Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ohjelmistotuotantoprojekti

Esimerkkituoteperhe

Projektisuunnitelma

22.04.2004

Ryhmä 6
Juha Andersson
Jarmo Kielosto
Leo Linnamaa
Jan Tilles
Joose Vettenranta

Versiohistoria

Versio	***Päivämäärä***	***Muutokset***
1.4	22.04.2004	Aikataulua ja tehtävänjakoa täsmennetty
1.3	18.03.2004	Vaatimusdokumentin jälkeen muutettu aikataulu luvussa 5.2 sekä lisätty FP-arvio
1.2	01.02.2004	Ryhmän palaverissä käsitellyt lisäykset ja muutokset
1.1	29.01.2004	Ohjaajan kommenttien pohjalta muutettu
1.0	28.01.2004	Ensimmäinen versio ryhmän arvioitavaksi

Sisältö

1. Johdanto

Projektin aikana toteutetaan yksinkertainen tuoteperhe Java-ohjelmointikielellä. Tuoteperhe sisältää kolme sovellusta, jotka toimivat esimerkkiaineistona RITA-projektissa (Environment for Testing Framework-based Software Product Families) analysoijaohjelmistolle. Projektissa tuotetaan myös tarvittava dokumentaatio sekä testit analysointia varten. Vaikka itse tuoteperhe on yksinkertainen, projektin haastavuus liittyy oikeaan arkkitehtuuriin.

2. Organisaatio

Projektin asiakas on Raine Kauppinen (raine.kauppinen@cs.helsinki.fi) RITA-projektista. Asiakkaan teknisenä avustajana toimii Antti Tevanlinna (antti.tevanlinna@cs.helsinki.fi). Kurssin vastuuhenkilöinä toimivat Juha Taina ja Turjo Tuohiniemi (ohtu@cs.helsinki.fi) sekä projektiryhmän ohjaajana Ilja Ponka (ilja.ponka@cs.helsinki.fi).

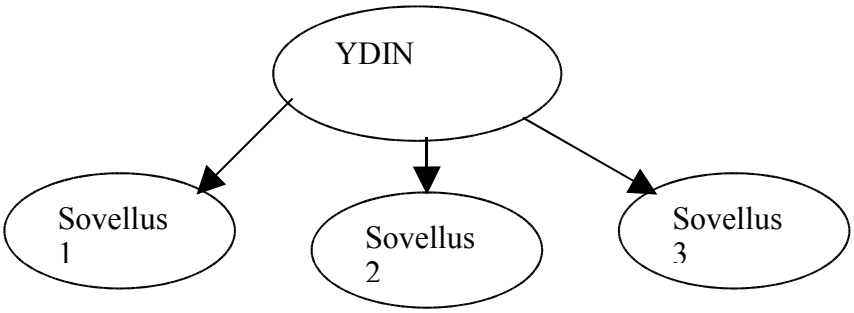
Projektiryhmän jäsenet ovat seuraavat:

Jäsen	**Tehtäväkuva**	**Sähköposti**	**Puhelinnumero**
Juha Andersson	Dokumenttivastaava	juha.andersson@edu.hel.fi	040 593 9903
Jarmo Kielosto	Testivastaava	jarmo.kielosto@cs.helsinki.fi	040 822 9662
Leo Linnamaa	Kehittäjä / sihteeri	leo.linnamaa@helsinki.fi	0400 463 268
Jan Tilles	Projektipäällikkö	tilles@mappi.helsinki.fi	040 837 7701
Joose Vettenranta	Ympäristövastaava	joose@iki.fi	044 561 0270

Jokaiselle on siis määritelty ensisijainen päävastuualue, mutta kaikki osallistuvat pääasiallisesti kaikkiin projektin vaiheisiin. Vastuualueet on jaettu projektin jäsenten toiveiden pohjalta. Projektipäällikkönä toimii Jan Tilles, joka vastaa tehtävien delegoinnista. Kokousten sihteerinä on Leo Linnamaa, joka vastaa myös ytimen kehittämisestä. Jarmo Kielosto tuntee JUnit-testiympäristöä ennestään, joten hänen vastuullaan on seurata, että testit tulevat asiallisesti toteutetuksi. Joose Vettenranta pitää yllä CVS-ympäristöä ja huolehtii sen varmuuskopioinnista toiselle palvelimelle sekä ylläpitää kurssin kotisivuja. Juha Andersson vastaa dokumenttien viimeistelystä.

