ja projektin läpivienti

Luvussa tarkastellaan projektissa käytettyä prosessimallia sekä projektin läpivientiä aikataulutuksen ja käytänteiden osalta.

Projekti oli määrä toteuttaa kevätlukukaudella 2010 niin, että sovellus on uusien ominaisuuksien toteuttamisen osalta valmis 23.4.2010 mennessä ja koko projektin valmis toukokuun puoleen väliin mennessä. Uusien ominaisuuksien kohdalla aikataulu piti, mutta projektikurssille asetettujen vaatimusten täyttäminen viivästyi ja viivästyy yhä (FIXME ei lopulliseen). Aikataulua käydään läpi lisää luvussa 12.2

12.1 Prosessimalli

Projekti oli määrä viedä läpi käyttäen ketterää ohjelmistokehitystä. Kesken jäänyt projektisuunnitelma [17] ehti määritellä projektissa käytettävän ketteriä menetelmiä. Suunnitelmassa määriteltiin käytettävän kahden viikon mittaisia toteutusvaiheita, joiden kehitystehtävät sovitaan kunkin vaiheen alussa pidettävässä projektipalaverissa projektiryhmän tekemän ehdotuksen pohjalta.

Projektiryhmän muodostama **ehdotus toteutusvaiheen tehtävistä** muodostui palaverikeskustelujen, kiinnittämättömien tehtävätikettien ja muun yhteydenpidon perusteella. Projektiryhmä pyrki arvioimaan vaiheeseen kiinnitettävät tehtävät käytettävissä oleviin resursseihin nähden järkevästi. Tehtävien kokojen arvioinnissa onnistuttiin pääosin hyvin, vaikka ensimmäisistä toteutusvaiheista jäi noin päivän tai kahden edestä valmistumattomia ominaisuuksia, josta lisää luvussa 12.3. Valmistumattomat ominaisuudet hyväksyttiin palavereissa tehtäväksi seuraavan vaiheen alussa.

Projekti vietiin läpi käyttäen **ketterää ohjelmistokehitystä**. Kun ominaisuus eli vaatimusmäärittelyssä Trac-tikettityyppi Feature valmistui, se hyväksytettiin tilaajalla. Hyväksyttäminen tehtiin tavallisesti siten, että projektiryhmä järjesti ominaisuuden nähtäville ja ilmoitti siitä projektin sähköpostilistalla. Pari kertaa tilaaja ei hyväksynyt kehitettyä ominaisuutta sellaisenaan, ja projektiryhmä korjasi ominaisuudesta löytyneet puutteet seuraavan toteutusvaiheen aikana. Kehitystyössä käytettiin neljää kahden viikon mittaista toteutusvaihetta, joiden lisäksi viimeistelylle oli varattu viides kolmen viikon mittainen viimeistelyvaihe. Viidenteen vaiheeseen ei kehitetty uusia ominaisuuksia, ellei niiden kehittämistä oltu jo aloitettu aiemmissa vaiheissa.

Projektille erityinen piirre oli tehtävien **määristä sopiminen vaiheiden aikataulun perusteella**. Käytännössä Verso-projektin sovelluksen kehittämiseen oli alusta asti varattu neljä kehitysvaihetta, johon sisältyvät tehtävät valikoituivat vähitellen. Tämän esittäminen ei olisi ollut lainkaan mielekästä, jos projektiryhmän motivaatio ei jo alusta asti olisi näyttänyt korkealta. Tasainen ja loppuun asti kestänyt kehitysrytmi ei kuitenkaan ollut itsestäänselvyys, ja projektin lyhyen keston vuoksi selkeämpi valinta olisi ollut tarkastella sovelluksen ominaisuuksia alusta asti inkrementeittäin. Verso-projektiin sovitut vaiheet muistuttivat kuitenkin enemmän aikataulutettuja kehityspyrähdyksiä (engl. *sprint*).

