Projektisuunnitelma

Kihla-ryhmä

Helsinki 2.2.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Markus Bunders
Harri Hämäläinen
Anni Kotilainen
Raine Leskinen
Panu Luosto
Janne Mäntyharju
Petrus Repo

Asiakas

Heikki Lokki

Johtoryhmä

Juha Taina

Kotisivu

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

Versiohistoria

Versio	*Päiväys*	*Muutokset*
PS1-RC1	1.2.2007	Alustava lopullinen, esitellään projektiryhmälle
PS1-0.3	1.2.2007	Riskit, ositus, projektin yleiskuvaus, materiaali Wikistä
PS1-0.2	29.1.2007	Ryhmän jakautuminen huomioitu vastuualueissa
PS1-0.1	26.1.2007	Alustava versio

Sisältö

1 Johdanto

Suomessa rengastetaan vuosittain 200 000 – 250 000 lintua, ja rengastettujen lintujen tapaamisia kirjataan vuodessa noin 40 000 kappaletta. Osa tavatuista linnuista on rengastettu ulkomailla, ja toisaalta Suomessa rengastettuja lintuja tavataan varsinkin muuttoreittien varrella eri puolilla maapalloa. Tietojen keräämisestä ja säilyttämisestä vastaavat Suomessa Helsingin yliopiston Luonnontieteellisen keskusmuseon rengastustoimiston 7 työntekijää. Lintuja rengastaa Suomessa vuosittain noin 600 vapaaehtoista harrastajaa. Rengastusluvan saamiseksi on muun muassa läpäistävä vaativa lajintuntemustentti.

Rengastukseen liittyvät tiedot ovat Helsingin yliopiston tietotekniikkaosaston ylläpitämässä tietokantapalvelimessa Oracle 10 -tietokannassa. Rengastajat toimittavat tallennusohjelmalla tallentamansa rengastus- ja tapaamistiedot rengastustoimistoon tietokantaan tallettamista varten. Vuodesta 1974 alkaen kaikki rengastukseen liittyvät tiedot ovat tietokannassa, ja vanhempia tietoja ollaan tallettamassa tietokantaan siirtämistä varten. Tietokannassa on nykyisin noin 7 000 000 rengastuksen ja 1 000 000 tapaamisen tiedot. Rengastustoimiston henkilökunta ylläpitää tietoja selaimeen perustuvalla käyttöliittymällä. Raportteja tutkijoille, rengastajille ja viranomaisille tuotetaan rengastustoimistossa muokkaamalla kyselykieleen perustuvia ohjelma- ja komentotiedostoja.

Projektiryhmän tehtävänä on toteuttaa ratkaisu kolmeen seuraavaan rengastustoimiston kuvailemaan ongelmaan.

1.1 Rengastajan toimittamien tietojen käsitteleminen

Rengastajat toimittavat tiedot rengastamistaan linnuista sekä talletusohjelmalla talletettuina tiedostoina että paperitulosteina. Paperitulosteet mapitetaan rengastustoimistossa yksikäsitteisen renkaan tunnuksen mukaan järjestettyinä. Paperitulosteita kertyy paljon ja niistä halutaan luopua. Paperitulosteiden sijaan halutaan tallettaa rengastajien lähettämät tiedostot siten, että myöhemmin voidaan palata tarkastamaan yksittäisten renkaiden tietoja rengastajan lähettämässä tiedostossa virheiden selvittelyjen yhteydessä.

Tehtävänä on suunnitella ja toteuttaa järjestelmä, jossa rengastajien lähettämät tiedostot säilytetään ja yksittäisen renkaan tiedot löytyvät helposti, nopeasti ja luotettavasti. Rengastajat lähettävät tiedostoja rengastamistaan linnuista yleensä 1–3 kertaa vuodessa ja tiedostojen kertymä vuodessa on noin 1000.

1.2 Rengastustoimiston luomien raporttien välitys rengastajille

Rengastettujen lintujen tapaamisista lähetetään kirje linnun tapaamishistoriasta tapaajalle, rengastajalle ja kaikille muillekin, jotka kyseistä lintua ovat rengastustoimiston tietojen mukaan käsitelleet. Mikäli lintu on tavattu ulkomailla, kyseisen maan rengastustoimisto saa myös kirjeen. Rengastustoimiston järjestelmästä tuotetaan kirjeitä 10 eri kielellä ja kirjeet lähetetään paperipostissa. Vuosittain tapaamiskirjeitä lähetetään 20 000 – 30 000 sivua. Merkittävä osa vastaanottajista toivoisi saavansa palautteen rengastustoimistosta sähköpostissa siten, että he voisivat itse halutessaan tulostaa saamansa kirjeet.

Tehtävänä on suunnitella ja toteuttaa palautekirjeiden luotettava ja vaivaton lähettäminen sähköpostitse sekä toiminnan muuttamisen vaikutus rengastustoimiston tietokan-

taan.

1.3 Tuotantokäytössä olevan tietojärjestelmän käyttöliittymän kehittäminen

Syksyllä 2006 on rengastustoimistossa otettu käyttöön uusi selaimeen perustuva käyttöliittymä tietokantajärjestelmään. Tietokantaan on talletettuna muun muassa runsaasti arvoja, joiden perusteella tarkastetaan kantaan talletettavien tietojen oikeellisuus. Tehtävänä on laajentaa käyttöliittymää kattamaan arvojen ylläpito niiltä osin kuin näytöt puuttuvat.

