1. Johdanto

Viimeaikainen informaatiotekniikan nopea kehitys on ollut opetuksen ja opetusmenetelmien tutkimuksen lähtökohtana opetuksen eri aloilla ja kaikilla opetuksen eri tasoilla. Uudet menetelmät ja työkalut sekä toimintaympäristöjen muuttuminen ovat johtaneet opetuksen laadun, määrän ja menetelmien uudelleenarvioimiseen.

Kun Internet, erilaiset sähköiset ilmoitustaulut, sähköposti ja multimedia ovat tulleet arkisiksi työkaluiksi tämän päivän opiskelijoille, on näiden informaatiotekniikan saavutusten käyttö opetuksessa tullut haasteeksi opettajille ja tutkijoille. Menetelmien vaikutus ja tehokkuus ei kuitenkaan ole vielä täysin selkiintynyt ja kattavasti dokumentoitu. On kuitenkin olemassa lupaavia esimerkkejä siitä, että tällaisten menetelmien käyttö lisää opiskelumotivaatiota ja parantaa opiskelutehokkuutta [16, 19, 21, 24]. Näin ollen tulevaisuuden näkymät ovat erittäin lupaavia ja mielenkiintoisia.

Eräs verkkoympäristössä tapahtuvan tietokoneavusteisen opetuksen merkittävistä eduista on, että se tuo fyysisesti toisistaan erillään olevat opiskelijat, opettajat ja opintomateriaalin yhteen. Opiskelija, joka on kotona tai työssä voi päästä käsiksi opetusmateriaaliin milloin vain, ja lähes mistä vain. Kommunikaatio opiskelijoiden ja opettajan välillä on mahdollista etäisyyksistä huolimatta ja opiskelija voi edetä opiskelussaan omaa tahtia. Voidaankin sanoa, että opiskellaan yksilöllisesti, mutta ei yksin!

Myös Teknillisessä korkeakoulussa kiinnostus tietokoneavusteista opetusta kohtaan on kasvanut. Tästä yhtenä esimerkkinä on tietojenkäsittelyopin laboratorion TRAKLA-järjestelmä [16]. TRAKLA on alunperin suunniteltu opetuksen tukijärjestelmäksi massakurssille. Järjestelmä on kuitenkin kehitetty jo 90-luvun alussa, joten sen käyttöliittymä ei vastaa tämän päivän vaatimuksia. Tämän työn taustalla onkin kiinnostus tämän tyyppisten järjestelmien kehittämiseen nykypäivän tekniikkaa, kuten Internetiä ja WWW:tä, hyväksikäyttäen.

1.1 Työn sisältö

Tässä työssä on tarkoitus pohtia niitä keinoja ja menetelmiä, joilla tietokoneavusteisten opetusohjelmien käyttöliittymiä voitaisiin toteuttaa verkkoympäristössä hyödyntäen uusinta tekniikkaa. Luvussa 2 kerrotaan lyhyesti työn lähihistoria, asetetaan tutkimusongelma ja tavoitteet sekä perusteet, joilla näiden tavoitteiden täyttymistä voidaan arvioida. Luku 3 käsittelee työn kirjallisuustutkimusosaa, jossa kartoitetaan muita samantyyppisiä järjestelmiä sekä pyritään esittämään ne keskeiset ominaisuudet, joita tämän tyyppisillä järjestelmillä on. Luku 4 käsittelee World Wide Webiä ja sen keskeistä käsitteistöä siltä osin, kun se on työn jatkon kannalta oleellista. Luku 5 käsittelee työn toteutusosassa käytettyjä menetelmiä ja ohjelmointiympäristöä. Luvussa 6 käsitellään TRAKLA-järjestelmää ja sen toimintojen siirtämistä WWW-ympäristöön. Luku 7 sisältää varsinaisen työn toteutusosan teknisen kuvauksen. Järjestelmä oli testikäytössä keväällä 1997. Järjestelmästä saatu palaute esitetään luvussa 8. Lopuksi esitetään työn yhteenveto luvussa 9.

2. Tutkimusongelma ja tavoitteet

Tässä kappaleessa esitetään TRAKLA-järjestelmän lähihistoria, työn tutkimusongelma ja asetetaan tutkimuksen keskeiset tavoitteet. Lisäksi selostetaan ne kriteerit, joiden perusteella työn tavoitteiden saavuttamista voidaan arvioida.

2.1 Taustaa

Vuoden 1990 keväällä TKK:n Tietojenkäsittelyopin laboratorion kurssi Tik-76.122 Tietorakenteet ja algoritmit [1, 46] muuttui. Kurssilla ollut harjoitustyö poistettiin ja tilalle tulivat kotitehtävät. Tehtäviä ratkoessaan opiskelijat simuloivat käsin kurssin aihepiiriin kuuluvia algoritmeja. Tätä kautta kotitehtävien tarkoitus oli antaa opiskelijalle parempi yleiskuva kurssiin liittyviin algoritmeihin ja tietorakenteisiin, verrattuna siihen, että he olisivat harjoitustyön kautta perehtyneet syvällisesti vain 1-2 algoritmiin.

Tavoitteessa onnistuttiin työryhmän jättämän raportin [16] mukaan hyvin. Muutoksen välitön seuraus oli kuitenkin se, että tarkastettavia kotitehtäviä tuli valtava määrä ja kurssin aikaresurssit loppuivat kesken. Kotitehtävistä saatujen hyvien tulosten vuoksi niiden käyttöä haluttiin kuitenkin jatkaa. Tästä syystä käynnistettiin projekti tutkimaan kotitehtävien automaattisen tarkastuksen mahdollisuutta. Projekti sai työnimekseen TRAKLA (TietoRakenteet ja Algoritmit; KotiLaskujen Arvostelu). Ensisijainen tavoite oli siirtää kotitehtävien rutiininomainen tarkastustyö tietokoneavusteiseksi. Samalla mahdollistettiin henkilökohtaisten tehtävien räätälöinti jokaiselle opiskelijalle, jolloin vastausten kopioiminen ei ollut enää mahdollista. Näissäkin tavoitteissa onnistuttiin ja varsinainen TRAKLA-ohjelmisto toteutettiin ohjelmatyöprojektina.

Opiskelijalle näkyvä käyttöliittymä toteutettiin aluksi sähköpostin avulla. Koska suurin osa tehtävistä oli visualisoitavissa graafisesti, heräsi jo alkuvaiheessa ajatus käyttöliittymän parantamisesta. Projekti graafisen käyttöliittymän osalta toteutettiinkin diplomityönä Macintosh-ympäristössä. Samalla tuli mahdolliseksi TRAKLA-ohjelmiston sisäisen esityskielen piilottaminen käyttäjältä käyttöliittymän avulla. Projekti sai työnimekseen TraklaEdit [17]. Suppean opiskelijoiden käytössä olevan Macintosh-laitekannan vuoksi käyttöliittymän käyttö jäi kuitenkin vähäiseksi.

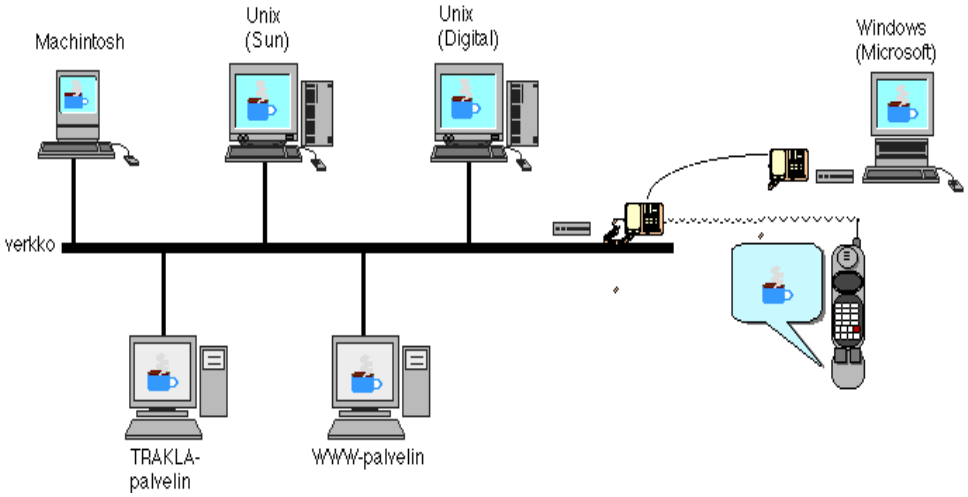
Projekti Macintosh-pohjaisen TraklaEditin siirtämiseksi muihin laiteympäristöihin sai alkunsa vuoden 1996 syksyllä. Projekti alkoi olemassa olevien resurssien tunnistamisella ja kartoituksella sekä eri toteutusmahdollisuuksien vertailulla. Koska toimiva käyttöliittymä oli jo olemassa, sen suora siirtäminen muihin ympäristöihin vaikutti lupaavalta vaihtoehdolta. Tämä olisi kuitenkin johtanut siihen, että suuria osia ohjelmasta olisi jouduttu toteuttamaan ainakin kahteen uuteen kohdejärjestelmään (Unix ja MS-DOS). Samalla ohjelman ylläpitäminen olisi tullut vaikeaksi.

Koska kurssin tiedotus toimi osin WWW:n kautta (World Wide Web), tuntui luonnolliselta, että tehtäville tarkoitettu käyttöliittymäkin olisi WWW-pohjainen. Tällaisen käyttöliittymän toteututukseen oli tarjolla erilaisia Internet-ohjelmointiin

soveltuvia ohjelmointiympäristöjä [2, 36]. Koska ensisijaisena tutkimusongelmana oli löytää laiteriippumaton ratkaisu, tämä vaihtoehto tuntui vastaavan asetettuihin vaatimuksiin. Projektia päätettiin jatkaa tutkimalla, kuinka tämän tyyppinen ohjelmistoprojekti voitaisiin toteuttaa laiteriippumattomasti WWW-ympäristössä.

2.2 Tutkimusongelma ja tavoitteet

Lähtökohtana tutkimusongelman ja tavoitteiden asettelussa oli siis laiteriippumattoman graafisen käyttöliittymän kehittäminen TRAKLA-ohjelmistolle. Lisäksi tavoitteiksi asetettiin mahdollisten muiden vastaavien sovellusten kartoittaminen sekä tietoturvakysymysten ratkaiseminen. Käytössä oli useita eri laitearkkitehtuuria edustavia tietokoneita (kuva 1), joilla tuli pystyä käyttämään toteutettavaa käyttöliittymää.



Kuva 1. Ensisijaisena tavoitteena on laiteriippumaton graafinen käyttöliittymä.

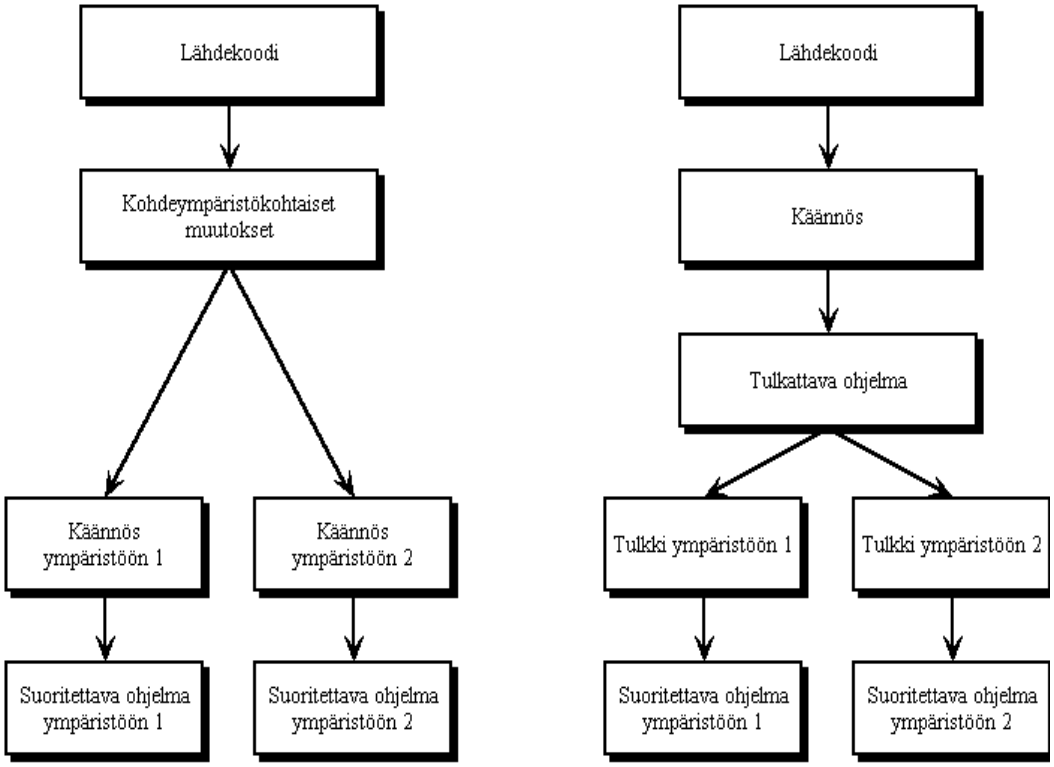
2.2.1 Laiteriippumattomuus

Perinteisillä ohjelmointikielillä (esimerkiksi C, C++ tai Pascal) tehty ohjelma käännetään suoraan tietokoneen ymmärtämäksi ajettavaksi ohjelmakoodiksi, jota tietokone voi suorittaa. Tällaista ohjelmakoodia ei kuitenkaan yleensä voida suorittaa kuin sen laitearkkitehtuurin ja käyttöjärjestelmän mukaisessa tietokoneessa, jolle ohjelma on käännetty. Jotta samaa ohjelmaa voitaisiin suorittaa jossain muussa laitearkkitehtuurissa, tulee ohjelma kääntää uudelleen. Johtuen eri laiteympäristöjen eroista, tällainen käännös uuteen ympäristöön vaatii tyypillisesti muutoksia myös alkuperäiseen lähdekieliseen ohjelmaan, varsinkin jos uudessa kohdeympäristössä on eri käyttöjärjestelmä. Kun näitä ongelmia halutaan välttää, pyritään ohjelman laiteriippumattomuuteen, jolla jatkossa tarkoitetaan ohjelman riippumattomuutta käytössä olevasta laitearkkitehtuurista ja käyttöjärjestelmästä.