3. Työn yleiskuvaus ja vaatimukset

Ohjelmistoperhe jakautuu neljään erilliseen osaan: Ensimmäinen osa on **perusydin** (core), joka toimii tuoteperheen sovellusten pohjana. Kolme muuta osaa ovat **sovelluksia**, jotka hyödyntävät ydintä. Ytimeen sijoitetaan perustoiminnot, kuten esimerkiksi pisteiden näyttäminen ja laskeminen.



Ohjelmiston ydin on tuoteperheen sovellusten keskeinen osa. Ytimessä on koottuna sovellusten yhteiset toiminnot mahdollisimman kattavasti. Rungon rakenteeseen liittyy erityisiä arkkitehtuurisia vaatimuksia, jotka asiakas määrittelee yhdessä projektiryhmän kanssa. Arkkitehtuuriset vaatimukset liittyvät asiakkaan kehittämään RITA-projektiin.

Sovellusten aihepiiriksi asiakas ehdotti harrastuspelien kirjanpito- ja pistelaskuohjelmia. Projektiryhmä päätyi seuraaviin peleihin:

- mökkitikka
- snooker
- keilailu

Vaatimuksen mukainen tuoteperhearkkitehtuuri on käyttää tuoteperheen ytimenä oliopohjaista **ohjelmistokehystä** (framework). Projektin aikana tuotettavat kolme sovellusta muodostavat siis tuoteperheen. Ytimeen kootaan sovellusten sisältämät yhteiset toiminnot sekä osia käyttöliittymästä.

Asiakas painottaa tuoteperheen ytimen arkkitehtuuria, jonka tulee olla toteutettu ”by the book”, mahdollisimman täsmällisesti ja siististi. Vaatimuksena on niin ikään, että ohjelmiston tulee olla testattavissa RITA-työkalun avulla.

Kaikkien tuoteperheen jäsenten tulee olla Java-kielellä toteutettuja graafisia ja ikkunoituja sovelluksia. Niiden on lisäksi erotuttava toisistaan 5-10 luokan osalta. Ajankäytön vuoksi sovellusten tulee olla yksinkertaisia, mutta silti niiden pitää sisältää riittävästi variaatiota analysointia varten. Niinpä sovellusten toteutusten vaikeustaso vaihtelee: yksi sovelluksista on hyvin yksinkertainen (mökkitikka), yksi hieman haastavampi (snooker) ja yksi monimutkaisempi (keilaus).

Testitapauksia suunniteltaessa vaatimuksena on luoda myös sellaisia testejä, jotka voidaan ajaa kaikissa tuoteperheen sovelluksissa. Testien suunnittelussa käytetään apuna RITA-työkalua. Testaus toteutetaan JUnit-työkalulla. Yksilö- ja integrointitestit erottaan lisäksi toisistaan.

Asiakkaan vaatimus on niin ikään, että ohjelmisto ei saa käyttää **java.lang.reflect**-paketin luokkia.

4. Kokoarvio

Projektin aihe on epätavallinen siinä mielessä, että vastaavia ohjelmistotuotantoprojekteja ei ole toteutettu aiemmin. Siksi projektin kokoarviota on hankala vielä arvioida. Ryhmällä on kokemusta pienistä Java-kielellä toteutetuista graafisista sovelluksista; esimerkiksi yksinkertainen tekstieditori, joka toiminnallisuudeltaan on todennäköisesti vähintään yhtä monimutkainen kuin tulevat sovellukset, vaatii noin 300 riviä koodia.

Karkea arvio voidaan esittää. Jos ydin koostuisi 1000 koodirivistä ja sovellukset käyttäisivät sitä tehokkaasti lisäten omia koodirivejä noin 300, olisi ohjelmiston alarajana pidettävä määrä noin 2000 riviä koodia. Vastaavasti jos ydin olisi 1500 riviä koodia ja sovellukset laajempia (esimerkiksi 750 riviä, 50% ytimestä), kokonaismäärä olisi suuruusluokkaa 3500 riviä.