2 Projektin organisaatio

Ohjelmistotuotantoprojektiryhmä Kihlan seitsemän jäsentä on jaettu kahteen pienryhmään. Jakautuminen päätettiin suorittaa, koska asiakkaalta saadut tehtävät olivat pilkottavissa sekä uuden pienehkön järjestelmän toteuttamiseen että tuotantokäytössä olevan järjestelmän parantelemiseen. Uuden järjestelmän toteuttava ryhmän nimeksi annettiin ”Raiteilla” (Ruby on Rails -ryhmä) ja olemassa olevan järjestelmää eteenpäinkehittävä ryhmä nimettiin ”Jokeriksi” (Java-ryhmä).

Ensisijaisesti Raiteilla-ryhmässä toimii neljä henkilöä ja Jokeri-ryhmässä kolme. Projektipäällikkö on vastuussa kummankin ryhmän aikataulusta sekä vastuualueiden delegoinnista. Jokaisella ryhmän jäsenellä on oma vastuualueensa, josta hänellä on ylin päätäntävalta. Muuten ryhmien toiminta on tasa-arvoista. Kaikki saavat esittää toisilleen kysymyksiä, ja erityisesti rakentavan palautteen antaminen on suotavaa.

Mikäli vastuuhenkilö ei väliaikaisesti kykene hoitamaan tehtäväänsä, hänen paikkansa ottaa varavastaava, kunnes tilanne palautuu ennalleen. Tällainen tilanne voi olla esimerkiksi sairaustapaus projektin kriittisessä vaiheessa. Mikäli vastuuhenkilön estynyt tilanne kuitenkin pitkittyy, asia käsitellään projektiryhmän kokouksessa tai ohjaajan kanssa.

Vastuualueet jakautuvat kahden pienryhmän kesken seuraavasti:

Molemmille ryhmille yhteiset vastuualueet

Petrus Repo	Projektipäällikkö, varakoodivastaava.
Markus Bunders	Vaatimusmäärittelyvastaava, varatarkastusvastaava.
Harri Hämäläinen	Suunnitteluvastaava, varaprojektipäällikkö.
Anni Kotilainen	Tarkastus- ja käyttöliittymävastaava, varasuunnitteluvastaava.
Raine Leskinen	Testausvastaava, varavaatimusmäärittelyvastaava.
Panu Luosto	Dokumenttivastaava, varatestausvastaava.
Janne Mäntyharju	Koodivastaava, varadokumenttivastaava.

Raiteilla-ryhmän vastuualueet

Harri Hämäläinen	Ryhmän suunnittelu- ja koodivastaava.
Petrus Repo	
Anni Kotilainen	
Raine Leskinen	

Jokeri-ryhmän vastuualueet

Janne Mäntyharju	Ryhmän koodivastaava, avustava dokumenttivastaava.
Panu Luosto	Ryhmän suunnitteluvastaava.
Markus Bunders	

Vastuuhenkilöiden tehtävät ovat lyhyesti määriteltynä seuraavanlaiset:

Projektipäällikkö vastaa projektisuunnitelmasta, projektin aikataulusta, henkilöiden allokoinnista tehtäviin ja riskienhallinnasta. Toimii puheenjohtajana kokouksissa, ellei kokous liity erityisesti jonkun muun vastuualueeseen.

Vaatimusmäärittelyvastaava toimii asiakkaiden ja projektiryhmän yhdyshenkilönä, vastaa vaatimusmäärittelyn osavaiheiden onnistumisesta, määrää vaatimusdokumentin rakenteen ja toimii puheenjohtajana vaatimusmäärittelyyn liittyvissä kokouksissa. Hän vastaa myös siitä, että kaikki tuotteeseen kohdistuvat vaatimukset saadaan kirjattua ylös.

Suunnitteluvastaava vastaa suunnittelutason rajapinnoista ja suunnitteludokumentin yhdenmukaisesta rakenteesta. Hän pitää huolen, että suunnittelu tehdään projektin kannalta riittävällä tarkkuudella ja että tietokannan määrittely täyttää sille asetetut tavoitteet.

Koodivastaava vastaa siitä, että ohjelmakoodin ulkoasu on yhteneväinen, ryhmän jäsenet tekevät yksikkötestauksen, rajapinnat ovat yhtenevät myös kooditasolla ja että koodi vastaa suunnittelua sekä arkkitehtuuri- että komponenttitasolla.

Testausvastaava vastaa testauksen kattavuudesta. Hän huolehtii, että kaikki käyttötapaukset testataan, kaikki käyttäjän vaatimukset testataan, kaikki kirjatut poikkeustilanteet testataan ja että asiakkaalle annetaan mahdollisuus hyväksymistestaukseen.

Dokumenttivastaava vastaa siitä, että dokumenttien ulkoasu on yhteneväinen, dokumentit ovat luettavassa kunnossa ja että dokumenttien sisältö on kattava. Dokumenttivastaava pitää yllä projektin kotisivua ja huolehtii tiedon järjestämisestä Wikiin.

Tarkastus laatii tarkastuslistat, vastaa tarkastusten järjestelyistä ja toimii tarkastustilaisuuksissa puheenjohtajana.

