VERSTAS-PROJEKTI

Projektiraportti

Antti Hakala

Tomi Laamanen

Paavo Nieminen

Jukka Toivanen

7.1.2003

Jyväskylän yliopisto

Tietotekniikan laitos

- Tekijät: Antti Hakala, Tomi Laamanen, Paavo Nieminen, Jukka Toivanen
- Yhteystiedot: `verstas@korppi.jyu.fi, anthakal@cc.jyu.fi, laamanen@cc.jyu.fi, nieminen@cc.jyu.fi, jitoivan@cc.jyu.fi`
- Projektitila: Agora C223.4
- Puhelin: 014-2604966
- Kotisivu: `http://kotka.it.jyu.fi/verstas/`

- Työn nimi: Verstas-sovellusprojektin projektiraportti
- Työ: Tietotekniikan sovellusprojekti

Tiivistelmä:

Tämä dokumentti on Jyväskylän yliopistossa syksyllä 2002 toteutetun Verstas-sovellusprojektin *projektiraportti* (aiemmin valmistuneissa dokumenteissa viitataan tähän nimikkeellä *projektikäsikirja*). Verstas-projekti toteutti Numerola Oy:lle mallinnustyökaluja yhdistävän ja kokoavan käyttöliittymän. Projektin tuotokseen sisältyi myös *Elisp-ohjelma* Emacs-tekstieditorin ohjaamiseksi sekä graafinen *sovellus Fortran-ohjelmien kääntämiseen* ja ajamiseen. Raportissa kuvataan projektin etenemistä, verrataan toteutunutta projektia sekä sovellusta alkuperäisiin suunnitelmiin ja arvioidaan asetettujen tavoitteiden toteutumista.

Avainsanat: sovellusprojekti, projekti, käyttöliittymä, mallinnusympäristö, organisaatiomuisti

Versiohistoria

Versio	Pvm	Tekijä	Kuvaus
0.1	01.12.2002	PN	Ensimmäinen raakaversio tehty projektisuunnitelman pohjalta
0.2	5.12.2002	PN	Suunniteltu dokumentin rakenne projektiohjeen avulla
0.3	12.12.2002	PN	Kirjoiteltu vähän alustavia ajatuksia sinne sun tänne
0.4	16.12.2002	PN, JT	Lisää ajatuksia sinne sun tänne, toteutunut ja suunniteltu aikataulu Gantt-kaaviona
0.5	17.12.2002	AH, TL, PN, JT	Henkilökohtaiset kokemukset ja ajankäyttö
0.6	17.12.2002	PN	Kirjoitettu loput tekstistä ja yritetty jäsennellä selkeästi. Pieniä osia jäi vielä puuttumaan.
0.7	18.12.2002	PN, AH, TL	Pieniä tarkennuksia sisältöön ja korjattu havaitut sanamuotovirheet. Lisätty viitteisiin toteutuksessa käytetyt manuaalit.

Tekijöiden lyhenteet:

- AH Antti Hakala
- TL Tomi Laamanen
- PN Paavo Nieminen
- JT Jukka Toivanen

Hyväksytty

Pvm ..... Allekirjoitus .....

Pvm ..... Allekirjoitus .....

Sisältö

1 Johdanto

Mallinnus eli *simulointi* tarkoittaa todellisen maailman ilmiöiden kuvaamista matemaattisesti ja vastauksien etsimistä tutkimuksessa ja tuotekehityksessä esiin nouseviin kysymyksiin: Minkä muotoinen osa toimii koneessa parhaiten? Kuinka suurta rasitusta tukipilari kestää? Minkä verran venttiilin materiaali taipuu sisäisen paineen vaikutuksesta? Mallinnus tapahtuu *tietokoneohjelmilla*, sillä ilman niitä olisi käytännössä mahdotonta löytää ratkaisua matemaattisiin yhtälöihin, joita tarkasteluissa nousee esiin.

Numerola Oy on vuonna 1998 perustettu yritys, joka tekee asiakassuuntautunutta tutkimusta ja ohjelmistokehitystä laskennallisen mallinnuksen ja optimoinnin alalla. Meneillään on myös oma *Numerrin-tuotekehitysprojekti*, jonka tarkoituksena on kehittää mallinnuksessa ja mallipohjaisessa optimoinnissa käytettävien simulointiohjelmistojen kehitysympäristö. Numerrin sisältää niin sanotun *laskentaytimen* sekä *työkaluja* (tietokoneohjelmia) joilla voidaan yksi kerrallaan suorittaa *mallinnusprosessin* eri vaiheet (mm. geometrian siirtäminen tietokoneen ymmärtämään muotoon, numeerinen laskenta sekä tulosten visuaalinen ilmentäminen ihmisen ymmärtämässä muodossa).

Verstas-projekti suunnitteli ja toteutti Numerola Oy:lle *kokoavan käyttöliittymän* näille mallinnus- ja simulointiohjelmistoille. Projekti toteutettiin Jyväskylän yliopiston tietotekniikan sovellusprojektina. Tässä raportissa kuvataan Verstas-projektin toteutunutta aikataulua ja projektityöskentelyä sekä verrataan toteutunutta projektia alkuperäiseen suunnitelmaan.

Luvussa 2 esitellään projektin käytössä olleet resurssit. Luvussa 3 kuvataan pääpiirteissään projektin alkuperäiset tavoitteet. Luvussa 4 kerrotaan projektin eri osa-alueiden toteutumisesta suunnitelmaan verraten sekä myös ajankäytöstä, tehtävien jakautumisesta ja projektin aikana havaituista riskitekijöistä. Luku 5 sisältää projektin jäsenten henkilökohtaisia kokemuksia projektista. Luvun 6 yhteenvedossa esitetään päättyneen projektin pohjalta johtopäätöksiä ja ajatuksia auttamaan ryhmän jäsenten ja tietotekniikan laitoksen myöhempiä projekteja. Lähdeluettelo sisältää kaikki projektityössä käytetyt lähteet (kirjat, artikkelit, WWW-sivut ja käyttöohjeet sekä projektin toteuttamat dokumentit).

1.1 Termit ja käsitteet

Tässä aliluvussa listataan projektin kuluessa syntynyt terminologia ja selitetään käsitteet siten kuin projektin jäsenet ja tilaaja ymmärtävät ne projektin päättyessä. Termiluettelo on jaettu aihepiirin mukaan ja järjestetty kunkin aihepiirin sisällä aakkosjärjestykseen.

1.1.1 Sovellukset, kirjastot ja tiedostot

Alisovellus: Numerolan toteuttama sovellusohjelma, jolla toteutetaan yksi mallinnusprosessin vaiheista.

Dmg: Alisovellus, joka muuttaa 2D-geometrian laskentaverkoksi.

Emacs: Laajennettava ja muokattava tekstieditori.

Exodus: Alisovellus, työkalu 3-ulotteisen laskentageometrian luomiseen.

Genesis: Alisovellus, työkalu 2-ulotteisen laskentageometrian luomiseen.

Makefile: Make-sovelluksen käyttämä tiedosto ohjelman lähdekoodin kääntämisen hallintaan.

Mbgen: Alisovellus, joka muuttaa 3D-geometrian laskentaverkoksi.

Numerrin-ydin: Numerolan toteuttama FEM-aliohjelmakirjasto.

Plot: Alisovellus visualisointiin.

TeeKL, AutoUI: Alisovellus, työkalu graafisen käyttöliittymän automaattiseen tuottamiseen ja hallintaan.

Verkkogeneraattori: Alisovellus, joka tuottaa geometriasta laskentaverkon. Verstaan alisovellusten käytössä on useita eri tilanteisiin sopivia verkkogeneraattoreita, kuten Verkotin, Dmg, Mbgen.

1.1.2 Ohjelmointikielet ja -ympäristöt

Elisp: Emacs lisp, Emacsin laajentamiseen käytettävä ohjelmointikieli.

Fortran: Erityisesti laskennallisten ongelmien ratkaisuun käytetty ohjelmointikieli.

Kylix: C++/Object Pascal -kehitysympäristö Linux-käyttöjärjestelmälle.

Object Pascal: Olio-ohjelmointilaajennos Pascal-ohjelmointikieleen.

1.1.3 Mallinnusprosessi ja laskenta

FEM: Finite Element Method, äärellisten elementtien menetelmä, eräs numeerinen menetelmä osittaisdifferentiaaliyhtälöiden numeeriseksi ratkaisemiseksi.

(Laskenta)geometria: Tietokoneella luotu tutkittavan fysikaalisen kappaleen muodon ja rakenteen kuvaus, esitettynä joukosta yksinkertaisia käyriä tai pintoja muodostuvana approksimaationa.

(Laskenta)geometrian luonti: Mallinnusprosessin vaihe, jossa kuvataan tutkittavan fysikaalisen järjestelmän geometriset ominaisuudet. Tuloksena on laskentageometria.