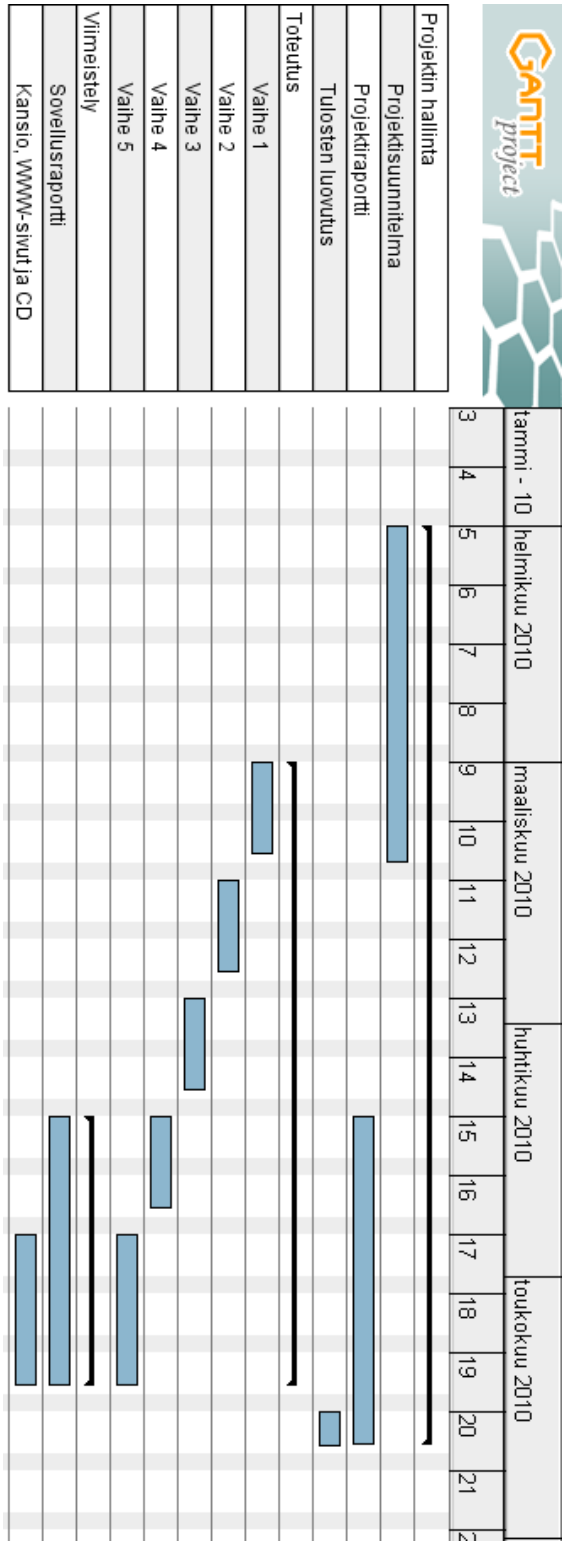
Prosessimallin toteuma vastasi vastasi hyvin suunnitelmaa, joskin suunnitelman prosessimallikuvaus jäi suppeaksi. Laajemmin suunnitelmassa kuvatut projektin Trac-tikettikäytänteet toteutuivat lähes sellaisenaan.

Projektissa tuotetulle **sovellukselle ei valmistettu erillistä vaatimusmäärittelyä**, vaan vaatimuksia ylläpidettiin tiketteinä Trac-järjestelmässä. Vaatimusmäärittelyn osalta projektissa nojattiin liiankin paljon tiketteihin. Yksittäisi tikettejä ylläpitäessä kokonaiskuva jää määrittelemättä, ja projektin kesto oli vain hädin tuskin tarpeeksi pitkä hyödyn saamiseksi ketteristä päätöksistä eli projektin aikana muodostuneiden tarpeiden pohjalta tehdyistä valinnoista toteutusvaiheiden tehtäviä kiinnittäessä. Trac oli työkaluna erityisen huono valinta kokonaisuuksien hahmottamiseen tikettien pohjalta, mikä hankaloitti valmiin sovelluksen hahmottamista ominaisuuksien osalta, kun kehitys ei ollut vielä valmis.

