VERSTAS-PROJEKTI

Sovellussuunnitelma

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-projektin sovellussuunnitelma
Työ	Tietotekniikan sovellusprojekti

Tiivistelmä:

Tämä dokumentti on Jyväskylän yliopistossa syksyllä 2002 toteutettavan Verstas-sovellusprojektin sovellussuunnitelma. Suunnitelmassa kuvataan toteutettavan sovelluksen arkkitehtuuria ja käyttöliitymää.

Avainsanat: mallinnusympäristö, käyttöliittymä, arkkitehtuuri, luokkakaavio, sekvenssikaavio

Versiohistoria

Versio	Pvm	Tekijä	Kuvaus
0.1	30.10.2002	JT, TL	Ensimmäinen raakaversio
0.2	15.11.2002	AH	Rajapinnat: korjattu ja lisätty kaavio, esimerkkejä ja huomioita.
0.3	22.11.2002	TL	Tehty Kari Kärkkäisen esittämät muutokset.

Tekijöiden lyhenteet

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

Hyväksytty

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

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

Sisältö

1 Johdanto

Verstas-projekti suunnittelee ja toteuttaa Numerola Oy:lle yhtenäisen ja kokoavan käyttöliittymän mallinnus- ja simulointiohjelmistoille. Projekti toteutetaan Jyväskylän yliopiston tietotekniikan sovellusprojektina. Projektiryhmään kuuluvat Antti Hakala, Tomi Laamanen, Paavo Nieminen ja Jukka Toivanen. Projektin vastaavana ohjaajana toimii Kari Kärkkäinen ja teknisenä ohjaajana Ville Tirronen.

Tämä suunnitelma kuvaa Verstas-projektin toteuttaman sovelluksen arkkitehtuurin ja käyttöliittymän. Sovellussuunnitelman luvussa 2 käydään läpi Verstas-sovelluksen arkkitehtuurin osat paketti- ja luokkakaavioin. Luku 3 esitttelee Verstas-sovelluksen ja sen alisovellusten rajapinnat. Luvussa 4 kuvataan Verstas-sovelluksen toiminnallisuutta sekvenssikaavioin. Luvussa 5 kuvataan sovelluksen käyttöliittymää.

1.1 Termit ja käsitteet

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.

Geometrian luonti: Mallinnusprosessin vaihe, jossa tutkittavan fysikaalisen järjestelmän ominaisuudet pyritään kuvaamaan tietokoneohjelman avulla. Tuloksena on ns. laskentageometria.

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

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

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.

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).

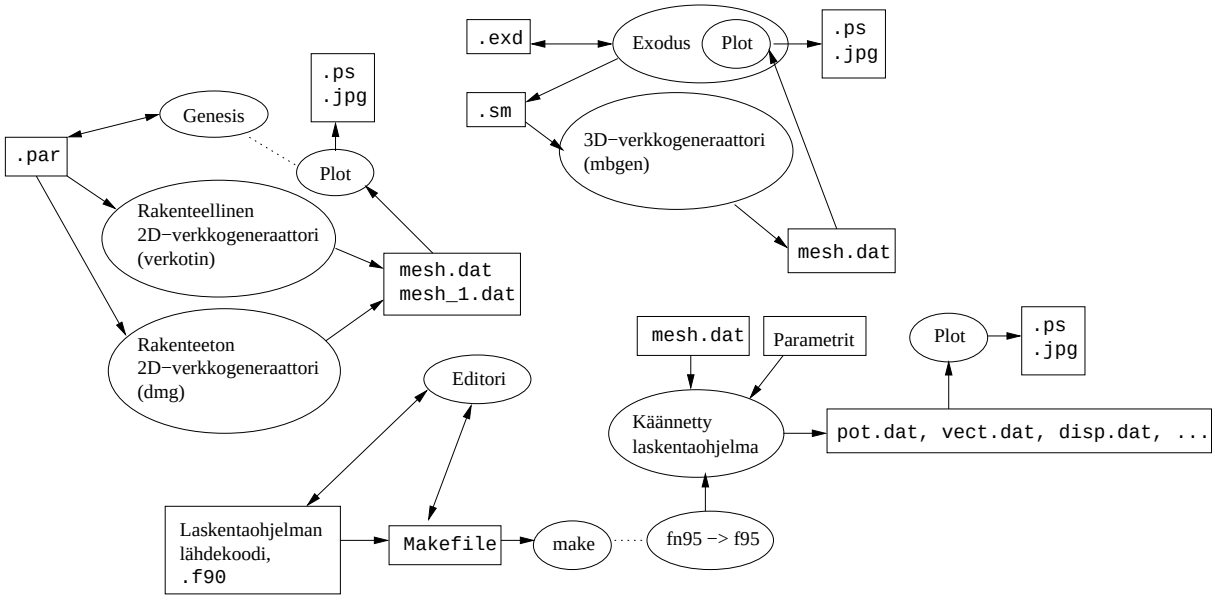
Raportti: Tutkimus-, päivitys- tai muu raportti, esim. pdf-muodossa.

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

1.2 Numerrin-järjestelmä

Nykyinen Numerrin-simulointiohjelmisto koostuu useista alisovelluksista. Verstas-sovellus kommunikoi erillisten alisovellusten kanssa ja huolehtii niiden yhteiskäytöstä sekä syöttö- ja tulostustiedostojen hallinnasta. Myöhemmin tullaan tekemään uusia alisovelluksia liitettäväksi Verstaaseen, joten kommunikoinnin suunnittelussa huomioidaan myöhempi laajennettavuus.