Laskenta: Mallinnusprosessin vaihe, jossa käännetty laskentaohjelma ajetaan syötteenään laskentaverkko sekä parametrit, tulosteena saadaan laskentatulokset.

Laskentaohjelma: Fortran-kääntäjällä laskentamallista käännetty ajettavissa oleva ohjelma.

Laskentatulos: Laskentaohjelman tuloste, matemaattisen mallin ratkaisu annetuilla lähtötiedoilla (geometria, parametrit), käytännössä ASCII-tekstinä esitettyä numeerista dataa.

Laskentaverkko: Verkkogeneraattorin tuloste, geometrian jako laskentaa varten osiin esim. kolmioihin tai tetraedreihin.

Laskentaverkon luonti: Mallinnusprosessin vaihe, jossa laskentageometriasta tuotetaan verkkogeneraattorilla laskentaohjelman tarvitsemaan laskentaverkko.

(Laskenta)malli: Matemaattinen malli, jonka ilmenemismuoto on fortran-lähdekoodi.

Mallin toteutuksen etsintä: Valittua matemaattista mallia vastaavan ohjelman etsiminen tai luominen.

Mallin valinta: Mallinnusprosessin vaihe, jossa päätetään (yleensä kirjallisuuteen tai tutkimukseen nojaten) myöhemmissä vaiheissa käytettävä matemaattinen malli.

Mallinnusprosessi: Nykyaikaisen tuotekehityksen ja tutkimuksen osa-alue, jossa prototyypin rakentamisen ja kokeellisen mittaamisen sijasta tehdään kokeiluja matemaattisella mallilla, joka kuvaa järjestelmän toimintaa annetuissa olosuhteissa.

Optimointi: Simuloinnin (automaattinen) toistaminen jollakin kriteerillä geometriaa ja/tai parametreja muuttamalla, tavoitteena parempi lopputulos.

Parametrit: Laskentaohjelman syöte, esim. tutkittavan ilmiön fysikaaliset ominaisuudet.

Visualisointi: Laskentatulosten esitys ihmisen ymmärtämässä muodossa (käytännössä graafisesti erilaisin tavoin), eräs mallinnusprosessin vaihe.

1.1.4 Verstas

Mallinnusympäristö: Kokonaisuus, jossa kaikki mallinnuksen vaiheet voidaan tehdä yhtenäisen käyttöliittymän avulla hallittavilla työkaluilla.

Numerrin-Verstas: Verstas, Verstas-ympäristö. Mallinnusympäristö, jossa kaikki mallinnusprosessin vaiheet voidaan toteuttaa käyttäen Numerolan työkaluja ja Numerrin-ydintä.

Projekti: Kokonaisuus, joka sisältää kaikki tiettyyn simulointitehtävään liittyvät tiedot (s.o. mallit, geometriat, parametrit, tulokset, raportit jne).

Raportti: Tutkimus-, päivitys- tai muu raportti, esim. pdf-muodossa.

Tietokanta: (Tässä projektissa) hakemistorakenne sisältäen uudelleen hyödynnettäviä mallinnukseen käytettäviä tiedostoja.

2 Projektin resurssit

Tämä luku kuvaa Verstas-sovellusprojektin henkilö- ja materiaaliresurssit sekä niiden käytön.

2.1 Henkilöresurssit

Projektiryhmä koostui neljästä opiskelijasta: Antti Hakala on neljännen vuoden tietotekniikan opiskelija, suuntautunut ohjelmistotekniikkaan. Tomi Laamanen on neljännen vuoden tietotekniikan opiskelija tietoliikenteen linjalla. Paavo Nieminen opiskelee tietotekniikkaa kolmatta vuotta, linjanaan tieteellinen laskenta. Jukka Toivanen on kolmannen vuoden tietotekniikan opiskelija, suuntautunut tieteelliseen laskentaan.

Tietotekniikan laitoksen puolelta projektin *vastaavana ohjaajana* toimi yliassistentti Kari Kärkkäinen ja *teknisenä ohjaajana* fil. yo Ville Tirronen.

Numerola Oy:n puolelta *edustajina* toimivat sovellusasiantuntija FT Raino Mäkinen, johtaja FT Pasi Tarvainen ja sovellusasiantuntija FM Eeva-Kaisa Rouhiainen.

Johtaja FT Kai Hiltunen Numerolalta testasi ja kommentoi sovellusta projektin loppuvaiheessa. Lisäksi projektin apuna olivat Numerolan puolelta virtauslaskennan asiantuntija DI Jarmo Korpijärvi, joka esitteli kaupallisen virtauslaskentaohjelmiston CFX:n toimintaa ja Anni Toivola, joka kehittää Genesis-alisovellusta.

Yhteistyö tilaajan ja ohjaajien kanssa oli helppoa, yhteydenottoihin vastattiin ja apua saatiin pyydettäessä. Kynnystä kysymysten esittämiselle ei ollut.

2.2 Tilat, tarvikkeet ja ohjelmat

Projektin käytössä oli lukittava *työhuone* Agora C223.4, jossa jokaisella projektiryhmän jäsenellä oli käytössään *oma työasema*. Projektin käytössä oli myös *tulostin*, *toimistotarvikkeita*, *kopiokone* sekä vanhempia *testikäyttöön tarkoitettuja tietokoneita*. Projektin käytössä oli lisäksi erillisellä varauksella käyttöön saadut *kokoustila* Agora C223.1, *videotykki* ja *kannettava tietokone*.

Kolmeen työasemaan oli asennettu *Linux*-käyttöjärjestelmä ja neljänteen *Windows 2000*. Linux-työasemiin oli asennettu *Kylix 3.0 Enterprise* ohjelmistokehitystyökalu, jota projektin toteutuksessa pääasiallisesti käytettiin. Työasemissa käytettiin lisäksi myös *Dia, ArgoUML, XFig ja Gimp* -ohjelmia kaavioiden piirtämiseen, *LyX*'iä TEX-dokumenttien tuottamiseen ja www-selaimia (*Opera, Mozilla*) tiedon hakemiseen. Ylläpitoon käytettiin *tekstieditoreita* ja lukuisia unix-käyttöjärjestelmän *apuohjelmia*. Ajankäyttövihkot toteutettiin *Open Office* -taulukkolaskentaohjelmalla. Projektin alkuvaiheessa asennettiin projektilaisten käyttöön *Numerrin-ohjelmiston* versio, jonka yhteyteen sovellusta lähdettiin kehittämään. Windows-työasemassa käytettiin lähinnä *Microsoft Project* -ohjelmaa aikataulun suunnitteluun, *Microsoft PowerPoint* -ohjelmaa opponoinneissa ja loppuesittelyssä käytetyn esitysgrafiikan luontiin sekä *VNC* -ohjelmaa jonkin Linux-työaseman etäkäyttämiseksi.

Projektin käytössä olleet tilat toimivat tarkoituksenmukaisesti ja huoneen työasemat ohjelmineen suurimmaksi osaksi moitteettomasti alkuvaiheen nikottelun jälkeen. Käytännössä lähes kaikki projektin tehtävät tapahtuivat Linux-ympäristössä ja aina kun koko projektiryhmä oli samaan aikaan töissä, joutui joku työskentelemään melko *hankalasti VNC-yhteyden kautta* jakaen jonkin Linux-työaseman resurssit toisen käyttäjän kanssa. Kuitenkin ajoittain tarvittiin myös Windows-konetta, joten muutosta käyttöjärjestelmien asennuksiin ei lähdetty pyytämään

mikrotuelta. Testikäyttöön tarkoitettujen tietokoneiden funktio jäi ryhmälle arvoitukseksi, sillä niihin ei päässyt kirjautumaan eikä mikrotuelta saanut valaisevia vastauksia niitä koskeviin kysymyksiin.

Versionhallintaan käytettiin *CVS* (*Concurrent Versions System*) -järjestelmää komentorivikomennoin. Versionhallinta oli tärkeä osa projektin onnistumista, sillä se mahdollisti dokumenttien ja ohjelmakoodin joustavan päivittämisen. Järjestelmän käyttöönotto ja käyttäminen osoittautui helpohkoksi, mutta projektin viimeisinä päivinä ei oltu enää varmoja, päivittyvätkö tiedot oikein. Ilmiön taustat jäivät epäselviksi. Kyse saattoi olla järjestelmän epämääräisestä käyttäytymisestä, mutta luultavammin käyttövirheestä tai jopa yksinkertaisesti päivitysten unohtumisesta.

2.3 Työaika

Ryhmä käytti projektiin liittyviin toimiin yhteensä yli 1430 tuntia aloitusluennon (13.9.2002) alun ja projektiraportin kirjoittamisen (18.12.2002) välillä. Lopullinen tuntimäärä loppuesittelyn ja viimeisen palaverin (20.12.2002) jälkeen tulee olemaan noin 1500. Ajankäyttö jakautui ryhmän jäsenten kesken melko tasaisesti. Taulukossa 1 esitetään ajankäyttö sillä tarkkuudella kuin tehtyjen ajankäyttövihkojen pohjalta on mahdollista. Vaakariveiltä voidaan lukea kunkin henkilön työaika osa-alueittain. Pystyriveiltä taas kuhunkin osa-alueeseen käytetty aika henkilöittäin. Kunkin sarakkeen ja rivin viimeisessä solussa on yhteenlaskettu aika. Projektin kokonaisajankäyttö on siten luettavissa oikeasta alakulmasta.