Olemassa olevat resurssit ja tekniikka ovat kehittyneet sille tasolle, että erilaisia geneeristä ja siirrettävää ohjelmakoodia tuottavia kääntäjiä on olemassa, tai paremminkin resurssit mahdollistavat tällaisen tekniikan käytön. Tästä yhtenä esimerkkinä on Java [2]. Tavoitteiden kannalta tärkeintä on, että niiden tuottamaa yhtä ja samaa ohjelmakoodia pystytään suorittamaan eri laiteympäristöissä ja käyttöjärjestelmissä, etenkin niissä, jotka ovat käyttöliittymän kannalta olennaisimmat.

Tämä vaatii kuitenkin tulkattavaa ohjelmaa, jota suoritetaan erillisellä tulkilla. Tulkin tehtävä on toimia rajapintana geneerisen ohjelmakoodin ja laitearkkitehtuurin tarjoamien resurssien ja palveluiden välillä.

Työssä on tarkoitus selvittää, kuinka laiteriippumattomuus toteutuu tällaisessa ratkaisussa ja mitä uusia vaatimuksia se asettaa käytettävän laitteiston suorituskyvylle. Tavoitteena on lisäksi pyrkiä ratkaisuun, jossa olemassa olevan laitekannan suorituskyky vastaisi uuden käyttöliittymän vaatimuksia.



Kuva 2. Tavallisen ja tulkattavan ohjelman vaiheet lähdekielisestä ohjelmasta ajettavaksi ohjelmaksi.

2.2.2 Vastaavien sovellusten kartoittaminen

Etäopetus ja tietokoneavusteinen opiskelu ovat viimeisen vuosikymmenen aikana kehittyneet valtavasti. Tähän on ehkä suurimpana syynä tekniikan kehittyminen ja sen suomien resurssien kasvu sille tasolle, että opiskelu tällaisilla metodeilla on mahdollista.

Toisena tavoitteena onkin selvittää, opetuksellisesta näkökulmasta katsottuna, onko TraklaEditin tyyppisiä sovelluksia olemassa ja kuinka ne on toteutettu. Tämä siksi, että laajemmassa perspektiivissä katsottuna TraklaEditin tapaiset ohjelmat voidaan mieltää myös itsenäisiksi, varsinaisista luennoitavista kursseista erillään oleviksi opetusohjelmiksi.

2.2.3 Tietoturva

Operoitaessa verkkoympäristössä, tietoturva on yleensä yksi merkittävistä kynnyskysymyksistä. Tämän tyyppisissä ratkaisuissa tietoturvakysymykset tulee selvittää usean eri kohderyhmän osalta. Tällaisia ryhmiä ovat mm. opiskelijat, järjestelmän ylläpito sekä kurssin opetuksesta vastaava henkilöstö. Esimerkiksi opiskelijan kannalta tulee selvittää, mitä henkilötietoja opiskelijasta joudutaan käyttämään ja kuinka nämä tiedot voidaan suojata ulkopuolisilta käyttäjiltä. Järjestelmän ylläpidon kannalta on puolestaan tärkeää osoittaa, että uusi järjestelmä ei aiheuta turvallisuusriskiä muille järjestelmän käyttäjille tai ohjelmistoille. Opetuksesta vastaavalle henkilöstölle tärkeää on, että kurssikohtaiset tiedostot ovat asianmukaisesti suojattu, jolloin esimerkiksi opiskelijoilla ei saa olla mahdollisuutta muokata arvosanatietokantaa.

Lisäksi tulee huomioida käytettävien ohjelmistojen sisäiset tietoturvarajoitukset. Esimerkiksi Javassa tietoturvan valvonta on jätetty ohjelmaa suorittavan tulkin sisäiselle tietoturvamanagerille (security manager, yleensä integroitu selainohjelman omaan tietoturvaa valvovaan osaan), jonka tehtävänä on valvoa, ettei ajettava ohjelma tee laittomia ohjelmakutsuja. Tällaisia ovat mm. paikallisen levyn luku- ja kirjoitusoperaatiot.

Mikäli ristiriitoja tavoitteiden, toteutuksen ja käytettävyyden välillä syntyy, ne pitäisi pystyä ratkomaan jo ennen varsinaista ohjelmiston toteutusvaihetta. Kolmas tärkeä tavoite on siis ratkaisun tietoturvan suunnittelu ja saattaminen vallitsevien standardien ja vaatimusten edellyttämälle tasolle.

2.3 Kriteerit

Jotta järjestelmää voitaisiin luontevasti verrata muihin samankaltaisiin järjestelmiin, on tarpeen määritellä joukko kriteereitä, joilla vertailua voidaan suorittaa. Samoilla kriteereillä on tarkoitus lisäksi myöhemmin tutkia kuinka työlle asetetut tavoitteet toteutuivat.

Seuraavassa kriteerit on jaettu kolmeen eri luokkaan. Näistä yleiset kriteerit voidaan mieltää mille tahansa ohjelmistotuotteelle asetettaviksi kriteereiksi. Koska kyse on opetuksen tueksi tarkoitetusta järjestelmästä, tulee lisäksi määritellä kriteerit opetuksellisesta näkökulmasta. Järjestelmän ominaisuuksille voidaan myös asettaa kriteereitä, jotka tosin riippuvat melko paljon opetusohjelmiston käyttötarkoituksesta ja luonteesta.

2.3.1 Yleiset kriteerit

Peruslähtökohdat kriteereille ovat hyvin samantapaiset kuin mille tahansa ohjelmistotuotteelle. *Toimivuus* ja *käytettävyys* ovat kaikille käyttäjäryhmille tärkeitä. Käytettävyyden yksi perusedellytyksiä on *helppokäyttöisyys*. Tähän liittyy ohjelman *loogisuus* ja *yksinkertaisuus* sekä *yhdenmukaisuus* muiden vastaavanlaisten käyttöliittymien kanssa. Tällaiset ominaisuudet tulevat korostetusti esille, koska käyttäjien, eli tässä tapauksessa opiskelijoiden, tulee omaksua ohjelman käyttö itsenäisesti

ja nopeasti, eikä tätä varten voidaan yleensä järjestää varsinaista koulutusta. Tällöin myös kunnollisten *ohjekirjojen* merkitys kasvaa.

2.3.2 Opetukselliset kriteerit

Jotta järjestelmällä olisi opetuksellista merkitystä, tulisi sen pystyä jollain tavalla vaikuttamaan oppimistehokkuuteen myönteisesti. Tämä vaikutus voidaan saavuttaa joko suorasti tai epäsuorasti. Esimerkiksi opetuksen tukijärjestelmien eräs keskeinen argumentti on, että niiden avulla opettajan työtä voidaan tehostaa siten, että aikaa jää enemmän varsinaiseen opetustyöhön. Tällöin kyseessä on järjestelmä, jossa rutiininomaiset tarkastusprosessit siirretään tietokonevavusteiseksi. Näin säästyvät resurssit voidaan kohdentaa paremmin varsinaiseen opetustyöhön.

Oppimistehokkuuteen voidaan puolestaan vaikuttaa suoraan kehittämällä järjestelmiä, jotka hyödyntävät mahdollisimman paljon erilaisia opiskelumotivaatiota ja muistamista tehostavia ominaisuuksia. Esimerkiksi *audiovisuaalisten ominaisuuksien käyttö* saattaa parantaa opetettavan asian muistamista merkittävästi. *Vuorovaikutteisuus* puolestaan harjaannuttaa opiskelijaa itsenäisesti opiskelemaan ja tekemään. Tätä kautta opetusohjelma voi tuoda merkittävää lisäarvoa opetustapahtumaan.

Eräs kriteeri, jolla tällaisten järjestelmien toimivuutta voidaan arvioida on järjestelmästä saatu *palaute*. Koska varsinaisia tutkimuksia opiskelutehokkuuden parantamisesta tällaisin keinoin ei juurikaan ole saatavilla, on tärkeää, että järjestelmästä saadaan opiskelijoilta palautetta. Palautetta voidaan saada myös vertaamalla opiskelijoiden saamia arvosanoja ennen järjestelmän käyttöönottoa saatuihin arvosanoihin.

2.3.3 Järjestelmän ominaisuuskriteerit

Erilaisille opetuksen tueksi suunnitelluille järjestelmille voidaan asettaa yksilöllisiä ominaisuuskriteereitä. Nämä kriteerit määräytyvät järjestelmälle asetetuista vaatimuksista ja tavoitteista. Voidaan kuitenkin löytää joukko yhteisiä ominaisuuskriteereitä, joita voidaan soveltaa kaikkiin järjestelmiin.

Saatavuudella tarkoitetaan tässä yhteydessä sitä, kuinka hyvin järjestelmää käyttävä kohderyhmä pääsee käsiksi järjestelmään, toisin sanoen, mitä erityisjärjestelyjä vaaditaan, jotta määritelty kohderyhmä voisi hyödyntää järjestelmää täysipainoisesti. Mitä helpommin ja mitä vähemmillä erityisjärjestelyillä järjestelmä saadaan kohderyhmän ulottuviin, sitä parempi järjestelmä.

Järjestelmän *päivitettävyys* on myös eräs tärkeimmistä ominaisuuksista. Mitä helpommin kurssimateriaali ja materiaalin päivitykset on toimitettavissa eri osapuolten saataville, sitä parempi järjestelmä.

2.4 Rajaukset

Koska varsinkin kirjallisuusselvityksen osalta tavoitteet on asetettu varsin väljästi ja tietokoneavusteiseen opetukseen (TAO) liittyvää materiaalia on runsaasti, katsot-

tiin aiheen rajaaminen tarpeelliseksi. Tämän työn kannalta keskeisin mielenkiinto kohdistuu tietokoneavusteiseen opetukseen suunniteltujen järjestelmien toteutustekniikoihin. Vaikka työssä sivutaankin TAO-ohjelmistojen sisällöllisiä ominaisuuksia, ne eivät ole työn kannalta keskeisiä ja niiden laajempi käsittely rajataan työn ulkopuolelle.

3. Kirjallisuustutkimus

Tämä luku käsittelee työn kirjallisuustutkimusta. Tarkoituksena on luoda yleiskatsaus tietotekniikan tietokoneavusteisen opetuksen nykyisiin menetelmiin ja välineisiin, sekä esittää ne työhön liittyvät tutkimukset sekä kirjallisuus- ja WWW-viitteet, jotka ovat aiheen kannalta keskeisimmät.

3.1 Yleistä

Tämän työn kirjallisuustutkimus jakautuu selkeästi kahteen osaan. Toisaalta on pyritty löytämään kirjallisuudesta (lähinnä opetusalan) julkaisuja tietotekniikan opetukseen suunnitelluista ja toteutetuista ohjelmistoista. Tässä yhteydessä TRAKLA ja TraklaEdit on nähty yhtenä kokonaisuutena (WWW-TRAKLA), ja pyrkimyksenä on ollut löytää vastaavantyyppisiä järjestelmiä. Toisaalta on pyritty etsimään suoraan WWW:stä sellaisia WWW-pohjaisia sovelluksia, joilla olisi opetuksellista merkitystä. Tällöin TraklaEditiä on pidetty ennemminkin itsenäisenä ohjelmana, jolla voidaan visualisoida erilaisia tietorakenteita ja algoritmeja.

Tällaiseen jaotteluun on päädytty kahdestakin syystä. Ensinnäkin painetussa kirjallisuudessa diskussio pyrkii esittämään teorioita ja ajatuksia opetuksellisesta ja didaktisesta näkökulmasta, esimerkiksi [21, 32]. Implementaatiotason diskussio on puhtaasti toissijainen. Käytettävistä ohjelmointikielistä puhutaan korkeintaan perusteltaessa tietotekniikan peruskurssien opetuksen apuna käytettävien ohjelmointikielten valintoja [8, 20]. Tästä syystä WWW-TRAKLA tulee nähdä yhtenä opetuksen tueksi suunniteltuna *järjestelmänä*, jotta vertailu olisi mielekästä ja mahdollista. Toiseksi WWW:n kehittyminen on vasta viime vuosina mahdollistanut WWW-TRAKLAn tapaisten järjestelmien kehittämisen. Niinpä vastaavanlaisten *toteutusten* löytyminen on erittäin epätodennäköistä ja kokonaisten järjestelmien vertailuun ei päästä. Sen sijaan WWW:n avulla voidaan löytää erilaisia interaktiivisia tietorakenteita ja algoritmeja visualisoivia komponentteja, joiden toimintaa ja toteutusta voidaan verrata suoraan TraklaEditiin.

3.2 Tietokoneavusteinen opetus

Tietokoneavusteinen opetus voidaan määritellä esimerkiksi seuraavasti.

"*Laajasti käsitettynä tietokoneavusteinen opetus tarkoittaa kaikkea opetusta, jossa tietokone on apuna. Se voi olla kouluttajan työväline tai esitysväline, opiskelijan työväline, itsenäinen oppimateriaalin välittäjä ja oppimisprosessin ohjaaja.*

Suppeampi tietokoneavusteisen opetuksen määritelmä rajoittuu sellaiseen tietokoneen opetuskäyttöön, jossa tietokone toimii oppimateriaalin välittäjänä ja oppimisprosessin ohjaajana. Tällöin on kysymys opetusohjelmista tai oppimisympäristöistä, jotka sopivat sekä itseopiskeluun että itsenäiseen ryhmäopiskeluun. Suomenkielistä lyhennettä TAO käytetään yleensä tarkoittamaan suppeampaa määrittelyä." [21]

Tässä työssä toteutettava järjestelmä voidaan nähdä joko laajemman tai suppeamman määritelmän mukaisesti tietokoneavusteisena opetusohjelmana. Vaikka TraklaEdit on ensisijaisesti suunniteltu TRAKLA-järjestelmän käyttöliittymäksi ja tätä kautta opiskelijalle kotitehtävien palautukseen tarkoitetuksi työvälineeksi, se voidaan nähdä myös kouluttajan työ- ja esitysvälineenä. Tällöin se voidaan nähdä laajemman määritelmän mukaisesti tietokoneavusteiseen opetukseen tarkoitetuksi opetusvälineeksi. Tässä työssä on kuitenkin tarkoitus ensisijaisesti nähdä WWW-TRAKLA kokonaisuutena, joka on siis suppeamman määritelmän mukainen oppimisympäristö.