Alisovellukset ottavat vastaan syötetiedostoja ja tulostavat tiedostoja (kts. kuva 1), jotka jälleen menevät syötteeksi uusille alisovelluksille. Tärkeä osa Verstaan toiminnallisuutta on tiedostojen hallitseminen ja liittäminen asiaankuuluviin alisovelluksiin.



Kuva 1: Tietovirrat Numerrin-järjestelmässä

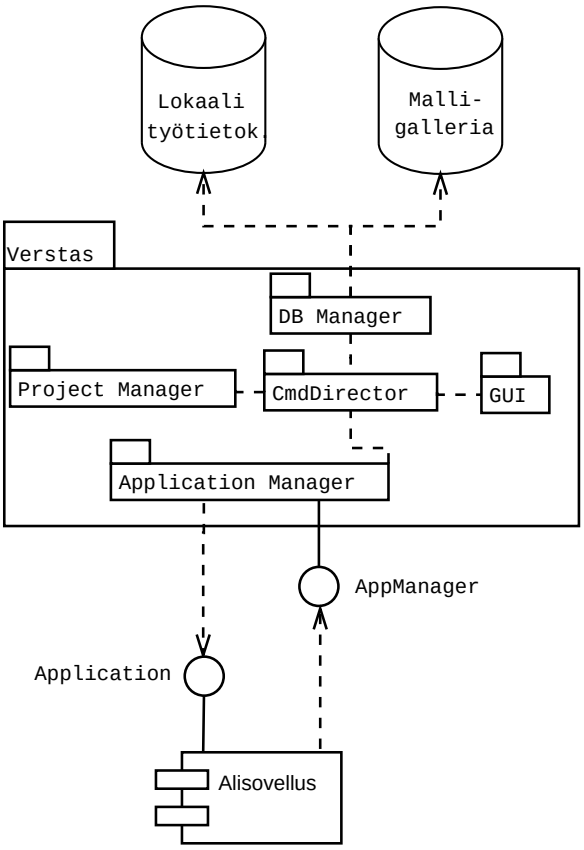
2 Verstas-arkkitehtuuri

Tässä luvussa kuvataan järjestelmän arkkitehtuuria paketti- ja luokkakaavioin. Lisäksi esitellään tärkeimpien luokkien metodit pääpiirteissään.

2.1 Arkkitehtuuri

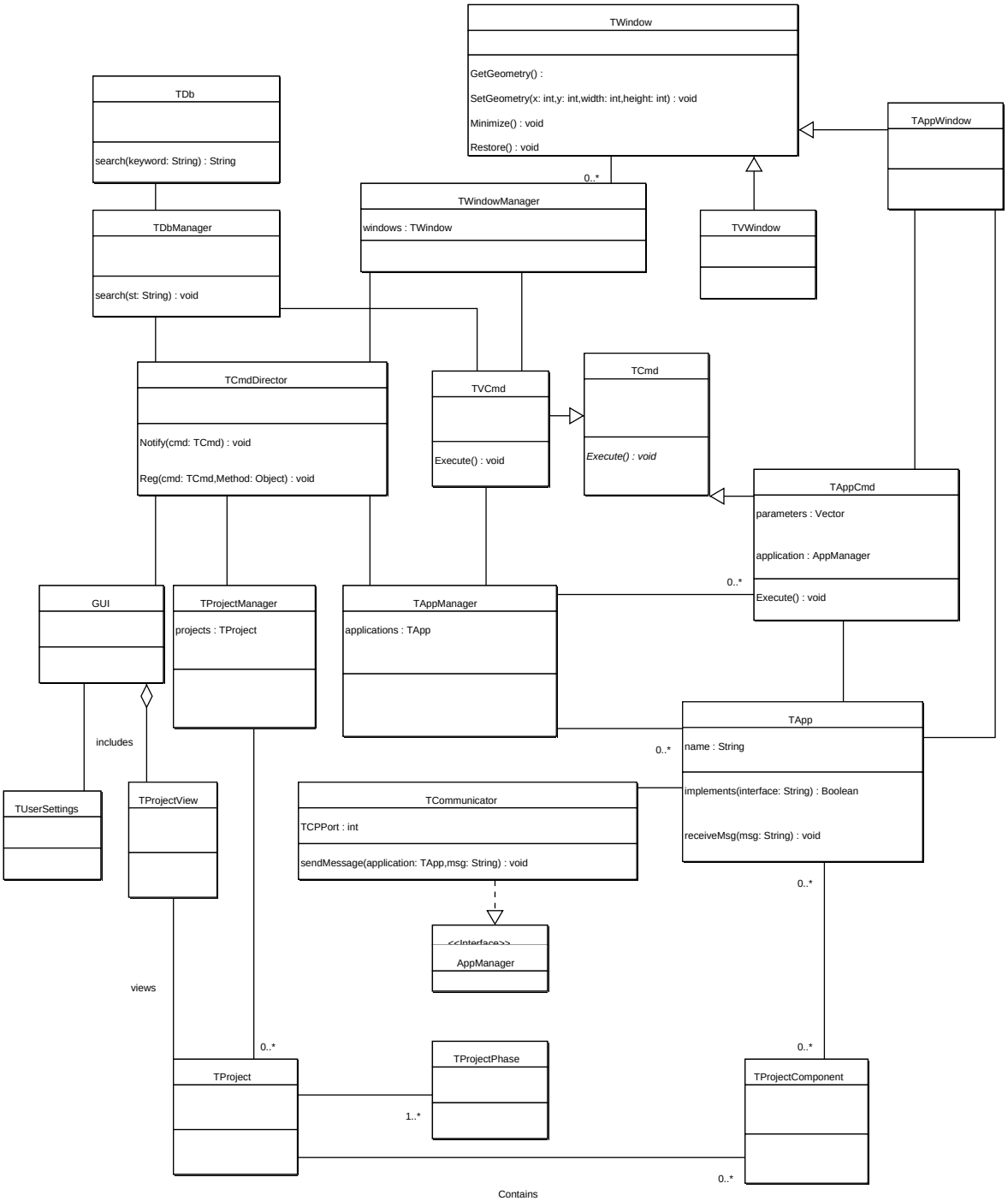
Kuvassa 2 on esitetty Verstaan arkkitehtuuria pakettikaavion muodossa. Verstas jakautuu osiin siten, että projektien hallinnasta, alisovellusten käynnistämisestä ja tietokantojen hallinnasta vastaa kustakin oma manageri. Managerien välistä viestiliikennettä hoitaa Mediator-mallin [3] mukaisesti oma komponentti. GUI on eritetty verstaan muista osista siten, että viestiliikenne hoidetaan välittäjäkomponentin kautta.