Aika on esitetty tunteina ja minuutteina. Osa-alueet on jaettu karkeasti *koulutukseen* (luennot ja ohjekirjojen lukeminen), *projektinhallintaan* (palaverit, raportointi, tiedotus), *suunnitteluun*, *toteutukseen* (ohjelmointi) ja *testaukseen* (komponentti- ja integraatiotestaus). Ajankäytön kirjaamisesta ei ollut sovittu yhtenäistä käytäntöä, joten merkinnät tehtiin vaihtelevalla tarkkuudella ja jako osa-alueisiin oli projektin jäsenillä erilainen. Paavo esimerkiksi merkitsi koulutuksen ja suunnittelun kohdalle aikaa, joka käytännössä sijoittui toteutuksen lomaan. Taulukosta voi siis lukea luotettavasti lähinnä yleislinjauksen kokonaistuntimääristä.

Henkilö	Koulutus	Projektinh	Suunn	Toteutus	Testaus	Yhteensä
AH	35:00	78:15	65:45	137:15	27:45	344:00
TL	30:00	74:30	108:00	145:30	18:30	376:30
PN	50:43	101:36	112:03	73:05	15:35	353:02
JT	31:30	101:30	54:30	135:00	34:30	357:00
Yht.	147:13	355:51	340:18	490:50	96:20	1430:32

Taulukko 1: Projektiryhmän aikaresurssien käyttö.

3 Projektin tausta ja tavoitteet

Tässä luvussa kerrataan projektin taustat ja tavoitteet.

3.1 Tilaajan tavoitteet

Numerola Oy:n tärkein tavoite projektin suhteen oli *saada käyttöliittymä* Numerrin-sovelluspaketille. Käyttöliittymältä toivottiin alunperin seuraavia asioita:

- käyttöliittymä pystyisi *hallitsemaan* Numerrin-alisovelluksia,
- se muodostaisi alisovellusten kanssa *yhtenäiseltä* näyttävän ja tuntuvan kokonaisuuden,
- *olisi mukautettavissas* erilaisten käyttäjien tarpeisiin sopivaksi,
- automatisoisi mallinnusprosessin *rutiininomaisia vaiheita*,
- *piilottaisi* suurta käyttöjärjestelmä- tai ohjelmointituntemusta vaativat toiminnot graafisen käyttöliittymän taakse ja
- veisi Numerrin-ohjelmistoa *kaupallisesti houkuttelevaan* suuntaan

Lisäksi Numerolassa toivottiin saatavan kokemuksia Kylix-ohjelmakehitysympäristöstä pitäen silmällä Numerrin-ohjelmiston jatkokehitystä. Näitä kokemuksia on kerätty luvun 6 yhteenvetoon. Käyttöliittymän *siirrettävyys Windows-ympäristöön* haluttiin varmistaa.

Varsinaiset toiminnalliset vaatimukset tarkentuivat ajan mittaan ja lopulliset tavoitteet tärkeysjärjestyksineen kirjattiin tilaajan ja ryhmän välisen projektisopimuksen liitteenä olleeseen käyttötapausdokumenttiin [20].

3.2 Opetukselliset tavoitteet

Sovellusprojekti kuuluu pakollisena opintojaksona Jyväskylän yliopiston tietotekniikan laitoksen tutkintoihin. Opiskelijat suorittavat projektin usein cum laude -opintojensa loppuvaiheessa. Tarkoituksena on antaa opiskelijalle käsitys työelämän ohjelmistoprojekteista, ryhmätyöstä ja työn tulosten esittelemisestä [11]. Erityisesti Verstas-projektin opetuksellisiksi tavoitteiksi määriteltiin projektin alkaessa (ks. [21], luku 2.2)

- tuntuman antaminen *projekti- ja ryhmätyöskentelyyn*,
- työvaiheiden *työmäärän ja aikataulujen arviointi,*
- projektin osapuolten välisen *kommunikoinnin ja yhteistyön* hahmottaminen,
- *tilaajan ja tuottajan* roolien oppiminen,
- *dokumenttien ja kaavioiden* laatiminen,
- *kokouskäytäntö* sekä
- *vastuu* projektityössä.

Tehtäviä jaettaessa pyrittiin varmistamaan, että jokainen ryhmän jäsen pääsee tutustumaan projektin kaikkiin vaiheisiin ja tehtäviin.

4 Projektin toteutus

Tässä luvussa esitellään projektin toteutustavat, työnjako ja toteutunut aikataulu. Projektin vaiheista kerrotaan aikajärjestyksessä ja kohdattuja hakaluuksia analysoidaan. Verrataan toteutunutta projektia alkuperäisessä projektisuunnitelmassa [21] esitettyyn.

4.1 Toteutustavat

Esitellään ensiksi projektityössä valitut toiminta- ja dokumentointikäytännöt. Nämä valittiin tilaajan alkuperäisten toiveiden sekä projektiohjeen [11] perusteella. Käytetyt työkalut ja ohjelmat on lueteltu tarkemmin luvussa 2.

4.1.1 Ohjelmointiympäristö

Sovelluksen ohjelmointikielenä oli *Object Pascal* ja ohjelmointiympäristönä *Kylix 3.0 Enterprise*. Emacs-moduuli toteutettiin *Elisp*:llä. Sovellus toteutettiin *Linux-ympäristöön*. Windows-toteutus priorisoitiin vaatimusmäärittelyä tehtäessä vähemmän tärkeäksi, eikä sitä lopulta ehditty ajan puutteen vuoksi toteuttaa. Alkuperäisestä projektisuunnitelmasta poiketen myöskään testausta ei suoritettu missään vaiheessa Windows-ympäristössä.

4.1.2 Tiedotus ja palaverit

Projektilla oli 11 *projektipalaveria* tilaajan kanssa. Palaverit pidettiin pääsääntöisesti viikon välein. Muutama viikottainen palaveri jätettiin toteutusvaiheessa väliin. Lisäksi projektin aikana pidettiin yksi *toteutusteknisiin kysymyksiin* keskittynyt palaveri 14.11.2002 sekä kaksi *simulointiohjelmistojen käyttöesittelyä* (Numerrin 23.9.2002 ja CFX 9.10.2002). Teknisen ja vastaavan ohjaajan kanssa pidettiin ajoittain *pienempiä neuvonpitoja*. Projektin etenemisestä tiedotettiin palaverien lisäksi projektin *WWW-sivulla* ja *sähköpostilistalla*, jonka kautta esitettyihin tiedusteluihin saatiin tilaajalta vastaukset ripeästi.

4.1.3 Dokumentaatio

Seuraavassassa listassa on esitelty aikajärjestyksessä Verstas-projektin toteuttamat dokumentit ja niiden tarkoitus:

- *Projektisuunnitelmassa* (viite [21]) kuvattiin Verstas-projektin taustaa, aikataulua sekä projektin toteutukseen liittyviä asioita ennakoivasti. Projektisuunnitelmassa käytettiin lähteenä teoksia [12] ja [13].
- *Käyttötapausdokumentin* (viite [20]) pohjalta rakennettiin myöhemmät suunnitelmat.
- *Vaatimusmäärittelyssä* (viite [19]) määriteltiin tuotettavan sovelluksen ominaisuudet ja toiminnallisuudet sekä sen ympäristölleen asettamat vaatimukset. Vaatimusmäärittelydokumentin lähteenä käytettiin teoksia [13] ja [16].
- *Sovellussuunnitelmassa* (viite [22]) kuvattiin toteutettavan sovelluksen arkkitehtuuri ja käyttöliittymä. Lähteenä käytettiin teoksia [3], [5], [6], [12] ja [13].

- *Testaussuunnitelmassa* (viite [24]) esitettiin sovelluksen testauksen toteutusstrategiat, testitapaukset ja testausympäristö. Lähteenä käytettiin teoksia [13], [19] ja [20].
- *Testausraportista* (viite [25]) voidaan nähdä testauksen tulokset ja sen perusteella testi voidaan tarvittaessa toistaa. Lähteenä käytettiin teoksia [13], [24], [19] ja [20].
- *Käyttöohje* (viite [26]) toteutettiin HTML-muodossa sovelluksessa käytettäväksi ja siitä on myös tulostettu versio.
- *Asennusohje* (viite [27]) sisältää sovelluksen asennusohjeet.
- *Sovellusraportissa* (viite [23]) dokumentoitiin sovellussuunnitelman toteutuminen. Lähteenä käytettiin teoksia [3], [5], [6], [12], [13], [19] ja [20].
- *Projektiraportti* (viite *[28]*) on tämä dokumentti, jossa raportoidaan projektin toteutus ja verrataan toteutunutta suunnitelmaan. Dokumentin lähdeluettelo koostuu kaikista projektin dokumenteissa käytetyistä lähteistä sekä kaikista projektin tuottamista dokumenteista. Lähdeluettelossa on myös listattu kaikki oppaat ja manuaalit, joita projektin hyväksi on käytetty.