3.3 Tietotekniikan opetusta tukevat ohjelmistot

Kiinnostus opetusta tukevia ohjelmistoja kohtaan on kasvamaan päin. Tämä kiinnostus on nähtävissä erityisesti esimerkiksi matematiikan ja kielten opiskelun aloilla, mutta myös tietotekniikassa. Motivaatio ja lähtökohdat tällaisten ohjelmistojen kehittämiselle ovat monisäikeisiä. Voidaan kuitenkin löytää joitakin yhteisiä piirteitä kiinnostuksen lähteistä.

Tyypillisiä perusteluja ovat resurssien vähyys tai väheneminen joko opiskelijamäärien kasvaessa tai resurssien pienenemisen vuoksi, käsiteltävän aiheen kompleksisuus, käytössä olevien menetelmien puutteellisuus sekä halu uuden tekniikan tuomien etujen hyväksikäyttöön. Se, kuinka hyvin nämä tavoitteet täyttyvät, ei kuitenkaan ole täysin tutkittua. Puhtaasti tutkimuksellisesta näkökulmasta asiaa lähestyviä julkaisuja on varsin vähän. Tähän on syynä se, että tuloksia on vaikea verrata ja opetettava asia usein muuntuu, kun opetus tapahtuu tietokoneen avulla [21]. Joissain julkaisuissa [16, 24, 19] oli kuitenkin esitetty empiirisiä tutkimustuloksia siitä, kuinka opintomenestys oli kehittynyt kursseilla, joilla tällaisia järjestelmiä oli käytetty. Tulokset olivat poikkeuksetta positiivisia ja rohkaisevia.

Tietotekniikan alueella järjestelmiä on pyritty kehittämään useiden eri kurssien tarpeita vastaaviksi. Erilaisia työkaluohjelmia ja menetelmiä, kuten XTANGO/POLKA [29, 30], AOF [42], CORAL [31], PROTAN [23] ja SIMPLE [22], on kehitetty tällaisten järjestelmien rakentamiseen ja ylläpitoon. Tyypillistä kuitenkin on, että lähtökohtana on ollut jokin tietty kurssi ja sen erityisvaatimukset, jolloin yleiset työkalut ovat olleet riittämättömiä ja on jouduttu kehittämään kokonaan uusi järjestelmä. Tällaisia järjestelmiä ovat mm. WebCaMILE (Web-based Collaborative and Multimedia Interactive Learning Environment) [9] tietokonegrafiikan opetukseen, ASA [13] ja DynaLab (Dynamic Laboratory) [7] johdatus algoritmeihin -kurssille algoritmien toiminnan visualisoimiseen, CALOS (Computer Aided Learning; Operating Systems) [12] käyttöjärjestelmien opetukseen, TIHACW (This Is How A Computer Works) [10] tietokoneen arkkitehtuurin opetukseen sekä Swan [27] opetustyökaluksi tietorakenteiden ja algoritmien visualisoimiseen sekä animaatiotyökaluksi opiskelijoille ohjelmien toiminnan ymmärtämiseen. Lisäksi on kehitetty työkaluja mm. interaktiivisten opintomateriaalien kehittämiseen (esim. H^tX [18,3]), sekä ohjelmointitehtävien palautukseen helpottamaan ja automatisoimaan tarkastusprosessia (esim. TRY [25,26] ja Ceilidh [4]).

Aiheen ajankohtaisuudesta kertoo se, että osa näistä järjestelmistä [9,12,27] on toteutuksen osalta vielä kesken tai niiden jatkokehitys on meneillään. Näille järjestel-

mille on tyypillistä, että ne ovat interaktiivisia ja pyrkivät animaatioiden ja simulaatioiden avulla visualisoimaan opiskelijoille kurssin keskeisiä ja vaikeimpia osia. Vanhemmat järjestelmät [11,25] puolestaan ovat keskittyneet opetusresurssien parempaan kohdentamiseen siirtämällä rutiininomaisia tehtäviä tietokoneen hoidettavaksi.

Toteutustavat voidaan jakaa karkeasti kahteen luokkaan: tietylle laitearkkitehtuurille (yleensä MS-DOS/Windows, Unix tai Macintosh) suunniteltuihin järjestelmiin sekä WWW-pohjaisiin järjestelmiin. Näistä uusimpia ovat WWW-pohjaiset järjestelmät, kuten WebCaMILE ja CALOS, joista jälkimmäinen hyödyntää lisäksi Javaa.

Seuraavassa on esitelty järjestelmien ominaisuuksia yleiseltä kannalta. Sen jälkeen on esitelty edellämainituista järjestelmistä hieman tarkemmin kaksi.

3.3.1 Opetuksen tukijärjestelmät

Vanhimmat järjestelmät [16,11,25] ovat tyypillisesti lähteneet kehittymään resurssipulan pakottamina. Tällaisilla järjestelmillä ei välttämättä ole suoranaista opetuksellista funktiota, vaan ne ovat pikemminkin tukemassa opettajan työtä siten, että aikaa jäisi enemmän varsinaiseen opetustyöhön. Tällaisten järjestelmien keskeisiä osa-alueita ovat

- ilmoittautumiskäytännön hoitaminen ja opiskelijarekisterin ylläpito
- yksilöllisten harjoitustöiden räätälöinti opiskelijakohtaisesti,
- harjoitustöiden automaattinen lähetys ja vastaanotto,
- harjoitustöiden palautukselle asetettujen muodollisten vaatimusten tarkastaminen (esim. palautettiinko työ ajoissa ja oliko työn lähettänyt opiskelija ilmoittautunut kurssille),
- palautettujen harjoitustöiden generointi tarkastettavaan muotoon (esim. ohjelmointitehtävän automaattinen kääntäminen ja esimerkkiajojen suorittaminen) ja
- harjoitustöiden automaattinen tarkastus, arvostelu, arvosanojen vienti rekisteriin ja töiden edistymisen seuranta.

Vaikka varsinainen opetuksellinen näkökulma vaikuttaisi puuttuvan, voidaan kuitenkin löytää joitakin pedagogisia ominaisuuksia, joilla on kasvatustieteellistä merkitystä. Esimerkiksi palautusaikojen takaraja ja sen automaattinen valvominen motivoi opiskelijaa säännölliseen opiskeluun. Tämä on erityisen tärkeää kumulatiivisissa aineissa, joissa uuden asian oppiminen edellyttää aiemman asian hallitsemista. Lisäksi rajoittamattomat palautuskerrat motivoivat opiskelijaa korjaamaan vastaustansa, mikäli se ei alunperin mennyt oikein. Näin automaattiselta tarkastusohjelmalta saatu välitön palaute lisää opiskelutehokkuutta ilman lisäkustannuksia.

Tässä yhteydessä on syytä huomata, että kaikki nämä ominaisuudet ovat sellaisia, joita TRAKLA-järjestelmä yksinään tukee. Tämäntyyppisiä opetuksen tukijärjestelmiä on siis olemassa muitakin [3,4,11,18,25,26] ja niiden tarpeellisuus on havaittu maailmalla jo kauan sitten.

3.3.2 Opetusohjelmat

Opetusohjelmien visuaalinen puoli ja etäopetukselliset ominaisuudet ovat alkaneet kehittyä voimakkaasti vasta tämän vuosikymmenen puolella. Samalla niiden käyttötarkoitusta on pyritty laajentamaan kattamaan koko opetustapahtuma. Näin samaa ohjelmaa voidaan hyödyntää sekä opetustyökaluna että itseopiskeluvälineenä. Tällaisten järjestelmien keskeisiä ominaisuuksia ovat

- vuorovaikutteinen kommunikaatio (interaktio) käyttäjän kanssa,
- visuaalisten komponenttien käyttö (teksti, kuva, videokuva, ääni) ja
- animaatioiden ja simulaatioiden hyväksikäyttö.

Käyttökohteita näille ominaisuuksille ovat harjoitustehtävät ja niihin liittyvät esimerkit, käyttöohjeet ja viittaukset mahdollisiin lisätietoihin, tietokoneavusteiset opastus- ja opetusohjelmat sekä opetustilanteessa käytettävät havainnollistavat apuvälineet. Tällaisten järjestelmien kehittäminen vaatii aluksi huomattavasti resursseja, mutta kerran toteutettuna järjestelmää voidaan ylläpitää lähes ilman lisäkustannuksia. Lisäksi ne tarjoavat mahdollisuuden opiskelijaehtoiseen työskentelyyn sekä välittömän palautteen saamiseen tehtävistä. Tämä mahdollistaa kurssien pitämisen eritasoisille opiskelijoille sekä opiskelijoiden etenemisen omaa tahtia. Näissä suhteissa TraklaEdit tuo lisäarvoa TRAKLA-järjestelmälle, joka on ensisijaisesti opetuksen tukijärjestelmä.

WWW-pohjaiset järjestelmät

Kiinnostus tällaisten järjestelmien toteuttamiseen WWW-pohjaisina on vahvasti lisääntynyt viime aikoina. Syynä on ollut Internetin voimakas yleistyminen ja WWW:n tarjoamien ominaisuuksien kehittyminen. Edellä esitettyjen ominaisuuksien lisäksi WWW tarjoaa paljon muitakin etuja, kuten

- saavutettavuuden lähes kaikkialta (Internet),
- siirrettävyyden,
- päivitettävyyden (muutokset yhteen paikkaan näkyvät kaikkialle),
- etäkurssit (verkko, modeemi),
- hajautetut opiskelijaryhmät ja niiden välisen kommunikaation,
- ilmaiset, tutut ja helppokäyttöiset selainohjelmat (koululaitoksille) ja
- WWW-dokumenttien ylläpito ei vaadi kalliita ohjelmia.

Huonoina puolina voidaan pitää seuraavia seikkoja:

- HTML-koodaus (dokumenttien ylläpito) on kehittymättömämpää kuin kaupallisilla sovelluksilla,
- HTML:n luonne on staattinen, joten dynaamisten sivujen luonti vaatii erikoistoimenpiteitä,
- verkkojen hitaus on rasite, etenkin modeemiyhteyksillä,
- Animaatiot tulkattavilla ohjelmointikielillä (kuten Java) ovat hitaampia kuin

perinteiset ohjelmat, sekä

- henkilökohtaisten kontaktien puute pitkälle viedyssä etäopetuksessa.

3.3.3 ASA ja DynaLab

Aikaisemmin mainituista järjestelmistä vertailujärjestelmiksi on valittu ASA [13] ja DynaLab [7], koska niiden käyttötarkoitus on tietorakenteiden ja algoritmien animoinnissa. Lisäksi ne ovat keskenään melko samankaltaisia niin toteutukseltaan kuin käyttötarkoitukseltaankin.

Kummassakin järjestelmässä on lähtökohtana ollut ohjelman toiminnan visualisoiminen opiskelijalle. Tällä on pyritty opetettavan algoritmin parempaan ymmärtämiseen ja opiskelijan motivoimiseen. Algoritmien visualisoimiseen on käytetty vuorovaikutteisia ohjelmia, jotka kykenevät esittämään algoritmin suorituksen vaihe vaiheelta. Tällöin käytettävät ohjelmat osoittavat missä kohdassa lähdekielistä ohjelmaa suoritus on menossa, näyttävät muuttujien ja tietorakenteiden sisällön, laskevat ajonaikaisesti ohjelman suoritusaikaa sekä esittävät kuinka ohjelman saamaa syötettä ja tulostetta käsitellään suorituksen aikana.

DynaLab-järjestelmä koostuu virtuaalikoneesta, sen emulaattorista, erillisistä animaattoreista eri laiteympäristöille (kuten X-Windows ja MS-Windows), editorista ohjelmien työstämiseen ja kääntämiseen sekä kirjastosta, joka sisältää valmiita animaatio-ohjelmia. DynaLabilla opiskelijoilla voidaan teettää esimerkiksi tehtäviä, joissa tarkoituksena on empiirisesti tutkia eri algoritmien tehokkuutta, rekursion ja iteraation eroja, parametrinvälitystä tai osoittimien käyttöä. Lisäksi järjestelmässä on mahdollista suorittaa ohjelmaa takaperin. Näin jotakin ohjelman kohtaa voidaan tutkia tarkemmin aloittamatta ohjelman suoritusta alusta.

ASA-järjestelmän erikoisuutena on se, että siinä opiskelijoilla voidaan teettää tehtäviä, joissa opiskelijan on ohjelman suorituksen aikana vastattava kysymyksiin, kuinka muuttujien arvot muuttuvat ajon aikana. Vastaukset talletetaan ja opettajalla on mahdollisuus tutkia ja pisteyttää annettuja vastauksia.

Kummallekin järjestelmälle on ominaista, että ne muistuttavat peristeisten ohjelmointikielten korjaustyökaluja (debugger). Järjestelmät ovat vuorovaikutteisia ja erittäin visuaalisia. Vaikka esimerkiksi ASA-järjestelmässä on mahdollisuus kirjata opiskelijoiden saamia pistemääriä, kumpikaan järjestelmä ei kuitenkaan ole ensisijaisesti opetuksen tukijärjestelmä, vaan opetusohjelma.

Toteutuksen puolesta kumpikin järjestelmä on laiteriippuva. Kumpikin järjestelmä toimii MS-Windows -ympäristössä, lisäksi DynaLab toimii myös X-Windows -ympäristössä. Järjestelmille onkin yhteistä, että kumpikin on suunniteltu laboratoriokäyttöön. Suunnittelun lähtökohtana onkin ollut, että joukko opiskelijoita tekee annettuja tehtäviä yhtä aikaa, samassa tilassa.