Ohjelmiston kokoarviota (2000-3500 riviä) tarkennetaan sekä vaatimus- että suunnitteludokumentissa. Kun ydin on saatu koodattua, on sovellusten koko helpompi arvioida täsmällisesti. Vastaavasti jos tuoteperheeseen halutaan myöhemmin toteuttaa uusia sovelluksia, on niiden kokojen arviointi helpompaa, kun ytimen koko sekä siitä tehtyjen yksittäisten sovellusten laajuus on tiedossa.

Seuraavat lasketaan mukaan koodirivistöön (LOC-laskuun):

Mikä?	**Tuleeko mukaan?**
Class-määritteet	Kyllä
Konstruktori-määritteet	Kyllä
Metodi-määritteet	Kyllä
Muuttuja-määritteet	Kyllä
Kommentit	Ei
{ } –merkit	Kyllä
If – else	Kyllä
Case, while, yms	Kyllä
Tyhjät rivit	Ei
Try-catch	Kyllä

Toimintopisteiden (FP) arviointi on mahdollista nyt, kun toiminnallisuus on analysoitu määrittelydokumentissa. Jukka Paakin Ohjelmistotuotanto-kurssin luentomateriaalissa (http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento4.pdf) pistelaskuun esitetään seuraava kaavio:

	yksinkertaisia	***k**	**normaaleja**	***k**	**vaikeita**	***k**	**pisteet**
syötteitä		***3+**		***4+**		***6+**	
tulosteita		***4+**		***5+**		***7+**	
kyselyjä		***3+**		***4+**		***6+**	
tiedostoja		***7+**		***10+**		***15+**	
liittymiä		***5+**		***7+**		***10+**	
FP							

Syötteitä ohjelmistoperheessä on seuraavasti:

- pelaajamäärän valitseminen (mökkitikka ja keilailu)
- pelaajatietojen syöttäminen (kaikki kolme sovellusta)
- pisteiden lisääminen (kaikki kolme sovellusta)

Kaikkia syötteitä voi pitää yksinkertaisina, FP on 8 * 3 = 24 pistettä.

Tulosteita ohjelmistoperheessä on seuraavasti:

- loppuraportin katsominen (kaikki kolme sovellusta)
- top 10 -lista (snooker)
- loppuraporttien selaaminen (keilailu)

Loppuraportin katsominen on yksinkertainen toiminto, top 10 -listan sekä loppuraporttien selaaminen normaaleja toimintoja toteutukseltaan. FP on 3 * 3 + 2 * 4 = 17 pistettä.

Kyselyjä ohjelmistoperheessä ei tehdä, koska tietokantayhteyksiä ei toteuteta. **Tiedostoja** on kaksi: top 10 -lista snookerissa ja loppuraporttien lista keilailussa. Kumpaakin voi pitää yksinkertaisena, joten FP on 2 * 7 = 14 pistettä.

Liittymien osuus on hieman tulkinnanvarainen: ohjelmisto ei liity suoranaisesti mihinkään liittymään, mutta se tulee olla testattavissa RITA-työkalulla, mikä vaikuttaa suunnitteluun. Siten FP:hen voisi lisätä 5 pistettä. Oheisessa taulukossa on koottu FP-pisteet:

Toiminto	FP
Syötteet	24
Tulosteet	17
Kyselyt	0
Tiedostot	14
Liittymät	5
Yhteensä	**60**

Toimintopisteitä voidaan vielä hienosäätää laskentakaavalla

FP = N*(0.65+0.01*∑(Fi))

(http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento4.pdf).