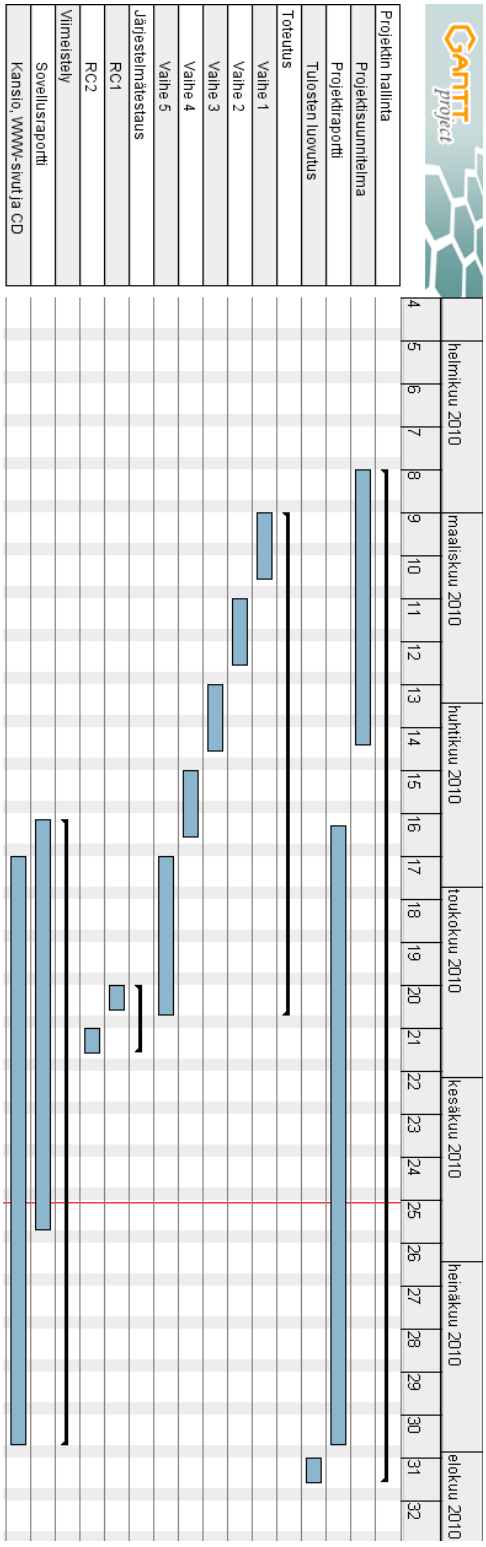
Koska projektin alussa **tilaajalla ei ollut tarkkoja vaatimuksia** sovelluksen ominaisuuksien tai alustavalintojen osalta, olisi jonkinlainen kokoavampi vaatimusmäärittely auttanut suunnittelua. Tässä olisi päässyt pitkälle jonkinlaista tuotteen työlistaa (engl. *product backlog*) tukevalla tikettijärjestelmällä, joka mahdollistaisi tehtävien suunnittelun korkeammalla tasolla. Tracin eduksi voi kuitenkin todeta sen toimineen yksittäisiä tehtäviä ajatellen hyvänä keskusteluympäristönä projektissa.

12.2 Aikataulu

Projektin oli määrä olla kokonaisuudessaan valmis toukokuun puoleen väliin mennessä, mutta se on myöhästynyt siitä pahasti, yli kahdella (FIXME päivitä) kuukaudella. Sovelluksen osalta kehitystyö valmistui noin kahdella viikolla venynyttä viimeistelyvaihetta aikataulussa pääosin valitun prosessimallin perusteella.



Kuva 12.1: Projektin läpiviennin suunniteltu aikataulu



Kuva 12.2: Projektin läpiviennin toteutunut aikataulu

Suunnitelman 12.1 ja toteuman 12.2 välillä **merkittävin ero liitttyy viimeistelyyn ja raportointiin**, erityisesti raportointiin. Sovellus- ja projektiraporteille olisi pitänyt varata huomattavasti suunniteltua enemmän aikaa. Jälkikäteen voi nähdä, että pysyäkseen aikataulussa viimeistelynkin osalta Verso-projektista olisi pitänyt karsia kehitystehtäviä vähintään yhden tai kahden toteutusvaiheen verran.

Aikataulukuvissa ei näy kehitystehtävien myöhästymistä, sillä myöhästyneet ominaisuudet hyväksyttiin osaksi edellistä seurannutta vaihetta. Erot eivät kuitenkaan ole suuria, kehitystehtäviä jäi tekemättä noin päivän tai kahden edestä vaiheiden aikana.

12.3 Tehtävien toteutusajan 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.

13 Riskit ja niiden seuranta

Luvussa kuvataan projektin ennakoitujen ja ennakoimattomien riskien toteutumista sekä vaikutuksia tuloksiin tai projektiin.