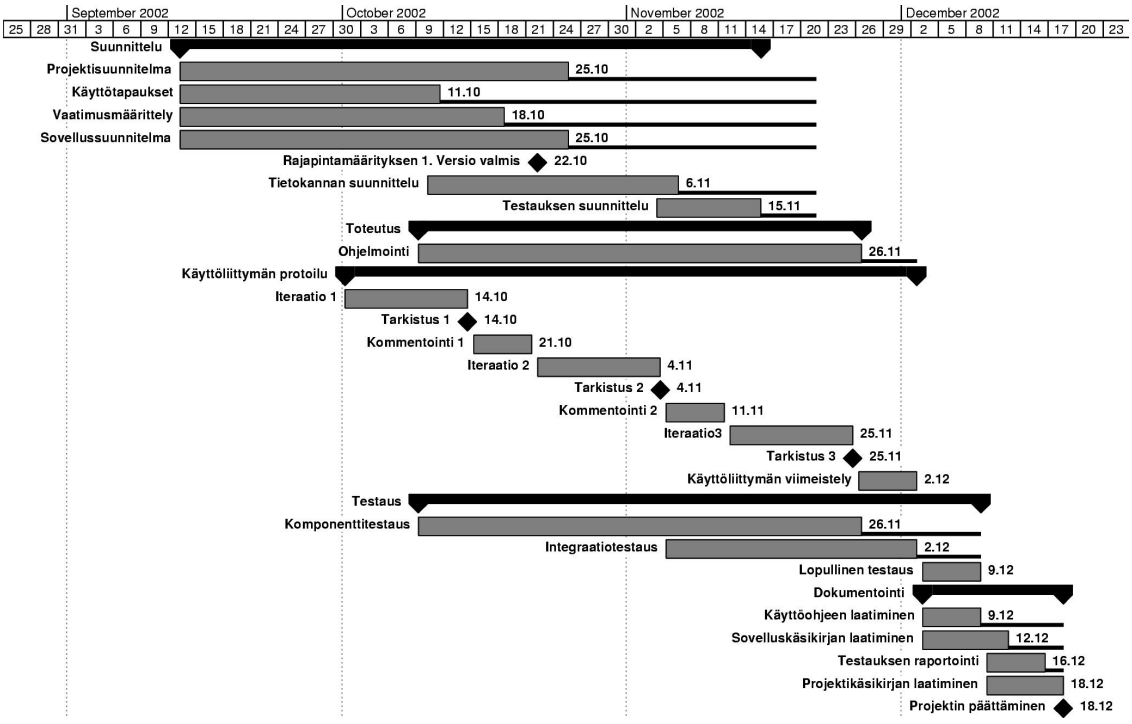
4.2 Aikataulu

Sovellusprojekti toteutettiin 13.09.2002-20.12.2002 välisenä aikana. Se päättyi 19.12.2002 pidettyyn loppuesittelyyn ja tuotetun materiaalin luovuttamiseen 20.12.2002. Kuvassa 1 on esitetty projektin alkuperäinen jako viiteen päävaiheeseen sekä kunkin osavaiheen tarkempi aikataulu. Kuvassa 2 taas on esitetty vastaavien vaiheiden toteutunut aikataulu. Päivämäärät ovat dokumenttien versiohistorian mukaisia.

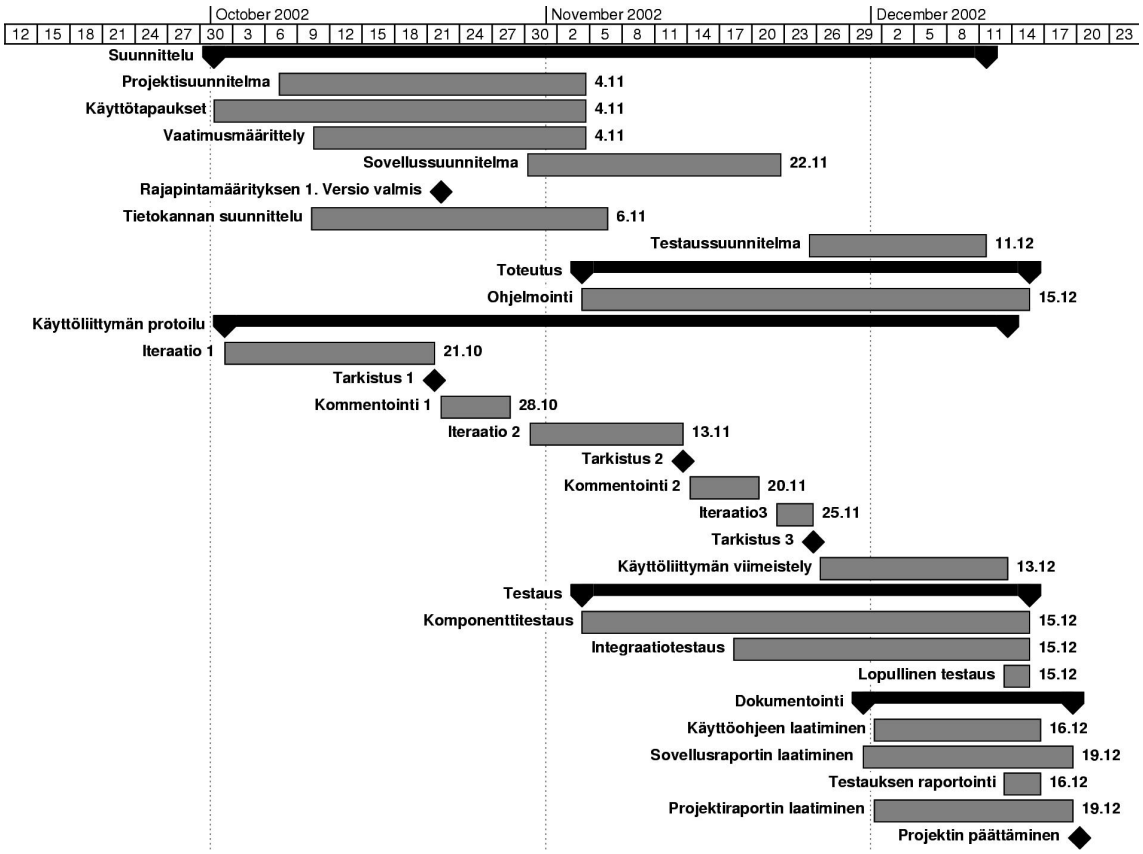
Huomataan, että projekti päättyi suunniteltuna ajankohtana, jolloin aikaansaatu sovellus oli dokumentoitu, vaatimusmäärittelyn mukainen ja järjestelmällisesti testattu. Toteutuksen aikana aikataulusta jäätiin reilusti jälkeen, joten jouduttiin käyttämään toteutukseen ja testaukseen vielä viimeistelyyn ja dokumentointiin varattu aika.

Viivästymiset alkoivat heti projektin alussa: Suunnitteludokumentit aloitettiin vasta kun sovellusprojektin aloitusluennosta oli ehtinyt kulua kaksi viikkoa. Sovellussuunnitelma valmistui jopa lähes kuukauden myöhässä eivätkä muutkaan dokumentit valmistuneet suunniteltuna ajankohtana. Ohjelmointia jatkettiin kolmisen viikkoa suunniteltua pidempään ja lopullinen testaus tapahtui viikon myöhässä. Käyttöliittymän iteraatioversiot toimitettiin useita päiviä myöhässä. Siltikin ne olivat puutteellisia ja niihin täytyi tehdä lisäpäivityksiä toimittamisen jälkeen. Yleisesti ottaen projekti laahasi alusta lähtien systemaattisesti noin viikon jäljessä aikataulusta, minkä seurauksena projektin loppuvaiheessa tuli kiire.

Lopussa ryhmä pystyi kuitenkin joustamaan työmäärän suhteen. Viivästymiset eivät päässeet kasautumaan ja projekti pystyttiin päättämään suunniteltuna ajankohtana. Alkuperäisiä syitä viivästymisille pohditaan luvussa 4.4.



Kuva 1: Projektin alkuperäinen aikataulu Gantt-kaaviona



Kuva 2: Projektin toteutunut aikataulu Gantt-kaaviona

4.3 Projektin kulku

Seuraavassa esitetään lyhyt kuvaus projektin vaiheiden toteutumisesta siinä järjestyksessä kuin ne suurin piirtein ajallisesti tapahtuivat. Tarkempi kuvaus kohdatuista hankaluuksista on luvussa 4.4.