Käyttöliittymävastaava vastaa ohjelmiston käytettävyydestä, käyttöliittymäsuunnitteluprosessin rakenteesta sekä sen dokumentoinnin yhtenäisyydestä ja kattavuudesta. Hänen vastuualueeseensa kuuluu myös se, että suunniteltu käyttöliittymä toteuttaa vaatimusmäärittelyssä määritellyn toiminnallisuuden.

3 Riskianalyysi

Seuraavissa kolmessa taulukossa esitellään terminologia, jota projektiryhmä käyttää riskien ja riskeihin varautumistoimenpiteiden luokittelussa.

Riskin toteutumistodennäköisyydelle projektiryhmä käyttää seuraavaa luokitusta. Todennäköisyysvälit ovat puhtaasti suuntaa antavia.

Termi	**Todennäköisyys**
Lähes varma	$0{,}9 < p \leq 1{,}0$
Varma	$0{,}7 < p \leq 0{,}9$
Keskiverto	$0{,}5 < p \leq 0{,}7$
Alle keskiverto	$0{,}3 < p \leq 0{,}5$
Mahdollinen	$0{,}1 < p \leq 0{,}3$
Epätodennäköinen	$0{,}0 \leq p \leq 0{,}1$

Projektiryhmä luokittelee riskit niiden seurauksien vakavuuden mukaan seuraavassa taulukossa esitettyihin kategorioihin.

Termi	**Vaikutus**
Katastrofaalinen	Riskin toteutuminen lopettaa projektin.
Erittäin vakava	Riskin toteutuminen vahingoittaa projektia ja voi estää sen jatkumisen.
Vakava	Riskin toteutuminen haittaa projektia ja voi estää sen pysymisen aikataulussa.
Keskiverto	Riskin toteutuminen haittaa projektia ja voi estää kaikkien haluttujen ominaisuuksien toteuttamisen.
Lievä	Riskin toteutuminen aiheuttaa projektiin lisätyötä, mutta ei estä projektia valmistumasta aikataulussa toivotuin ominaisuuksin.
Erittäin lievä	Riskin toteutuminen aiheuttaa pientä epämukavuutta projektissa.

Projektiryhmä käyttää seuraavan taulukon mukaista jaottelua priorisoitaessa riskeihin liittyviä varautumistoimenpiteitä.

Termi	**Varautumistoimenpiteet**
Ykkösluokka I	Riskin toteutumisen todennäköisyyttä pienennetään aktiivisesti koko projektin kestävin vastatoimin. Riskin toteutuessa on varasuunnitelma.
Kakkosluokka II	Riskin toteutumisen todennäköisyyteen pyritään vaikuttamaan projektin alussa tehtävillä päätöksillä. Riskin toteutuessa on varasuunnitelma.
Kolmosluokka III	Riskin toteutumisen todennäköisyyteen ei vaikuteta. Riskin toteutuessa on olemassa varasuunnitelma.
Nelosluokka IV	Riskiä ei huomioida.

3.1 Projektin riskit

Seuraavassa on jaoteltu projektin riskit kolmeen ryhmän sen mukaan, ovatko ne sidottuja jompaankumpaan pienryhmään vai ovatko ne molemmille ryhmille yhteisiä.

3.1.1 Yleiset riskit

1. Jäsenen keskeytys
    - Esimerkki: Ryhmän jäsen päättää keskeyttää kurssin henkilökohtaisten kiireiden tai muun vastaavan syyn vuoksi.
    - Seuraus: Vakava. Projektista katoaa vastuu henkilö, muiden työtaakka kasvaa, projekti mahdollisesti viivästyy.
    - Ehkäisy: Projektiryhmä on avoin toisilleen, jäseniä kannustetaan ja pyritään luomaan yleinen hyvä ilmapiiri.
    - Todennäköisyys: mahdollinen
    - Prioriteetti: IV
2. Aikataulu ei pidä
    - Esimerkki: projekti venyy, koska projektiryhmä ei käytä riittävästi aikaa projektille, tai projektin eteneminen pysähtyy esteeseen, tai projekti osoittautuu arvioitua suuremmaksi.
    - Seuraus: Vakava, projektia ei ehditä toteuttaa suunnitellussa mittakaavassa.
    - Ehkäisy: Projektia hallitaan projektipäällikön toimesta, joka aikatauluttaa projektin ja seuraa projektin etenemistä. Käydään jokaisen viikon ensimmäisessä kokuksessa lista kyseisen viikon tehtävistä ja varmistetaan, että kunkin tehtävän vastuuhenkilö varaa kalenteristansa riittävän määrän tunteja tehtävän suorittamiseksi.
    - Todennäköisyys: keskiverto
    - Prioriteetti: II
3. Asiakkaan vaatimukset ymmärretään väärin

- Esimerkki: Asiakas esittää vaatimuksensa, jotka ryhmä uskoo ymmärtävänsä, mutta jotka siitä huolimatta tulkitaan virheellisesti.
- Seuraus: Vakava. Ryhmä käyttää aikaa tehtäviin, jotka eivät ole projektin kannalta oleellisia.
- Ehkäisy: Asiakkaalle esitetään demoja välivaiheista, jolloin väärinymmärretty vaatimus saadaan kiinni mahdollisimman varhaisessa vaiheessa. Toisaalta vaatimusmäärittelyyn kiinnitetään paljon huomiota ja se tehdään asian vaatimalla tarkkuudella.
- Todennäköisyys: alle keskiverto
- Prioriteetti: I

