

Yhteenvetodokumentti

PUSU-ryhmä

Helsinki 13.12.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (9 op)

Projektiryhmä

Jussi Hynninen

Jaakko Juvonen

Paavo Koskinen

Mikko Leino

Janne Salo

Vesa Tuomiaro

Asiakas

Johannes Korpela

Johtoryhmä

Kimmo Simola

Juhani Haavisto (ohjaaja)

Kotisivu

<http://www.cs.helsinki.fi/group/pusu/>

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	13.12.2007	Ensimmäinen versio

Sisältö

1 Johdanto	1
2 Sanasto	1
3 Dokumenttien tiivistelmät	2
3.1 Vaatimusmäärittelydokumentti	2
3.1.1 Asiakkaan vaatimukset	3
3.1.2 Järjestelmävaatimukset	3
3.1.3 Käsiteltävän datan kuvaus	5
3.1.4 Käyttötapaukset ja sidosryhmät	5
3.2 Suunnitteludokumentti	7
3.3 Testaussuunnitelma	7
3.4 Käyttö- ja ylläpito-ohje	8
4 Projektin päättöanalyysi	8
4.1 Aikataulu	8
4.2 Työvaiheet	9
4.2.1 Ongelmaan tutustuminen ja projektin aloitus	9
4.2.2 Vaatimusmäärittely	10
4.2.3 Ohjelmiston suunnittelu	10
4.2.4 Testauksen suunnittelu	11
4.2.5 Toteutus ja yksikkötestaus	11
4.2.6 Integrointi- ja järjestelmätestaus	11
4.2.7 Projektin päättötoimet	11
4.3 Työkalut ja menetelmät	12
4.4 Onnistumiset ja haasteet	12
4.5 Yhteenveto	13

1 Johdanto

RSS-syötteet toimivat nykypäivänä asiakaslähtöisesti siten, että asiakasohjelmat hakevat tietyin väliajoin palvelimelta uuden RSS-dokumentin, esimerkiksi uutisia. Tämä toiminta on erittäin tehokasta. Ensinnäkin asiakasohjelmien pitää osata veikata, koska tietoa kannattaa hakea ja hakiessaan RSS-dokumentin kaikki tarjolla olevat uutiset lähetetään kerralla, myös vanhat jo haetut. Tästä seuraa tyypillisesti se, että suurin osa palvelimelta haetuista artikkeleista on duplikaatteja, mikä aiheuttaa turhaa dataliikennettä. Erityisesti tämä ongelma korostuu silloin, kun RSS-syötteitä haetaan mobiililaitteilla hitaan ja kalliin datayhteyden ylitse.

PUSU-projektiryhmä on luonut RSS-syötteille uudenlaisen Push-palvelinohjelmiston, joka poistaa edellä mainitun duplikaattiongelman. Järjestelmään lisätään internetissä saatavilla olevia RSS-syötteitä ylläpitokäyttöliittymän kautta, minkä jälkeen niiden sisältämiä artikkeleita lähetetään automaattisesti eteenpäin asiakasohjelmille. Asiakasohjelma ilmoittaa järjestelmälle, mitä syötteitä hän haluaa seurata ja kuinka usein hänelle saa lähettää artikkeleita. Järjestelmä pitää kirjaa asiakasohjelmista siten, että se osaa lähettää ainoastaan uudet artikkelit. Täten asiakasohjelmalle ei lähetetä ollenkaan artikkelien duplikaatteja ja edellä mainittu turha tiedonsiirto järjestelmän ja asiakasohjelman välillä poistuu.

Järjestelmällä on ylläpitokäyttöliittymä, jonka kautta voidaan hallita muun muassa järjestelmän tukemia RSS-syötteitä. Järjestelmä on toteutettu Java-kielellä. Kontrolli- ja tiedonsiirtoprotokollana asiakasohjelman ja järjestelmän välillä käytetään SIP-protokollaa. Protokollaa ei toteutettu erikseen projektia varten, vaan käytettiin valmista avoimen lähdekoodin NIST-SIP -toteutusta (versio 1.2). RSS-syötteiden hakemiseen ja parsimiseen järjestelmä käyttää Informa-kirjastoa (versio 0.7.0).

Tämä dokumentti sisältää yhteenvedon projektin aikana tehdystä ohjelmistosta ja dokumenteista sekä analyysin projektin onnistumisesta.

2 Sanasto

Järjestelmä PUSU-projektin tuottama palvelinohjelmisto ja ylläpitokäyttöliittymä.

Artikkeli Synonyymi RSS-artikkelille.

Asiakas Järjestelmän tilaaja. Tässä projektissa Johannes Korpela.

Asiakasohjelma Ohjelma, joka tilaa järjestelmältä RSS-syötteen/syötteitä.

Asiakkuus Asiakkuuteen kuuluu kaikki asiakasohjelman tilaukset.

Client Synonyymi asiakasohjelmalle.

Tilaus Tilaus on asiakkaan ilmaisema tahto vastaanottaa jonkin syötteen artikkeleita. Asiakas muodostaa tilauksen lähettämällä SUBSCRIBE-pyyntöä.

