# Projektisuunnitelma

## Ohjelmistotuotantoprojekti Todistustoimitin

Jaakko Nenonen, Teppo Kankaanpää, Aki Nyrhinen, Visa Röyskö, Juha-Matti Tapio

7. helmikuuta 2003
Helsingin yliopisto
Tietojenkäsittelytieteen laitos

# Sisältö



## Liitteet

1. GANTT-kaavio

## Historia

| Päiväys | Versio | Kuvaus | Tekijät |
|---|---|---|---|
| 27.1.2003 | 0.1 | Esiversio | Kaikki |
| 28.1.2003 | 0.2 | Esiversio | Kaikki |
| 30.1.2003 | 0.3 | Luonnos | Kaikki |
| 6.2.2003 | 1.0 | Hyväksytty | jmtapio |

# 1 Johdanto

Tämä dokumentti on ohjelmistotuotantoprojekti-kurssin todistustoimitin-ryhmän projektisuunnitelma. Se sisältää yleiskuvauksen projektista, aikataulun, projektin työnjaon ja riskianalyysin. Projektin tarkoituksena on tuottaa ohjelmisto todistusten helppoon laadintaan tietokoneella käyttäen erilaisia sääntökokoelmia.

# 2 Yleiskuvaus

Viime syksynä luennoitiin laitoksella kurssi Proof Theory and Proof Search, jossa käsiteltiin luonnollisen logiikan tapaista formalismia, jonka etuina tähän nähden on se, että todistamisprosessi vaatii vähemmän intuitiivista päättelyä. Lauseen todistaminen tapahtuu ”käänteisessä järjestyksessä”, lähtien lopputuloksesta ja jakaen sitä yhä pienempiin osiin sääntöjä käyttäen kunnes päädytään aksioomiin tai erityisiin kohtiin, joissa ollaan päädytty ”identtisesti tosiin” lauseisiin, jolloin todistus on valmis.

Aiemmin kurssin käytössä on ollut vain tekstipohjaisia ohjelmia, joilla todistaminen on melko sekavaa ja vaikeaa. Projektin tarkoituksena on tuottaa graafinen helppokäyttöinen ohjelma, jolla todistuksia voidaan laatia mukaillen sitä, millä tavalla todistukset kirjoitettaisiin paperille.

## 2.1 Projektin osanottajat

| Nimi | Asema | E-mail |
|---|---|---|
| Marja Huovinen | ohjaaja | mphuovin@cs.helsinki.fi |
| Teppo Kankaanpää | opiskelija | teppo.kankaanpaa@helsinki.fi |
| Aki Nyrhinen | opiskelija | aki.nyrhinen@cs.helsinki.fi |
| Jaakko Nenonen | opiskelija | jaakko.nenonen@cs.helsinki.fi |
| Visa Röyskö | opiskelija | visa.roysko@cs.helsinki.fi |
| Juha-Matti Tapio | opiskelija | juha-matti.tapio@cs.helsinki.fi |
| Raul Hakli | asiakas | raul.hakli@cs.helsinki.fi |
| Matti Nykänen | asiakas | matti.nykanen@cs.helsinki.fi |
| Pietu Pohjalainen | vastuuhenkilö | ohtu@cs.helsinki.fi |

Projektissa mukanaolevat viisi opiskelijaa (projektiryhmä) toteuttavat työn asiakkaan asettamien vaatimusten mukaisesti. Asiakkaat ovat tietojenkäsittelytieteen laitoksen henkilökuntaa, jotka ovat mukana kurssin järjestämisessä. Asiakkaan yhteyshenkilö on pääasiassa Matti Nykänen. Projektin ohjaaja seuraa ja auttaa projektin etenemisessä. Ohjelmistotuotantoprojektin vastuuhenkilö toimii projektin vastuuhenkilönä ohjelmistotuotantoprojekti-kurssin puolella.

## 2.2 Projektiryhmän yhteydenpito

Projektiiryhmä kokoontuu pääsääntöisesti kahdesti viikossa, tiistaisin 16-18 ja perjantaisin 14-16 tietojenkäsittelytieteen laitoksen salissa B451 lukuunottamatta perjantaita 28.3, jota ennen sovitaan uusi paikka. Myös ohjaaja on paikalla kokouksissa.