4. Asiakkaalle tuotetaan vakavaa haittaa
   - Esimerkki: koko rengastustiedot sisältävä tietokanta onnistuu katoamaan tai rengastustietoja vuotamaan ulkopuolisten käsiin.
   - Seuraus: Vakava. Mahdollisesti koko rengastustoiminnan laatu Etelä-Suomen alueella heikkenee.
   - Ehkäisy: Käytetään kehitystietokantaa ennen tuotantoon siirtämistä. Huolehditaan tietoturvasta.
   - Todennäköisyys: epätodennäköinen
   - Prioriteetti: II
5. Projektin työmäärä ei jakaudu tasaisesti
   - Esimerkki: Projektiryhmästä osa tekee ohjeellista tuntimäärää huomattavasti enemmän töitä viikossa ja vastaavasti osa henkilöistä ohjeellista tuntimäärää vähemmän.
   - Seuraus: Keskinkertainen. Ylitöitä tekevät väsyvät liiaksi, ja heidän motivaationsa laskee kurssin edetessä.
   - Ehkäisy: Käydään viikon tehtävät yhdessä läpi. Delegoidaan tehtäviä pois ylityöllistetyiltä jäseniltä. Tarvittaessa tarkastetaan tehtävien prioriteetit ja viimeisenä vaihtoehtona jätetään prioriteetiltaan vähemmän tärkeitä tehtäviä tekemättä.
   - Todennäköisyys: keskiverto
   - Prioriteetti: I
6. Dokumentoinnille jää liian vähän aikaa.
   - Esimerkki: Iteroinnin toisen syklin alussa havahdutaan siihen, että ensimmäisen syklin dokumentit on toteutettu liian heppoisasti, minkä seurauksena toisessa syklissä kuluu suunniteltua enemmän aikaa dokumentointiin.
   - Seuraus: Vakava. Joko toteutukselle tai dokumentoinnille ei jää riittävästi aikaa toisessa iteraatiossa.
   - Ehkäisy: Dokumentointivastaava (yhdessä projektipäällikön kanssa) seuraa riittävällä tarkkuudella aikataulun mukaisesti ”toteutetun” dokumentoinnin kattavuutta. Samat toimenpiteet kuin kohdassa ”aikataulu ei pidä”.
   - Todennäköisyys: alle keskiverto
   - Prioriteetti: I

3.1.2 Raiteilla-ryhmän riskit

1. Railsin ja olemassa olevan Oracle-tietokantarakenteen yhteensovittaminen tuottaa vaikeita ongelmia.
   - Esimerkki: Tuotantokäytössä olevan tietokantajärjestelmän koko on 10 gigatavua, minkä vuoksi pienimääräisen esimerkkidatan tuominen omaan testikantaan vaatii paljon manuaalista työtä. Mahdollisesti dokumentoimaton rakenne voi myös aiheuttaa inhimillisen tulkintavirheen, kun testitietokantaa toteutetaan.
   - Seuraus: Vakava. Toteutusta ei voida kunnolla aloittaa ennen kuin tämä pohjatyö on tehty.
   - Ehkäisy: Pyydetään tuotantokäytössä olevasta tietokannasta erillinen kopio, jota voidaan viime kädessä käyttää oman testikantamme rinnalla.
   - Todennäköisyys: mahdollinen
   - Prioriteetti: II
2. Olemassa olevan järjestelmän raportinluontityökalusta (dokumentoimaton Fortran-koodi) ei saada riittävän selkeää kuvaa tai sen toimintaa on tulkittu puutteellisesti.
   - Esimerkki: Lukuisat poikkeukset raportin tietojenkeräilyn säännöstössä aiheuttavat inhimillisen tulkintavirheen, jolloin uuden järjestelmän toteutus kerää raporttiin tietoa virheellisin perustein.
   - Seuraus: Vakava. Oleellista järjestelmän osaa ei voida toteuttaa tai se toteutetaan väärin.
   - Ehkäisy: Käydään Fortran-koodista tulkittu säännöstö tarkasti läpi asiakkaan kanssa. Viimeisenä vaihtoehtona muokataan olemassa olevaa Fortran-koodia suorittamaan tietojen keräily uuteen järjestelmään (XML).
   - Todennäköisyys: alle keskiverto
   - Prioriteetti: II
3. Ryhmän jäsenten kokemattomuus Ruby on Rails -ympäristöstä johtaa puutteelliseen suunnitteluun ja/tai toteutukseen.
   - Esimerkki 1: Toteutus ei ole riittävissä määrin MVC-periaatteiden mukainen.
   - Esimerkki 2: Toteutuksessa ei noudateta riittävissä määrin ”Don’t Repeat Yourself” -periaatetta.
   - Esimerkki 3: Toteutuksessa luodaan itse rakenteita ja moduuleja, jotka olisivat tarjolla suoraan Rails-kehyksestä.
   - Seuraus: Vakava. Lopputulos on monimutkainen tai epäselkoinen
   - Ehkäisy: Järjestetään Rails-tutustumisilta. Koodivastaava seuraa koko suunnitteluja toteutusvaiheen ajan toteutettuja ratkaisuja sekä ehdottaa menetelmiä eri ongelma-alueiden ratkaisemiseksi.
   - Todennäköisyys: keskiverto
   - Prioriteetti: I