Kommunikointi alisovellusten kanssa eristetään muusta verstaasta siten, että viestien välityksen toteutustapa voidaan tarvittaessa vaihtaa. Kommunikointia varten määritetään oma rajapinta (kts. kohta 3) sekä alisovelluksille että Verstaan viestiliikennekomponentille.



Kuva 2: Verstaan arkkitehtuuri pakettikaaviona

2.2 Luokat



Kuva 3: Luokkakaavio

Kuvassa 3 on esitetty verstaan tärkeimmät luokat luokkakaavion muodossa. Kaaviossa ei ole pyritty kuvaamaan lopullisia luokkien välisiä riippuvuuksia koodissa, vaan ennemminkin osien välisiä loogisia suhteita. Metodeista ja attribuuteista on esitelty vain tärkeimmät.

Seuraavassa esitellään Verstaan tärkeimmät luokat pääpiirteissään.

2.2.1 TProject

`TProject` kuvaa yhtä projektia. Projektiin liittyy erinäinen määrä `TProjectComponent`-olioita, jotka kuvaavat projektiin kuuluvia komponentteja. Komponentit voidaan avata ja uusia komponentteja voidaan luoda vastaavassa alisovelluksessa. Komponentteja voidaan hakea gallerioista `TDbManagerin` avulla.

2.2.2 TProjectManager

`TProjectManager` kuvaa avoinna olevien projektien joukkoa. Täsmälleen yksi projekteista voi kulloinkin olla aktiivinen ja vain siihen voi tehdä muutoksia.

2.2.3 TAppManager

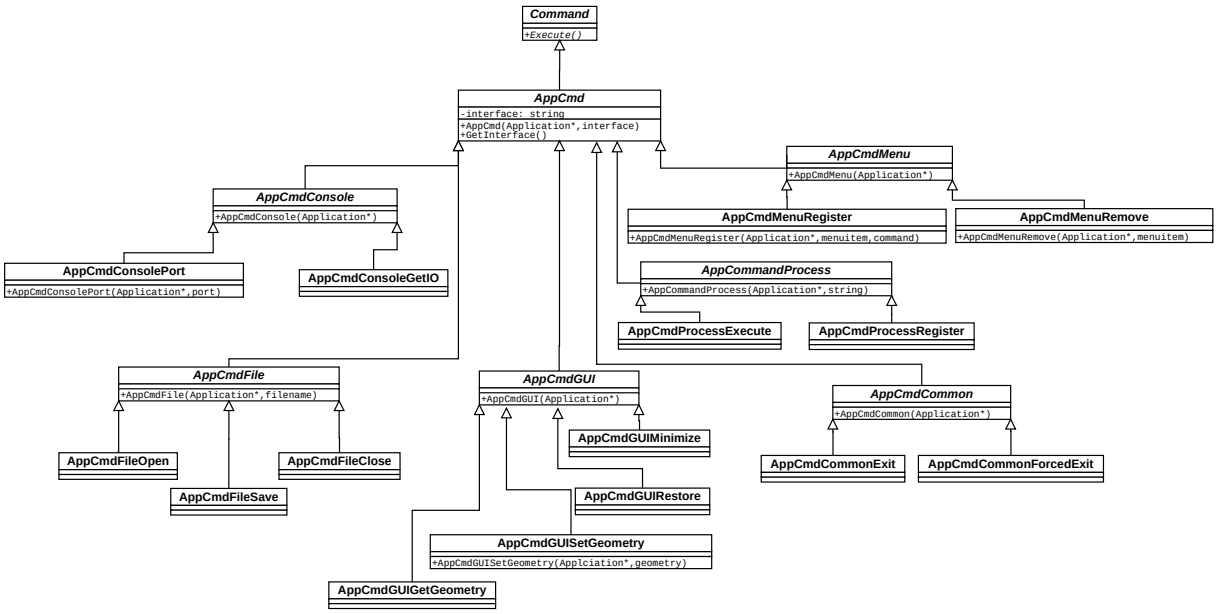
`TAppManager` käynnistää alisovelluksia ja hallitsee niitä. `TAppManager` tietää kaikki Verstaan käytössä olevat alisovellukset sekä sen, kuinka ne voidaan käynnistää. `TAppManager` myös osaa palauttaa tiedoston nimen perusteella tiedon siitä, missä alisovelluksessa se oletusarvoisesti tulee avata.