RSS XML-pohjainen standardi usein uutisten, blogien yms. julkaisemiseen. Termi viittaa aina RSS:n versioon 2.0, ellei toisin mainita.

Atom RSS:n kaltainen julkaisuformaatti, joka tarjoaa RSS:ää laajemmat ominaisuudet.

RSS-dokumentti RSS-muotoinen dokumentti.

RSS-artikkeli RSS-dokumentin sisältämä yksittäinen artikkeli. Koostuu item-elementistä ja sen sisällöstä.

RSS-syöte Palvelimen tarjoama RSS-dokumenttien virta.

SIP Protokolla loogisen yhteyden muodostamiseen tietoverkossa. Järjestelmä käyttää tätä asiakasohjelmien kanssa kommunikointiin.

SUBSCRIBE-pyyntö SIP:n laajennos, jolla asiakasohjelma voi pyytää tietoa vastaanottajan tilamuutoksista. Järjestelmässä asiakasohjelmat käyttävät SUBSCRIBE-pyyntöä tilatessaan RSS-syötteitä.

NOTIFY-pyyntö SIP:n laajennos, jolla SUBSCRIBE-pyyntöön vastaanottaja voi ilmoittaa tilamuutoksista pyytäjälle. Järjestelmä käyttää tätä mm. RSS-dokumenttien lähettämiseen asiakasohjelmalle.

Informa LGPL-lisenssin alainen Java-kirjasto, joka toteuttaa mm. RSS- ja Atom-syötteiden noutamiseen liittyvän toiminnallisuuden¹.

JAIN SIP Yhteinen nimitys eräälle Javan SIP-rajapinnalle² ja sen toteutukselle.

Log4j Apache-projektin tekemä kirjasto, joka toteuttaa lokitiedostojen kirjoittamiseen liittyvän toiminnallisuuden³.

PostgreSQL BSD-lisenssin alainen tietokantajärjestelmä.

3 Dokumenttien tiivistelmät

Tässä luvussa esitellään lyhyesti projektin aikana tuotetut dokumentit.

3.1 Vaatimusmäärittelydokumentti

Vaatimusmäärittelydokumentti määrittelee järjestelmään toteutettavat ominaisuudet ja rajapinnat. Lisäksi se määrittelee XML-dokumenttityypin, jota asiakasohjelmat käyttävät kommunikointiin ohjelmiston kanssa.

¹<http://informa.sourceforge.net/index.html>

²<http://www.jcp.org/en/jsr/detail?id=32>

³<http://logging.apache.org/log4j/1.2/index.html>

3.1.1 Asiakkaan vaatimukset

- Palvelin toimittaa asiakasohjelmistoille personoituja RSS-syötteitä niin, että yksittäinen artikkeli toimitetaan asiakkaalle vain kerran
- Palvelin- ja asiakasohjelmistot kommunikoivat käyttäen SIP-protokollaa ja sen SUBSCRIBE/NOTIFY-toimintoa
- Palvelin pitää kirjata asiakkaista, heidän preferensseistään ja osaa lähettää uutisia asiakkaille heidän preferenssiensä mukaan
- Palvelimessa pitää olla yksinkertainen tapa, jolla uutistoimittaja voi helposti lisätä uutisia tietokantaan

3.1.2 Järjestelmävaatimukset

Oheisessa taulukossa on listattu vaatimusmäärittelydokumentin sisältämät järjestelmävaatimukset.