3.1.3 Jokeri-ryhmän riskit

1. Ongelma-alue on pirstoutunut pieniin osiin
   - Esimerkki: Asiakkaalta ei saada ongittua alkuvaiheessa riittävästi vaatimuksia, jotta varsinainen suunnittelu tai toteutus voisi alkaa.
   - Seuraus: Vakava. Java-ryhmällä on tyhjäkäyntiä.
   - Ehkäisy: Pyritään olemaan aktiivisesti yhteydessä asiakkaaseen ja viimeisenä vaihtoehtona annetaan ihmisille tehtävää Rails-ryhmästä.
   - Todennäköisyys: keskiverto
   - Prioriteetti: I
2. Edellisen projektiryhmän ohjelmakoodit ja käytetyt tekniset ratkaisut ovat tuntemattomia tai huonoja.
   - Esimerkki: Käyttöliittymän Java-koodi osoittautuu sotkuisaksi.
   - Seuraus: Keskinkertainen: Työmäärä kasvaa ja projekti saattaa viivästyä.
   - Ehkäisy: Koodin ja teknisten välineiden kartoitus heti projektin alussa, jotta huonoihin osioihin voidaan puuttua tarpeeksi ajoissa.
   - Todennäköisyys: keskiverto
   - Prioriteetti: II

4 Laitteisto- ja ohjelmistoympäristön vaatimukset

Ohjelmisto vaatii Ruby-tulkin, RubyGEMS -paketin ja Rails-kehyksen. Projektin vaatimat mahdolliset muut GEM-asennukset selviävät yksityiskohtaisen suunnittelun yhteydessä.

Rengastustoimiston tuotantokäytössä oleva tietokantajärjestelmä on Oracle. Tuotantotietokannassa on raakamuotoista dataa yhteensä noin 10 gigatavua.

Työskentely-ympäristönä käytetään jotain Ruby on Rails -ystävällistä ohjelmistoa, kuten RadRails (Windows) tai TextMate (Mac OS X). Viime kädessä kehitysohjelmistona voi käyttää myös tekstieditoria, mutta oleellista on, että Rails-konventioiden mukaisesti hajautettu tiedostohierarkia on kehittäjän kannalta helposti tarkasteltavissa.

5 Koko- ja kustannusarviot

Kihla-järjestelmän ydinohjelmisto voidaan jakaa seuraaviin osiin (vertaa Johdanto-luvun listaan):

1. Sisäänkirjautuminen. Järjestelmä vaatii, että jokaisella käyttäjällä on käyttäjätunnus ja salasana.
2. Tuntikirjanpidon syöttö. Tiedot syötetään kenttä kerrallaan.
3. Oman tuntikirjanpidon katselu ja muokkaus.

4. Oman tuntikirjanpidon merkintöjen poisto.
5. Oman ryhmän tuntikirjanpitojen yhteenvetojen katselu. Tiedot näytetään joko viikoittain tai tehtyjen tehtävien mukaan ryhmiteltyinä.
6. Muiden ryhmien tuntikirjanpitojen yhteenvetojen katselu. Tässä ovat samat näytöt kuin edellä.
7. Ryhmän ilmoitustaulun katselu ja viestien kuittaus.
8. Uuden viestin lisäys ryhmän ilmoitustaululle.
9. Oman viestin poisto ilmoitustaululta.
10. Omien tietojen näyttö ja muokkaus.
11. Ryhmän tietojen näyttö ja muokkaus (ohjaaja ja vastuuhenkilö).
12. Käyttäjien tietojen näyttö ja muokkaus (vastuuhenkilö).
13. Käyttäjän ryhmän vaihto (vastuuhenkilö).
14. Ryhmän lisäys (ohjaaja ja vastuuhenkilö).
15. Ryhmän tietojen muokkaus (ohjaaja ja vastuuhenkilö).
16. Ryhmän poisto (vastuuhenkilö).
17. Käyttäjän poisto (vastuuhenkilö).

6 Toimintopisteanalyysi

Toimintopisteanalyysilla kartoitetaan projektin tuotteen kokoa. Ennen toimintopisteiden aikaa tuotteen kokoa arvioitiin mm. koodirivien määrällä. Koodirivien määrä ei kuitenkaan ole yhtenäinen mittayksikkö, sillä tarvittavien rivien ja merkkien määrät vaihtelevat huomattavasti eri kielten välillä. Toimintopiste-idea perustuu siihen, että tuotteen toiminnalliset ominaisuudet pisteytetään samaan tapaan kuin tuotteelle asetetut vaatimukset toteutusympäristön suhteen. Lisää tietoa tästä löytyy muun muassa Juha Tainan kirjoittamasta artikkelista "A brief introduction to function points" (`http://www.cs.helsinki.fi/group/ohtu/resurssit/fp.html`).