2.2.4 TWindowManager

`TWindowManager` hallitsee ikkunoita. Ikkunat voivat olla joko Verstaan omia (luokka `TVWnd`) tai alisovellusten ikkunoita (luokka `TAppWnd`). Ikkunat toteuttavat samat metodit, kuten `SetGeometry`, `Minimize` ja `Restore`. Ikkunoiden välinen ero on metodien toteutuksessa: `TVWnd` toteuttaa metodit itse, kun taas `TAppWnd` luo alisovelluksia käskyttäviä `TAppCmdGUI`-komentoja.

`TWindowManager` tarjoaa tuen useiden työpöytien käytölle. Tällöin ikkunat kuuluvat aina jollekin työpöydälle, mutta voidaan myös määrittää näkymään kaikilla työpöydillä. Manageri tarjoaa tavallisimmat ikkunoiden järjestelykomennot, kuten Tile, Cascade ja Minimize all, jotka toteutetaan työpöydittäin.

2.2.5 TCmd



Kuva 4: Alisovelluskomentoluokkien periytyminen

`TCmd` on Command-mallin [3] mukainen komentoluokka, jonne kapseloidaan kommennon suorittamisen vaatima toiminnallisuus. `TCmd` Luokasta peritään luokka `TAppCmd`, josta edelleen kaikki alisovellusten käskyttämiseen tarvittavat komennot (kts. kuva 4), sekä luokka `TVCmd`, josta peritään verstaan sisäiset komennot. Katso myös kohta 4.2.

Kaikki komentoluokat suoritetaan `Execute`-metodilla joka on määritelty komentohierarkian ylimmällä tasolla. Ylimmän tason komentoluokka `TCmd` on abstrakti, mutta sen `Execute`-metodissa on määritelty kaikille komennoille yhtenäinen toiminnallisuus (komennon suorituksen ilmoittaminen `TCmdDirectorille`). `TCmd`-luokan `Execute`-metodi taas kutsuu abstraktia `DoExecute` metodia, johon alemmat komentoluokat määrittävät toiminnallisuutensa. Tätä tapaa jakaa metodin toiminnallisuutta luokkahierarkiassa kutsutaan Template Methodiksi (kts. [3]).

2.3 Assosiaatioiden toteutus

Verstaan sisällä tulee tilanteita, joissa eri osat liittyvät toisiinsa monesta moneen -assosiaatioilla. Nämä toteutetaan aina siten, että vastaava manageri tarjoaa kaikki riittävät saantimetodit. Tällöin assosiaatioon liittyvä tieto talletetaan vain assosiaation toiseen osapuoleen.

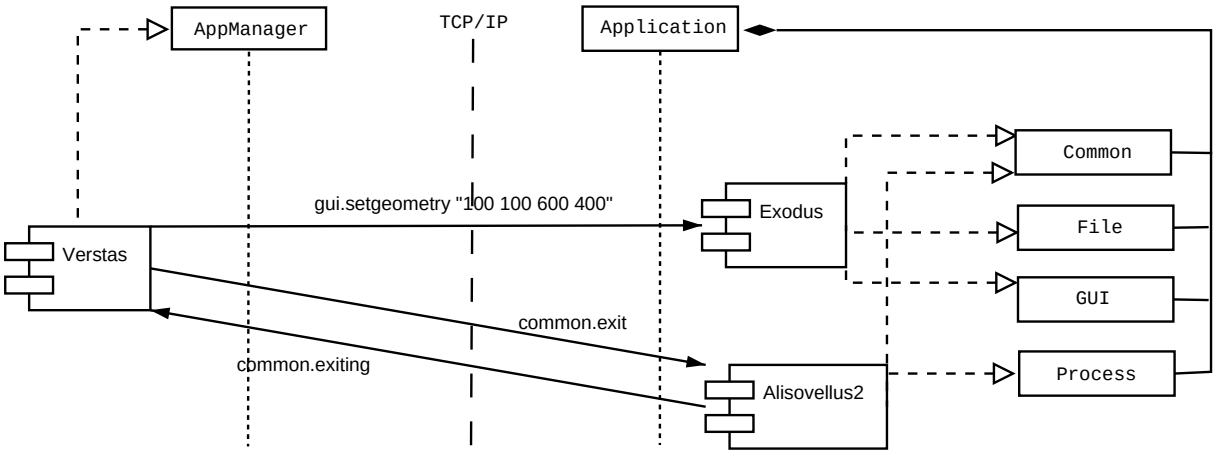
Esimerkki: `TApp` tietää kaikki edustamassaan alisovelluksessa auki olevat tiedostot sekä sen, mihin vaiheeseen alisovellus kuuluu. Siispä tiedostot eivät tiedä missä alisovelluksessa ne ovat auki eivätkä vaiheet tiedä mitä alisovelluksia niissä on auki. `TAppManager` tarjoaa funktiot `GetApps(TFile)` ja `GetApps(TProjectPhase)`, joilla vastaavat tiedot voidaan kysyä. Tämän lisäksi `TAppManager` tarjoaa toteutusvaiheen edetessä mahdollisesti tehtävien muutosten helpottamiseksi myös funktiot `GetFile(TApp)` ja `GetPhase(TApp)`.

3 Rajapinnat

Rajapintojen määrittely jakautuu kahteen osaan: `Application` ja `AppManager`. `Application`-rajapinta on rajapinta joka alisovelluksen on toteutettava, jotta se on se liitettävissä Verstaaseen. `AppManager` on rajapinta, jonka Verstas toteuttaa ja jonka komentoja käyttäen alisovellus ilmoittaa tapahtumistaan.

`Application`-rajapinta jakautuu alirajapintoihin `common`, `file`, `gui`, `console` ja `process`. Alisovellus ilmoittaa käynnistyessään (`AppManager`-rajapinnan `common.starting`-komennolla) mitkä alirajapinnat se toteuttaa. Näin Verstas saa tiedon mm. siitä onko alisovelluksella graafista käyttöliittymää, joka sen on otettava huomioon työpöytää järjestettäessä.