Tunnus	Toiminto	Kuvaus
JV1	RSS-syötteiden noutaminen	Järjestelmä noutaa asetuksissa määrättyt RSS-syötteet verkosta
JV2	Tilauksen vastaanottaminen	Asiakasohjelman lähettämään SUBSCRIBE-pyyntöön vastataan SIP-protokollan mukaisesti. Asiakasohjelman ilmoittamat käyttäjäasetukset tallennetaan. Heti tämän jälkeen palvelin lähettää asiakasohjelmalle jokaista tilattua syötettä kohden NOTIFY-pyyntö.
JV3	RSS-syötteiden lähettäminen	Järjestelmä lähettää RSS-syötteitä asiakasohjelmille.
JV4	Tarjottavien RSS-syötteiden listaaminen	Järjestelmä lähettää XML-dokumentin, jossa listataan järjestelmän tarjoamat RSS-syötteet.
JV5	Tilauksen lopettaminen	Asiakasohjelman asiakkuus poistetaan järjestelmästä.
JV6	RSS-syötteen lisääminen järjestelmään	Järjestelmän ylläpitäjä voi lisätä järjestelmään uuden noudettavan RSS-syötteen.
JV7	Artikkelien kirjoittaminen järjestelmän omaan uutissyötteeseen	Järjestelmän ylläpitäjällä on mahdollisuus kirjoittaa artikkeleja järjestelmän omaan uutissyötteeseen, joka on tilattavissa samalla tavalla kuin muutkin järjestelmän kautta saatavilla olevat RSS-syötteet
JV8	Järjestelmän asetusten muokaus	Järjestelmän ylläpitäjä voi muokata järjestelmän asetuksia WWW-käyttöliittymän kautta
JV9	Asiakasohjelmien hallinta	Järjestelmän ylläpitäjä voi halutessaan perua haluamiensa asiakasohjelmien syötetilauksia.
JV10	Lokitietojen tallentaminen	Järjestelmä tallentaa toiminnastaan tietoja.
JV11	Tilauksen jatkaminen	Asiakasohjelman lähettämään SUBSCRIBE-pyyntöön vastataan SIP-protokollan mukaisesti. Tilaukselle tallennetaan asiakasohjelman ilmoittama uusi päättymishetki. Heti tämän jälkeen palvelin lähettää asiakasohjelmalle jokaista tilattua syötettä kohden NOTIFY-pyyntö.
JV12	Tilauksen muuttaminen	Asiakasohjelman lähettämään SUBSCRIBE-pyyntöön vastataan SIP-protokollan mukaisesti. Tilaukselle tallennetaan asiakasohjelman ilmoittamat uudet käyttäjäasetukset. Heti tämän jälkeen palvelin lähettää asiakasohjelmalle jokaista tilattua syötettä kohden NOTIFY-pyyntö.
JV13	RSS-syötteen poistaminen järjestelmästä	Järjestelmän ylläpitäjä voi poistaa noudettavan RSS-syötteen järjestelmästä.
JV14	Noudettavien RSS-syötteiden asetusten muuttaminen	Järjestelmän ylläpitäjä voi muuttaa järjestelmän kautta tarjolla olevien RSS-syötteiden asetuksia WWW-käyttöliittymän kautta.

3.1.3 Käsiteltävän datan kuvaus

Asiakasohjelmilta saatavien SUBSCRIBE-pyyntöjen runko (body) sisältää XML-muotoista dataa, joka kertoo asiakasohjelman preferenssit järjestelmälle. Näitä preferenssejä ovat tilattavat RSS-syötteet ja niiden päivitystä koskevat asetukset. XML-data noudattaa seuraavaa dokumenttityypin määrittelyä (DTD):

```
<!ELEMENT flags (list?,feed+)>
<!ELEMENT list EMPTY>
<!ELEMENT feed (name,min_interval,since,until?,max_items)>
<!ATTLIST feed type (rss | atom) "rss">
<!ELEMENT name (#PCDATA)>
<!ELEMENT min_interval (#PCDATA)>
<!ELEMENT since (#PCDATA)>
<!ELEMENT until (#PCDATA)>
<!ELEMENT max_items (#PCDATA)>
<!ATTLIST max_items from (beginning | end) "end">
```