Toimintopisteet saadaan laskettua kaavasta: $FP = c(\alpha + \beta \sum F_i)$. Seuraavassa käytetään empiirisiin kokoeisiin perustuvia vakioita $\alpha = 0{,}65$ ja $\beta = 0{,}01$. Kertoimen c arvo saadaan eri osa-alueiden tehtävien lukumäärän summana, jota painotetaan osa-alueiden vaikeusasteiden mukaan. Projektin tehtävien vaikeustaso määritellään karkeasti kolmiportaisella asteikolla *helppo*, *keskinkertainen* ja *vaikea*. Näitä vastaavat kertoimet heijastavat yleistä käsitystä osa-alueiden oletetuista yleisistä vaikeusasteista. Tämän projektin osalta kerroin c määritellään seuraavien arvioiden perusteella.

Osa-alue	Lukumäärä	Vaikeustaso	Kerroin
käyttäjäsyötteet	5	helppo	3
käyttäjätulosteet	5	keskinkertainen	5
käyttäjäkyselyt	0		
tiedostot	4	keskinkertainen	10
ulkoiset rajapinnat	2	vaikea	10
Yhteensä	$3 \cdot 5 + 5 \cdot 5 + 10 \cdot 4 + 2 \cdot 10 = 100$		

Summa $\sum F_i$ saadaan puolestaan taulukon 1 kysymysten perusteella. Kysymystä vastaavan ominaisuuden prioriteetti arvioidaan asteikolla 0–5, missä 0 tarkoittaa merkityksetöntä ja 5 erittäin oleellista ominaisuutta. Näiden perusteella saadaan lopulta toimintopistearvoksi $FP = 100 \cdot (0{,}65 + 0{,}01 \cdot 28) = 93$.

Koodirivien määrää voidaan arvioida kertomalla FP kielelle määritellyllä vakiolla, joka oliokielille on 30. Tällöin arvioitu koodirivien määrä on noin 2800 riviä. Laskettu arvio koskee uuden järjestelmän toteutuksen vaatimaa työmäärää. Koska järjestelmä on tarkoitus toteuttaa Ruby on Rails -sovelluskehyksellä, jossa koodirivin eteen tehty suunnittelutyö on kuitenkin huomattavasti perusoliokieltä suurempi, ei tätä arviota voida pitää kovinkaan luotettavana.

7 Projektin ositus

Projektin prosessimalliksi on valittu iteratiivinen kahden syklin malli. Projektin kesto on 15 viikkoa siten, että viikot 1–7 ovat ensimmäistä iteraatiosykliä, viikko 8 on tarkoitettu väliviikoksi, jota voidaan tarvittaessa käyttää aikataulun kuromiseksi, ja viikot 9–15 ovat iteraation toinen sykli.

Ensimmäisen syklin (viikot 1–7) aikana tuotettavat dokumentit ovat:

- Yhteiset dokumentit:
  - projektisuunnitelma ”PS1”
- Ryhmä Raiteilla:
  - vaatimusmäärittelyn yleistaso ”VD1-Rails”
  - suunnitteludokumentti ”SD1-Rails”
  - järjestelmätestauksen suunnittelu ”TS1-Rails”
- Ryhmä Jokeri:
  - vaatimusmäärittelyn lopullinen versio ”VD1-Java”
  - suunnitteludokumentti limittäisen toteutuksen kannalta riittävän pienissä paloissa ”SD1-Java”

	Kysymys	**Prioriteetti**
1.	Vaatiiko järjestelmä luotettavaa tietojen varmuuskopiointia ja palautusta?	2
2.	Tarvitaanko tietoliikenneyhteys?	4
3.	Sisältääkö järjestelmä hajautettuja funktioita?	2
4.	Onko tehokkuus kriittisessä roolissa?	1
5.	Sijoittuuko järjestelmä jo olemassa olevaan paljon käytettyyn ympäristöön?	0
6.	Vaatiiko järjestelmä reaaliaikaista tietojen syöttämistä?	4
7.	Vaatiiko reaaliaikainen tietojen syöttäminen useita näyttöjä ja operaatioita?	2
8.	Päivitetäänkö järjestelmän kannalta kriittisiä tiedostoja?	1
9.	Ovatko syötteet, tulosteet, kyselyt ja tiedostot monimutkaisia?	3
10.	Onko sisäinen prosessointi monimutkaista?	2
11.	Onko koodin tarkoitus olla uudelleen käytettävää?	1
12.	Sisältääkö suunnitelma konversion ja asennuksen?	1
13.	Onko järjestelmä suunniteltu käytettäväksi monessa eri organisaatiossa omina asennuksinaan?	1
14.	Onko järjestelmän tarkoitus muuttaa ja helpottaa käyttäjän toimia?	4
	Yhteensä	28

Taulukko 1: Summan $\sum F_i$ laskeminen toimintopisteanalyysiä varten.