Käytännössä alisovellus ottaa käynnistyessään TCP/IP-yhteyden Verstaaseen ja rajapintojen komennot välitetään tekstimuodossa Verstaan ja alisovelluksen välillä.



Kuva 5: Rajapinnat ja ajonaikainen toiminta

3.1 Application

Toteuttaakseen `Application`-rajapinnan alisovelluksen tulee toteuttaa:

1. **Komentoriviparametri**t:
   `-iconic` Alisovellus käynnistyy pienennettynä.

   Lisäksi alisovellus pystyy käynnistettäessä asettamaan ikkunoidensa geometrian niillä komentoriviparametreillä jotka alisovellus antaa saadessaan komennon `common.getargs`. Ikkunat palautuvat samaan tilaan kuin missä ne olivat alisovelluksen saadessa komennon.

2. **Luvuissa 1.1-1.5 esitellyistä rajapinnoista vähintään Common-rajapinta.**

Rajapinnat on esitelty seuraavasti:

`komento “parametri1” “parametri2”...`	toimenpide

Komennon saadessaan alisovelluksen on toimittava toimenpiteen mukaan. Jos alisovellus toteuttaa rajapinnan, on sen toteutettava rajapinnan kaikki komennot ja täytettävä ilmoitusvelvollisuudet. Ilmoitukset on tehtävä mainitulla `AppManager`-rajapinnan ilmoituksella (ilmoitettu hakasuluissa).

3.1.1 Common

Komennot:

`common.exit`	Alisovellus sulkeutuu
`common.forcedexit`	Alisovellus sulkeutuu kysymättä käyttäjältä mitään.
`common.getargs`	Alisovellus palauttaa komentoriviparametrien muodossa tietoja, joiden avulla alisovellus pystyy käynnistyessä palauttamaan geometriansa (kts. Esim. 1, kohdat 5-6).
`common.ping`	Kysely alisovellukselta, alisovelluksen tulee vastata `common.pong`.
`common.pong`	(Verstaan vastaus alisovelluksen `common.ping`:iin. Ei vaadi toimenpiteitä alisovellukselta)

Ilmoitusvelvollisuudet:

- Alisovellus käynnistyy [`common.starting`]
- Alisovellus sulkeutuu [`common.exiting`]

3.1.2 File

`file.open "filename"`	Alisovellus avaa tiedoston.
`file.save "filename"`	Alisovellus tallentaa tiedoston.
`file.close "filename"`	Alisovellus sulkee tiedoston.

Huomioita:

- Verstas olettaa ettei alisovelluksessa voi olla sama tiedosto auki monesti.
- Jos alisovelluksessa tallennetaan tiedosto "vanha" nimelle "uusi" (SaveAs): Alisovellus ilmoittaa Verstaalle `file.closed "vanha"`, `file.opened "uusi"`, `file.saved "uusi"`.
- Jos alisovelluksessa luodaan uusi tiedosto jota ei ole vielä talletettu, alisovelluksen tulee ilmoittaa `file.modified "untitled"`. Jos tiedosto talletetaan, toimitaan yo. tavalla.

Ilmoitusvelvollisuudet:

- tiedoston avaaminen [`file.opened`]
- tiedoston tallentaminen [`file.saved`]
- tiedoston sulkeminen [`file.closed`]

- tiedoston muuttaminen [`file.modified`]

3.1.3 GUI

`gui.setbounds "Width Height X Y"`	Alisovellus asettaa ikkunansa annetun alueen sisään. (`X`,`Y`) on alueen yläkulma, `Width` alueen leveys ja `Height` alueen korkeus pikseleissä.
`gui.iconify`	Alisovellus pienenee.
`gui.restore`	Alisovellus palautuu.

3.1.4 Process

`process.execute "command"`	Alisovellus suorittaa merkkijonolla tunnistettavan komennon, jonka se on aikaisemmin ilmoittanut verstaalle.

3.2 AppManager

AppManager toteuttaa seuraavat rajapinnat:

3.2.1 Common

`common.starting "name" "interface1" "interface2"...`	Alisovellus kertoo käynnistyneensä, ilmoittaa tunnisteensa ja rajapinnat jotka se toteuttaa. `Common`-rajapintaa ei ilmoiteta, koska se on kaikille alisovelluksille pakollinen.
`common.exiting`	Alisovellus kertoo sulkeutuvansa.
`common.ping`	Alisovellus "pingaa" Verstasta. Verstaan on vastattava komennolla `common.pong`.
`common.pong`	Alisovelluksen vastaus Verstaan komentoon `common.ping`.
`common.args`	Alisovellus antaa merkkijonon, joka voidaan käynnistettäessä liittää sen komentoriviparametreihin (kts. esim 1, kohdat 6 ja 11).

3.2.2 File

`file.opened "filename"`	Alisovellus kertoo avanneensa tiedoston.
`file.saved "filename"`	Alisovellus kertoo tallentaneensa tiedoston.
`file.closed "filename"`	Alisovellus kertoo sulkeneensa tiedoston.
`file.modified "filename"`	Alisovellus kertoo ettei tiedoston tallennettu versio vastaa alisovelluksessa olevaa.

3.2.3 Message

`message.error "errormessage"`	Alisovellus pyytää Verstasta näyttämään käyttäjälle virheilmoituksen.
`message.inform "message"`	Alisovellus pyytää Verstasta näyttämään käyttäjälle viestin.