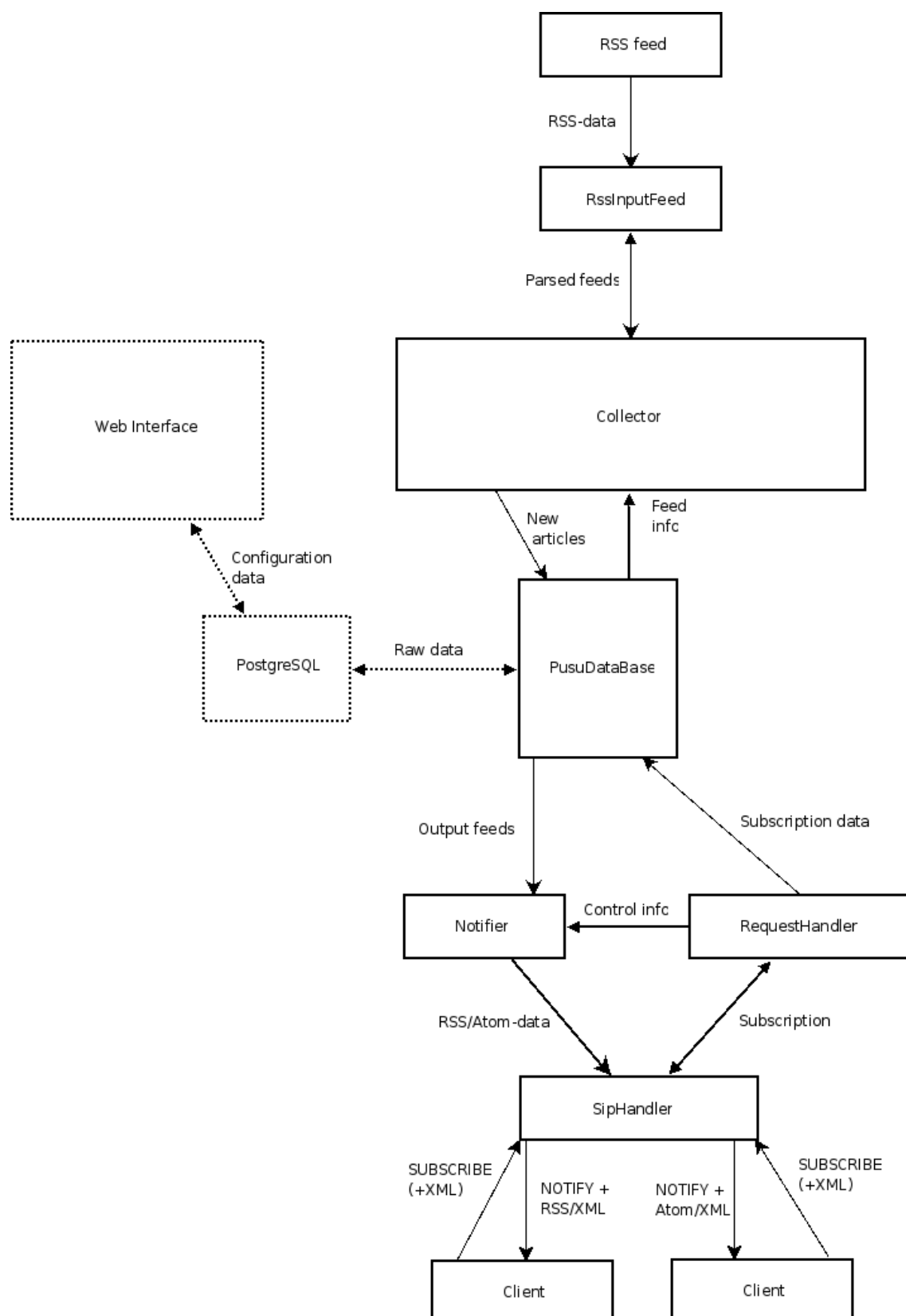
3.1.4 Käyttötapaukset ja sidosryhmät

Sidosryhmät

- Syötteiden tilaajat (asiakasohjelmat)
- Uutispalvelut (joista järjestelmä hakee syötteet)
- Järjestelmän hallinnoijat

Käyttötapaukset

Tunnus	Kuvaus	Muunnelmat	Kuvaus
KT1	Saatavilla olevien syötteiden kysely	KT1b	Saatavilla olevien syötteiden kysely tilausta lopettamatta
KT2	Syötteiden toimitus	-	-
KT3	Tilauksen vastaanottaminen	KT3b	Tilauksen uusiminen/muuttaminen
KT4	Tilauksen lopettaminen	-	-
KT5	Syötteiden hallinnointi	KT5b	Syötteen poistaminen
KT6	Tiedotteiden lisääminen ("serverin oma syöte")	-	-
KT7	Syötteiden nouto järjestelmään	-	-
KT8	Järjestelmän hallinnointi	-	-



Kuva 1: Arkkitehtuurisuunnitelma

3.2 Suunnitteludokumentti

Suunnitteludokumentti sisältää järjestelmän yleisen arkkitehtuurikuvauksen, tietokantasuunnitelman, järjestelmän tärkeimpiä toimintoja kuvaavat sekvenssikaaviot sekä käyttöliittymäsuunnitelman ja liitteet.

Arkkitehtuurimallikappaleessa ja kuvassa 1 on kuvattu järjestelmän osakomponentit ja niiden väliset suhteet sekä komponenttien välillä kulkevan tiedon korkealla tasolla

Tietokantasuunnitelma sisältää järjestelmän tietomallin, kaavion ja tarkan kuvauksen tietokannan tauluista sekä create table -lauseet tietokannan luomista varten.

Palvelimen (ts. järjestelmä pl. ylläpitokäyttöliittymä) toiminta vaatimusmäärittelydokumentissa annetuissa käyttötapauksissa (saatavilla olevien syötteiden kysely, syötteiden kysely tilausta lopettamatta, syötteiden toimitus, tilaus, tilauksen lopettaminen, syötteiden nouto järjestelmään) on kuvattu suunnitteludokumentissa sekvenssikaavioiden avulla. Lisäksi järjestelmän käynnistykselle on oma sekvenssikaavionsa.

Käyttöliittymäsuunnitelmassa on hahmoteltu käyttöliittymän toteutuksen yleisiä suunta-
viivoja. Tarkempi kuvaus varsinaisesta toteutuksesta löytyy käyttöliittymän php-skriptien dokumentaatiosta. Ylläpitodokumentti puolestaan sisältää tietoa käyttöliittymän konfiguroinnista ja käytöstä.

Suunnitteludokumentin liitteitä ovat luokkakaavio ja Javadoc-dokumentaatio. Luokkakaavio paitsi kuvaa palvelinohjelmiston luokat ja pakkaukset ja niiden väliset suhteet, myös toimii tarkemman tason arkkitehtuurikuvana. Luokat on tarkemmin dokumentoitu Javadoc-dokumentaatioissa. Javadoc on lisäksi järjestelmän sisäisten rajapintojen dokumentaatio. Toteutuksen tarkkojen yksityiskohtien dokumentaationa toimivat ohjelmakoodin kommentit niiltä osin kun käytetyt ratkaisut sitä vaativat. Järjestelmän ulkoiset rajapinnat joko perustuvat tunnettuihin standardeihin (SIP, RSS) tai on dokumentoitu vaatimusmäärittelydokumentissa (tilaus- ja syötelistaus-XML).