Toisen syklin (viikot 9–15) aikana tuotettavat dokumentit ovat:

- Yhteiset dokumentit:
  - projektisuunnitelman tarkennus ”PS2”
- Ryhmä Raiteilla:
  - vaatimusmäärittelyn lopullinen versio ”VD2-Rails”
  - suunnitteludokumentin lopullinen versio ”SD2-Rails”
  - testaussuunnitelman päivitys ja lopullinen versio ”TS2-Rails”
- Ryhmä Jokeri:
  - suunnitteludokumentin lopullinen versio SD2-Java
  - testaussuunnitelma TS2-Java
  - toteutus ja yksikkötestaus

7.1 Ajoittuminen ensimmäisessä syklissä

7.1.1 Vaatimusmäärittelyn ositus

Vaatimusmäärittelyn osavaiheet 1. syklissä ovat seuraavat:

- Ryhmä Raiteilla:
  - vaatimusten kartoitus: viikot 2–3
  - vaatimusten tarkennus: viikko 3
  - vaatimusten validointi: viikko 4
  - käyttöliittymäprototyypin teko: viikko 3
    - käyttöliittymäprototyypin esittely asiakkaalle: viikko 4
  - mallien teko: viikko 3
    - järjestelmän toiminnan kuvaaminen kaavioiden avulla
- Ryhmä Jokeri:
  - vaatimusmäärittelyä pienissä osissa lisäävän mallin mukaisesti siten, että kartoitusta, määrittelyä ja validointia tehdään limittäin viikot 2–7
    - lopullisen vaatimusmäärittelydokumentin muodollinen tarkastus (FTR) viikolla 7
  - mallien teko
    - malli apinoidaan nykyisellään olemassa olevasta järjestelmästä

7.1.2 Suunnittelun ositus

Suunnittelun osavaiheet ovat ensimmäisessä syklissä yleisarkkitehtuuri, mahdolliset rajapinnat, tietokanta, käyttöliittymä sekä WWW-järjestelmä. Rajapintasuunnittelun tarve

ilmenee vasta, kun vaatimusmäärittelystä saadaan ulos tarkempi kuva järjestelmästä sekä järjestelmän ympäristöstä. Sunnittelun osavaiheita tehdään pienissä osissa lisäävän mallin mukaisesti. Osittelu jakautuu ensimmäisessä syklissä seuraavasti.

Ryhmä Raiteilla:

- käyttöliittymä: viikot 3–5
  - yleisilme sekä XHTML/CSS-ulkoasun rakenne
- yleisarkkitehtuuri: viikko 4
- rajapinnat: viikot 4–5
- tietokanta: viikot 4–5
- rengastajan WWW-järjestelmä: 5–6
  - Rails: mallit, ohjaimet ja näkymät

Ryhmä Jokeri kehittää tuotantokäytössä olevaa Java-pohjaista järjestelmää. Olemassa olevaan järjestelmään tehdään lukuisia pieniä parannuksia. Siksi Jokeri-ryhmä tekee ensimmäisessä syklissä suunnittelua rinnakkain vaatimusmäärittelyn kanssa lisäävän mallin mukaisesti. Jokeriryhmän suunnittelun yksityiskohdista tiedetään tarkemmin vasta vaatimusmäärittelyn yhteydessä. Merkittävä osa suoritettavasta kehitystyöstä koostuu pienikokoisista yksittäisistä tehtävistä, joten suunnittelu on jaettu erittäin karkealla tasolla.

Ryhmä Jokeri:

- Käyttöliittymä: viikot 4–7
- Tietokanta: viikot 4–7
- Rajapinnat (mahdollinen): viikot 4–7

7.1.3 Toteutuksen ja yksikkötestauksen ositus

Toteutus jakautuu osavaiheiltaan Rails-järjestelmän ja web-käyttöliittymän toteutukseen. Jokaisen osajärjestelmän valmistuttua täytyy kyseiselle komponentille kirjoittaa yksikkötestit ja funktionaaliset testitapaukset. Toteutukseen varattu aika ensimmäisessä syklissä on 3 viikkoa. Mikäli toteutus pitkittyy, periodien väliviikolla (aikataulun 8. viikko) voidaan tarvittaessa kuroa aikataulua umpeen.

Ryhmä Raiteilla:

- Osajärjestelmien toteutus: viikot 5–7
  - data2XML
  - salasanapalautin
  - luotujen raporttien hallinta niin rengastajan kuin toimiston päässä
  - Rengas99-datatiedostojen hallinta

  - rajapinta Ringcheck-ohjelmaan
- käyttöliittymän HTML ja CSS: viikot 6–7
- ensimmäisen syklin täysprototyypin viimeistely ja testien täydentäminen: viikko 9

Ryhmä Jokeri:

- pienet yksittäiset kehitystyöt: viikot 5–7
- uusien osajärjestelmien toteutus: vasta 2. iteraatiossa
  - aikataulutusta tarkennetaan, mikäli uusi osajärjestelmä osoittautuukin Jokeri-ryhmän vaatimusten kartoitusvaiheessa aiemmin odotettua suuremmaksi.

7.1.4 Integrointi- ja järjestelmätestaus

Integrointi- ja järjestelmätestaus ovat osittain päällekäin toteutusvaiheen kanssa. Raiteilla-ryhmä määrittelee ja toteuttaa järjestelmäympäristöön soveltuvan korkean tason ”QA-pienkielen”, jonka avulla kirjoitetaan integrointitestaukselle Ruby on Rails -testitapaukset.

Ryhmä Jokeri aloittaa varsinaisen toteutustyön vasta toisessa syklissä, joten heidän osaltansa järjestelmätestausta ei vielä tässä yhteydessä tehdä.

Ryhmä Raitella:

- järjestelmätestauksen testitapausten luonti: viikot 4–5
- integrointitestaus: viikot 6–7
- järjestelmätestaus: viikko 9.