Vertailtaessa järjestelmiä kohdassa 2.3 esitettyihin kriteereihin havaitaan, että opetukselliset kriteerit täyttyvät kummassakin esimerkkitapauksessa. Kumpikin järjestelmä pyrkii visuaalisten ominaisuuksien käytöllä luomaan vuorovaikutteisen ja tätä kautta motivoivan oppimisympäristön. Sen sijaan järjestelmät eivät täysin vastaa asetettuja ominaisuuskriteereitä. Saatavuus ASA:n ja DynaLabin tyyppisisssä jär-

jestelmissä ei ole paras mahdollinen. Tyypillisesti ohjelmistot on asennettu tiettyihin koneisiin opetusta varten varattuihin tietokoneluokkiin. Lisäksi ohjelmiston päivitys joudutaan tyypillisesti suorittamaan jokaiselle laboratorion koneelle erikseen.

3.4 Internet ja tietotekniikan opetus

Kirjallisuustutkimuksen toinen osa käsittelee Internetiä ja sieltä löytyviä tietotekniikan opetukseen löytyviä resursseja.

Internetin eri mahdollisuuksien käyttö opetuksen apuvälineenä on viime vuosina kasvanut räjähdysmäisesti. Vaikka käsitteinä etäopetus ja etäopiskelu ovatkin syntyneet ennen Internetin vallankumousta, niiden mahdollisuudet ovat kasvaneet huimasti nimen omaan Internetin suosion ansiosta. Tästä hyvänä esimerkkinä ovat erilaiset, pelkästään verkossa toimivat koulut ja oppilaitokset. Tai oikeastaan niiden suuri määrä. Lukumäärää on vaikea mennä arvioimaan, mutta joka tapauksessa puhutaan sadoista ellei tuhansista palvelimista. Ei siis ihme, että verkossa on jopa palvelimia [37], jotka toimivat pelkästään julkaistakseen linkkejä verkkokoulujen palvelimiin, sekä arvosteluja palvelinten ja opetuksen laadusta.

Tutkituista resursseista on pyritty esittelemään ne, jotka parhaiten sopivat työn kuvaan. Koska erilaisten toteutusten kirjo on melko laaja, on lisäksi pyritty esittämään toteutusten yhteiset piirteet sekä niiden hyvät ja huonot puolet.

Parhaana opetusalan linkkikokoelmana voidaan pitää *joulukuun lista* -nimellä (The December List) kulkevaa linkkipalvelinta, joka ylläpitää arvosteluja ja linkkejä mm. erilaisiin opetukseen suuntautuneisiin palvelimiin. Valitettavasti kokoelma oli kuitenkin niin laaja (reilusti yli sata linkkiä), että karsintaa jouduttiin tekemään arvostelujen pohjalta ja näinollen moneen, ehkä potentiaaliseenkin, palvelimeen ei ennätetty tutustua. Löydettyjen resurssien osalta parhaat viitteet löytyivät kuitenkin Gamelanin kokoelmasta, joka sisältää erittäin runsaasti Javalla toteutettuja sovelluksia.

3.4.1 Joulukuun lista

Joulukuun lista -nimellä tunnettu Internetissä toimiva suosittu linkkikokoelma [37] on December Communications, Inc. -yhtiön [38] ylläpitämä palvelu, josta löytyy linkkejä liittyen Internetin välityksellä tapahtuvaan kommunikaatioon. Kokoelmasta löytyy linkkejä aina kulttuurista teknologiaan ja erilaisiin sovelluksiin. Kokoelman alasivuilta löytyy myös linkkejä ympäri maailmaa erilaisiin verkossa toimiviin kouluihin.

Valitettavasti Internetin kaupallinen puoli tulee tässä kohtaa harmittavalla tavalla esille. Suurien lupausten takana on tyypillisesti vain pelkkää sanallista esittelyä tarjotuista kursseista ja hyvin vähän itse opetusmenetelmistä. Konkreettisia demoja ei löydy juuri lainkaan ja nekin harvat, jotka löytyvät ovat alkeellisia tai eivät ole tietotekniikan alueelta.

Koska hakua ei voida pitää kattavana johtuen siitä, että kaikkiin linkkeihin ei ollut mahdollisuuksia tutustua ja osa linkeistä olisi vaatinut rekisteröitymään ja ostamaan palvelun, jää arviointi tältä osin puutteelliseksi. Yleisvaikutelma on kuitenkin se, että tietotekniikan opetukseen suunniteltujen WWW-pohjaisten ohjelmistojen löytyminen verkkoympäristössä on epätodennäköistä. Tai ainakin järjestelmät ovat vielä kehitysasteella, joten niihin ei vielä löydy verkosta viitteitä.

Suurimmassa osassa järjestelmiä WWW toimi ainoastaan välityskanavana varsinaisille etäopetusohjelmille, jotka on suunniteltu jollekin tietylle laitearkkitehtuurille. Lisäksi WWW:tä käytetään kommunikaatiovälineenä opiskelijoiden ja opettajien välillä.

Vaikka tietotekniikan opetukseen ei kunnollisia viitteitä löytynytkään, voidaan löydettyjen resurssien pohjalta kuitenkin tehdä joitakin johtopäätöksiä verkossa tapahtuvan opetuksen laadusta. Linkkejä erilaisiin opetuksen tueksi suunniteltuihin sovelluksiin löytyi aina hypermedialla toteutetuista kemian oppikirjoista [51] interaktiiviseen sammakonleikkelyyn [52]. Toteutukset pyrkivät käyttämään kaikkia olemassa olevia resursseja hyväkseen, kuten hypertekstiä, kuvia, ääntä ja animaatioita. Laadullisesti järjestelmiä oli monen eri tasoisia. Täytyy kuitenkin muistaa, että kaikkien ominaisuuksien käyttö vaatii aina resurssia käyttävältä laitteistolta paljon enemmän kuin vaatimattomampi järjestelmä. Tässä suhteessa järjestelmien laadullinen arviointi tulisi aina tehdä opiskelijan käytettävissä olevia resursseja silmälläpitäen.

3.4.2 EarthWeb ja Gamelan

EarthWeb [40] on 1994 perustettu Internetissä toimiva linkkipalvelin, jonka tarkoituksena on tehdä pioneerityötä WWW-pohjaisten ohjelmien ja verkkoteknologian parissa. Sen toimintaa rahoittaa Warburg Pincus Ventures, LP. Yksi sen merkittävimmistä osa-alueista on Gamelan [41] - Internetin virallinen Java-hakemisto Java-pohjaisiin resursseihin, johon se on valtuutettu JavaSoftin [49] toimesta. Palvelimesta löytyy maailman suurin hakemisto Java-resursseihin, jotka on Gamelanin toimesta myös arvioitu. Sieltä löytyy myös keskustelua, uutisia ja julkaisuja koskien Javaa.

Resurssit on jeattu puumaiseksi hierarkiaksi, jossa jo pelkästään tietotekniikan opetukseen liittyviä Javalla toteutettuja sovelluksia eli ohjelmasia (joissain yhteyksissä käytetään myös nimitystä sovelma) löytyi yli 100 kappaletta jaettuna kuuteen eri alaryhmään. Näistä seuraavassa on poimittu esimerkinomaisesti lähemmän tarkastelun piiriin muutama Java-ohjelmanen kahdesta eri ryhmästä: binääripuut ja lajittelualgoritmit.

Yhteistä tarkasteltaville sivuille on niiden pyrkimys vuorovaikutteisuuteen. Esimerkiksi lajittelualgoritmeissa [39] oli esillä sekä valmiilla satunnaisella syötteellä käynnistyvä, että käyttäjän määriteltävissä olevalla syötteellä algoritmin toimintaa visualisoiva versio. Samalta sivulta löytyi lähes kaikki tunnetuimmat lajittelualgoritmit ja niiden toimintaa ja nopeutta pystyi vertailemaan vierekkäin. Myös animaation näyttävyyteen oli kiinnitetty huomiota varsinkin binääripuita [45] visualisoivalla sivulla. Esimerkkinä mainittakoon värikäs 3D-animaatio, joka visualisoi splay-puun toimintaa.

Opetuksellisesta näkökulmasta tämän tyyppiset sovellukset tarjoavat näyttäviä visualisointeja opetettavalle aiheelle. Esimerkiksi luokkaopetuksen ohessa opiskelijoille voidaan tarjota visuaalisia virikkeitä opetettavasta aiheesta, sekä viitteitä resursseihin, joihin he voivat tutustua myös jälkikäteen. Yksityiskohtiin meneviksi opetusohjelmiksi niistä ei tällaisenaan kuitenkaan ole. Tämä johtuu pitkälti siitä, että niiltä puuttuu kokonaan opetuksen tukijärjestelmille tyypilliset ominaisuudet. Esimerkiksi opitun asian testaaminen ei ole järjestelmällä mahdollista. On kuitenkin todennäköistä, että lähitulevaisuudessa tällaiset animaatiot tulevat lisääntymään ja kehittymään myös paremmin opetuksellisia vaatimuksia vastaaviksi.

Vaikka tämän tyyppisistä animaatioista ei varsinaisiksi opetusohjelmiksi olisikaan, on niillä yksi keskeinen etu esimerkiksi verrattuna kappaleessa 3.3.3 (ASA ja DynaLab) oleviin esimerkkiopetusohjelmiin - ne ovat laiteriippumattomia. Lisäksi niiden ominaisuuksiin kuuluu helppo saatavuus ja päivitettävyys. Tällöin niiden “jakelu” opiskelijoille on erittäin helppoa: opettajan ei tarvitse kuin kertoa millä osoitteella kyseinen animaatio verkosta löytyy. Opiskelija voi tällöin itse käydä tutustumassa annettuun resurssiin. Vaatimuksena on ainoastaan verkkoyhteys ja sopiva selainohjelma.

Edellä esitettyjen ominaisuuksien vuoksi tämän tyyppiset animaatiot ja muut sovellukset sopivat erityisen hyvin lisämateriaaliksi kursseille. Lisäksi niitä voidaan hyödyntää kurssilla visuaalisena havaintomateriaalina esimerkin omaisesti. On kuitenkin huomattava, että mikään ei estä kehittämästä näitä animaatioita myös paremmin opetuksellisia kriteereitä vastaaviksi. Esimerkiksi ASAn ja DynaLabin kaltaiset järjestelmät voitaisiin varsin hyvin toteuttaa myös verkkoympäristössä.

3.5 Yhteenveto

Tietokoneavusteisessa opetuksessa käytetyt ohjelmistot voidaan jakaa kahteen ryhmään. Tässä työssä niistä on käytetty nimityksiä *opetuksen tukijärjestelmät* ja *opetusohjelmat*. Näistä ensinmainittu ryhmä edustaa ohjelmistoja, joiden ensisijainen tarkoitus on pyrkiä käyttämään tietotekniikkaa hyväksi siten, että mahdollisimman paljon erilaisia automatisoitavissa olevia, opetettavaan aiheeseen liittyviä rutiineja voidaan jättää tietokoneen hoidettavaksi. Tällaisia ovat esimerkiksi ilmoittautumiskäytännön hoitaminen, opiskelijarekisterin ylläpito, harjoitustöiden automaattinen lähetys, vastaanotto ja tarkastus sekä töihin liittyvien muodollisten vaatimusten seuraaminen. Toinen ryhmä koostuu opetusohjelmista, joilla pyritään suoraan parantamaan opetettavan asian oppimista. Tällaisten ohjelmien ominaisuuksia ovat mm. vuorovaikutteisuus ja visuaalisen havaintomateriaalin käyttö.

Tällä hetkellä ehkä suurin kiinnostus on jälkimmäistä ryhmää edustavia ohjelmia kohtaan. Etenkin Internetin ja WWW:n yleistymisen myötä sekä näiden medioiden kautta avautuvien mahdollisuuksien kasvaessa on tällaisten järjestelmien kehitys otettu haasteena vastaan eri opetusalojen ja tutkijoiden taholta. Kuitenkaan järjestelmiä, joissa yhdistyisivät sekä opetuksen tukijärjestelmille, että opetusohjelmille tyypilliset ominaisuudet yhdessä vuorovaikutteisten WWW-sivujen kanssa, ei ole vielä olemassa. Tässä suhteessa tämä työ ja siitä saadut kokemukset luovat pohjaa tuleville opetusohjelmaratkaisuille.

WWW:n ja Internetin edut opetuksen tukena tulevat esille lähinnä tiedottamisen ja kurssimateriaalin helpon jakelun muodossa. Päivitykset tällaiseen materiaaliin saadaan kaikkien osapuolien nähtäville nopeasti ja helposti eikä niitä tarvitse tehdä kuin yhteen paikkaan. Lisäksi materiaaliin on helppo liittää viittauksien avulla myös kolmansien osapuolien WWW-sivuja aiheesta. Kun vielä lisäksi hyödynnetään WWW:tä vuorovaikutteisten opetusohjelmien avulla, on käytössä tehokas tietokoneavusteinen oppimisympäristö.

4. World Wide Web (WWW)

Sekä kirjallisuustutkimuksen, että työn toteutusosan kannalta yksi merkittävimmistä osa-alueista on WWW (Worl Wide Web). Seuraavassa on tarkoitus esittää ne keskeisimmät termit ja käsitteet, jotka ovat välttämättömiä työn toteutuksen kuvauksen seuraamiseksi ja ymmärtämiseksi.

4.1 Yleistä

Yleisesti ottaen paras tapa ottaa selvää WWW:hen liittyvistä asioista on WWW itse. Tämä johtuu pitkälti kyseisen median luonteesta ja ominaisuuksista. Toinen merkittävä syy tähän on WWW:n dynaaminen tila, joka ei tunnu stabiloituvan vielä pitkään aikaan. Painettu kirjallisuus ei pysy tällaisen kehitysvauhdin perässä.

Tästä on kaksi välitöntä seurausta. Ajantasalla oleva tieto löytyy ja näin ollen se täytyy yleensä aina hakea suoraan verkosta, ainakin jos halutaan varmistua tiedon tuoreudesta. Huonona puolena on, että viittaukset verkosta löytyviin dokumentteihin eivät ole kovin pysyviä. Ts. viittaukset vanhenevat, kun dokumentit vaihtavat olinpaikkaa. Näin dokumentin löytäminen saattaa vaikeutua. Viittauksen informaatioarvoa kuitenkin lisää se, että haluttu dokumentti voidaan ehkä paikallistaa uudesta olinpaikastaan dokumentin nimen perusteella. Tähän voidaan käyttää erilaisia Internetistä löytyviä hakuohjelmia.