3.3 Testaussuunnitelma

Testaussuunnitelma sisältää menetelmät, käytännöt ja testitapaukset tuotetun ohjelmiston järjestelmällistä testausta varten. Testaus on pyritty automatisoimaan mahdollisimman pitkälle, jolloin inhimillisen virheen mahdollisuus jää mahdollisimman pieneksi.

Ohjelmiston yksikkö- ja integrointitestaus on toteutettu käyttäen JUnit-kirjastoa. Kirjasto ja siihen liittyvät työkalut mahdollistavat testitapausten suorittamisen täysin automatisoidusti. Testitapauksia ajetaan säännöllisesti, seuraten testattavan koodin kehitystä ja puuttuen huomattuihin virheisiin mahdollisimman aikaisin.

Yksikkötestauksessa on keskitytty pääasiassa halutun toiminnallisuuden ja rajapintamäärittelyn todentamiseen. Jokainen testattava yksikkö asetetaan sitä varten erikseen luo-

tuun testiympäristöön, jossa koodin toimintaa voidaan tarkastella helposti. Testien lausekattavuudeksi on valittu noin 80%. Lausekattavuus on mitattu automaattisesti käyttäen Eclemma-analyysityökalua.

Integroititestaus on suoritettu samoilla menetelmillä ja työkaluilla kuin yksikkötestaus. Testaus keskittyy kuitenkin eri komponenttien väliseen yhteistoimintaan ja pyrkii todentamaan pääasiassa rajapintojen käyttöä komponenttien välillä. Testaus on toteutettu käyttäen *bottom-up* -lähestymistapaa, jossa komponentteja yhdistellään toisiinsa yksi kerrallaan kunnes koko ohjelmisto on täysin integroitu.

Viimeisenä vaiheena ohjelmistolle on suoritettu järjestelmätestaus. Järjestelmätestauksen testitapaukset on johdettu suoraan määrittelydokumentin käyttötapauksista ja ne pyrkivät todentamaan koko ohjelmiston toimintaa yleisimmissä tapauksissa. Järjestelmätestausta ei ole automatisoitu, sillä testitapaukset on suoritettava todellisessa ympäristössä käyttäen todellisia asiakasohjelmia (tässä tapauksessa Nokia Internet Tablet N800-laitteita, joihin on asennettu asiakkaan toimittama ohjelmisto). Mitään suorituskyky- tai rasitustestejä ei ole suoritettu sopivan testiympäristön puuttuessa.

3.4 Käyttö- ja ylläpito-ohje

Käyttö- ja ylläpito-ohje antaa tarvittavat tiedot ohjelmiston käyttäjille siitä, kuinka järjestelmä asennetaan ja konfiguroidaan, ja tarjoaa jatkokehittäjille muutamia vinkkejä mistä kannattaa katsoa ja mitä voisi olla järkevä tehdä.

Asennuskappale vie ohjelmiston käyttäjän läpi tyypillisestä asennusskriptin ajosta ja opastaa käyttäjän suojaamaan ylläpitokäyttöliittymän salasanalla. Tämän jälkeen esitellään järjestelmän konfiguraatiodostot, ja kuinka niitä käytetään.

Käyttöohje taas näyttää miten järjestelmää käskytetään ja minkälaisia toimintoja siitä löytyy. Kappale jakaantuu järjestelmän tapaan luonnollisesti kahteen osaan, palvelimeen ja ylläpitokäyttöliittymään.

Ylläpitokappaleessa kerrotaan ensin mitä ohjelmistoja tarvitaan, jotta järjestelmää voidaan käyttää. Sitten kerrotaan, mitä jäi suunnitelmasta toteuttamatta ja mitkä suunnitelmat muuttuivat toteutusvaiheen aikana. Tämän jälkeen osoitetaan lähdekoodista niitä kohtia, joita voisi olla mielekästä muuttaa. Lopuksi annetaan muutamia jatkokehitysideoita.

4 Projektin päättöanalyysi

4.1 Aikataulu

Projektin aloitusvaiheessa valittiin käytettäväksi prosessimalliksi hiukan sovellettu vesiputousmalli sekä laadittiin alustava projektin aikataulu (kaavio alla). Aikataulua muu-

tettiin projektin edetessä vain hiukan. Kaikki projektinvaiheet pystyttiin aloittamaan ja lopettamaan määräajallaan.