Tässä N on toimintopisteiden summa ja Fi kompleksisuustekijä, joka voi saada arvot 0-5 seuraavasti:

0= ei koskaan

1= harvoin

2= toisinaan

3=keskimääräisesti

4=merkittävästi

5=oleellisesti

Seuraavassa kaaviossa on laskettu kompleksisuustekijöiden vaikutus:

Kompleksisuustekijä	Fi
1. Onko järjestelmä vikasietoinen? Tarvitaanko luotettavaa tietojen varmistus- ja palautusmenettelyä?	0
2. Tarvitaanko tietoliikenneominaisuuksia?	0
3. Onko hajautettua prosessinhallintaa?	0
4. Onko suorituskyky kriittinen elementti?	0
5. Käytetäänkö järjestelmää olemassaolevassa raskaassa käytössä olevassa koneympäristössä ?	2
6. Tarvitaanko interaktiivista tietojen syöttöä?	5
7. Täytyykö interaktiivinen tietojen syöttö synkronoida usealle näytölle tai operaatiolle?	0
8. Päivitetäänkö tiedostoja interaktiivisesti?	2
9. Ovatko syötteet, tulosteet, tiedostot tai kyselyt monimutkaisia?	1
10. Onko ohjelman toiminta monimutkaista?	1
11. Onko koodi tarkoitettu uudelleenkäytettäväksi ?	5
12. Ovatko ohjelmiston muunnokset ja asennus mukana suunnitelmassa ?	3
13. Onko ohjelmisto suunniteltu toimivaksi useina asennuksina eri organisaatioissa ?	1
14. Onko sovellus suunniteltava käyttäjäystävälliseksi ?	2
Yhteensä	**22**

Näin laskukaava tuottaa seuraavan arvon: FP = 60 * (0.65 + 0.01 * 22= 52,2 pistettä. FP-arviota voi siis pitää 52:na. Oliokielissä, joka Javakin on, arvioidaan yhden FP:n vastaavan 20-60 riviä koodia. Paakkisen (http://www.cs.helsinki.fi/u/paakki/ohtuk03-luento4.pdf) mukaan C++-kielessä FP-vastaavuus on 64 LOC/FP. Koska Javan syntaksi on hyvin samankaltainen C++:n kanssa mutta koska Javassa on automatisoitu esimerkiksi roskienkeruu (ja siihen liittyen olioiden tuhoaminen), voisi Javan LOC / FP-arvoa pitää pienempänä.

QSM:n (Quantitave Software Management) sivuilla esitetään toisaalta Javan keskimääräiseksi LOC/FP-arvoksi peräti 63:a ja alhaiseksi (low) arvoksi 53 (http://www.qsm.com/FPGearing.html). Projektin kokoarvioksi kannattaa siis ottaa vaihteluväli eikä tarkkaa arvoa. Alarajaksi tässä määritellään 55 ja ylärajaksi 60.

Näin ohjelmistoperheen koko FP-analyysin kautta arvioituna olisi sadan rivin tarkkuudella **2600 - 3100** riviä.

5. Aikataulu

5.1. Prosessimalli ja tuotettavat dokumentit

Projekti noudattaa työn osituksessa sovellettua **vesiputousmallia**. Vesiputousmalli on yleinen ja tunnettu lineaarinen malli, josta ryhmä toteuttaa **analyysi-, suunnittelu-, toteutus-, testaus- ja viimeistelyvaiheet**. Tärkeimpänä vaiheena pidetään analyysi- ja suunnitteluvaihetta, jotka on limitetty osin päällekkäin. Seuraava vaihe aloitetaan aina, kun edellinen on riittävän pitkällä.

Ryhmä tuottaa projektin aikana seuraavat dokumentit. Tehtävänjakoa on tarkennettu käyttöohjeen osalta 22.4.2004.

- Projektisuunnitelma, jonka vastuuhenkilö on Juha Andersson
- Vaatimusdokumentti, jonka vastuuhenkilöt ovat Jan Tilles ja Leo Linnamaa
- Suunnitteludokumentti, jonka osana on testaussuunnitelma, vastuuhenkilöinä Joose Vettenranta sekä Jarmo Kielosto
- Käyttöohje, jonka vastuuhenkilöinä ovat Jan Tilles (käyttäjän näkökulma) ja Leo Linnamaa (sovelluskehittäjän näkökulma)
- Testausyhteenveto, jonka vastuuhenkilö on Jarmo Kielosto
- Loppuraportti, josta koko ryhmä vastaa

Projektiryhmä päätti käyttää dokumentoinnissa MS Wordin doc-formaattia. Asiakkaalle dokumentit lähetetään PDF-muodossa. Kaikki dokumentit sekä syntyvät lähdekoodit talletetaan ryhmähakemistoon, jonka ylläpitämisessä käytetään apuna CVS:ää. Lisäksi ryhmä pitää kirjaa työtunneistaan sekä kokouksistaan. Kaikki dokumentit liitetään projektikansioon, joka palautetaan projektin päätyttyä.