Seuraavassa on käytetty lähteinä useita WWW:stä löytyviä dokumentteja. Osa näistä dokumenteista on toki löydettävissä myös painetussa muodossa, lähinnä erilaisten raporttien muodossa. Paperimuodossa löytyviin lähteisiin on viitteet merkitty myös kirjallisuusluetteloon. Puhtaat WWW-viittaukset dokumentteihin, joilla ei ole yksiselitteistä kirjoittajaa, on sijoitettu sivun alaviitteisiin. Tämä myös siksi, että kyseisten viitteiden pysyvyydestä ei ole niin suuria takeita.

4.2 WWW, hyperteksti ja hypermedia

Kirjaimet WWW tulevat englanninkielen sanoista World Wide Web (lyhyesti pelkkä Web), joka vapaasti tulkittuna tarkoittaa maailmanlaajuista hakuteosta. WWW:n alullepanijana tunnetaan *Tim Berners-Lee*, joka CERNissä (the European Laboratory for Particle Physics) toimiessaan kehitteli hajautettua hypermediajärjestelmää (“distributed hypermedia system”). Tätä nykyä Tim Berners-Lee jatkaa pioneerityötään W3 konsortiossa MIT:ssä[1] .

Hypertekstin etuna on, että hypertekstidokumentissa esiintyvästä yksittäisestä sanasta tai aiheesta voi yleensä saada lisätietoa hiirtä napsauttamalla. Tällaista viitettä hypertekstidokumentissa sanotaan linkiksi. Linkki on osoite, joka viittaa toiseen dokumenttiin, joka käsittelee linkkisanan kuvaamaa aihetta. Itseasiassa tällainen linkitetty dokumentti voi olla ja yleensä onkin jonkin toisen osapuolen kirjoittama ja

1. W3 konsortio on MIT:n (Massachusetts Institute of Technology) tietojenkäsittelyopin laboratorion johtama teknologiayhteisö (**http://www.w3.org/pub/WWW/**).

ylläpitämä. Tällaista linkkiä voisikin verrata esimerkiksi sivun alaviitteeseen tai dokumentin kirjallisuusviitteeseen sillä erotuksella, että viitteeseen pääsee käsiksi välittömästi!

Edellytyksenä Webiin pääsylle on selainohjelma (browser). Selainohjelma kykenee lukemaan erityisiä hypertekstidokumentteja (ks. “HTML - Hypertext Markup Language” sivulla 18) sekä hakemaan verkosta linkkien välityksellä lisätietoa muista lähteistä. Tällaisia lähteitä ovat hypermediapalvelimet (hypermedia server, WWW server), joita ylläpitävät informaation tarjoajat.

Hyperteksti on vain osajoukko koko hypermediasta. Hypermediaa on itse asiassa mikä tahansa media, joka sisältää linkkejä toisiin medioihin. Tämä tarkoittaa sitä, että selainohjelmat eivät tuo kuvaruudulle pelkkää tekstiä, vaan kykenevät myös esittämään kuvia ja animaatioita sekä tuottamaan ääntä.

4.3 Kielet, protokollat ja käsittteet

WWW:n myötä on syntynyt tarve määritellä koko joukko uusia kieliä, termejä ja käsitteitä. Kielillä tässä yhteydessä tarkoitetaan nimen omaan kieliä, joilla pystytään määrittelemään jollain hypermedian osa-alueella tarvittava esitysmuoto.

On kuitenkin muistettava, että Internet ja kaikki sen sisältämät osa-alueet ovat koko ajan suuren muutoksen alla. Uusia ideoita ja suuntauksia tulee jatkuvasti ja vanhoja standardeja parannellaan ja niihin lisätään uusia ominaisuuksia jatkuvalla syötöllä. Ei siis ihme, että uusien käsitteiden ja termien viidakossa ei oikein tahdo pysyä perässä.

Etenkin hyvien suomennosten löytäminen on tuottanut ja varmasti tulee jatkossakin tuottamaan vaikeuksia. Ennenkuin termi on maailmalla vakiintunut tarpeeksi, on turha yrittää kääntää sitä suomeksi. Seuraavassa termejä on pyritty suomentamaan (ei siis pyritty korvaamaan suomenkielisellä termillä) sen verran, kuin on ollut järkevää ja tarpeellista.

4.3.1 HTML - Hypertext Markup Language

WWW-dokumentit on kirjoitettu erityisellä HTML (Hypertext Markup Language) - eli hypertekstikielellä. Tästä kielestä laajempi versio kulkee nimellä SGML[1] (Standard Generalized Markup Language), jota käytetään määrittelemään tiettyjä eritystarkoituksiin sopivia kieliä. Tällaista dokumentin tyypin määritelmää kutsutaan DTD:ksi (Document Type Definition). HTML on siis SGML:llä tehty DTD (määrittely).

HTML-dokumentti rakentuu lohkoista tai komponenteista, jotka rajataan erityisillä merkinnöillä. Kuvissa 3 ja 4 on esimerkkejä tällaisista merkinnöistä. Lohkon alkumerkintää vastaa samanniminen loppumerkintä erotuksena “/”-merkki. Merkinnät kertovat dokumenttia lukevalle selainohjelmalle dokumentin loogisen rakenteen. Dokumentin fyysinen ulkoasu riippuu käytössä olevasta ympäristöstä.

1. Lisätietoja saa osoitteella **http://etext.virginia.edu/bin/tei-tocs?div=DIV1&id=SG**

Loogisella kuvauksella voidaan merkitä mm. kappalerakenteet (<P>), otsikoiden suhteelliset pistekoot (<Hn>, n=1,2,3...), taulukot (<TABLE>), muotoilunohjaukset jne. Kuvassa 3 on yksinkertainen HTML-muotoon määritelty teksti, joka koostuu viidestä lohkosta: html, head, body, h1 sekä P. Huomaa, että osa lohkoista on sisäkkäin ja osa peräkkäin. Lisäksi hypertekstidokumenttiin voidaan liittää kuvia, animaatioita ja äänitiedostoja. Tällaisten lisäominaisuuksien liittäminen vaatii kuitenkin katseluohjelmalta eli selainohjelmalta vastaavien komponenttien tukea. Esimerkiksi Javalla tuotettu animaatio-osa (Java applet) HTML-dokumentissa voitaisiin kuvata kuvan 4 esittämällä tavalla. Tällaisille komponenteille voidaan välittää myös parametreja, kuten tavallisille ohjelmille. Selainohjelma, joka ei tue Javaa, jättäisi <APPLET> ja </APPLET> merkintöjen välissä olevan osan huomiotta.

HTML-dokumentteja voidaan tuottaa millä tahansa “tavallista” tekstitiedostoa käsittelevällä tekstinkäsittelyohjelmalla. Ts. HTML-dokumentit tulee olla ASCII-muodossa .

```
<html>
<head>
<TITLE>A Simple HTML Example</TITLE>
</head>
<body>
<H1>HTML is Easy To Learn</H1>
<P>Welcome to the world of HTML.
This is the first paragraph. While short it is
still a paragraph!</P>
<P>And this is the second paragraph.</P>
</body>
</html>
```

Kuva 3. Yksinkertaisen dokumentin kuvaus HTML-merkkauskielellä.

```
<APPLET CODE=”myJava.class” WIDTH=400 HEIGHT=100>
<PARAM NAME=picture VALUE=4>
</APPLET>
```

Kuva 4. HTML-dokumenttiin liitetyn Java ohjelmasen (applet) määrittely.

4.3.2 URL - Uniform Resource Locator

Kuten edellisessä kappaleessa jo mainittiin, Internetistä löytyy WWW:n avulla paljon muitakin resursseja kuin pelkkää tekstimuotoista informaatiota. Pääsy tällaisiin resursseihin, niiden paikantaminen ja niiden löytyminen edellyttää määrämuotoista osoitetta, jonka kaikki kommunikaatio-osapuolet ymmärtävät. Tähän tarkoitukseen käytetään ns. URL-viittauksia. URL (Uniform Resource Locator) [5] määrittelee resurssin paikantamiseen tarvittavan formaalin informaation syntaksin ja semantiikan. Tällainen URL on siis Internetin standardi, joka määrittelee jonkin verkosta löytyvän objektin, esimerkiksi tiedoston tai uutisryhmän.

URL koostuu palasista, joista alkuosa määrittelee käytettävän protokollan. Kuvassa 5 on lueteltu tällä hetkellä olemassa olevia standardoituja protokollia.

- ftp File Transfer Protocol (tiedonsiirtoprotokolla)
- http Hypertext Transfer Protocol (hypertekstin siirtoprotokolla)
- gopher The Gopher Protocol
- mailto Electronic mail address (sähköpostiosoite)
- news USENET news (kansainväliset uutisryhmät)
- nntp USENET news using NNTP access
- telnet Reference to interactive sessions
- wais Wide Area Information Servers
- file Host-specific file name (paikallinen tiedosto)
- prospero Prospero Directory Service

Kuva 5. URL-viittauksissa käytettyjä protokollia.

URLin loppuosa määräytyy käytettävästä protokollasta. On kuitenkin olemassa yleinen syntaksi protokollille, jotka käyttävät IP-pohjaista (Internet protocol) tiedonsiirtoa. Tällaisen URLin loppuosa muodostetaan seuraavasti:

//<käyttäjätunnus>:<salasana>@<isäntäkone>:<portti>/<url-polku>

Isäntäkoneen osoite voidaan antaa täydellisesti määriteltynä selväkielisenä koneen nimenä tai IP-osoitteena. Portti määrittelee yhteyteen käytettävän portin, joskin useimmilla protokollilla on myös oletusportti, jota käytetään, mikäli määrittely puuttuu. Url-polku määrittelee yksityiskohdat siitä, kuinka tietty resurssi löydetään. Se kuinka polku määritellään, riippuu käytettävästä protokollasta.

4.3.3 Protokollat

Käytetyistä protokollista HTTP on se, jota käytetään hypertekstidokumenttien siirtoon. HTTP:tä käytettäessä url-polun loppuosa viittaa haettavaan dokumenttiin. Kyseessä on yleensä looginen polku isäntäkoneen hakemistorakenteessa. Isäntäkoneeksi annetaan WWW-palvelimen nimi. HTTP käyttää oletuksena porttia 80, joten sitä ei yleensä tarvitse antaa. Myöskään käyttäjätunnusta ja salasanaa ei julkisissa palvelimissa tarvitse käyttää.

Muita yleisesti käytettyjä protokollia ovat FTP tiedostojen siirtoon, MAILTO sähköpostiosoitteille sekä NEWS uutisryhmille. Yleisimmät selainohjelmat osaavat vähintään nämä protokollat.

4.3.4 Lomakkeet ja CGI-scriptit

Selainohjelmat kykenevät tavallisen HTML-muotoisen hypertekstin lisäksi näyttämään myös HTML-lomakkeita (HTML forms). Tällaiset lomakkeet mahdollistavat käyttäjältä tulevan syötteen keräämisen ja prosessoinnin. Lomakkeet voivat sisältää esimerkiksi valintalistoja, painikkeita ja tekstikenttiä, joita käyttäjä voi manipuloida hiirellä ja näppäimistöllä.

Käyttäjältä saatu syöte lähetetään WWW-palvelimelle, joka käynnistää erillisen ohjelman, joka on määritelty HTML-lomakkeessa. Tällaista ohjelmaa kutsutaan CGI-ohjelmaksi tai CGI-scriptiksi (Common Gateway Interface) [47].

CGI-scripti voi olla esimerkiksi Perl-kielellä kirjoitettu lyhyt ohjelmanpätkä, joka suorittaa joukon komentoja tai muita ohjelmia. Koska CGI-scriptejä ajetaan WWW-serverin oikeuksilla, ne aiheuttavat myös potentiaalisen turvallisuusriskin. Tästä syystä peruskäyttäjillä ei tyypillisesti ole oikeuksia asentaa CGI-scriptejä WWW-palvelimen alle, vaan niiden asentamiseen tarvitaan ylläpito-oikeudet. CGI-scriptin toteutusvaiheessa tulee erilaiset turvallisuuskysymykset ottaa tarkkaan huomioon. Tätä varten turvallisten CGI-scriptien toteuttamiseen löytyy WWW:stä runsaasti ohjeita ja vinkkejä [43,44,47,48].

4.4 Johtopäätökset

Koska WWW on kehitetty alunperinkin hajautettuja hypermediadokumentteja varten, tässä piilee myös sen vahvin puoli. Kun perinteinen kirjallisuus pyrkii esittämään asiat lineaarisesti, WWW:ssä käytetään hypertekstiä. Vaikka lähestymistapa ei sovikaan aivan kaikille dokumenteille, on se oikein käytettynä erittäin tehokas. Esimerkiksi asteittain tarkentuva ohjekirja voi olla hyvin toteutettuna erittäin käyttökelpoinen: ensimmäisellä tasolla käyttäjälle annetaan lyhyet kuvaukset ohjelman eri toiminnoista, joista halutessaan voi saada tarkempaa tietoa hypertekstin viittauksia seuraamalla. Dokumenttia on helppo päivittää ja laajentaa tarpeiden mukaan. Lisäksi hypertekstiä hyväksikäyttäen voidaan dokumentteja elävöittää ja selkeyttää.

WWW:n käyttö ei kuitenkaan rajoitu pelkästään hypertekstiin. Erilaisten rajapintojen avulla sitä voidaan käyttää moneen muuhunkin tarkoitukseen. Tästä on esimerkkinä mm. seuraavassa luvussa esitetyt vuorovaikutteiset WWW-sivut, joita voidaan toteuttaa eri tekniikoilla. Tällöin voidaan puhua jo todellisesta hypermediasta.

5. Java-ohjelmointi

Tämän luvun tarkoituksena on esitellä työn toteutukseen käytettyä Java-ohjelmointikieltä ja sen ominaisuuksia.