	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Projektin aloitus ja ongelmaan tutustuminen	X														
Projektisuunnitelma	X	X													
Vaatimusmäärittely	X	X	X	X											
Suunnittelu			X	X	X	X	X		X						
Testauksen suunnittelu			X	X	X	X	X		X						
Toteutus ja yksikkötestaus							X		X	X	X	X	X	X	
Integrointi- ja järjestelmätestaus												X	X	X	
Demo ja asiakkaan palaute														X	
Projektin viimeistely ja luovutus															X

Projektin aikana seurattiin eri työvaiheisiin käytettyä työmäärää ohjelmistotuotantoprojekteja varten suunnitellun järjestelmän avulla. Työtunnit jakautuivat kuvan 2 mukaisesti. Selvästi suurimman osan projektin työtunneista veivät yhteiset palaverit sekä suunnitteluvaihe. Huomionarvoinen seikka on, että ohjelmiston toteutukseen ja testaukseen käytettiin projektin aikana suurin piirtein saman verran työtunteja. Koska projektissa toteutettiin palvelinohjelmisto, käyttöliittymän suunnitteluun ja toteutukseen käytetty työmäärä jäi pieneksi.

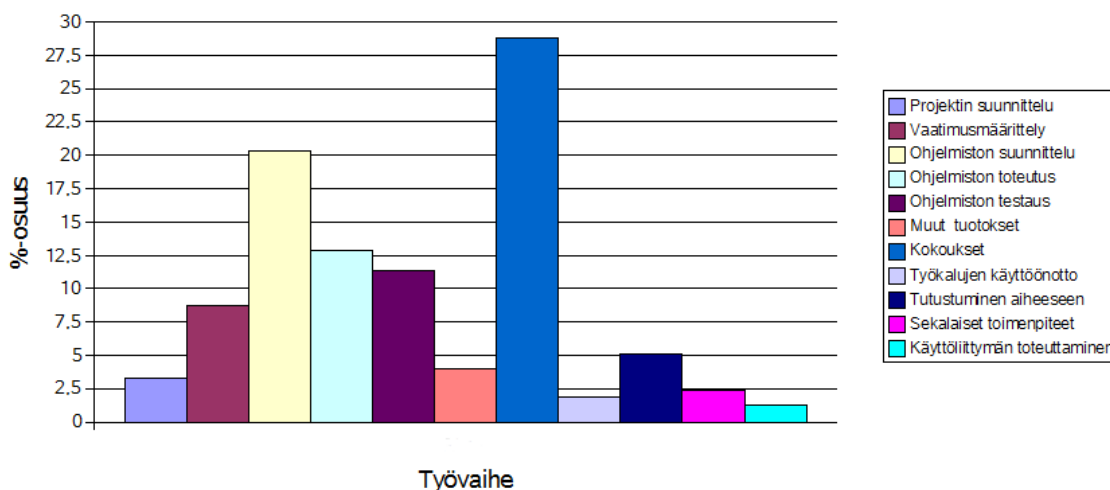
Projektiin käytettiin tunteja noin 160 tuntia henkilöä kohden, eli yhteensä projektiin käytettiin noin 960 työtuntia. Toteutunut tuntimäärä on huomattavasti pienempi, kuin ohjelmistotuotantoprojekteille asetettu maksimituntimäärä 240 tuntia henkilöä kohden. Maltillisesta työtuntimäärästä huolimatta toteutettu ohjelmisto oli laadukas ja projekti onnistui hyvin. Tämän mahdollistivat useat seikat, joista tärkeimpinä voidaan mainita sopiva aiheenrajaus sekä projektiryhmän ammattitaito.

4.2 Työvaiheet

Seuraavissa kappaleissa kuvataan lyhyesti kunkin projektin osavaiheen tärkeimmät päätökset ja toimenpiteet.

4.2.1 Ongelmaan tutustuminen ja projektin aloitus

Projekti aloitettiin tutustumalla aihepiiriin liittyviin termeihin sekä spesifikaatioihin. Eri-tyisesti tutkimuksen kohteena olivat projektin kannalta keskeiset SIP sekä RSS. Projekti-
kuvauksen perusteella voitiin päätellä, että järjestelmään tullaan tarvitsemaan muun



Kuva 2: PUSU-projektin työtuntien jakaantuminen

muassa asiakasohjelmien kanssa kommunikoiva SIP-komponentti sekä RSS-syötteitä käsittelevä komponentti. Projektiryhmä löysi näille komponenteille valmiit toteutukset, mikä helpotti tulevaa toteutustyötä. Projektin prosessimalliksi päätettiin yksimielisesti valita hiukan sovellettu vesiputousmalli, jossa projektin vaiheita oli osittain limitetty päällekkäin. Prosessimalli sekä suunniteltu aikataulu osoittautuivat myöhemmin kyseiseen projektiin erittäin sopiviksi - aikataulua jouduttiin korjaamaan projektin aikana ainoastaan hiukan.

4.2.2 Vaatimusmäärittely

Vaatimusmäärittelyvaiheen aikana järjestettiin asiakkaan kanssa tapaamisia, joissa asiakas esitteli aihepiiriä sekä kartoitettiin systeemille asetettavia vaatimuksia. Tärkein vaatimusmäärittelyn tulos oli asiakasohjelman sekä palvelimen välisen viestinnän määrittelyän formaatin päättäminen. Projektiryhmällä ja asiakkaalla oli pääpiirteissään yhtenäiset mielipiteet järjestelmän vaatimuksista ja vaatimukset saatiin kartoitettua kattavasti ja tarkasti.