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**	**Arvioitu haitta**	**Toteutunut haitta**
Muutostarve kehityskoneissa	suuri	keskinkertainen	keskinkertainen
Valitun alustan ongelmat	keskinkertainen	suuri	pieni
Jäsenten poissaolot	keskinkertainen	pieni	pieni
Testipalvelimen ongelmat	keskinkertainen	keskinkertainen	pieni
Jäsenten motivaation puute	pieni	keskinkertainen	(kesken)
ATK-tuen viive	keskinkertainen	(ei arviota)	keskinkertainen
Esikäyttäjien aktiivisuus	keskinkertainen	(ei arviota)	pieni

Taulukko 13.1: Arvioidut riskit, niiden todennäköisyydet, arvioidut ja toteutuneet haitat.

13.2 Muutostarve kehityskoneissa

Projektiryhmän käytössä oleviin kehityskoneisiin oli asennettu ryhmän tärkeimmät tarvitsemat kehitystyökalut ja sovelluskirjastot projektin alkuvaiheessa. Projektin edetessä työasemiin tarvittiin kuitenkin muutoksia esimerkiksi *git pusheja* tai LDAP-autentikointia tehdessä. Uusien työkaluvaatimuksien kanssa projektiryhmä

oli riippuvainen lähituen ylläpidosta, koska projektiryhmällä ei ollut kehityskoneisiin ylläpito-oikeuksia. Riskinä oli, että projektiryhmä joutuu odottamaan uusia työkaluja.

Riskiä yritettiin minimoida pyytämällä tiedossa olevia työkaluja parhaimmillaan noin viikko tai kaksi etukäteen. Riski toteutui pahiten sellaisissa tilanteissa, joissa tarvittava muutos kehityskoneisiin jäi ratkeamatta projektin aikana. Esimerkiksi juuri *git pushia* tai Python-kielen LDAP-laajennosta ei saatu projektin aikana toimimaan kehityskoneilla. Riskin vaikutus jäi näissäkin tilanteissa korkeintaan keskinkertaiseksi, sillä projektiryhmä kehitti tavat kokeilla ominaisuuksia virtuaalipalvelimen *versotest.it.jyu.fi* kautta.

Osa projektin aikana kehityskoneiden ylläpitoon liittyvistä vaikeuksista johtui siitä, että kehityskoneiden ja testipalvelimen ohjelmapakettivarastoissa oli eri versioita samoista ohjelmista, jotka toimivat hieman eri tavalla. Eri versiot vaikeuttivat myös sovelluksen asennusohjeen kirjoittamista.

13.3 Valitun alustan ongelmat

Sovelluksen alustaksi valikoitui Gitorious, koska sen nähtiin lupaavimmaksi tavoitteiden saavuttamisen kannalta projektiryhmän suorittamassa alustavertailussa [16]. Projektin kuluessa oli silti mahdollista, että Gitoriouksen käyttö alustana estyy jonkin ennustamattoman syyn perusteella. Tämmöinen syy olisi voinut olla Gitorious-sovelluksen jonkin fundamentaalin tietoturvaongelman ilmeneminen, joka estäisi sen käytön salaisten tietojen tai yliopiston käyttäjätunnuksien kanssa.

Riski ei liittynyt erityisesti Gitorious-sovellukseen tai sen pohjana toimivaan Ruby on Rails -sovelluskehykseen. Riski olisi toteutuessaan olennaisesti hankaloittanut sovelluskelle asetettujen tavoitteiden saavuttamista. Riskin vaikutus arvioitiin pieneksi, eikä se toteutunut lainkaan. Riskiä ei varsinaisesti voitu kontrolloida, mutta projektiryhmä seurasi koko projektin ajan kaikkia Gitoriouksen tiedotuskanavia ja olisi näin kuullut ongelmista nopeasti.

13.4 Jäsenten poissaolot

Riskinä oli, että jokin ryhmän jäsen on pois ilman tarvittavia järjestelyjä, jolloin muu projektiryhmä ei tietäisi poissaolevan toimien tilanteesta tai niiden hoitamisesta. Hankalan poissaolon olisi voinut muodostaa sairastuminen, mutta myös lomamatka. Riskin vaikutus olisi toteutuessaan ollut keskinkertainen.

Riskin vaikutus arvioitiin pieneksi, sillä kellään ryhmästä ei ollut projektin alussa tiedossa viikonloppua pidempiä lomamatkoja keväälle eikä esimerkiksi sairastumisista johtuvia muita poissaoloja ollut aihetta odottaa ennalta. Riskiä hallittiin projektin alusta loppuun kestäneellä hyvällä kommunikoinnilla jäsenten välillä. Lisäksi uusia toiminnallisuuksia toteuttaessaan ryhmäläiset toteuttivat ohjelmointia monilta osin pariohjelmointina, jolloin tietämys uusista tekniikoista levisi heti tehdessä.