5.1 Yleistä

Johdannossa mainitun siirrettävän ohjelmakoodin lisäksi toinen mielenkiintoinen kehityssuunta nykyaikaisessa ohjelmointikulttuurissa on komponenttipohjainen ohjelmointi. Ohjelmat rakentuvat komponenteista, jotka voivat olla tätä ohjelmaa varten suunniteltuja ja toteutettuja tai jostain toisesta ohjelmistoprojektista tai vaikkapa komponenttikirjastosta poimittuja. Samaa komponenttia voidaan siis käyttää uudelleen ja tätä kautta kehittää yhä parempia ja luotettavampia ohjelmia yhä nopeammin. Kyse on oikeastaan oliopohjaisesta ohjelmoinnista, jossa ajatusta on viety hiukan eteenpäin.

Verkkoympäristössä selainohjelmat voidaan nähdä eräänlaisina alustoina (Container) tällaisille komponenteille tai olioille. Ohjelma voidaankin tässä suhteessa nähdä kuvauksena, joka määrittelee olioiden rakenteen ja keskinäisen kommunikaation. Verkkoympäristössä tällaisia komponentteja voidaan ladata paikallisen levyn lisäksi myös verkosta, eikä ladattavien komponenttien tarvitse olla pelkkää tekstiä tai kuvia, vaan ne voivat olla myös ajettavaa ohjelmakoodia.

5.2 Java ja ActiveX

Java-ohjelmaset (applet) [2] ja ActiveX-kontrollit (controls) [33] hyödyntävät molemmat komponenttitekniikkaa. Lisäksi kumpikin tukee mahdollisuutta rakentaa vuorovaikutteisia WWW-sivuja. Valinta toteutustavasta tehdäänkin yleensä juuri näiden kahden järjestelmän välillä [36]. Eräs keskeinen ero on kuitenkin se, että Java on ohjelmointikieli, kun taas ActiveX-kontrolleja voidaan kirjoittaa useilla eri kielillä - tyypillisesti kuitenkin C++:lla. Ohjelmointikielenä Java muistuttaa kuitenkin pitkälle C++:aa, joten ohjelmistometodiikan osalta järjestelmät ovat samankaltaisia.

Ehkä keskeisin ero järjestelmien välillä on kuitenkin se, että Javalla voidaan tuottaa siirrettävää ja tulkattavaa ohjelmakoodia (tavukoodi), kun puolestaan ActiveX-kontrollit ovat konekoodia. Java-ohjelmanen tyypillisesti ladataan verkon yli selainohjelmaan, johon on ohjelmoitu Javan virtuaalikone, joka osaa tulkata ohjelman sisältämää tavukoodia. Jokaisella Javaa tukevalla selainohjelmalla on lisäksi luokkakirjasto, joka sisältää paljon tarpeellisia valmiiksi ohjelmoituja komponentteja. ActiveX:n tyypillinen kohdejärjestelmä on Intel-prosessoripohjainen järjestelmä, jossa on käyttöjärjestelmänä Windows 95 tai Windows NT. Koneriippuvuus tuo tässä kohtaa kuitenkin toisenlaista lisäetua - ActiveX-kontrolleja voidaan suorittaa myös muunlaisilla alustoilla kuin selainohjelmilla. Lisäksi konekoodiksi käännetyn ohjelman suorittaminen on huomattavasti nopeampaa kuin tulkattavan ohjelman.

5.2.1 Turvallisuus

Komponenttien lataaminen verkkosivuilta tuo uusia turvallisuusongelmia verrattuna niiden lataamiseen paikalliselta levyltä tai palvelimelta. Tyypillisesti paikalliselta levypalvelimelta ladatut ohjelmat luokitellaan turvallisiksi, ts. niiden oletetaan toimivan oikein eikä aiheuttavan tuhoa omalle järjestelmälle. Paikallisen järjestelmän ylläpito on helposti tavoitettavissa ongelmatilanteissa ja kuuluu yleensä samaan yhteisöön käyttäjän kanssa. Verkkomaailmassa turvallisen palvelimen määritteleminen on huomattavasti vaikeampaa. Ladattu komponentti saattaa sijaita toisella puolella maapalloa, eikä verkkopalvelimen ylläpitäjää välttämättä edes tiedetä. Yleensä kuitenkin suurten ja tunnettujen yhtiöiden ja yhteisöjen palvelimia pidetään luotettavina ja niistä ladattuja komponentteja luotettavina. Tässä piilee kuitenkin aina omat riskinsä, joten Javassa turvallisuus on otettu myös muilla tavoin huomioon (ks. “Java ja turvallisuus” sivulla 26).

5.2.2 Käytettävän ympäristön valinta

Tällä hetkellä näyttää siltä, että sekä Java että ActiveX tulevat puolustamaan paikkaansa tulevaisuudessa vuorovaikutteisten WWW-sovellusten ohjelmointiympäristöinä. Kummallakin on omat hyvät ja huonot puolensa, jotka suurelta osin vaikuttavat siihen, kumman näistä järjestelmien kehittäjät valitsevat toteutustavakseen.

Tässä työssä toteutukseen on valittu Java. Valinta on ilmeinen tutkimusongelman ja tavoitteiden määrittelyn myötä. ActiveX ei ainakaan toistaiseksi tarjoa laiteriippumattomuutta, joka oli keskeinen asia tavoitteiden kannalta.

5.3 Java-ohjelmointikieli

Java-ohjelmointikieli (tästälähtien pelkkä Java) tai ainakin sen perustana oleva idea on ehkä yksi viimevuosien merkittävimmistä edistysaskeleista Internettiin verkostuneessa maailmassa. Javan juuret juontavat 90-luvun alkuun, jolloin pieni Sun Microsystems -yhtiön työntekijöiden ydinryhmä lähti kehittelemään uutta oliopohjaista ohjelmointikieltä sulautetuille järjestelmille [50]. Alunperin *James Gosling* nimesi uuden kielen Oakiksi, mutta maailmanlaajuiseen levitykseen se päästettiin nimellä Java.

Tärkein Javan tuoma lisäarvo verrattuna muihin ohjelmointikieliin on mahdollisuus luoda ohjelmasia (applet) - pieniä ohjelmia, joita suoritetaan WWW-sivujen sisällä. Tällainen ohjelmanen voi kommunikoida käyttäjän kanssa WWW-selaimen sivulla käyttämättä resursseja palvelimesta sen jälkeen, kun se on sieltä verkon yli ladattu. Jotkut ohjelmaset voivat toki kommunikoida myös palvelimen kanssa, mutta se ei ole välttämätöntä. Tämän erityisominaisuutensa vuoksi Java soveltuu erittäin hyvin hajautettuihin ympäristöihin kuten WWW, mutta sillä on myös muita ominaisuuksia, jotka tekevät siitä varsin käyttökelpoisen.

5.3.1 Java-kääntäjä

Java mahdollistaa myös perinteisten sovellusten (ohjelma) ohjelmoinnin. Javaa käytetäänkin perinteisten ohjelmointikielten lisäksi paljon tapauksissa, joissa laiteriippuvuudella ei ole suurta merkitystä. Java tarjoaa käyttäjälleen turvallisuusominaisuuksiensa vuoksi mielekkään käännösympäristön, jossa ohjelmointi ja ohjelman testaus on helppoa. Jotkut tavanomaisista ohjelmointivirheistä voidaan kokonaan välttää esimerkiksi automaattisen roskankeruun ja turvallisten tyyppiviittausten vuoksi. Java tarjoaa myös tuen mm. monisäikeisille ohjelmille ja siinä on sisäänrakennettuna erityinen mekanismi poikkeustenhallintaan virheidenkäsittelyä varten. Java siis tarjoaa koko joukon tehokkaita sisäänrakennettuja ominaisuuksia - kielenä Java on kuitenkin varsin yksinkertainen.

Jotta Java-ohjelmia voitaisiin suorittaa mahdollisimman monessa eri laiteympäristössä, sen implementaatiotason riippuvuudet on pyritty minimoimaan. Esimerkiksi kokonaislukutyyppi int on sovittu 32-bittiseksi etumerkilliseksi (2:n komplementti) luvuksi riippumatta ohjelmaa suorittavasta prosessoriarkkitehtuurista. Kun ohjelmointikieli ja ajonaikainen ympäristö on tarkkaan määritelty, voidaan Java-ohjelmia ajaa missä tahansa laiteympäristössä. Edellytyksenä on, että laiteympäristölle on toteutettavissa Java-tulkki (virtuaalikone), joka kykenee suorittamaan Java-kääntäjän tuottamaa tavukoodia määritellyn standardin mukaisesti. Käytännössä tämä tarkoittaa vähintään 32-bittistä laitearkkitehtuuria.

Ulkoiselta olemukseltaan Java muistuttaa erittäin paljon C ja C++ -kieliä. Keskeiseltä filosofialtaan se on oliopohjainen ohjelmointikieli.

5.3.2 Java-kehitysympäristö ja -työkalut

Eräs merkittävä Javan etu on, että sen kehitystyökalut (Java Developer's Kit, JDK) ovat ainakin toistaiseksi ilmaisia. JDK sisältää Java-kääntäjän ja perinteisten ohjelmointityökalujen lisäksi paljon muitakin työkaluja, joista mainittakoon javadoc ja appletviewer -ohjelmat. Näistä ensinmainittu on hyödyllinen työkalu automaattiseen dokumentointiin. Ohjelmalla voi generoida lähdekielisestä ohjelmasta dokumentit suoraan HTML-muotoon. Tämä edellyttää, että ohjelma on kommentoitu javadocin ymmärtämällä tavalla. Appletviewer-ohjelma on pienimuotoinen Java-virtuaalikone (tulkki), jolla voi ajaa Java-ohjelmasia.

5.3.3 Komponenttikirjastot

Javan komponenttikirjastoja (API) on useita. Ydinkirjasto (Core API) sisältää vähimmäisjoukon komponentteja, jotka ohjelmoija voi olettaa löytyvän kaikista Javaa tukevista virtuaalikoneista. Lisäksi on JavaSoftin määrittelemä Standard Extension API, sekä koko joukko muiden osapuolien komponenttikirjastoja, jotka on määritelty API-standardin mukaisesti. Tällaisen API-määrittelyn etuna on, että mikäli ohjelmoija voi olettaa jonkin tietyn API:n löytyvän käyttöympäristöstä, hän voi käyttää kyseisen API:n komponentteja ilman, että niitä tarvitsee kääntää tai ladata verkon yli.

API voi koostua yhdestä tai useammasta paketista (package). Esimerkiksi ydinkirjaston API tarjoaa monenlaisia paketteja, joista mainittakoon java.applet (mahdol-

listaa Java-ohjelmasten luomisen Applet-luokan avulla), java.awt (Abstract Window Toolkit, sisältää valmiit toteutukset mm. valintalistoille, menuille, tekstikentille jne.), java.net (verkkotuki) sekä java.util (sisältää erilaisia tietorakenteita yms.).

5.3.4 Oliopohjainen ohjelmointi

Java on oliopohjainen ohjelmointikieli, kuten esimerkiksi C++. Seuraavassa on esitetty ne oliopohjaisen ohjelmoinnin keskeiset termit ja käsitteet, jotka ovat jatkon seuraamisen kannalta olennaisimmat.

Java-ohjelmoinnin peruskäsite on **luokka** (class). Luokat sisältävät toiminnallisia kuvauksia eli **metodeja** (methods) sekä muuttujien tyypinmäärittelyjä eli **kenttiä** (fields). Metodit koostuvat ajettavasta ohjelmakoodista, joka manipuloi kenttiä. Luokat tarjoavat rakenteen **olioille** sekä mekanismin tuottaa olioita luokan määrittelystä. Laskentaa voidaan suorittaa ainoastaan primitiivisillä **tyypeillä**, kuten kokonaisluvut (integer), liukuluvut (floating-point) jne. mutta oliopohjaisessa ohjelmoinnissa mielenkiintoista on nimen omaan se, miten olioita luodaan ja manipuloidaan.

Jokainen olio on luokkansa **instanssi**. Olion kenttien sisällöt ovat oliokohtaisia, ellei niitä erikseen määritellä staattisiksi. Ts. jokaisella saman luokan oliolla on samannimiset kentät, mutta erilliset instanssit niistä. Tällöin jokaisella erillisellä oliolla on oma **tilansa**. Kun olion metodia kutsutaan, tutkitaan kyseisen olion luokkaa, josta ajettava ohjelmakoodi etsitään. Jokaisella luokalla on nimi, joka on samalla myös luokkaa edustavan **olion tyyppi**. **Viittaus** olioon luodaan käyttämällä luokan nimeä viittauksen tyypinmäärittelyssä. Pelkkä viittaus ei kuitenkaan luo uutta oliota, vaan se täytyy tehdä esimerkiksi käyttämällä new-konstruktoria.

Jokainen Java-ohjelmoijan kirjoittama uusi luokka on peritty tavalla tai toisella. **Perinnällä** tarkoitetaan jonkin olemassaolevan **kantaluokan** ominaisuuksien siirtämistä uudelle **aliluokalle**. Aliluokka siis laajentaa kantaluokan määrittelyä perimällä kantaluokan ominaisuudet ja lisäämällä siihen omat määrittelynsä. Perimismekanismi synnyttää **luokkahierarkian**, joka on puurakenne ja jonka juurena on erityinen Javan komponenttikirjaston luokka Objects. Kaikki luokat siis periytyvät tästä luokasta, vaikka sitä ei suoraan olisi luokan määrittelyssä kerrottukaan.

Luokat voivat olla myös **abstrakteja**. Abstrakti luokka määrittelee vain osan implementaatiosta jättäen erityiset ominaisuudet (metodit) aliluokkien määriteltäviksi (abstrakteiksi). Abstrakti luokka on käyttökelpoinen tilanteessa, jossa suurin osa tietyn tyyppisten olioiden käyttäytymisestä voidaan mallintaa yleisellä tasolla, mutta pieni osa käyttäytymisestä on aliluokkakohtaista. Tällöin jokaisen abstraktista luokasta perityn aliluokan on implementoitava abstrakteiksi määritellyt kantaluokkansa metodit.