3.2.4 Menu

`menu.register "menuitem" "command"`	Alisovellus lisää Verstaaseen pikavalinnan sekä kertoo komennon, jonka haluaa liittää siihen. Alisovelluksen on toteutettava `process`-rajapinta, jonka kautta ilmoitus valikon käytöstä tulee (`process.execute "command"`).
`menu.remove "menuitem"`	Alisovellus poistaa pikavalintansa Verstaasta.

Esim 1.

1. Käyttäjä käynnistää Exoduksen Verstaasta. Alisovellus käynnistyy ja lähettää verstaalle ilmoituksen `common.starting "exodus" "gui" "file"`.
2. Käyttäjä työskentelee jonkin aikaa alisovelluksella.
3. Verstas kokeilee vieläkö alisovellus kuuntelee ja antaa alisovellukselle komennon `common.ping`.
4. Alisovellus vastaa `common.pong`.
5. Käyttäjä tallentaa projektin, jolloin Verstas antaa alisovellukselle komennon `common.getargs`.
6. Alisovellus palauttaa `common.args "--args 100,100,400,200,100,300,200,80"` (tiedot ikkunoidensa sijainneista ja koista). Verstas tallentaa tiedon projektitiedostoon.
7. Käyttäjä sulkee Verstaan, jolloin Verstas antaa alisovellukselle komennon `common.exit`.
8. Alisovellus palauttaa `common.exiting` ja sulkeutuu.
9. Verstas sulkeutuu.
10. Käyttäjä käynnistää Verstaan ja avaa edellisen projektin.
11. Verstas käynnistää Exoduksen: `exodus -args 100,100,400,200,100,300,200,80`.
12. Exodus käynnistyy samaan tilaan kuin missä se oli saadessaan komennon `common.getargs`.

Esim. 2:

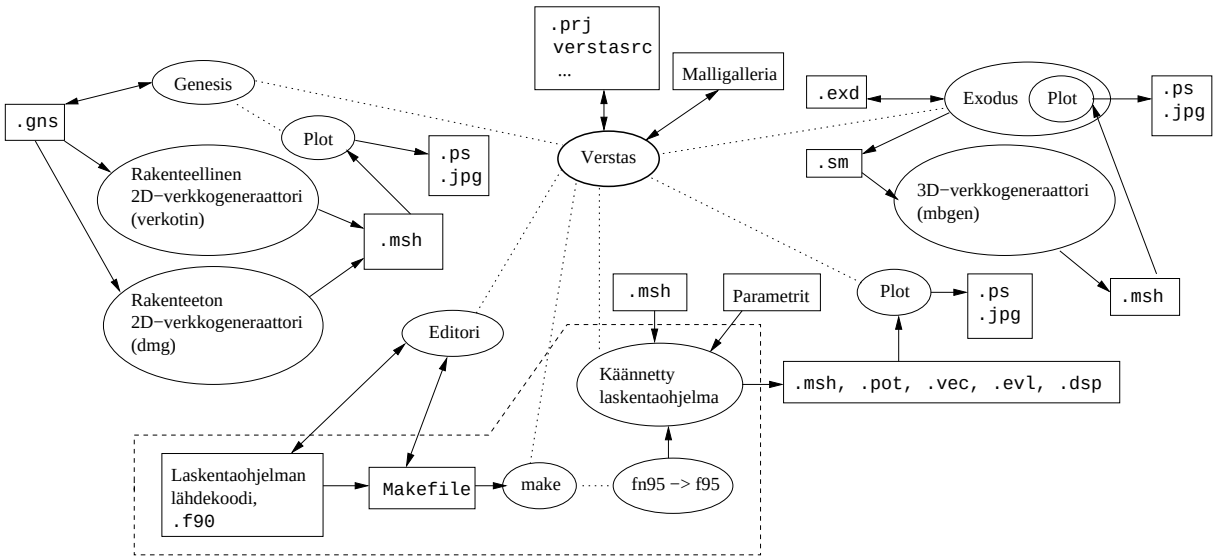
1. Alisovellus rekisteröi verstaaseen pikavalinnan komennolla: `menu.register "Run" "run_cmd"`.
2. Käyttäjä valitsee Verstaasta alisovelluksen pikavalinnan Run.
3. Verstas lähettää alisovellukselle komennon `process.execute "run_cmd"`.
4. Alisovellus suorittaa toimenpiteen joka oli tarkoitus yhdistää Run-pikavalintaan.

4 Toiminnallisuus

Tässä luvussa kuvataan Verstas-järjestelmän toiminnallisuutta tietovirtakaavion ja esimerkkitapauksia kuvaavien sekvenssikaavioiden avulla.