4.2.3 Ohjelmiston suunnittelu

Ohjelmiston suunnitteluun käytettiin eniten aikaa projektin aikana. Tärkeimmät ohjelmiston arkkitehtuuria ja rajapintoja koskevat päätökset tehtiin yhteisessä suunnittelutilaisuudessa. Perusteellisen suunnittelutyön tuloksena oli laadukas ja kattava dokumentti, jonka perusteella oli mahdollista toteuttaa vaatimusmäärittelyn mukainen järjestelmä. Oli erittäin merkittävää, että suunnitteluvaiheessa saatiin määriteltyä ja dokumentoitua erit-

täin tarkasti kaikki järjestelmän luokat ja niiden väliset rajapinnat. Suunnitteludokumentin laatu varmistettiin FTR-tilaisuudessa.

4.2.4 Testauksen suunnittelu

Testauksen suunnitteluvaiheessa määriteltiin ohjelmiston testauksen menetelmät sekä hyväksymiskriteerit. Merkittävin päätös oli, että jokaiselle luokalle luodaan ennen toteuttamista yksikkötestit, jotka varmistavat luokan rajapinnan mukaisen toiminnan. Tämä päätös tehtiin, koska jo suunnitteluvaiheessa oli nähtävissä että projektin toteutusvaiheessa on käytettävissä runsaasti aikaa. Päätöksen seurauksena ohjelmisto tuli testattua perusteellisesti.

4.2.5 Toteutus ja yksikkötestaus

Toteutusvaihe aloitettiin jakamalla jokaiselle projektiryhmäläiselle yksikkötestit ja luokat, jotka heidän tuli toteuttaa. Tarkoituksena oli, että luokan ja sen yksikkötestin kirjoittajat ovat eri henkilöt, jotta testeistä saataisiin mahdollisimman puolueettomat ja perusteelliset. Yksikkötesteistä tehtiin varsin perusteellisia. Tästä kertoo esimerkiksi se, että noin puolet ohjelmiston lopullisesta koodimäärästä muodostui testeistä ja se, että testaukseen ja toteutukseen käytetyt tuntimäärät olivat suurin piirtein yhtä suuret. Toteutuksen edistymistä ja testauksen kattavuutta seurattiin tarkoitusta varten projektin wikiin laaditun taulukon avulla. Toteutusvaiheen aikana järjestelmän arkkitehtuuriin tehtiin joitakin muutoksia, jotka päivitettiin myös suunnitteludokumenttiin.

4.2.6 Integrointi- ja järjestelmätestaus

Integrointitestaus suoritettiin luomalla testitapaukset komponenteille sekä erilaisille komponenttien yhdistelmille. Komponenttien yhdistelemiseen käytettiin suunnitelman mukaisesta *bottom-up* -menetelmää.

Järjestelmätestaus suoritettiin käyttämällä asiakkaan tarjoamaa asiakasohjelmaa, jota ajettiin Nokia N800 -tableteilla. Järjestelmätestausta haittasi jonkin verran asiakasohjelman määrittelynvastainen toiminta sekä konfigurointimahdollisuuksien puute. Ongelmista huolimatta järjestelmätestit saatiin suoritettua onnistuneesti.

4.2.7 Projektin päättötoimet

Tärkeimpiin projektin päättötoimiin kuuluivat järjestelmän asennussysteemin luominen, asiakasdemon järjestäminen sekä kaikkien oleellisten projektin tuotosten yhteen kokoaminen.

4.3 Työkalut ja menetelmät

Projektin aikana käyttöön valikoituivat jo ennestään tutut tai muutoin hyväksi havaitut työkalut projektin eri vaiheissa. Heti alussa päätettiin ottaa käyttöön SVN versiohallintajärjestelmänä, jossa säilytettiin lähes kaikkia projektin tiedostoja. SVN repositoryä käytettiin pääasiassa joko TortoiseSVN:n (Windowsissa) tai Subclipsen (Eclipsessä) kautta sekä suoraan komentoriviltä.

Lähes kaikki dokumentit tuotettiin yhteistyössä käyttäen projektille luotua wiki-ympäristöä. Lopulliseen muotoonsa dokumentit muunnettiin käyttäen LaTeX-ladontaohjelmistoa. Koodin dokumentaatio tuotettiin automaattisesti Javan JavaDoc-kääntäjällä sekä TexDoclet-laajennoksella, jonka avulla dokumentti saatiin LaTeXin ymmärtämään muotoon.

Suunnitteluvaiheessa luokka- ja sekvenssikaaviot tuotettiin käyttämällä mm. Poseidon for UML -ohjelmaa tai piirtämällä käsin sopivalla kuvankäsittelyohjelmalla. Kaavioita säilytettiin lähdemuodossa SVN repositoryssä sekä kuvina wiki-dokumenteissa.