Projektilla on postituslista, jonka osoite on `ohtuk03-toto-list@cs.Helsinki.FI`. Ryhmän jäsenet ja ohjaaja ovat listalla. Projektilla on myös kotisivu osoitteessa `http://www.cs.helsinki.fi/group/toto/`. Kotisivulle tulevat ainakin kokouspöytäkirjat ja hyväksytyt välivaiheiden dokumentit. Lisäksi käytössä on huoneessa D328 oleva kaappi numero 17, jossa olevaan mappiin varastoidaan dokumentit hyväksytyssä vaiheessa, ja projektin päätyttyä myös muokatut versiot, jos muutoksia on tehty.

# 3 Ohjelmisto

Projektin tarkoituksena on rakentaa kaavamuokkain joka toteuttaa ainakin kaksi kurssilla esitettyä sääntökokoelmaa, klassisen ja intuitionistisen ilman niinsanottua cut-sääntöä ja kvanttoreita. Sääntökokoelmissa on listattu joukko tapoja joilla kaavaa voidaan muokata. Kuhunkin kaavaan voidaan soveltaa useampaa sääntöä, käyttäjän tehtävänä on valita mitä. Ohjelman tulisi näyttää ruudulla kaava, josta klikkaamalla käyttäjä voi valita, mihin suuntaan todistus etenee seuraavaksi. Valinnan jälkeen ohjelma laskee todistuksen seuraavan askeleen, josta jälleen voidaan jatkaa todistamista.

Tärkeintä ohjelmassa (tietenkin sen lisäksi että se toimii oikein) on selkeä käyttöliittymä, koska ohjelmaa tullaan käyttämään opetuksen apuna kun kurssia luennoidaan jatkossa. Yhdellä hiiren näpäytyksellä voisi valita kaavasta konnektiivin, jota puretaan auki seuraavaksi. Koska todistuksesta tulee puumainen ja pidemmissä lauseissa varsin laaja, todistuksen näyttäminen ruudulla tulisi olla myös hyvin suunniteltu.

Ohjelmaan on hyvä sisällyttää myös jonkinlainen undo- ja redo-mahdollisuus, koska todistaminen on silti, vaikka näitä sääntöjä käyttäisikin, eri todistuspolkujen kokeilemista. Käyttäjä voi peruuttaa todistusaskeliaan ja valita toisia, tai palata aikaisempaan todistukseen jos uusi tie näyttää huonommalta.

Ohjelman olisi mahdollistettava uusien sääntökokoelmien käyttäminen suhteellisen pienellä ohjelmointivaivalla. Erityisesti niinsanottu cut-sääntö ja kvanttorit vaativat käyttäjältä muutakin syötettä kuin säännön valinnan. Ohjelman tulisi ottaa tämä huomioon niin, että näiden ominaisuuksien toteuttaminen olisi suhteellisen helppoa (ts. ei vaatisi koko ohjelman uudelleenkirjoittamista).

## 3.1 Toteutuksesta

Ohjelmisto koodataan Javalla. Sääntökokoelmien tallennus ja tulkkaus saatetaan tehdä LISPillä, Prologilla tai muulla symbolisen ohjelmoinnin kiellellä, jos se näyttää sopivan toteutustavaksi paremmin kuin puhdas Java. Todennäköisesti tällöin käytettäisiin Javan päälle rakennettua tulkkia eikä ulkoista ohjelmaa. Edellisen takia käyttöympäristöksi käy sekä Windows että Linux. Ohjelma ei saa olla kovin raskas, jotta se toimisi keskitason kotikoneella ja laitoksen mikroluokissa nopeahkosti, koska sen tulee olla tavallisen opiskelijan käytettävissä. Dokumentaatio tehdään LaTeX-tiedostoina ja Javadoc:na. Versionhallintaan käytetään CVS:ää. Kaavioiden piirtämiseen käytettyä ohjelmaa ei ole vielä päätetty, sopivan ohjelman selvittäminen on eräs projektin osatehtävä.

## 3.2 Ohjelmiston jakautuminen

Alla on kokoarvion pohjaksi eräs ehdotus sille, miten ohjelma voisi jakautua komponentteihin:

Ohjelmisto jakautuu pääosin toiminnalliseen osaan, tietorakenneosaan ja käyttöliittymäosaan, jossa ensimmäisen tehtävänä on toteuttaa haluttu kalkyyli, toisen tehtävänä tallentaa lasketut lausekkeet johonkin helposti käsiteltävään muotoon ja toteuttaa undo-toiminto, ja viimeisen tehtävänä näyttää ne.