4.1 Tietovirrat Verstas-järjestelmässä

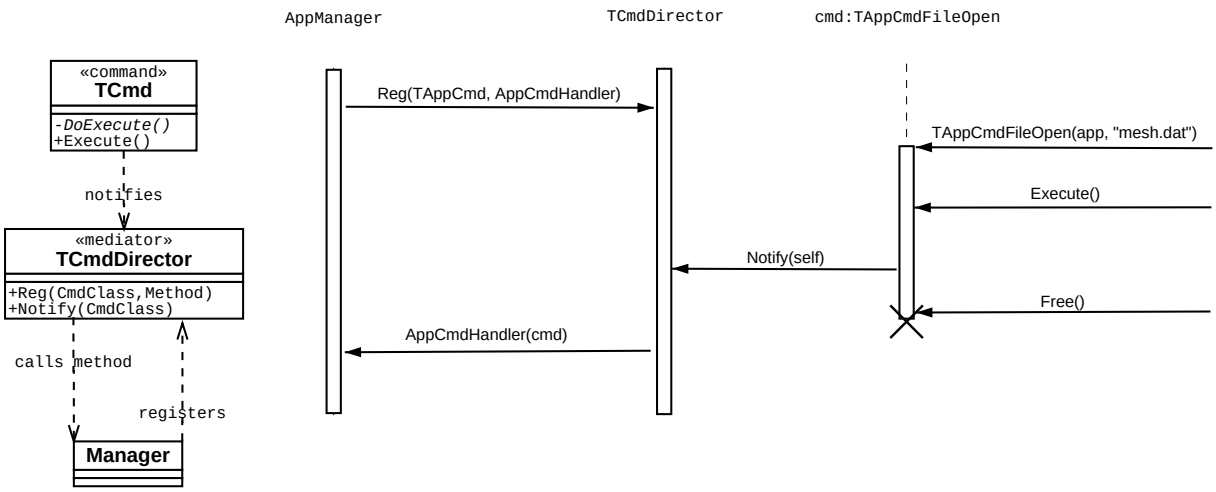
Kuvassa 6 on esitelty tietovirrat, kun Numerrin-pakettia käytetään Verstaan avulla. Verstas hoitaa alisovellusten käynnistämisen ja varmistaa, että projektiin kuuluvat tiedostot avautuvat automaattisesti oikeassa alisovelluksessa. `Makefilen` tuottamisesta ja maken käynnistämisestä sekä laskentaohjelman suorittamisesta vastaa oma itsenäinen Solver-sovellus, jonka projekti toteuttaa.



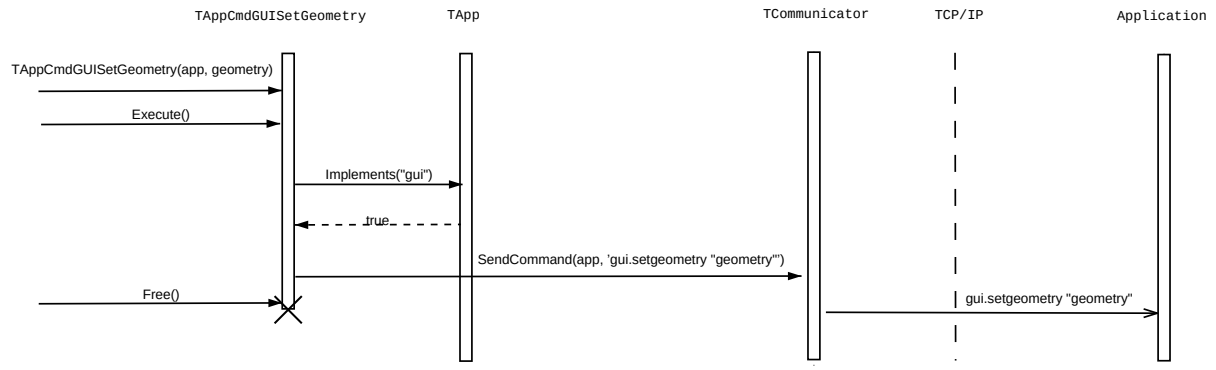
Kuva 6: Tietovirrat Verstas-järjestelmässä

4.2 Komentojen välittäminen

Suoritettuja komentoja voidaan välittää tiedoksi kaikille asiasta kiinnostuneille luokille. Tiettyä komentoa kuunteleva olio rekisteröi komentoa vastaavan käsittelijä `TCmdDirector` -luokalle. Olio voi rekisteröityä kuuntelemaan jotakin konkreettia komentoa kuten `TAppCmdFileOpen` (tiedostonavauskomento alisovellukselle) tai abstraktia yläluokkaa kuten `TVCmd` (kaikki verstaan sisäiset komennot). Kun komento suoritetaan, siitä viedään aina tieto `TCmdDirectorille`, joka välittää tiedon edelleen kaikille, jotka ovat rekisröityneet kuuntelemaan kyseistä komentoa.



Kuva 7: Komennon välittäminen käsittelijän rekisteröineille kuuntelijoille



Kuva 8: Komennon välittäminen alisovellukselle

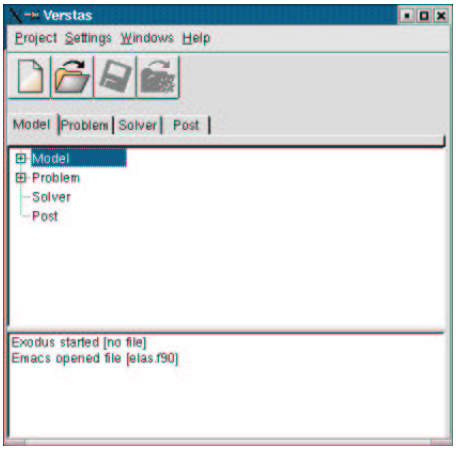
5 Käyttöliittymä

Tässä luvussa esitellään Verstas-järjestelmän alustavaa käyttöliittymää, jonka pohjalta käyttöliittymän kehitystyötä tehdään.

5.1 Pääikkuna

Verstas-sovelluksen pääikkunan Project-valikosta voidaan luoda, avata, tallentaa ja sulkea projekteja, lisätä projektiin komponentteja, käynnistää alisovelluksia tai etsintätoiminto sekä poistua sovelluksesta. Projektin luonti, avaaminen, tallentaminen ja komponentin lisääminen voidaan tehdä myös pikavalintapalkin painikkeilla. Settings-valikosta siirrytään käyttäjäasetuksiin. Windows-valikosta järjestellään, pienennetään ja palautetaan ikkunoita tai aktivoidaan haluttu alisovelluksen ikkuna. Help-valikosta voidaan käynnistää avustustoiminto ja näyttää yleisiä tietoja Verstas-sovelluksesta.