Toteutus- ja testausvaiheessa pääasiallinen kehitysympäristö oli Eclipse. Koodin yksikkö- ja integrointitestaus suoritettiin käyttämällä JUnit-kirjastoa ja sen työkaluja. Testien lausekattavuus mitattiin käyttämällä EclEmmaa ja Cloveria. Koodin metriikat laskettiin käyttämällä Metricsiä.

Projektissa käytetyt kirjastot (Informa, JAIN/NIST SIP, Log4j) valittiin luomalla projektin alkuvaiheessa muutama pieni demo-ohjelma, joiden avulla saatiin hieman tuntumaa kirjastojen käyttöön ja toimintaan. Lähes kaikki työkalut valittiin ensisijaisesti automaatiota silmälläpitäen. Joissain tapauksissa työkalujen valintaan vaikuttivat myös yhteensopivuus kehitysympäristön kanssa sekä helpokäyttöisyys.

4.4 Onnistumiset ja haasteet

Projekti oli kaiken kaikkiaan erittäin onnistunut. Projektiryhmä pysyi koko projektin ajan aikataulussa tehden laadukasta työtä. Projektin aikana tuotettu ohjelmisto oli korkealaatuinen ja tarkasti vaatimusmäärittelyn mukainen. Erityisesti ohjelmiston testaus onnistui projektin aikana hyvin. Tästä kertoo esimerkiksi se, että luokkien yksikkötestit kirjoitettiin suurimmaksi osaksi valmiiksi jo ennen itse luokan toteuttamista. Eräs onnistunut osa-alue projektissa oli myöskin ulkopuolisten kirjastojen ja apuohjelmien laaja ja menestyksekkäs käyttäminen.

Projektiin liittyi ainoastaan muutamia haasteita. Vaatimusmäärittelyvaiheen alussa projektiryhmän työskentelyä häirtasivat jonkin verran epäselvyydet liittyen projektin asiakkaan vaihtumiseen. Tämä vaihdos osoittautui kuitenkin myöhemmin vaatimusmäärittelyvaiheen aikana hyödylliseksi, sillä uudella asiakkaalla oli erittäin vankka tietämys aihealueesta.

Käytetyistä kirjastoista Informa osoittautui erityisen haasteelliseksi. Dokumentaatiosta huolimatta kirjasto ei kykene jäsentelemään Atom-syötemuotoa eikä tuottamaan halutun muotoisia RSS-dokumentteja. Lisäksi kirjastosta puuttuu kokonaan tuki Atom-syötemuodon tuottamiseen. Omia hankaluuksiaan tuotti myös kirjaston tapa ilmoittaa uusista artikkeleista sekä syötteiden noudon yhteydessä tapahtuneista virheistä.

Asiakkaan toimittama varsinainen asiakasohjelma osoittautui myös ongelmalliseksi. Ohjelma saatiin testauskäyttöön vasta projektin loppupuolella eikä se noudattanut täysin vaatimusmäärittelyssä kuvattua protokollaa, vaikeuttaen etenkin järjestelmätestausta. Lisäksi ohjelma sisältää joitain asetuksia, joita ei pysty muuttamaan ilman uudelleenkäynnistämistä. Tästä syystä kaikkia palvelimen ominaisuuksia ei voitu testata todellisella asiakasohjelmalla.

4.5 Yhteenveto

Ohjelmistotuotantoprojektien päätavoite on ensisijaisesti kognitiivinen - tarkoituksena on saada mahdollisimman hyvä kuva siitä, millaista on olla mukana käytännön työelämän ohjelmistoprojekteissa. Kurssin aikana opiskelijoilla on mahdollisuus ja tarkoitus kokeilla käytännössä Ohjelmistotuotanto-kurssilla opittuja menetelmiä. Toissijaisista tavoitteista tärkeimpiä ovat ohjelmiston ja dokumentoinnin laatu.

Suurimmalla osalla projektiryhmäläisistä oli jo ennestään runsaasti kokemusta käytännön projektityöskentelystä. Heille kurssin tarjoama hyöty ei ollut niin suuri kuin muille ryhmäläisille. Kurssin päätavoite täyttyi kuitenkin kaiken kaikkiaan erittäin hyvin: projektiryhmä toimi tiiviisti yhdessä käyttäen hyväksi todettuja ohjelmistotuotannon menetelmiä. Jokaisella ryhmäläisellä oli jaettu oma vastuualueensa, mutta tämä jako oli lähinnä nimellinen. Käytännössä jokainen ryhmän jäsen oli tärkeässä asemassa koko projektin ajan. Ryhmän osaaminen näkyi esimerkiksi siinä, että projekti pysyi hyvin aikataulussa, ja että projektin aikana tuotetut dokumentit olivat laadukkaita.

Kaiken kaikkiaan projekti oli erittäin onnistunut. Se tarjosi jokaiselle ryhmäläiselle mahdollisuuden oppia jotakin uutta ja kehittää itseään työelämää varten. Projektiryhmä osallistui kurssin aikana myöskin ohjelmistotuotantoprojektitutkimukseen, mikä antoi hyvän tilaisuuden saada työskentelystään monipuolista palautetta.

5 Liitteet

Liite 1 Testausraportti