Tietorakenneosa koostuu pääasiassa puurakenteesta, johon on tallennettu jokainen lauseke johon käyttäjä on todistusta vienyt, niin että yksi puun solmu sisältää lausekkeen, sen isäsolmu on lauseke josta se on johdettu, ja lapsisolmuina ovat kaikki ne lausekkeet jotka käyttäjä on todistuksen aikana tästä lausekkeesta johtanut.

Käyttöliittymä jakautuu esimerkiksi puumaiseksi rakenteeksi jossa yhden solmun sisältönä on siihen kuuluvan ikkunan sisältö ja linkit muihin ikkunaan mahdollisesti sidoksissa oleviin ikkunoihin. Ikkunaan on rakennettu jokin tapa löytää se lauseke jota käyttäjä on klikannut ikkunassa, ja klikattu lausekkeen osa. Näiden avulla voidaan löytää puurakenteesta lauseke, jota seuraavaksi on muokattava, ja se, miten sitä on muokattava. Jokin toinen osa hoitaisi tämän ikkunarakenteen sijoittelun jollain tavalla.

Toiminnallinen puoli voisi jakautua rajapintaan käyttöliittymän kanssa ja itse säännöt toteuttavaan osaan joka voi sitten olla tehty vaikka LISP-kieltä tai vastaavaa apuna käyttäen.

## 3.3 Ohjelmiston koko

Ohjelmiston koon arvioiminen tässä vaiheessa projektia on erityisen hankalaa, etenkin kun projektissa on osia joiden toteuttamisesta ja suunnittelusta yhdelläkään ryhmän jäsenellä ei ole kokemusta. Tämän takia seuraavaan arvioon tulee suhtautua vähintään skeptisesti.

Kalkyyliosan koko riippuu paljon tehdystä valinnasta: valitaanko kovakoodattu ratkaisu, vai tehdäänkö pieni ohjelmointikieli kalkyylien kuvaamiseksi, jota kalkyyliosa pystyy tulkitsemaan. Eroa kuitenkin tasoittaa se, että valmiin tuotteen täytyy osata ainakin kaksi kalkyyliä. Ensimmäisen vaihtoehdon kokokoarvio on noin 300 riviä kalkyyliä kohti, eli yhteensä 600 riviä, jälkimmäisessä vaihtoehdossa enemmän, noin 1000-1500 riviä. Jos päädytään hybridiratkaisuun jossa käytetään jonkin kolmannen osapuolen tarjoamaa ohjelmointikieltä, ja kuvataan kalkyylit kyseisellä kielellä, kalkyylikohtainen rivimäärä voi vähentyä tuntuvastikin. Kolmannen osapuolen ohjelmointikielen kyky manipuloida todistusrakennetta täytyy kuitenkin toteuttaa, ja sen monimutkaisuutta on mahdotonta arvioida ohjelmointikielen toteutuksellisia yksityiskohtia tuntematta. Paremman tiedon puutteessa on parasta arvioida kalkyylitoteutuksen rivimäärä noin 1500 riviin.

Tietorakenneosa on ohjelman kolmijaon pienin komponentti. Se sisältää pääasiassa rakenteiden kuvaukset ja yksinkertaiset operaatiot joilla tietorakenteita käsitellään. Siihen kuuluu myös kaavojen jäsennys merkkijonoista ja tallennus merkkijonoiksi. Vastaavat operaatiot toteutetaan myös kaavaa korkeamman tason kokonaisuuksille: sekvenssille ja todistukselle. Myös undo-ominaisuus kuuluu läheisesti tietorakenteen piiriin. Kaavajäsentäjä toteutuu sadalla rivillä, ja korkeamman tason rakenteet vähemmällä. Jäsentäjät vaativat siis noin 200 riviä. Vastaavat tallennusoperaatiot ovat hyvin yksinkertaisia, ja niistä selvitään yhteensä 50 rivillä. Rakenteiden kuvaukseen ja operaatioihin kuluu 200 riviä, ja undo-toteutukseen 300. Yhteensä tietorakennetoteutuksen rivimääräksi tulee 750.