Dokumenttien kieli on suomi. Ylläpidettävyyden vuoksi Java-koodien luokkien, metodien sekä muuttujien nimet päätettiin kirjoittaa englanniksi: näin koodia voi tarvittaessa tarkastella helpommin jatkossa kuka hyvänsä. Ohjelmointityylissä noudatetaan seuraavia konventioita:

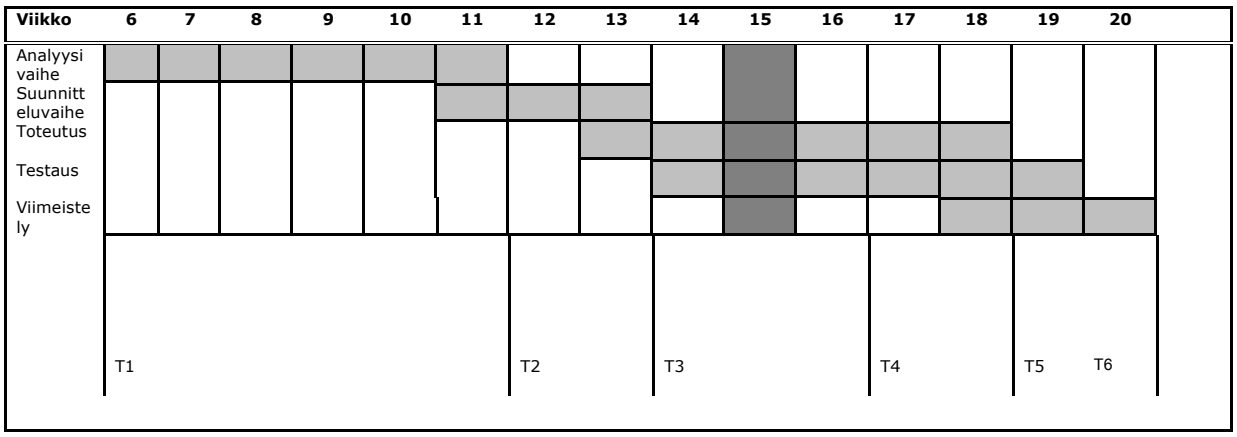
Nimeäminen	Kaikki muuttujat, luokat, metodit yms nimetään englanniksi. Monisanaiset kuvaukset kirjoitetaan yhteen siten, että ensimmäinen sana alkaa pienellä ja sitä seuraavat isolla, esim. isValidString
Kommentit	Käytetään JavaDoc –tyylistä dokumentointi
{	Merkitään alkavaksi edellisen rivin lopusta, esim: if (ehto) { tai public static void metodi () {

}	Merkitään alkavaksi samalle sisennykselle kuin rivi jolta se alkaa. Loogiset seuraukset jatkuvat samalle riville, kuten “} else {“ tai “} catch (Exception e) {“
{ }	Käytetään aina selvyyden vuoksi

5.2. Projektin eteneminen

Projektin eteneminen on kuvattu seuraavassa GANTT-kaaviossa, joka on päivitetty 22.4.2004. Kuvaus alkaa viikosta 6, jolloin projekti eteni suunnitteluvaiheeseen.

aikataulu



T1:
Projektisuunnite
lma valmis
T2:
Määrittelydokumentti
jäädytetty 12.3.2004
T3: Suunnitteludokumentti/testaussuunnitelma
valmis
T4:
Toteutusvaihe
päättyy ke
28.4.
T5: Käyttöohje ja testausyhtee-
veto valmis pe 7.5.
T6: Deadline
loppuraportin palautus
10.5.