4.3.1 Aloitus ja tehtävien selkiytyminen

Tehtävän alkuperäinen muoto säikähdytti ryhmän laajuudellaan. Projektissa tulisivat yhdistymään käyttöliittymän tekeminen, tiedostojärjestelmän ja eräänlaisen tietokannan hallinta, useiden samanaikaisesti käynnissä olevien ohjelmien hallinta ja rajapinnan luominen tätä tarkoitusta varten. Myös sovelluksen lopullinen käyttöympäristö, *mallinnusprosessi*, oli tuntematon alue koko ryhmälle, paitsi Jukalle, joka oli ollut kesällä harjoittelussa Numerolassa.

Vastaavan ohjaajan sekä Numerolan antamien käyttöesimerkkien ja Jukan kokemuksen avulla mallinnuksen yleiskuva hahmottui ryhmälle ja olemassa olevan Numerrin-ohjelmiston tietovirrat pystyttiin hahmottamaan.

4.3.2 Suunnittelu

Suunnittelu käsitti projektityön, sovelluksen ja testauksen erillisen suunnittelun. Suunnitteludokumenttien tarkoitus on esitetty luvussa 4.1.3.

Rajapinnan suunnittelu olisi ollut hyvä tehdä nopeasti projektin alettua, mutta se voitiin tehdä vasta kun aihealue oli ehditty sisäistää. Tilaajalla oli selkeä käsitys rajapinnalta vaadittavista ominaisuuksista, joten käyttötapausten [20] perusteella tehdyn abstrahoinnin avulla saatiin lyötyä rajapinta lukkoon riittävän ajoissa.

4.3.3 Toteutus

Toteutus käsitti sovelluksen osien ohjelmoinnin. Itse *Verstas-pääsovelluksen* lisäksi toteutettiin *laskenta-alisovellus* ja *Elisp-ohjelma*, jolla Emacs-tekstieditoria voidaan ohjata kuten Numertimen alisovelluksia.

Pääsovellus jaettiin aikataulua ajatellen kahteen osaan: *Käyttöliittymä* ja *loppuosa*. Käyttöliittymän suunnittelu puolestaan oli alkuperäisessä suunnitelmassa jaettu kolmeen perättäiseen iteraatioon.

Versioita ei saatu toimitettua ajallaan palautteenantoa varten eikä niissä sittenkään ollut tarpeeksi ominaisuuksia varsinaisen käytettävyyden kommentointia varten. Tilannetta yritettiin parantaa integroimalla Verstaan koko toiminnallisuus mahdollisimman nopeasti käyttöliittymään. Tämä aiheutti ohjelmien toteutuksessa kiirettä ja virheitä, jotka kostautuivat vielä toteutusvaiheen lopussa. Niinpä sovellus jäi käyttöliittymän osalta hiomattomaksi.

4.3.4 Testaus

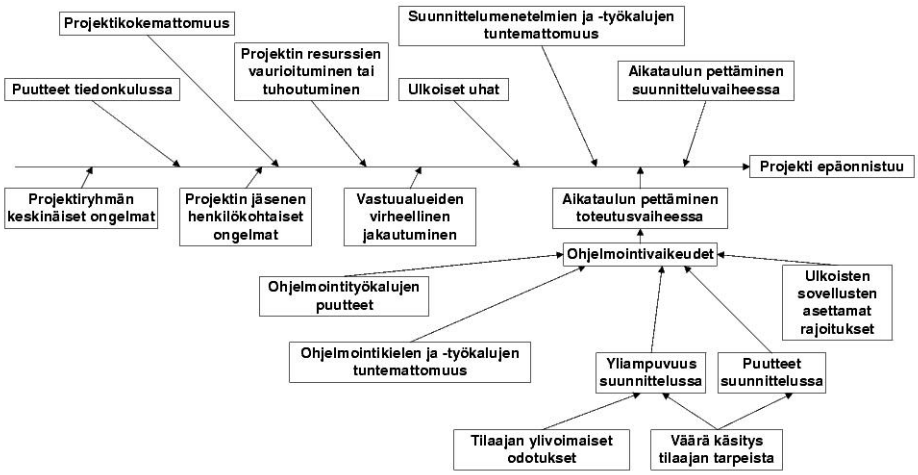
Projektin lopuksi sovellus *testattiin järjestelmällisesti* testaussuunnitelman [24] mukaan. Testausvaihe oli hyödyllinen, sillä siinä löydettiin ohjelmasta selkeitä puutteita vaatimusmäärittelyyn nähden. Puutteet ehdittiin korjata siten, että testaussuunnitelman mukainen testikierros saatiin menemään läpi ilman virheitä ennen projektin loppua. Jos testauksessa olisi havaittu suurempia virheitä, ei niihin olisi ehditty juurikaan puuttua aikaisempien viivästymisien takia.

4.3.5 Raportointi

Toteutuksen ja testauksen jälkeen jäljellä oli enää *raporttien tekeminen* projektista, itse sovelluksesta sekä sovelluksen testauksesta. Raporttien kirjoittaminen suunnitelmien ja toteutuksen pohjalta oli suoraviivaista ja sujui ilman ongelmia. Raporttien tarkoitus on esitetty luvussa 4.1.3.

4.4 Riskit ja viivästymisten aiheuttajat

Projektin alussa arvailtiin riskitekijöitä, jotka saattaisivat aiheuttaa projektin epäonnistumisen tai huonontaa tuloksen laatua. Nämä kerättiin pelkistettyyn riskikaavioon (kuva 3) ja arvioitiin niiden todennäköisyyttä asteikolla 1-3 (ks. [21], luku 4.3). Tässä luvussa analysoidaan projektia haitanneita todellisia riskejä ja verrataan niitä suunnitelmassa esiintyneisiin.



Kuva 3: Projektisuunnitelmassa esiintynyt riskikaavio

4.4.1 Kokemattomuus

Projektikokemattomuus sekä *ohjelmointikielen- ja työkalun tuntemattomuus* olivat selkeitä riskejä, joihin myös vääjäämättä törmättiin. Projektityyppiseen *työtapaan* ja *tehtävien jakoon* totuttelu vei aikaa ennen kuin kaikki resurssit saatiin käyttöön. Alussa ryhmän jäsenet keskittyivät usein samaan tehtävään. Suunnitelmia muokattiin pitkään pelkän muokkaamisen vuoksi, koska ei ollut selvää *mikä minkäkin suunnitteludokumentin tarkoitus* loppujen lopuksi on. Tarkoitus selvisi vasta kun suunnitelmaa oli ehditty tehdä pitkän aikaa. Suunnitelmista tuli siten hieman ylimalkaisia ja silti ne valmistuivat myöhässä.

Aikaa meni Object Pascalin ja Kylix-kehitysympäristön opetteluun ennen kuin varsinainen ohjelmointi oli mahdollista aloittaa. Jälkikäteen ajatellen olisi voinut olla viisasta aloittaa ohjelmointi aikaisemmin ja limittää siten suunnittelu- ja toteutusvaiheita. Selkeissä ongelmakohdissa kysyttiin apua tekniseltä ohjaajalta, joka vastasikin yleensä pian ja vastaukset antoivat mukavasti suuntaa toteutukselle. Riittävästä ohjauksesta huolimatta aikataulu viivästyi.

Projektisopimus laadittiin aikaisemmin Numerolalle tehdyn Genesis-työprojektin sopimuksen pohjalta. Tämä oli virhe, sillä sopimus kuitenkin muokattiin asteittain vastaamaan lähes suoraan Jukka-Pekka Santasen esittämää yliopiston lakimiehen avustuksella suunniteltua ja päivitettyä mallia [14]. Tähän hukkaantui lukemattomia työtunteja arvokkaasta suunnitteluvaiheesta.

Alkuvaiheen palavereissa hairahduttiin keskustelemaan yksityiskohdista jopa epäolennaisuuksia myöten ja käsiteltiin turhan paljon dokumentteja. Dokumenttien tarkastus- ja palautteenantokäytäntö olisi ollut syytä sopia välittömästi.

4.4.2 Muut riskit

Tilaajan ylivoimaisena odotuksena voitaneen pitää alkuperäistä vaatimusta kaikkien alisovellusten ikkunoiden sijoittumisesta suuren pääikkunan sisään. Huomattavan paljon aikaa käytettiin *selvittelytyöhön* mahdollisuuksista toteuttaa alisovellusten ikkunointi tällä tavoin. Varsinaisesti päästiin töihin vasta kun luovuttiin ajatuksesta suurikkunan sisällä toimivasta Verstaasta. Myös käytettävän editorin ratkaisemiseen kului resursseja alkuvaiheessa. Lopullinen vaatimus-

määrittely ei enää ollut yliampuva eikä ohjelmointivaiheessa tarvinnut luopua mistään tärkeäksi arvotetusta ominaisuudesta.