Käyttöliittymäosa on toistaiseksi osista tuntemattomin, ja siksi sille on myös vaikeinta esittää kokoarviota. Käyttöliittymä päädytään rakentamaan mahdollisimman matalalta tasolta asti itse, sillä kirjaston tarjoamat komponentit sopivat projektin tarkoitusperiin huonosti. Tämä tarkoittaa sitä, että jokaista tietorakenteiden luokkaa kohti täytyy olla kohtuullisesti käyttöliittymäkoodia tuottamassa näkymän kyseisen luokan ilmentymästä, ja vastaavasti koodia joka pystyy yhdistämään ruudulla olevan koordinaatin tiettyyn tietorakenteiden olioon. Näihin kuluvan yhteensä noin 1000 riviä (hattu). Lisäksi käyttöliittymään täytyy ohjelmoida toteutus jokaiselle ohjelman todelliselle toiminnolle, kuten todistuspuiden (osien) tallennukselle, lataamiselle, tulostamiselle, jne. Näihin arvioin kuluvan toiset 1000 riviä. Yhteensä käyttöliittymän kooksi tulee siis 2000 riviä.

Koko ohjelmiston kokoarvio on siis 1500 riviä (kalkyylit) + 750 riviä (tietorakenteet) + 2000 riviä (käyttöliittymä) = 4250 riviä.

# 4 Toimintasuunnitelma

## 4.1 Prosessimalli

Projektissa käytetään perusmallina vesiputousmallia, joka koostuu siis perättäin etenevistä vaatimusmäärittely-, suunnittelu-, toteutus- ja testausvaiheista. Malliin on kuitenkin otettu mukaan extreme programming -mallista jatkuvan testauksen (testausrutiinit kirjoitetaan ennen kuin itse toiminnan toteuttava koodi kirjoitetaan) ja pariohjelmoinnin piirteet, jonka takia testausvaihe alkaa käytännössä samaan aikaan kuin toteutusvaihe. Toteutusvaiheen jälkeen on vielä aikaa pelkälle testaamiselle.

## 4.2 Työn jakautuminen

Projektiryhmä tuottaa itse ohjelmiston lisäksi seuraavat dokumentit:

- Projektisuunnitelma
- Vaatimusmäärittelydokumentti
- Suunnitteludokumentti
- Toteutusdokumentti
- Testaussuunnitelma
- Testausdokumentti
- Käyttöohje
- Ylläpitodokumentti

- Loppuraportti

Suuremmille dokumenteille on määritelty erilliset vastuuhenkilöt seuraavassa kappaleessa. Dokumenttien valmistumisen päivämäärät on mainittu alla Aikataulut-kappaleessa. Loppuraportti tehdään projektin päätyttyä. Projekti tuottaa myös työtuntilistat projektin etenemisen seurantaa varten, kokouspöytäkirjat ja mahdollisesti sisäisiä muistioita.

Ainakin projektisuunnitelma, vaatimusmäärittely ja käyttöohje hyväksytetään asiakkaalla ja projektisuunnitelma myös ohjelmistotuotantoprojektin vastuuhenkilöllä. Loppuihin dokumentteihin riittää projektiryhmän hyväksyntä.

Projektille ei ole määritelty tämän tarkempaa työnjakoa. Töitä jaetaan henkilöille, kun kokouksessa jokin osatyövaihe on jaettu edelleen sopivankokoisiin osiin (esimerkiksi dokumentin kappaleisiin tai ohjelmoitaviin moduuleihin) jotka voidaan jakaa tehtäväksi jollekin. Jokainen ryhmän jäsen osallistuu siis potentiaalisesti jokaisen dokumentin ja ohjelmakomponentin kirjoittamiseen.

## 4.3 Vastuualueet

- Projektipäällikkö: Jaakko Nenonen
- Raportointivastaava: Jaakko Nenonen
- Dokumenttivastaava: Juha-Matti Tapio
- Vaatimusmäärittely: Aki Nyrhinen
- Suunnittelu: Teppo Kankaanpää
- Toteutus: Jaakko Nenonen
- Testaus: Visa Röyskö
- Käyttöohje: Juha-Matti Tapio

Projektipäällikön tehtävänä on valvoa projektin kulkua ja toimia kokousten puheenjohtajana. Raportointivastaava raportoi projektin kulusta yliopistolle toimittamalla aikataulut projektin vastuuhenkilölle.

Dokumenttivastaava huolehtii dokumenttien yleisilmeen yhtenäisyydestä sekä oikolukee dokumentit.