Riski ei toteutunut lähes lainkaan, vaikka Juho oli sairaana pari päivää ja Heikki päivän.

13.5 Testipalvelimen ongelmat

Riskinä oli, että esikäytössä oleva sovellus toimisi 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 viivästyttää kehitystyötä.

Riskin vaikutukset arvioitiin keskinkertaisiksi sen projektia viivästyttävän luonteen vuoksi. Riskiä hallittiin projektin aikana järjestämällä testipalvelimen lokitiedostojen kirjoitus kuntoon, jotta mahdollisissa vikatilanteissa mahdollisimman paljon tietoa tapahtuneesta olisi käytettävissä myöhemmin. Luvusssa 8.2 kuvataan lisäksi automaattisten virhetietoa sisältävien poikkeustiedotteiden käyttöä.

Riski toteutui vain pienenä, koska ongelmatilanteiden ratkaisussa ei yleensä mennyt pitkään.

Projektisuunnitelmassa [17] kuvattiin riskin hallitsemiseksi harjoitteet erilaisten vikatilanteiden etukäteen harjoituttamiseksi. Suunnitellusta kuitenkin poikettiin, sillä vastaantulleet ja mahdollisesti yleistyvät vikatilanteet korjattiin yleensä koko pro-

jektiryhmällä ja tietämys korjaustoimenpiteistä levisi saman tien. Tarvetta erilliselle harjoittamiselle ei lopulta ollut.

13.6 Jäsenten motivaation puute

Riskinä oli, että projektiryhmän yleinen motivaatio ei ole projektin aikana tarvittavalla tasolla projektille asetettujen saavuttamiseksi. Projektin alussa projektiryhmä koostui toisilleen täysin vieraista opiskelijoista, joita kohtasi projektin alussa jyrkkä oppimiskäyrä ryhmäkemian vasta muodostuessa.

FIXME: Kirjoita loppuun, kun dokumentit ovat lähempänä valmista.

13.7 Esikäyttäjien vaikutus

Riskinä oli, että projektissa tuotetun sovelluksen julkaiseminen esikäyttöön aiheuttaa projektiryhmälle niin suuren ylläpito- tai neuvontataakan, että se viivästyttää projektia. Riskin suuruutta ei arvioitu etukäteen lainkaan, eikä se toteutunut, sillä esikäyttäjien aktiivisuus oli vähäinen, kuten kappaleessa 8.3 kuvataan.

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.

14.2 Tero Hännisen kokemuksia omin sanoin

Projektin alussa oli pientä jännitystä siitä, mitä tuleman pitäisi. Oppimiskäyrä oli siinä mielessä jyrkkä, että suurin osa työkaluista oli vieraita. Samasta syystä projekti oli erittäin opettavainen: itse tulin oppineeksi todella paljon ainakin seuraavista aiheista: Git, Ruby, Ruby on Rails, web-kehitys ja Linux-terminaali. Mainitut työkalut ovat mielestäni fiksuja ja tehokkaita ja olenkin todella tyytyväinen, että ne tuli opittua tehokkaasti projektin myötä. On paljon mukavampaa opetella asioita kun

on todellinen tarve siihen ja mahdollisuus soveltaa kaikkea oppimaansa heti – sen sijaan, että opettelisi vain oppiakseen.

Projekti onnistui mielestäni hienosti. Kaikki ryhmän jäsenet tekivät hommia hyvin, yhteistyö sujui ja henki oli hyvä. Toisessa lähdekoodin katselmoinnissa tekninen ohjaaja kehui koodia varmaotteiseksi opiskelijoiden kirjoittamaksi. Lopputulos ylitti tilaajan tavoitteen: tavoitteena oli saada jonkinnäköinen prototyyppi, mutta tuloksena oli toimiva ja käytettävä sovellus (tähän toki auttoi se, että sovelluksen pohjaksi löytyi paljon tarjoava Gitorious).