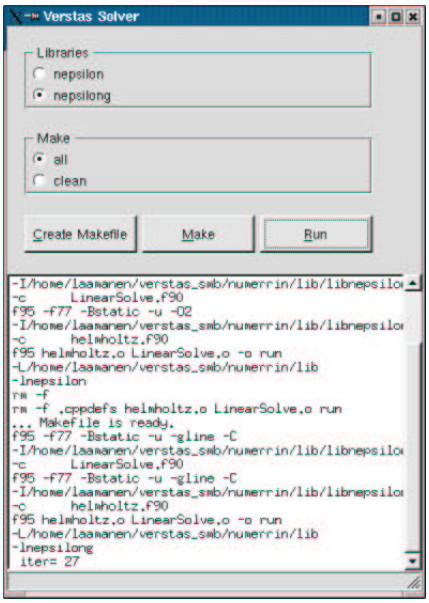
Välilehdillä siirrytään mallinnusvaiheesta toiseen, vain aktiivista vaihetta vastaavat alisovellukset ovat näkyvissä. Puunäkymässä näkyvät hierarkisesti projektin kuhunkin mallinnusvaiheeseen ja alivaiheeseen liitetyt komponentit. Komponentteja voidaan lisätä ja poistaa sekä käynnistää komponentti sitä vastaavaan alisovellukseen. Teksti-ikkunassa näkyvät alisovelluksilta tulevat ilmoitukset.



Kuva 9: Verstaan käyttöliittymä

5.2 Laskenta-alisovellus

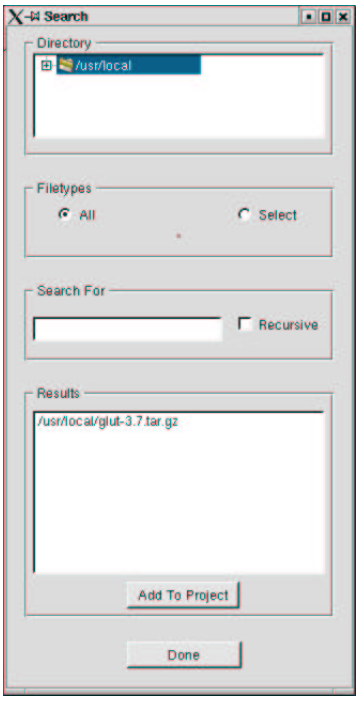
Laskenta-alisovellus Solver on ulkoisia ohjelmia hyödyntävä sovellus, jolla hallitaan kääntämistä ja laskentaa. `Makefile`-tiedosto luodaan Create Makefile -painikkeella Libraries-kohdasta valitun kirjaston mukaisesti. Make-ohjelma käynnistetään Make-painikkeella (all: kääntäminen, clean: käännöksessä syntyneiden tiedostojen poistaminen). Käännetty laskentaohjelma käynnistetään Run-painikkeella, joka muuttuu laskennan ajaksi Stop-painikkeeksi, jolla laskenta voidaan tarvittaessa keskeyttää hallitusti tai pakotetusti. Ulkoisten ohjelmien tulosteet virheineen näkyvät teksti-ikkunassa.



Kuva 10: Laskenta-alisovelluksen käyttöliittymä

5.3 Etsintädialogi

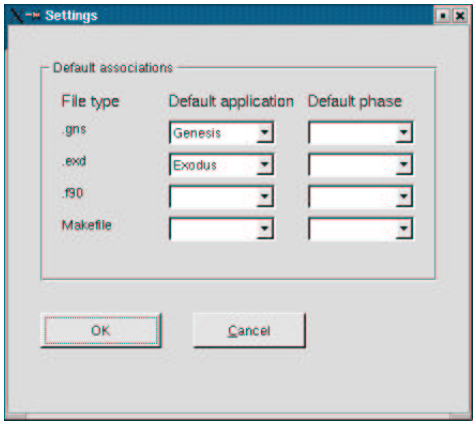
Etsintädialogilla voidaan etsiä valmiita mallinnuskomponentteja tiedostonimen tai tiedostossa olevan kuvauksen perusteella. Etsittävien tiedostojen tyyppi voidaan valita tai etsiä kaiken tyyppisiä tiedostoja. Etsinnän tulokset näkyvät Results-ikkunassa, josta halutut komponentit voidaan valita ja lisätä projektiin Add To Project -painikkeella.



Kuva 11: Asteittain tarkentava etsintädialogi

5.4 Asetukset

Asetukset-dialogista valitaan kutakin tiedostotyyppiä vastaava alisovellus ja mallinnusvaihe.



Kuva 12: Asetuksien dialogi

6 Yhteenveto

Tässä dokumentissa määriteltiin Verstas-projektin sovelluksen toteutusta. Verstas-sovellusta toteutettaessa edetään sovellussuunnitelman mukaisesti. Sovellussuunnitelmassa on kuvattu Verstas-sovelluksen arkkitehtuuria, rajapintoja, toiminnallisuutta sekä käyttöliittymää. Luokka- ja komponenttijakoa on dokumentissa tarkasteltu luokka- ja komponenttikaavioin. Järjestelmän toiminnallisuutta on kuvattu sekvenssikaavioin.

Viitteet

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

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

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

[4] 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.

[5] Santanen Jukka-Pekka, *Opinnäytteiden kirjoittaminen, lyhyt oppimäärä*, saatavilla WWW-muodossa <URL: `http://www.mit.jyu.fi/santanen/info/kirjoittamisesta.html`>, viitattu 30.10.2002.