Erillisillä työvaiheilla on omat vastuuhenkilönsä, jotka vastaavat kyseisen työvaiheen etenemisestä ja viime kädessä vaiheesta tuotetusta dokumentista, esimerkiksi sen yhtenäisyydestä ja rakenteesta. Koska projektissa käytetään extreme programmingin jatkuvaa testausta, jokainen komponenttia ohjelmoiva ryhmän jäsen ohjelmoi ennen sen ohjelmointia sille testausrutiinin. Testausvastaava laatii kuitenkin testauksen päälinjat ja pitää huolta testauksen kattavuudesta. Projektin lopussa on myös aikaa pelkälle testaamiselle, joten testauksen puutteita voidaan korjata vielä tässä vaiheessa. Käyttöohjeesta vastaava on vastuussa mm. käyttöohjedokumentin yhtenäisyydestä ja rakenteesta.

## 4.4 Aikataulu

Projektin kulku on esitetty liitteessä 1 olevassa Gantt-kaaviossa ja tärkeitä päivämääriä on koottu alla olevaan taulukkoon.

Projekti päättyy 13.5.

| Tapahtuma | Päivämäärä |
|---|---|
| Projektin ensimmäinen kokous | 17.1 |
| Projektisuunnitelma valmis | 28.1 |
| Määrittelydokumentti valmis | 11.2 |
| FTR-kokous suunnitteludokumentista | 11.3 |
| Suunnitteludokumentti valmis | 18.3 |
| Toteutus valmis | 15.4 |
| Käyttöohje valmis | 22.4 |
| Testaus valmis | 6.5 |
| Testausdokumentti valmis | 6.5 |
| Loppuraportti valmis | 13.5 |
| Projekti päättyy | 13.5 |

# 5 Riskianalyysi

Riskien analyysissä on kunkin esille tulleen mahdollisen riskin osalta mietitty miten siihen varaudutaan ennakolta, miten siihen reagoidaan sen tapahtumisen jälkeen, mikä sen vaikutus on projektille ja miten vakava se on. Menetelminä on käytetty sovellettuja RMMM- ja RiskIt-menetelmiä.

| | |
|---|---|
| Riski | Laitoksen levyjärjestelmässä tai mikroluokissa on katko. |
| Ennakointi | Tehdään varmuuskopioita esimerkiksi kotikoneille.<br>Jätetään aikatauluihin joustoa. |
| Reaktio | Käytetään kotikoneille tehtyjä varmuuskopioita ja<br>tehdään työt tarvittaessa kotona. |
| Vaikutus | Tiedonkulku hidastuu ja projekti voi viivästyä. |
| Vakavuus | Marginaalinen. |

| | |
|---|---|
| Riski | Jäljittämätön bugi käyttöliittymässä. |
| Ennakointi | Testataan huolellisesti ja kaiken aikaa. |
| Reaktio | Täydennetään yksikkötestausta ja suoritetaan koodin auditointi. |
| Vaikutus | Heikkolaatuinen lopputulos tai viivästys aikataulussa. |
| Vakavuus | Kriittinen. |

| | |
|---|---|
| Riski | Joku projektin jäsenistä keskeyttää tai sairastuu. |
| Ennakointi | Mahdollisesti varamiesmenettly. |
| Reaktio | Työtehtävät jaetaan uudelleen ja työmäärää tarpeen mukaan karsitaan. |
| Vaikutus | Projekti viivästyy tai lopputulosta karsitaan. |
| Vakavuus | Kriittinen. |
| | |
| Riski | Lopputulos ei vastaa sitä, mitä asiakas todella haluaa. |
| Ennakointi | Kommunikoidaan asiakkaan kanssa jatkuvasti. |
| Reaktio | Pyritään muokkaamaan tulosta mahdollisuuksien rajoissa. |
| Vaikutus | Projekti viivästyy tai lopputulos ei ole toivottu. |
| Vakavuus | Kriittinen. |
| | |
| Riski | Käyttöliittymän toteutus osoittautuu arveltua vaikeammaksi. |
| Ennakointi | Perehdytään aiheeseen huolella ja harjoitellaan. |
| Reaktio | Toimintoja karsitaan tai aikataulua venytetään. |
| Vaikutus | Huonompi lopputulos tai projektin viivästyminen. |
| Vakavuus | Kriittinen. |
| | |
| Riski | Työnjako on epäselvä tai huonosti rajoitettu. |
| Ennakointi | Jaetaan työt selkeästi. |
| Reaktio | Jaetaan työt selkeämmin. |
| Vaikutus | Jokin työ voi jäädä tekemättä ja joku työ saatetaan tehdä useasti. |
| Vakavuus | Marginaalinen. |
| | |
| Riski | Extreme Programming -työtavat havaitaan liian raskaiksi tai toimimattomiksi. |
| Ennakointi | Seurataan työtapojen toimivuutta. |
| Reaktio | Luovutaan kyseisistä tavoista ja tehdään projekti normaalisti loppuun. |
| Vaikutus | Toteutus saattaa viivästyä. |
| Vakavuus | Marginaalinen. |
| | |
| Riski | Ehdotettu käyttöliittymä havaitaan sekavaksi vasta toteutettuna. |
| Ennakointi | Käyttöliittymä toteutetaan aikaisessa vaiheessa jolloin sitä voi ehkä vielä miettiä/säätää uudelleen, tai käytetään demoja. |
| Reaktio | Suunnitellaan uudestaan tai parannetaan vanhan dokumentaatiota. |
| Vaikutus | Huonolaatuinen lopputulos ja/tai projektin viivästyminen. |
| Vakavuus | Marginaalinen. |