Projektin suunnitteluvaiheen lopussa törmättiin aikaisemmin muodostuneeseen *väärinkäsitykseen*: Projektiryhmä luuli, että Numerola toteuttaa laskenta-alisovelluksen. Numerolassa sen sijaan oltiin yhtä varmoja siitä, että laskentaohjelman kääntäminen ja ajaminen kuuluu ryhmän toteuttamaan sovellukseen. Väärinkäsitys kumpusi luultavasti eroavuudesta tilaajan alkuperäisen tehtävänkuvauksen ja käyttötapausten yhteydessä suoritetun abstrahoinnin välillä (verstaan ja alisovellusten eriyttäminen toisistaan). Eteenpäin päästiin ryhmän joustaessa ja luvatessa myös laskennan kuuluvan projektin tuotokseen. Tämä toi projektille lisää työtunteja ja kenties laski kokonaisuuden tasoa hieman. Tehtävä olisi ehkä kannattanut rajata alussa suppeammaksi.

Ohjelmointityökalun puutteita arvioitiin projektin aikana monesti. Kylix valittiin alussa pääasiassa kahdesta syystä: Käyttöliittymän tekemisen helppous sekä Numerolan tarve saada kokemuksia Kylix'n käytöstä. Käyttöliittymä olikin helppo tehdä, mutta se on kovin pieni osa koko Verstaan toiminnallisuudesta. Vaikeuksia tuli toimittaessa säikeiden, prosessien ja TCP/IP:n kanssa - eli käytännössä hyvin monessa paikassa. Object Pascal tuntuu sallivan omituisuuksia, joiden havaitseminen ei esimerkiksi C++:ssa tai Javassa jää ohjelmoijan vastuulle. Esimerkkinä mainittakoon olion luominen luokasta, jolla on abstrakteja metodeita.

Loput ennalta nähdyistä riskeistä vältettiin. Projektiryhmän keskinäisiä ja jäsenten henkilökohtaisia ongelmia ei esiintynyt. *Vastuualueet* jakautuivat tasaisesti ja *tiedonkulku* toimi projektin alusta loppuun hyvin. Myöskään projektin resurssit eivät vahingoittuneet ulkoisten uhkien takia, mikä oli alunperinkin aika teennäinen riski. Verstaan sekä *ulkoisten alisovellusten* ja jopa Emacs-editorin yhteistoiminta alkoi sujua yllättävänkin hyvin suhteessa ennakko-odotuksiin.

4.4.3 Riskienhallinnan arviointi

Projektisuunnitelmassa pystyttiin tiedostamaan ja luettelemaan kaikki todelliset viivästyksien aiheuttajat ja muut haittatekijät, joihin projektin aikana törmättiin. Niiden edellä kuvailtuja *todellisia ilmenemismuotoja* oli kuitenkin mahdoton arvata etukäteen (juuri projektikokemattomuuden takia). Etukäteen oli myös hyvin vaikea suunnitella yksityiskohtaisia toimintamalleja riskiskenaarioiden ennaltaehkäisyyn ja niistä toipumiseen. Jos kyseessä olisi ollut oikea ja kriittinen projekti, olisi riskejä pitänyt analysoida paljon tarkemmin ja niiden ennaltaehkäisyyn olisi täytynyt kehittää selkeämpiä toimintamalleja alusta alkaen noudatettavaksi. Tämä olisi sinänsä vienyt paljon resursseja, joten tällaisessa pikaprojektissa mahdollisuutta ei ollut.

Kohdatut aikatauluongelmat eivät ehtineet kasautua, sillä projektin loppuvaiheessa pystyttiin lisäämään työtunteja jonkin verran. Mitään laiskuutta ei projektin aikana esiintynyt kenenkään osalta. Pikemminkin projektiryhmän ilmapiiri oli erityisen motivoitunut ja tekemiseen suuntautunut.

4.5 Työnjako

Projektin vastuujako toteutui kuten projektisuunnitelmassa (ks. [21], luku 4.2) päätettiin. Kukaan ei joutunut tekemään liian paljon töitä eikä vastuualueita tarvinnut vaihtaa projektin aikana. Päävastuu dokumenttien ja ohjelmakoodien tuottamisesta pyrittiin jakamaan mahdollisimman tasaisesti eri jäsenille. Käytännössä toteutustyö ei jakautunut suoraan vastuujaon mukaan. Töitä tehtiin ristiin, ettei tyhjäkäyntiaikaa tai liiallista rasitusta olisi tullut kenellekään. Loppujen lopuksi jokainen ryhmän jäsenistä ehti osallistua lähes jokaisen osan tekemiseen. Poikkeuksia

olivat Tomin tekemä laskentasovellus ja Antin toteuttama eLisp-koodi, joihin muut eivät juurikaan kajonneet.

Ryhmällä oli kiertävä projektipäällikkyysvuoro. Projektipäällikkö vastasi tiedotuksesta ja aikataulun pitämisestä sekä toimi palavereissa puheenjohtajana. Ryhmän jäsenet toimivat vuorotellen palaverien sihteerinä.

5 Henkilökohtaiset kokemukset

Tässä luvussa on jokaisen projektiryhmän jäsenen kohdalta lyhyt henkilökohtainen kertomus projektityön sujumisesta ja opituista asioista.

5.1 Antti Hakala

Sovellusprojekti oli suhteellisen erilainen kurssi verrattuna muihin suorittamiini. Työtä riitti reilusti dokumentoinnissa, palavereissa ym. ja piti vielä opetella kaksi uutta ohjelmointikieltäkin (eLisp + object pascal). Sovelluksen vaatimusten rajaus oli alussa hieman tuskaista ja uusien työkalujen käytön opettelu vei myös aikaa, mutta tämän jälkeen niillä sai homman kuitenkin hoidettua (LyX etenkin). Ryhmä toimi hyvin yhteen ja jokainen osasi hieman eri asioita, mikä oli hienoa.

Vaikeat ja helpot kohdat

Vaikeinta projektityössä oli kaiken mahdollisen dokumentointi. Yllättävän vaikeaa oli eLispin opettelu näin lyhyessä ajassa. Rajapinnan määrittely sen sijaan sujui ehkä jopa yllättävänkin helposti.

Mitä opin

- Object pascalissa voi typecastata olion kokonaisluvusta.
- Object pascalissa voi tehdä ilmentymän luokasta, jolla on abstrakteja metodeja.
- LyX on aika kätevä.

Mitä tekisin toisin jos aloittaisin alusta

Harkitsisin ehkä toteutuskieltä uudemman kerran. Näin pienessä ajassa ja näillä resursseilla ei paljon parempaa tulosta kuitenkaan voi toivoa. Aikataulu olisi voinut olla erilainen ja olisimme voineet seurata sitä paremmin.

Ohjeita tuleviin aikoihin

Seuraavassa projektityössä olisi syytä saada parempaa palkkaa. Tulevien tietotekniikan sovellusprojektiryhmien ei kannata säikähtää vähästä.

5.2 Tomi Laamanen

Kokonaisuutena positiivinen kokemus. Työmäärä paisui ehkä hieman liian suureksi projektille varattuun aikaan nähden. Aihe oli suhteellisen mielenkiintoinen, joskin suunnittelun aikana liikuttiin paikoitellen aika epämääräisissä sfääreissä. Dokumenttien suhteen mentiin viimeistelyssä välillä liiallisuuksiin (käyttötapaukset) ja palavereissa kului välillä aikaa epäolennaiseen saivarteluun. Projektiryhmä toimi mielestäni hyvin yhteen ja jokainen teki sitä mitä parhaiten osasi. Ohjelmointivaiheessa Jukan panos oli olennainen. Vastaava ohjaaja ja tekninen ohjaaja hoitivat asiansa hyvin ja tilaajalta tuli vastaus kysymyksiin nopeasti.

Vaikeat ja helpot kohdat

Kylix tuotti vaikeuksia varmaankin kaikille, se ei välttämättä ole vielä täysin sopiva tuotantokäyttöön.

Mitä opin

Opin projektissa erityisen paljon ryhmätyöskentelyä ja kommunikointitaitoja. Kylixiä oli myös tietysti ihan hyvä oppia.

Mitä tekisin toisin jos aloittaisin alusta

Kyllähän suunnittelu olisi pitänyt yrittää aloittaa aikaisemmin. Itse yrittäisin olla vähemmän sarkastinen ja itsepäinen.

Ohjeita tuleviin aikoihin

Teoksessa [3] esitettävää leikkaamiseen ja liimaamiseen tukeutuvaa ohjelmointimenetelmää ei tule aliarvioida nopeassa ohjelmistonkehityksessä. Regexp on kova juttu.