Ryhmä Jokeri:

- Järjestelmä- ja integraatiotestaus vasta 2. syklissä
  - Aikataulutusta tarkennetaan, mikäli uusi osajärjestelmä osoittautuukin Jokeri-ryhmän vaatimusten kartoitusvaiheessa aiemmin odotettua suuremmaksi.

7.2 Ajoittuminen toisessa syklissä

Toisessa syklissä Raiteilla-ryhmä suunnittelee ja toteuttaa ensimmäisessä syklissä tuotettuun järjestelmään hallintatoiminnot (ns. admin-paneelin). Lisäksi järjestelmän käyttöliittymää kehitetään eteenpäin lisäämällä siihen AJAX-toiminnallisuutta. Lähtökohtaisesti järjestlemää tulee kuitenkin voida käyttää, vaikka selaimessa ei olisi lainkaan JavaScript-tukea.

Jokeri-ryhmä aloittaa mahdollisten uusien osajärjestelmien toteutuksen toisessa syklissä. Samoin Jokeri-ryhmän vaatimusmäärittelydokumentille suoritetaan tarkastustilaisuus toisen syklin ensimmäisellä viikolla.

Projektisuunnitelmaa tarkennetaan toisen syklin alussa.

7.3 Aikataulu

Aikataulu on liitteenä.

8 Seuranta- ja raportointimenetelmät

Projektiryhmä pitää seurantakokouksen lähtökohtaisesti joka maanantai koko kevään ajan. Seurantakokouksessa tarkistetaan, että projektissa on pysytty aikataulussa ja seurattu projektisuunnitelmaa. Seurantakokouksen pöytäkirjat talletetaan projektin dokumentteina.

Projektiryhmä pitää tarkastuksen kummankin ryhmän vaatimusmäärittely- sekä suunnitteludokumenteille. Tarkastuksissa seurataan Sommervillen luvun 22.2. ja Ohjelmistotuotantokurssin luentokalvojen tarkastusmallia.

Projektiryhmä raportoi suullisesti viikoittain seurantakokouksen yhteydessä laadunvalvontaryhmälle (eli projektin ohjaajalle) projektin edistymisestä. Samassa yhteydessä ryhmän kukin jäsen kertoo ryhmän muille jäsenille, mitä hän on edellisen viikon aikana tehnyt ja mitä tulee alkavalla viikolla tekemään.

Liitteet

- Raiteilla-ryhmän projektiaikataulu.
- Jokeri-ryhmän projektiaikataulu.

Liite 1: Raiteilla-ryhmän projektiaikataulu

AIKATAULUSUUNNITELMA Ryhmä: Raiteilla **1.2.2007 16:56**

1. iteraatio		**TODO: Gantt-kaavio viikko**	TR: **sisäinen deadline**	**esittely / demo**	FTR: **katselmointi**
PS1	Projektisuunnitelma	1-3	2.2.2007		
VD1-Rails	Vaatimusmäärittely (Rails)	2-3	5.2.2007		8.2.2007 (+ asiakas)
	käyttäjänäkymän kälin eka proto (Rails)	3	1.2.2007	6.2.2007 (+ asiakas)	
SD1-Rails	Suunnittelu (Rails)	4-6	26.2.2007		
	Toteutus, prototyypit (Rails)	5-7			
TS1-Rails	Testaussuunnitelma (Rails)	6	26.2.2007		
	Järjestelmä- ja integrointitestaus (Rails)	6-7			
	Rails: ensimmäinen täysproto valmis	7	2.3.2007		
TAUKO	Tauko: periodien väliviikko	8			
	Projektin esittely asiakkaalle	9		12.3.2007 (+ asiakas)	
2. iteraatio					
PS2	Proj.suunnitelman päivitys	9			
	Järjestelmä- ja integrointitestaus (Rails)	9			
VD2-Rails	Vaatimusmäärittely (Rails)	9-10			
SD2-Rails	Suunnittelu (Rails)	10-12			
	Toteutus, prototyypit (Rails)	12-13			
TS2-Rails	testaussuunnitelman päivitys (Rails)	12-13			
	Järjestelmä- ja integrointitestaus (Rails)	13-14			
	ohjelmiston RC1 valmis, demo as.:lle	14			
	projektin viimeistely ja luovutus	15	6.5.2007		

Liite 2: Jokeri-ryhmän projektiaikataulu

AIKATAULUSUUNNITELMA Ryhmä: Jokeri 1.2.2007 16:56

1. iteraatio		**viikko**	TR: **sisäinen deadline**	FTR: **katselmointi**
PS1	Projektisuunnitelma	1-3	2.2.2007	
VD1-Java	Vaatimusmäärittely (Java)	3-7		vkolla 7
	Käyttöliittymäprotot	4-7		
SD1-Java	Suunnittelu limittäin vaat.määr. kanssa	4-7		
	Toteutus	5-7		
TS1-Java	Testaussuunnitelma (Java)	7	26.2.2007	
TAUKO	Tauko: periodien väliviikko	8		
2. iteraatio				
PS2	Proj.suunnitelman päivitys	9		
	Toteutus	9-13		
SD2-Java	Suunnittelu	9-10		
TS2-Java	Testaussuunnitelman päivitys (Java)	12-13		
	Järjestelmä- ja integrointitestaus (Java)	13-14		
	Ohjelmiston RC1 valmis, demo asiakkaalle	14		
	Projektin viimeistely ja luovutus	15	6.5.2007	