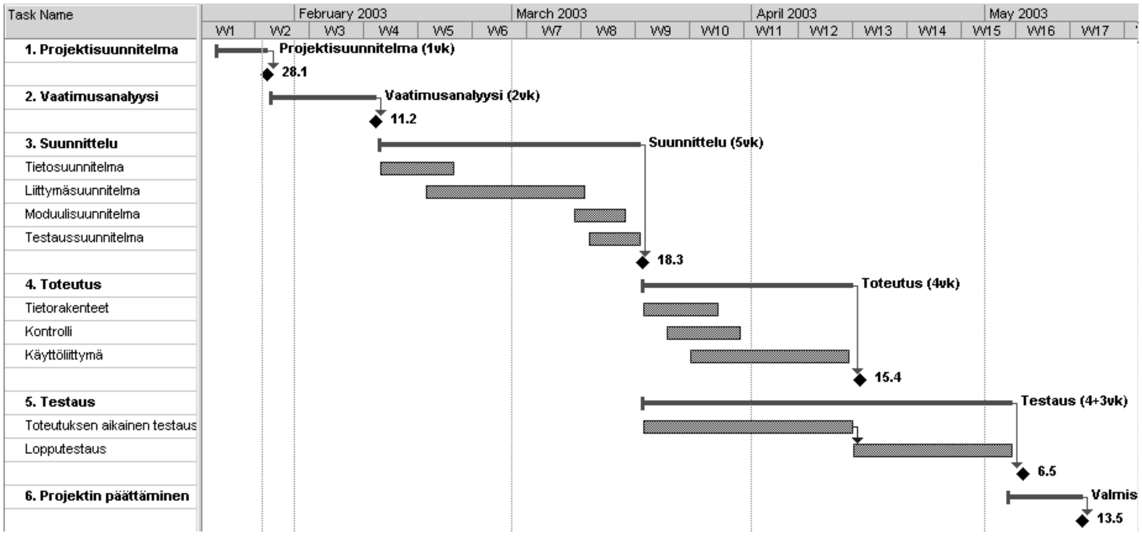
| | |
|---|---|
| Riski | Jos säännöt on toteutettu LISPillä, ja LISP-tulkissa on bugi. |
| Ennakointi | Testataan huolellisesti. |
| Reaktio | Vaihdetaan LISP-tulkkia ja toivotaan että vaihtoehto toimii paremmin. |
| Vaikutus | Ei välttämättä suurta vaikutusta. |
| Vakavuus | Marginaalinen. |

| | |
|---|---|
| Riski | Kielivalinnat osoittautuvat työläiksi. |
| Ennakointi | Seurataan työn edistymistä. |
| Reaktio | - |
| Vaikutus | Projekti viivästyy. |
| Vakavuus | Marginaalinen. |

| | |
|---|---|
| Riski | Työkalut osoittautuvat heikkolaatuisiksi tai huonosti soveltuviksi. |
| Ennakointi | Varataan joustoa aikatauluun. |
| Reaktio | Vaihdetaan työkaluja. |
| Vaikutus | Projekti viivästyy tai työmäärä kasvaa. |
| Vakavuus | Marginaalinen. |

# Liitteet

## 1. GANTT-kaavio




Kuva 1: Aikataulu