5.3 Paavo Nieminen

Projekti oli hieno kokemus. Se oli itselleni ensimmäinen kosketus oman alan työkenttään - toisaalta ohjelmistotekniikkaan ja toisaalta tieteelliseen laskentaan. Molemmat alueet säväyttivät monipuolisuudellaan ja mielenkiintoisuudellaan. Ohjelmiston suunnittelumenetelmiä oli erityisen valaisevaa oppia. Alussa kului energiaa tutustumisessa Linux-käyttöjärjestelmään työkaluineen, kohtalaisen tuntemattomaan tieteellisen laskennan alaan sekä projektityöhön ja työnjakoon. Projektityö oli näistä kaikkein helpoin oppia. Omituisinta oli olla ensimmäisenä projektipäällikkövuorossa kun ei ollut harmainta aavistusta projektipäällikön tarkoituksesta. Tämä ja kaikki muutkin asiat selvisivät pikkuhiljaa työn edetessä ja nyt projektin loppuessa hommaan on päässyt jyvälle ihan mukavasti. Voisipa aloittaa alusta uudelleen tietäen sen mitä nyt.

Vaikeat ja helpot kohdat

Ohjelmointi osoittautui yllättävän vaikeaksi. Teoriassahan homman pitäisi olla yksinkertaista ja sitä on yliopistossa monta vuotta opeteltu. Käytännössä vastaan tulee kuitenkin aikaraja eli deadline ja sen kohdalla teoreettinen helppous sitten kaatuu ohjelman mukana. Vaikeuksia voitaisiin nähdäkseni (teoriassa) välttää kunnollisen suunnittelun avulla, mutta myös ohjelman suunnittelulla on aikaraja. Siihen taas liittyy itse ajankäytön arviointi, joka mielestäni oli Verstas-projektin alussa jopa mahdoton tehtävä. Näiden vaikeuksien jälkeen sitten kaikki muu osoittautuikin ennakko-odotuksista poiketen melkoisen helpoksi ja selkeäksi.

Mitä opin

Opin että tekemällä tosiaan oppii. Lukemattomista opituista asioista ehkä tärkein liittyy ohjelmiston suunnitteluun. Suunnitelmaa voi pyöritellä käsissään maailman ääriin asti saamatta

ohjelmaa koskaan tehtyä, ellei suunnittelua tee määrätietoisesti ja tarkoitushakuisesti. Erityisesti on syytä tietää suunnittelun tarkoitus - ehkä jopa ennen kuin ryhtyy suunnittelemaan. Ja puolittain tehdyn suunnitelman pohjalta sitten vasta onkin vaikea tehdä kunnon ohjelmaa.

Mitä tekisin toisin jos aloittaisin alusta

Ryhmä olisi käsittääkseni voinut toimia paremmin vain siinä tapauksessa, että se olisi ensin toteuttanut vastaavan projektin ja sen jälkeen ryhtynyt tekemään Verstasta. Silloin olisi luultavasti osattu aloittaa määrätietoinen ja olennaisiin seikkoihin keskittyvä suunnittelu sekä myös uuteen työvälineeseen tutustuminen välittömästi projektin alettua. Olisi myös osattu huomattavan paljon paremmin arvioida työvaiheisiin kuluvien tuntien määriä.

Jälkikäteen ajatellen olisi saattanut olla viisasta keskittyä toteuttamaan käyttöliittymän edellytyksenä oleva sisäinen mekanismi (prosessien viestinvälitys, tietokanta, tietorakenteet) kunnolla ja jättää itse käyttöliittymä vähemmälle huomiolle.

Ohjeita tuleviin aikoihin

Sovellusprojektiryhmän kannattaa heti alussa ryhtyä toimimaan kohti päämäärää. Työaikaa kannattaa ryhtyä käyttämään alusta asti, mutta siitä kannattaa karsia pois epäolennainen harhailu. Nyrkkisääntöä en osaa sanoa, vaan muutamia erityisiä huomioita:

Ensimmäisen toimenpiteen tulee olla aikataulun suunnittelu. Kokemattoman ryhmän on mahdotonta arvata työmääriään, joten aikataulu kannattanee veikata aikaisempien projektien kertomusten pohjalta ja erityisesti pelivaraa tulee lisätä reilusti joka ikiseen väliin.

Tietotekniikan sovellusprojektissa projektisopimustekstin muokkailu on ylimääräistä ja epäolennaista hommaa. Sopimus on syytä tehdä kerralla suoraan mallin pohjalta. Mistään hinnasta kenenkään pyynnöstä ei kannata lähteä kaivamaan mallia mistään muualta.

5.4 Jukka Toivanen

Viileä häck. Sairaan hyvä porukka, jolla epäilemättä tehtäisiin minkälainen softa tahansa ja milloin tahansa. Varsinkin alkuvaiheessa oli hankala asennoitua järjettömältä tuntuvaan näennäisen hyödyttömien suunnitteludokumenttien generointiin, johon meni runsaasti hyödyllisestikin käytettävissä olevaa aikaa. Palaverit olivat turhan pitkiä ja joskus niissä meni mielessä lähestulkoon selvät asiat täysin uusiksi.

Vaikeat ja helpot kohdat

Yllättävän vaikeaa oli sovelluksen suunnittelun alkuvaihe, jossa joutui lähtemään liikkeelle täysin tyhjältä pöydältä. Koodailuvaihe oli ajoittain helppoa ja ajoittain hankalaa.

Mitä opin

Opin kaiken käyttötapauksista.

Mitä tekisin toisin jos aloittaisin alusta

Suunnittelun olisi voinut tehdä paremminkinkin, mutta vain, jos olisi tiedetty huomattavasti aiemmin mitä oikein ollaan tekemässä. Object pascalilla heittäisin vesilintua.

Ohjeita tuleviin aikoihin

Tuleviin projekteihin antaisin itselleni iskulauseeksi "Älä häslää".

Tietotekniikan sovellusprojekteja ohjeistaisin: "Tilaaja on aina oikeassa, paitsi kun se on väärässä (ja tällöin tulisi nostaa rohkeasti kissa pöydälle)".

6 Yhteenveto

Tämä dokumentti käsitteli Jyväskylän yliopiston tietotekniikan laitoksella syksyllä 2002 toteutetun Verstas-sovellusprojektin suunnitelmaa ja toteutusta toisiinsa verrattuna. Yhteenvedossa esitetään tärkeimmät aikaisemmissa luvuissa tehdyt havainnot sekä niiden pohjalta johtopäätöksiä ja ajatuksia avuksi ryhmän jäsenten ja tietotekniikan laitoksen tuleviin projekteihin. Havainnot ja ohjeet ovat nopean luettavuuden vuoksi listan muodossa.

6.1 Yleiskuva ja tavoitteiden saavuttaminen

Neljästä opiskelijasta koostuva Verstas-projektityöryhmä suunnitteli ja toteutti Numerola Oy:lle *kokoavan käyttöliittymän mallinnus- ja simulointiohjelmistoille*, *Elisp-ohjelman Emacs-tekstieditorin ohjaamiseksi* sekä *graafisen sovelluksen Fortran-ohjelmien kääntämiseen ja ajamiseen.* Projekti toteutettiin tietotekniikan laitoksen ohjaamana ja valvomana.

Projekti päättyi suunniteltuna ajankohtana, mutta toteutunut aikataulu poikkesi suunnitelmasta suhteellisen paljon (ks. luku 4.2). Viivästymiset johtuivat suurimmaksi osaksi projektiryhmän jäsenten *kokemattomuudesta* projektityössä ja aikataulun suunnittelussa (ks. luku 4.4). Kukaan ryhmän jäsenistä ei ollut aikaisemmin ollut mukana vastaavanlaisessa projektissa.

Projektissa ei kohdattu ylitsepääsemättömiä ongelmia ja se *onnistui parhaalla olosuhteiden mahdollistamalla tavalla.* Projektiryhmä on työn päättyessä sitä mieltä, että tilaaja on saanut projektin hintaa vastaavan työpanoksen ja tuotteen, vaikka kaikki luvussa 3.1 kuvaillut tuotteelle esitetyt tavoitteet eivät välttämättä täydellisesti toteutuneet. Itse ryhmä kokee saaneensa päätoimista opiskelua ajanlaskun ja tietosisällön perusteella enemmän kuin kurssista on opintoviikkoja luvassa. Opetukselliset tavoitteet (ks. luku 3.2) saavutettiin erinomaisesti ja opintojakso oli työmäärästä huolimatta ryhmän jäsenille mieluinen (ks. luku 5).

6.2 Havainnot Kylix-ohjelmistokehitystyökalusta

Kylix ei välttämättä ollut paras mahdollinen toteutustyökalu tässä projektissa. Alla on listattu tärkeimpiä syitä. Täytyy huomata, että Kylix 3.0 ei virallisesti tue projektin työasemissa ollutta Red Hat 7.3 -jakelupakettia linuxista.

- Kylixin avustustoiminto on heikko. Uuden avustussivun aukeaminen kestää sietämättömän pitkään.
- Debuggeri ei toimi hyvin säikeiden kanssa.
- Kylix on aavistuksen epävakaa ja kaatuu joskus.
- Object Pascal sallii omituisuuksia, kuten kokonaisluvun muuttaminen olioreferenssiksi ja olion luominen luokasta, jolla on abstrakteja metodeja.
- Indy-TCP/IP-komponentti, jota sovelluksessa käytettiin, ei tuntunut luotettavalta. Sitä ei saa kunnolla vapautettua, jos joku yhteys on päällä.