Projektin läpiviennin kannalta kriittistä oli avoin ja aktiivinen viestintä. Tässä ei mielestäni ollut ongelmia, mutta erityisesti projektipäällikkömme Heikki hoiti viestintää vahvasti sekä tilaajien, ohjaajien että tietohallintokeskuksen palveluiden suuntaan. Toki viestintä toimi hyvin myös toiseen suuntaan. Myös projektiryhmän sisällä tieto kulki todella hyvin kaikkien työskennellessä samassa huoneessa ja perustamamme IRC-kanavan ansiosta.

14.3 Juho Niemisen kokemuksia omin sanoin

Ennen sovellusprojektin alkua olin kuullut, että kurssin on työläs eikä muita kursseja ehdi oikeastaan suorittaa sen aikana. Olin myös kuullut, että jos ohjaajaksi osuu Jukka-Pekka Santanen, niin kurssin työläin osio tulee olemaan dokumenttien laatiminen. Nämä molemmat kuulopuheet osottautuivat oikeiksi oman sovellusprojektikurssini kohdalla. Projektihuoneessa tuli vietettyä aikaa jokaisena mahdollisena työpäivänä, joten muiden kurssien suorittaminen jäi iltatöiksi. Ja kuten arvata saattaa, ei sellaisesta hyvää seuraa. Dokumenttien laatiminen oli kurssilla todellakin iso urakka, mutta ei aivan niin iso kuin ennalta pelkäsin.

Sovellusprojektista aikana oli hyvä meininki. Sovellusta tehtiin innolla, ja ominaisuuksia kehitettiin ihan omasta aloitteesta. Pieniä korjauksia tehtiin sinne tänne vain tekemisen ilosta. Valitettavasti joitain isojakin puutteita sovelluksen viimeiseen versioon jäi, joten työn kohdentamista olisi voinut harkita tarkemmin (eikä vain suorittaa niitä kivoja hommia).

Projektin lopussa pidettiin loppuesittely, joka oli hyvä ja opettavainen kokemus puheviestinnän kannalta. Projektista jäi hyvä mieli, kun huomasi, että muitakin kiinnostaa projektissa tehty työ. Loppujen lopuksi koko sovellusprojekti oli hyvin positiivinen kokemus ja siitä oppi paljon. Git, Ruby on Rails, Trac, Linuxin komen-

tokehote, palaveripöytäkirjat, sovellusraportti ja käytettävyystestaukset ovat kaikki uusia asioita, jotka tulivat tutuksi projektin aikana. Kokemus ei varmasti olisi ollut niin hyvä, jos projektin aihe ei olisi ollut niin mielenkiintoinen, tai asiakas olisi ollut hankala tai projektiorganisaatio ei olisi tarjonnut niin hyviä tiloja ja palvelua projektiryhmälle.

14.4 Marko Peltolan kokemuksia omin sanoin

FIXME: kesken.

14.5 Heikki Salon kokemuksia omin sanoin

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.

FIXME: kirjoita uusiksi kun muuten valmis.

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, "1. koodinkatselmoinnin muistio"

[3] Juho Nieminen, "1. palaverin pöytäkirja"

[4] Marko Peltola, "2. koodinkatselmoinnin muistio"

[5] Juho Nieminen, "2. palaverin pöytäkirja"

[6] Marko Peltola, "4. palaverin pöytäkirja"

[7] Ville Tirronen, "Alustavaa lukemistoa projektiin liittyen"

[8] "About Gitorious"

[9] Wikipedia, "Gitorious"

[10] Gitorious, "Gitorious"

[11] Gitorious, "Gitorious Coding Style Guide"

[12] Tero Hänninen, "Käytettävyyspäivän muistio"

[13] Juho Nieminen, "1. käytettävyystestauksen muistio"

[14] Juho Nieminen, "1. käytettävyystestauksen muistio"

[15] "Ruby Style Guide"

[16] Tero Hänninen, "Vertailu alustoista"

[17] Heikki Salo, "Projektisuunnitelma"

[18] Juho Nieminen, "Sovellusraportti"

[19] Tero Hänninen, "System test plan"

[20] Tero Hänninen, "System test report for version RC1"

[21] Tero Hänninen, "System test report for version RC2"

[22] Heikki Salo, “Suunnitelma esikäyttäjien hyödyntämisestä”.

[23] Antti-Juhani Kaijanaho, “Teknisen ohjaajan huomioita koodista”.

[24] Gitorious, “Usability suggestions to dashboard vs. profile, activity feed and projects”

[25] Verso-projekti, “Verso-projektin tehtävähallinta”