Jos kuitenkin halutaan määritellä vain metodien nimiä, joita jotkut luokat tukevat, mutta ei antaa niille mitään implementaatiota, voidaan käyttää **rajapintoja** (interface). Rajapinta on siis tässä suhteessa määrittely pelkästään mallinnuksen kannalta, kun taas luokka on mallinnusta ja implementaatiota sekaisin. Kaikki rajapinnan me-

todit ovat luonteeltaan abstrakteja, mutta koska rajapinta ei voi sisältää implementaatiota, sen metodeja ei ole tarvetta määritellä erikseen abstrakteiksi.

5.4 Java ja turvallisuus

Javan kohdalla turvallisuuskysymyksiä voidaan tutkia kahdelta eri puolelta. Toisaalta Java voidaan nähdä **ohjelmoijan** kannalta turvallisena ohjelmointiympäristönä, joka tarjoaa erittäin tehokkaat kehitystyökalut. Toisaalta taas Javalla tuotetut ohjelmat voidaan nähdä niiden **käyttäjän** kannalta jossain määrin turvattomina, kuten jatkossa tullaan esimerkkien avulla osoittamaan.

5.4.1 Turvallinen ohjelmointiympäristö

Kuten edellisissä kappaleissa on todettu, Java on ohjelmointiympäristönä varsin tehokas ja sitä kautta myös turvallinen. Esimerkiksi Javan sisäänrakennettu roskankeruu mahdollistaa turvallisen muistinhallinnan, koska ohjelmoijan ei tarvitse huolehtia olioiden viemän muistin vapauttamisesta. Lisäksi Java tarjoaa mm. viittausten tyypin tarkastuksen sekä erillisen virheidenkäsittelymekanismin (exception handling). Myöskään osoittimia ei ole. Tällöin monilta tyypillisiltä ohjelmointivirheiltä voidaan välttyä kokonaan. Kun vielä otetaan huomioon Javan tarjoaman ydinkomponenttikirjaston tuomat edut sekä se, että Java on kieliopiltaan kohtuullisen yksinkertainen ohjelmointikieli, voidaan sitä pitää *ohjelmoijan kannalta erittäin turvallisena ohjelmointikielenä*.

5.4.2 Turvaton verkkoympäristö

Kuten kappaleessa 5.2.1 todettiin, tuo verkkoympäristö mukanaan omat turvallisuusongelmansa. Mikäli verkkosivua ja sen tuottajaa, jolta Java-ohjelmanen ladataan, ei voida pitää luotettavana, tulee turvallisuudesta pystyä huolehtimaan myös ajettavan koodin tasolla. Javassa tämä on hoidettu erillisellä virtuaalikoneen turvallisuusmanagerilla (security manager), joka määrittelee ne ohjelmakutsut, jotka ajettava Java-ohjelmanen saa suorittaa. Tästä syystä esimerkiksi paikallisen levyn lukeminen ja kirjoittaminen on Java-ohjelmasissa estetty. Kyseessä on siis menettely, jossa Java-ohjelmasta ajetaan omassa “hiekkalaatikossaan”, jonka ulkopuolelle sillä ei ole asiaa. Mikäli ohjelmanen yrittää hiekkalaatikkonsa ulkopuolelle, herää turvallisuusmanageri ja aiheuttaa virheilmoituksen sekä ohjelman suorituksen keskeytymisen.

Tämäntyyppinen lähestymistapa aiheuttaa kuitenkin muita ongelmia. Esimerkiksi kuinka hiekkalaatikon rajat voidaan määritellä siten, että toisaalta ei aiheuteta potentiaalisia turvallisuusriskejä ohjelmaa suorittavalle järjestelmälle, mutta toisaalta ei myöskään estetä sellaisten järjestelmäkutsujen suorittamista, jotka olisivat joidenkin turvallisten sovellusten kannalta tarpeellisia. Esimerkiksi paikallisen levyn luku- ja kirjoitusoperaation estäminen vaikeuttaa monien hyödyllisten Java-sovellusten toteuttamista.

Turvallisuusaukkoja

Java-ohjelmasten turvallisuusaukot voidaan jakaa kahteen eri luokkaan. Ensimmäiseen luokkaan kuuluvat sellaiset turvallisuusaukot, jotka jollain tavalla rikkovat tai kiertävät selkeästi ohjelmasta suorittavan virtuaalikoneen turvallisuusmanagerin määrityksiä. Toiseen luokkaan kuuluvat sellaiset turvallisuusaukot, joissa pysytellään turvallisuusmanagerin sallimien toimintojen puitteissa.

Eräs esimerkki ensimmäiseen luokkaan kuuluvasta turvallisuusaukosta on Internetin välityksellä julki tuotu Java-ohjelmanen [34], joka pystyy tutkimaan, löytyykö paikalliselta levyltä tietyn nimistä ohjelmaa tai hakemistoa, vaikka tällaisen toiminnon suorittaminen ei ole sallittu selainohjelmien turvallisuusmanagereissa. Tällaisen turvallisuusaukon olemassaolo on varsin kiusallista esimerkiksi käyttäjän kannalta, joka selailee eri kaupallisten yritysten sivuja. Mikäli yritys haluaisi hyödyntää turvallisuusaukkoa, se voisi yrittää tutkia käyttäjän levyltä millaisia ohjelmia hänelle on asennettu ja hyödyntää näin hankittua tietoa omassa mainonnassaan.

Esimerkki toiseen luokkaan kuuluvasta turvallisuusaukosta on Java-ohjelmanen [35], joka lähettää WWW-sivulle tulleen käyttäjän nimellä sähköpostia tämän tietämättä. Kyseinen Java-ohjelmanen käyttää ainoastaan turvallisuusmanagerin sallimia systeemikutsuja. Kyseessä on menettely, jossa Java-ohjelmanen lähettää unix-koneille tyypillisen finger-komennon ohjelmasta suorittavalle koneelle ja saa (joskus) näin tietoonsa ohjelmasta suorittavan henkilön käyttäjätunnuksen.

Edellämainituista turvallisuusongelmista puhuttaessa tulee aina muistaa, että ne koskevat ainoastaan epäluotettavina pidettäviä verkkopalvelimia. Tällöin ohjelmasen tekijän tulee tieten tahtoen pyrkiä käyttämään hyväksi tai vahingoittamaan verkkokäyttäjää. Selainohjelmissa Java-ohjelmasten suoritus voidaan lisäksi estää tapauksissa, joissa väärinkäytöstä voidaan epäillä.

5.5 Johtopäätökset

Elävien ja vuorovaikutteisten WWW-sivujen tekemiseen on kehitetty erilaisia ohjelmointiympäristöjä. Näiden yhteydessä on alettu puhua Internet-ohjelmoinnista, jolla siis tarkoitetaan sovelluksia, jotka ohjelmina ovat osa WWW-sivua ja jotka kommunikoivat käyttäjän kanssa WWW-sivun ja selainohjelman välityksellä. Java on yksi tällainen Internet-ohjelmointiin tarkoitettu ohjelmointikieli.

Java on alunperin suunniteltu sulautettuja järjestelmiä varten. Yhdessä hajautettuja hypertekstidokumentteja varten suunnitellun WWW:n kanssa ne muodostavat varsin käyttökelpoisen kokonaisuuden. Vaikka Javan juuret eivät nykypäivänä olekaan enää kovin selkeästi nähtävissä, voidaan tulevaisuuden visiona nähdä Javan paluu juurilleen. Ovathan tänä päivänä erilaiset kannettavat matkapuhelimet (tässä esimerkkinä sulautetusta järjestelmästä) verkko-ominaisuuksineen tulossa markkinoille. Tällöin matkapuhelimen selainohjelma voisi olla Javalla toteutettu sekä Java-virtuaalikoneella varustettu, jolloin Java-sovelluksia voitaisiin ajaa tällaisissa kannettavissa laitteissa.

Ominaisuuksiensa puolesta Java soveltuu erityisen hyvin erilaisiin asiakas/palvelin (client/server) -tyyppisiin ratkaisuihin. Raskasta palvelinsovellusta voidaan suorittaa esimerkiksi unix-työasemassa, kun taas Javalla toteutettua kevyttä asiakassovel-

lusta voidaan ajaa lähes missä tahansa laiteympäristössä. Tällöin saavutetaan mm. seuraavia etuja:

- asiakassovellusta voidaan ajaa useissa eri laitearkkitehtuureissa ilman erityisjärjestelyjä, kuten lähdekielisen ohjelman kääntämistä uuteen laitearkkitehtuuriin,
- asiakassovelluksen päivitys on nopeaa ja tehokasta, koska se ladataan aina verkosta,
- käyttäjätunnuksia palvelinkoneeseen ei välttämättä asiakkaalle tarvita,
- tuki grafiikalle, audiolle sekä animaatiolle,
- tehokkaat ohjelmankehitystyökalut.

Huonona puolena voidaan pitää kehitysympäristön epävakautta. Varsin uutena ohjelmointikielenä Java-työkalut sisältävät pikkuvikoja ja ne ovat joiltain osin keskeneräisiä. Päivityksiä on saatavana, mutta niiden käyttöönottoa hidastaa saatavilla olevien Java-yhteensopivien selainohjelmien hidas markkinoille tulo.

6. WWW-TRAKLA

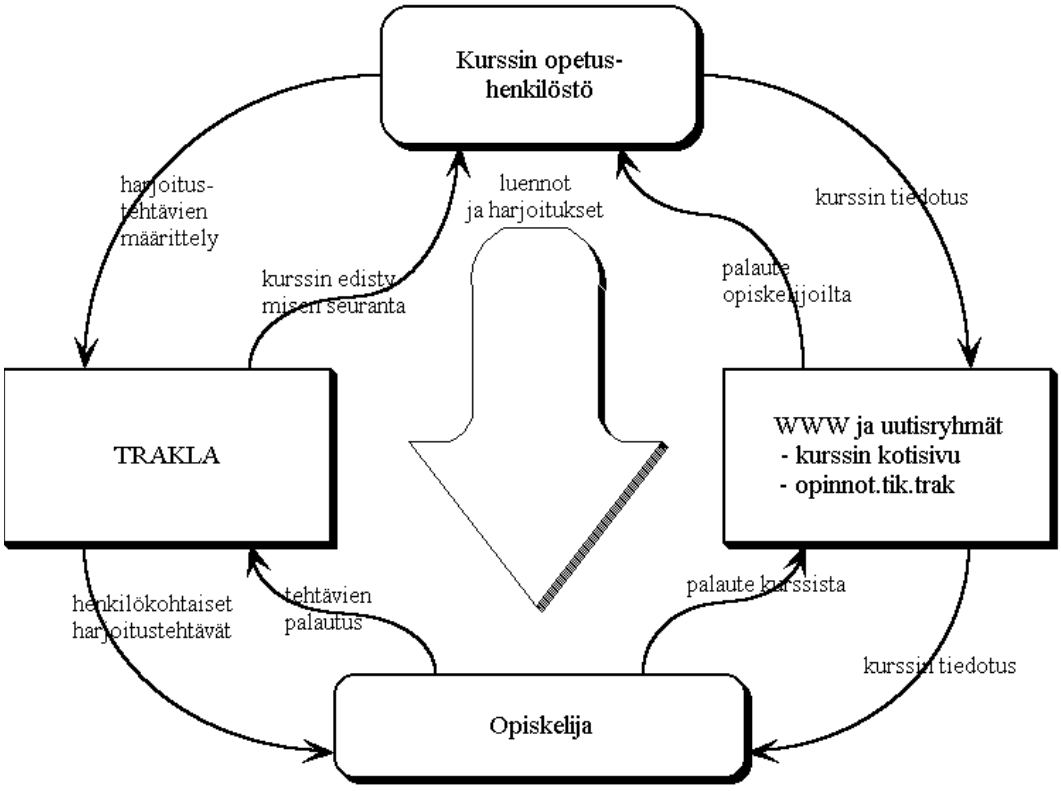
Tässä luvussa kuvataan TRAKLA-järjestelmän peruspiirteet, sille toteutettavan käyttöliittymän ominaisuudet sekä luodaan yleiskatsaus niiden yhdessä muodostamaan järjestelmään WWW-TRAKLA.

6.1 Yleistä

TRAKLA (TietoRakenteet ja Algoritmit; KotiLaskujen Arvostelu) [16] on Tietorakenteet ja algoritmit -kurssille opetuksen tueksi suunniteltu ohjelmisto. Ohjelmisto jakaa kurssille ilmoittautuneille opiskelijoille sähköpostin välityksellä henkilökohtaisia harjoitustehtäviä, sekä kykenee tarkastamaan opiskelijoiden lähettämiä vastauksia automaattisesti.

Kuvassa 6 on esitetty Tietorakenteet ja algoritmit -kurssin tärkeimmät resurssit *ennen* WWW-pohjaisen TraklaEditor-ohjelman käyttöönottoa. Oleellista on, että kurssin tiedotus ja harjoitustehtävät näkyvät opiskelijoille erillisinä osa-alueina.