Kylixin C++ -IDE saattaisi olla kokeilemisen arvoinen.

6.3 Ohjeita tulevia sovellusprojekteja varten

Ohjelista on järjestetty jotakuinkin siihen järjestykseen, jossa ohjeita tulisi lähteä noudattamaan projektin alussa. Suurin osa ohjeista liittyy projektin aloitukseen, joka onkin kokemusten perusteella kaikista tärkein ja helpoiten ylenkatsottu vaihe.

- Tehtävän näennäistä vaikeutta ei pidä alkuunkaan säikähtää. Kaiken oppii kyllä, kun lähtee rohkeasti tekemään.
- Luentoja tai ensimmäistä palaveria *ei tule jäädä odottamaan.* Projektissa on paljon tehtävää jo ennen kontaktia tilaajaan. Erityisesti projektin toteutusta voi ryhtyä pohtimaan ryhmän keskinäisissä aivoriihissä.
- Projektiryhmän tulisi *heti ensi töikseen* sopia täsmällisesti ja jopa kirjallisesti (projektisuunnitelmassa) työnjako.
- *Ajankäyttövihkot* sekä niihin liittyvä *laskentakäytäntö ja -tarkkuus* kannattaa alustaa yhtenäiseksi. Ajankäytön jako osa-alueisiin kannattaa tehdä mieluummin tarkaksi kuin ylimalkaiseksi.
- *Projektisopimus* tulee tehdä mallin [14] eikä minkään muun mukaiseksi.
- Kokematon ryhmä *ei mitenkään pysty näkemään riskien ilmenemismuotoja ennalta!* Korkeintaan voi tutustua vaikkapa aikaisempien projektien projektiraportteihin. Riskejä ei kannata pohtia liian pitkään. Pikemminkin tulee luottaa siihen että hankaluuksia tulee ja varautua tähän *tehtävän rajauksella* ja *pelivaralla aikataulussa.*
- Suunnitteludokumentteja on syytä alkaa tekemään projektiohjeen perusteella heti, eikä kannata odottaa mitään projektiluentoja.
- Suunnitteludokumenttien tarkoituksesta kannattaa lähteä etsimään tietoa, mutta ei kannata jäädä odottelemaan tiedon löytymistä vaan *lähteä tekemään suunnitelmia.* Tekemällä nimittäin oppii parhaiten ja lisäksi suunnitteluvaiheessa *on erittäin kova kiire* (vaikkei siltä tunnu).
- Jos sovellusprojektin alussa tuntuu, ettei ole kiire, tietää luultavasti keskittyvänsä epäolennaiseen. Pitäisi olla hiki päässä suunnittelemassa.
- Dokumentteja *ei tule puida palavereissa* yhtään. Ne kannattaa toimittaa paperiversiona etukäteen ja palaute ottaa keskitetysti paperilla. Palavereissa on syytä keskustella yleisemmistä asioista ja korkeintaan hyväksyä dokumentit. Käytännöt olisi hyvä sopia pikimmiten projektin alkaessa.
- *Käyttötapaukset* ja *vaatimusmäärittely* on hyvä paikka aloittaa sovelluksen suunnittelu, mutta näihin dokumentteihin ei saisi käyttää liian paljon aikaa. Asiassa helpottaisi tutustuminen projektiohjetta tarkempaan lähdekirjallisuuteen tai materiaaliin jopa ennen projektin alkua.
- Sovellussuunnitelman on syytä olla niin tarkka, *ettei ohjelmoidessa tarvitse enää suunnitella* pienintäkään asiaa. Tämän vuoksi suunnitteluvaiheessa on todella kiire.

- Versionhallinta auttaa pitämään dokumentit järjestyksessä.
- LyX on kätevä työkalu dokumenttien tekoon.
- Tilaaja on aina oikeassa, paitsi kun se on väärässä. Tällöin tulee viivyttelemättä kertoa tilaajalle, missä asiassa ja miksi se on väärässä.
- *Työkaluun tulee tutustua* kunnolla ennen varsinaisen ohjelmointivaiheen alkua. Jotain sillä pitää vähän tehdä, jos se ei ole ennestään tuttu.
- *Tehtävä kannattaa rajata suppeaksi*, koska toteutukseen käytettävä aika on todella rajallinen. Mieluummin pieni hyvä tuote kuin iso ahdistus. Verstas kävi rajalla.
- Palavereissa olisi hyvä tunnistaa epäolennainen saivartelu ja lopettaa sellainen mahdollisimman lyhyeen.

Viitteet

[1] *An Introduction to Programming in Emacs Lisp*, saatavilla WWW-muodossa <URL: `http://www.gnu.org/manual/emacs-lisp-intro/html_node/index.html`>, viitattu 18.12.2002

[2] Balaji, V, *Mkmf user's guide*, saatavilla WWW-muodossa <URL: `http://www.gfdl.gov/~vb/mkmf.html`>, viitattu 18.12.2002.

[3] Becks, Ari, *Delphi - sovelluksentekijän opas*, Gummerus kirjapaino Oy, Jyväskylä, 1995.

[4] *CVS - Concurrent Versions System*, saatavilla WWW-muodossa <URL: `http://www.loria.fr/~molli/cvs/doc/cvs_toc.html`>, viitattu 18.12.2002.

[5] Cantù, Marco, *Mastering Delphi 5*, 1999.

[6] Gamma, E, Helm, R, Johnson, R, Vlissides, J, *Design Patterns - Elements of Reusable Object-Oriented Software*, 2001.

[7] *GNU Emacs Lisp Reference Manual,* saatavilla WWW-muodossa <URL: `http://www.gnu.org/manual/elisp-manual-21-2.8/html_node/elisp.html`>, viitattu 18.12.2002

[8] *Lyx-LaTex Tips & Tricks*, saatavilla WWW-muodossa <URL: `http://www.educat.hu-berlin.de/~voss/lyx/index.phtml`>, viitattu 18.12.2002.

[9] Ross, Rick, *Creating Fork and Exec Classes with Kylix*, saatavilla WWW-muodossa <URL: `http://www.rick-ross.com/papers/borcon2001/6204.html`>, viitattu 18.12.2002.

[10] Ross, Rick, *Creating Message Passing IPC Classes with Kylix*, saatavilla WWW-muodossa <URL: `http://www.rick-ross.com/papers/borcon2001/6206.html`>, viitattu 18.12.2002.

[11] Santanen, Jukka-Pekka, *Tietotekniikan Sovellusprojektien ohje,* saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/opetus/sovellusprojektit/projohje.html`>, viitattu 17.12.2002.

[12] Santanen, Jukka-Pekka, Gradupohja *Microsoft Wordin template-tiedosto ja lyhyt ohje tyylien käyttöön*, saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/progradut/tyylipohjat/gradupohja.dot`>, viitattu 30.10.2002.

[13] Santanen, Jukka-Pekka, *Opinnäytteiden kirjoittaminen, lyhyt oppimäärä*, saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/santanen/info/kirjoittamisesta.html`>, viitattu 17.12.2002

[14] Santanen, Jukka-Pekka, *Projektisopimusmalli*, saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/palvelut/sovellusprojektit/sopimus.html`>, viitattu 17.12.2002

[15] Schoonover, Bowie, Arnold: *GNU Emacs, UNIX Text Editing and Programming*, Addison-Wesley, 1992

[16] Software Engineering Standards Commitee of the IEEE Computer Society, *IEEE Recommended Practice for Software Requirements Specifications IEEE Std 830-1998,* 1998.

[17] *Starting an external application with Kylix*, saatavilla WWW-muodossa <URL: `http://community.borland.com/article/0,1410,27500,00.html`>, viitattu 18.12.2002.

[18] Tarvainen, Tapani, *UNIX ja shell-ohjelmointi kevät 2001*, saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/opiskelu/kurssit/unixshell01/index.html`>, viitattu 18.12.2002.

[19] Verstas projekti, *Vaatimusmäärittely*, 2002

[20] Verstas projekti, *Käyttötapaukset*, 2002

[21] Vestas projekti, *Projektisuunnitelma*, 2002

[22] Vestas projekti, *Sovellussuunnitelma*, 2002

[23] Vestas projekti, *Sovellusraportti*, 2002

[24] Vestas projekti, *Testaussuunnitelma*, 2002

[25] Vestas projekti, *Testausraportti*, 2002

[26] Vestas projekti, *Verstas-sovelluksen käyttöohje*, 2002

[27] Vestas projekti, *Asennusohje*, 2002

[28] Vestas projekti, *Projektiraportti*, 2002

[29] *XEmacs Lisp Reference Manual*, saatavilla WWW-muodossa <URL: `http://www.xemacs.org/Documentation/21.5/html/lispref.html#SEC_Top`>, viitattu 18.12.2002