Seuraavassa kaaviossa on kuvattu eri vaiheiden minimi- ja maksimikestot, viimeiset päivät, jolloin kunkin vaiheen tulee olla valmis sekä eri vaiheisiin käytettävät tuntimäärät:

Projektin vaiheet	Kesto min. vko	Kesto max. vko	Valmis viimeistään	Katselmoinnit	Tunnit min	Tunnit max
Analyysivaihe	3	5	**Sunnuntai 7.3**	Vko 8. tai 9.	255	425
Suunnitteluvaihe	2	3	**Sunnuntai 28.3**	Vko 15 alussa	170	340
Toteutus	2	3	**keskiviikko28.4.**	Vko 18lopussa	170	255
Testaus	4	5	**Perjanntai 7.5.**	Vko 19 lopussa	255	340
Viimeistely	2	3	**Maanantai 10.5**		170	255

Tuntiarvio

Perustuu siihen että jokainen projektin jäsen tekee noin 17 tuntia työtä per viikko.

Projektin kesto on 14 viikkoa. Viikko 15 on mahdollisesti lyhennetty viikko pääsiäisestä johtuen.

Projektin vaatima minimituntimäärä: 1020 tuntia

Projektin vaatima maksimituntimäärä: 1500 tuntia

Minimituntimäärä edellyttäisi, että kaikki työvaiheet sujuisivat minimiajoissaan. Maksimimäärä on puolestaan arvioitu siten, että valtaosasta vaiheita ylitetään minimiaika; 300 tunnin henkilökohtaista määrää voi pitää kattona (15 x 20 h/vko). Testausjaksoon on varattu pitkä aika, jotta voidaan varmistaa, että ohjelmisto itsessään toimii sekä se, että ohjelmisto toimii RITA-työkalun kanssa.

Asiakas ei ole tavattavissa viikolla 14. Sen vuoksi suunnitteludokumentti katselmoidaan todennäköisesti viikon 15 alussa. Asiakkaaseen ollaan kuitenkin sitä ennen yhteydessä, jotta toteutusvaihe voidaan aloittaa viikolla 14. Tämä edellyttää, että asiakas hyväksyy ryhmän esittämät luokkakaaviot, joiden pohjalta koodaaminen voidaan toteuttaa.

Projektin on tarkoitus olla valmis viimeistään 10.5. Laitoksen antama ehdoton takaraja on 31.5. Mahdollinen yhteinen demotilaisuus järjestetään 6.5.2004. Tilaisuudessa jokainen projektiryhmä esittelee noin 20 minuutin ajan omaa tuotettaan demonstroimalla sen toimintaa. Projektiryhmälle järjestetään palautetilaisuus, jossa ohjaaja, ohjelmistotuotantoprojektien vastuuhenkilö ja asiakkaat arvioivat projektin onnistumista.

6. Työskentelytavat

Projektin jokaiselle jäsenelle on jaettu vastuualueet, jotka on lueteltu luvussa 2. Alussa ryhmä pyrkii tekemään mahdollisimman paljon yhteistyötä aikataulun sekä suunnittelun osalta. Koska RITA-työkalu on erittäin tärkeä osa projektia, vaikuttaa se myös suunnitteluun. Siksi testivastaava tutustuu työkaluun heti projektin alussa.

Ryhmä kokoontuu säännöllisesti seuraavasti:

- ma 16-18 luokassa C455 ja
- to 16-18 luokassa C475.

Seurantakokoukset pidetään viikosta 7 alkaen joka toinen torstai kello, paitsi viikon 15 kokous maanantaina 5.4.

Alkuvaiheessa ryhmä työskentelee pääsääntöisesti yhdessä (yhteisiä palavereja sekä yhteisesti

sovittuja itsenäisesti tehtäviä osia), mutta ohjelmointivaiheessa tehtäviä ositetaan. Ohjelmointityö tehdään pääsääntöisesti itsenäisesti, mutta sovelluksen ydin mietitään yhdessä niin pitkälle, että ytimen ohjelmoimisen osittaminen on mahdollista. Vaikkakin analyysi- ja suunnitteluvaihe tehdään pääosin yhdessä, niin sitä mukaan kun siitä saadaan eroteltua osia, joita voi toteuttaa myös yksinään, niitä aletaan toteuttaa itsenäisesti. Projektiryhmä toteuttaa kuitenkin sisäistä iteraatiota siten, että dokumenteista ja ohjelmakoodeista lähetetään jokaiselle kopiot, joista toiset antavat palautetta. Asiakkaan kanssa suunnitteluvaiheessa pyritään niin ikään iteratiiviseen työskentelyyn, kunnes suunnitteludokumentti on saatu jäädytettyä.

Ohjelmointi etenee siten, että ohjelmiston ydin toteutetaan ryhmätyönä ensin niin valmiiksi, että sovelluksia voidaan alkaa toteuttaa. Toteutusten vastuuhenkilöt ovat seuraavat: mökkitikka Juha Andersson, snooker Joose Vettenranta, keilailu Leo Linnamaa. Vastuutusta on tarkennettu 19.4.2004 palaverissa (jolloin mökkitikkasovellus oli valmis) siten, että Juha Andersson vastaa sekä snookerin että keilailun käyttöliittymän koodaamisesta ja Joose Vettenranta ja Leo Linnamaa pelilogiikasta. Jan Tilles puolestaan vastaa suunnitteludokumentin viimeistelystä.

Projektiryhmä pitää kirjaa työtunneistaan ja raportoi niistä projektipäällikölle, joka ilmoittaa ne kurssin vastuuhenkilöille laitoksen ilmoittamina määräaikoina (2.2.2004, 23.2.2004, 8.3.2004, 22.3.2004, 5.4.2004, 26.4.2004, 10.5.2004).

7. Riskianalyysi

Projektiryhmä pyrkii varautumaan riskeihin jo ennakolta siltä osin, kuin se on mahdollista. Riskien todennäköisyyttä on arvioitu asteikolla **pieni, mahdollinen, suuri**.

7.1. Projektiryhmään liittyvät riskit

Riski	**Ennaltaehkäisy**	**Hallinta**	**Todennäköisyys**
Henkilö sairastuu.	Sairauksia on mahdoton ennakoida. Järkevä suunnittelu auttaa työn organisoimiseen.	Mikäli poissaolo kestää pidempään, jaetaan tehtäviä muille.	mahdollinen
Henkilö keskeyttää kurssin.	Tehtävien tasainen jako sekä yleinen kannustava ilmapiiri.	Kriisipalaveri. Sovitaan, ovatko tehtävät jaettavissa vai muutetaanko tavoitteita.	pieni
Henkilön ohjelmointitaidot eivät ole riittävät.	Järkevä tehtävien jakaminen.	Ryhmän muut jäsenet opastavat.	pieni

7.2. Projektin hallintaan liittyvät riskit

Riski	**Ennaltaehkäisy**	**Hallinta**	**Todennäköisyys**
Aikataulu pettää.	Ryhmä seuraa ja huolehtii aikataulussa pysymisestä sekä konsultoi ohjaajaa aikataulun realistisuudesta.	Palaveri asiakkaan ja ohjaajan kanssa, miten toimia.	Mahdollinen, koska aihe on vieras ja arkkitehtuurin suunnittelu on vaikeaa.

7.3. Tekniikkaan liittyvät riskit

Riski	**Ennaltaehkäisy**	**Hallinta**	**Todennäköisyys**
Työvälineitä ei hallita.	RITA-työkalu on haaste. Testivastaava tutustuu siihen hyvissä ajoin.	Konsultoidaan asiakasta ongelmakohdissa.	Mahdollinen, koska työkalu on tuntematon.
Tiedot saavuttamattomissa.	Ympäristövastaava ottaa päivittäin varmuuskopiot.	Varmuuskopioiden palautus.	Pieni, koska kopiopalvelin eri paikassa ja laitoksella omat varmuuskopiot.

7.4. Tuotteeseen ja asiakkaaseen liittyvät riskit

Riski	**Ennaltaehkäisy**	**Hallinta**	**Todennäköisyys**
Ohjelmiston ydin liian suppea tai laaja.	Huolellinen suunnittelu.	Neuvotellaan asiakkaan kanssa.	Suuri, koska tehtävä on haastava.
Sovellukset eivät eroa riittävästi toisistaan.	Huolellinen suunnittelu.	Neuvotellaan asiakkaan kanssa.	Pieni
Vaatimukset muuttuvat kesken.	Asiakkaan asiantuntemus on hyvä. Vaatimukset lienevät selkeät.	Neuvotellaan asiakkaan kanssa.	pieni