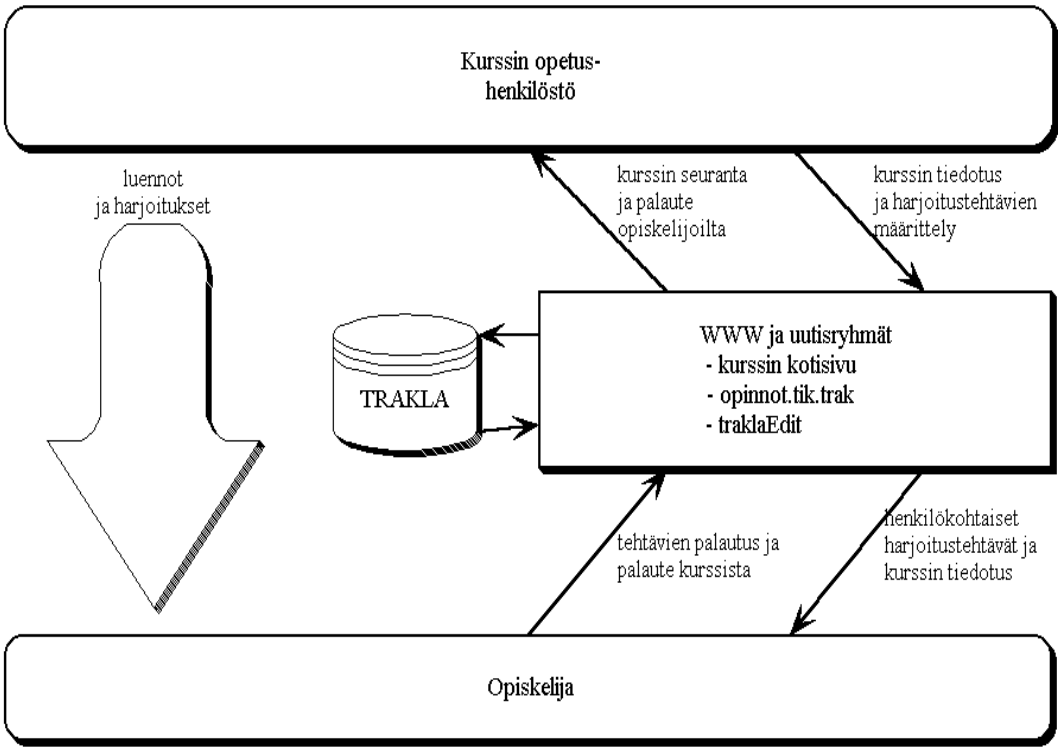
TRAKLA-ohjelmiston käyttöönoton jälkeen tiedotuskanavana käytettiin uutisryhmiä sekä WWW:tä. TRAKLA:n käyttöliittymä oli kuitenkin sähköpostipohjainen, joten tehtävien palautusta varten opiskelijan tuli saattaa vastauksensa TRAKLAn ymmärtämään muotoon.



Kuva 6. Tietorakenteet ja algoritmit kurssin resurssit ennen WWW-TRAKLAn käyttöönottoa.

Työn toteutusosassa toteutettiin WWW-pohjainen käyttöliittymä ja opetusohjelma *TraklaEdit* opetuksen tukijärjestelmälle *TRAKLA*. TraklaEdit on Java-ohjelmointikielellä toteutettu, hypertekstidokumenttien avulla käynnistyvä ja niitä hyödyntävä Java-ohjelmanen. Koko järjestelmästä käytetään tästä eteenpäin nimitystä *WWW-TRAKLA*. Kuvassa 7 esitetään pääsy kurssin eri resursseihin järjestelmän käyttöönoton jälkeen (vrt. kuva 6). Opiskelijan kannalta koko kurssin tiedotus ja muut resurssit voidaan nyt paikallistaa yhdestä ja samasta paikasta. Kurssin tiedotus toimii edelleen uutisryhmien ja WWW:n kautta, mutta nyt myös harjoitustehtäviä pääsee tekemään vuorovaikutteisten WWW-sivujen kautta, eikä tehtävien palautus vaadi erikoistoimenpiteitä, vaan ne voidaan TraklaEditiä käyttäen palauttaa suoraan TRAKLA-ohjelmistolle. Lisäksi opiskelijan ei tarvitse perehtyä TRAKLA-järjestelmään tai sen sisäiseen esityskieleen, koska kurssille ilmoittautuminen, tehtävistä saatujen pisteiden kyseleminen jne. voidaan hoitaa WWW-sivujen kautta. TRAKLA-ohjelmistolta saadut viestit opiskelijalle tulevat sähköpostilla opiskelijan ilmoittautumisen yhteydessä antamaan sähköpostiosoitteeseen.

Kurssin opetushenkilöstön kannalta WWW-TRAKLAn käyttöönotto edellyttää WWW-dokumentoinnin hallitsemista. Vanhaan järjestelmään verrattuna muutos näkyy siinä, että tehtäväpohjat tulee kirjoittaa HTML-muodossa ja lisäksi tehtäväkohtaisesti tulee määritellä kuhunkin tehtävään liittyvä TraklaEditorin toiminta (ks. “Parametrisointi ja toiminnallisuus” sivulla 37).

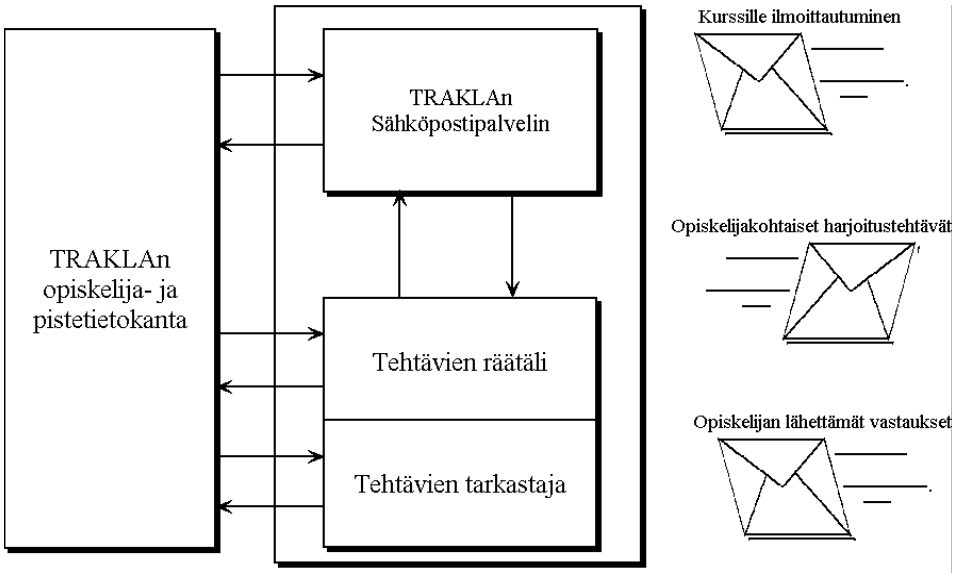


Kuva 7. Tietorakenteet ja algoritmit -kurssin resurssit WWW-TRAKLAn käyttöönoton jälkeen.

6.2 TRAKLAn toiminta

Kuvassa 8 on kaavio TRAKLAn toiminnasta. Opiskelija ilmoittautuu kurssille lähettämällä TRAKLAlle määrämuotoisen sähköpostiviestin, jossa hän ilmoittaa nimensä , opiskelijanumeronsa sekä sähköpostiosoitteensa. Kurssin tehtävät on jaettu tehtäväkierroksiin, joilla on omat määräaikansa. Tehtäväkierroksen alkaessa TRAKLA generoi (räätälöi) jokaiselle opiskelijalle henkilökohtaiset harjoitustehtävät (tehtävien rakenne on sama, mutta tietosisältö muuttuu). Opiskelijan tulee palauttaa tehtävät määräajan loppuun mennessä sähköpostilla TRAKLAlle, joka tarkastaa ja pisteyttää tehtävät. Lopuksi tehtävien pisteytys lähetetään opiskelijalle.

TRAKLA on suunniteltu massakurssia silmällä pitäen. Vastaavien tehtävien tuottaminen ja tarkastaminen paperiversioina olisi kohtuuttoman työläs operaatio. Kuitenkin kurssin keskeisiä tavoitteita on opettaa erilaisten yksittäisten algoritmien ja tietorakenteiden toimintaa. Tehokkaimmaksi oppimismenetelmäksi on havaittu lyhyiden tehtävien ratkaiseminen ja tätä kautta opetettavien algoritmien ja tietorakenteiden toiminnan ymmärtäminen.



Kuva 8. Yleiskuva TRAKLAn toiminnasta.

6.2.1 Tehtävien suunnittelu

Jotta tehtävien automaattinen generointi ja tarkastaminen onnistuisi, vaaditaan tehtävien luonteelta erityispiirteitä. Jokaista tehtävää varten tulee suunnitella tehtävärunko ja tehtävän parametrisoidut (muuttuva tietosisältö) osat. Lisäksi sekä räätälöintiä että tarkastusta varten tulee laatia ohjelmat, jotka riippuvat luonnollisesti toisistaan. Näin ollen tarkastusohjelma sisältää epäsuorasti tehtävän semanttisen merkityksen, koska räätälöinti voidaan tehdä pelkästään tehtävärungossa annettujen ohjeiden perusteella. Esimerkiksi lyhyille algoritmeille tämän tyyppinen lähestymistapa sopii erityisen hyvin. Algoritmin toiminta on ohjelmoitavissa yksiselitteisesti, joten annetulla syötteellä (muuttuva tietosisältö) saatu tuloste (opiskelijan palauttama vastaus) voidaan helposti tarkistaa. Koska kyseessä on annetun algorit-

min manuaalinen simulointi, tehtävät eivät ole triviaaleja, vaan vaativat annetun algoritmin toiminnan ymmärtämisen.

6.2.2 TRAKLA ja tietoturva

TRAKLA-järjestelmän tietoturva perustuu tiedostojen asianmukaiseen suojaukseen käyttöjärjestelmätasolla, sekä sähköpostin käyttöön. Ennen kurssin alkua opiskelijoiden on ilmoittauduttava kurssille, jolloin heidän opiskelijanumeronsa, nimensä ja sähköpostiosoitteensa kirjataan tietokantaan. Ilmoitettua opiskelijanumero/sähköpostiosoite -paria ei voi muuttaa kuin ottamalla yhteyttä kurssista vastaavaan henkilökuntaan. Tästä eteenpäin opiskelija tunnistetaan opiskelijanumeron perusteella. TRAKLA-järjestelmä vastaa kaikkiin saamiinsa viesteihin lähettämällä vastauksen tietokannassa olevaan opiskelijanumeroa vastaavaan sähköpostiosoitteeseen. Sähköpostin käytöllä ja ilmoittautumiskäytännöllä pyritään siis siihen, että opiskelija saa kaikista hänen tunnuksellaan lähetetyistä viesteistä kuittaukset ja vastaukset omaan postilaatikkoonsa.

Vaikka sähköpostia ei voidakaan pitää täysin aukottomana viestinvälitysjärjestelmänä, sopii se tähän tilanteeseen hyvin. Välitettävät viestit eivät ole arkaluontoisia, koska esimerkiksi opiskelijan pistetietoja voidaan yleisen käytännön perusteella pitää julkisina ja vastaavasti opiskelijan vastaus ei sisällä muille opiskelijoille hyödyllisiä vastauksia (henkilökohtaiset harjoitustehtävät).

6.3 TraklaEditin toiminta

Jo TRAKLAn suunnitteluvaiheessa heräsi järjestelmän jatkokehitysideoita, joista yksi oli graafisen käyttöliittymän toteuttaminen sähköpostilla tapahtuvan kommunikaation sijaan. Suurin osa TRAKLAlle toteutetuista tehtävistä on luonteeltaan graafisia, ts. niille löytyy olemassa oleva graafinen tulkinta tehtävän visualisoimiseksi. Erillinen käyttöliittymä mahdollistaa myös käyttäjän tekemien virheiden eliminoinnin kommunikoitaessa TRAKLA-serverin komentokielellä. Lisäksi tehtäväkohtainen sisäinen esityskieli voidaan piilottaa käyttäjältä.

Ensimmäisen TraklaEdit-version [17] toteutus tapahtuikin diplomityönä *Juha Hyvösen* toimesta, laiteympäristönä Macintosh. Toteutuksessa jokaista TRAKLAn räätäli/tarkastaja-ohjelmaparia vastaan toteutettiin editori, joka visualisoi annetun tehtävän. Opiskelijan sähköpostilla saama tehtäväviesti siirrettiin tiedostona TraklaEditille. Viesti sisälsi erityisiä ohjauskomentoja editoria varten selväkielisen tehtävänannon lisäksi. Opiskelija ratkaisi tehtävän graafisella, hiiripohjaisella käyttöliittymällä ja editori kykeni tallettamaan saadun ratkaisun TRAKLAn ymmärtämässä muodossa tiedostoon. Tiedosto lähetettiin tämän jälkeen sähköpostilla TRAKLAlle tarkastusta varten.

Ratkaisun pahin kompastuskivi oli laiteriippuvuus, koska kaikille opiskelijoille ei voitu järjestää pääsyä Macintosh-tietokoneelle. Vaikka TraklaEdit olikin toteutettu siirrettävyyttä silmälläpitäen, ohjelmiston siirtäminen useaan laiteympäristöön (ohjelma olisi pitänyt siirtää ainakin Unix ja MS-DOS ympäristöihin) osoittautui vaikeaksi. Lisäksi ohjelmiston ylläpidettävyys olisi vaikeutunut huomattavasti.

Tämän työn eräs keskeisistä tavoitteista oli laiteriippumattomuus toteutettavassa käyttöliittymässä. Tätä tarkoitusta varten käyttöliittymä haluttiin toteuttaa WWW-ympäristössä. Tähän tarjosi mahdollisuuden Java-ohjelmointikieli. Samalla kuitenkin saavutettiin joukko muita tällaisen ohjelmistoprojektin onnistumisen kannalta keskeisiä ominaisuuksia, kuten kirjallisuustutkimuksen yhteydessä jo todettiin. Näistä keskeisimpiä ominaisuuksia ovat saavutettavuus, päivitettävyys ja tehokas oppimateriaalin välitys. Esimerkiksi tehtävien vastaanotto ja lähetys eivät vaadi opiskelijalta erityistoimenpiteitä, vaan tehtävä haetaan automaattisesti selainohjelman ikkunaan ja vastaus palautetaan automaattisesti takaisin TRAKLA-järjestelmälle yhden painikkeen painalluksella.

Lisäksi käyttöliittymän suunnittelussa pyrittiin järjestelmän käyttö ja laajentaminen tekemään mahdollisimman joustavaksi. Kaikki TraklaEditin ominaisuudet on parametrisoitu, jolloin uusien tehtävin suunnittelu ja toteutus sekä vanhojen tehtävien muokkaaminen onnistuu HTML-pohjia muokkaamalla. Lisäksi käyttöliittymän hyödyntäminen onnistuu ilman TRAKLA-järjestelmääkin, esimerkiksi kouluttajan esitysvälineenä.

6.4 WWW-TRAKLA

Vaikka kurssin tehtävien ratkaiseminen WWW-sivuilla onkin työn keskeisin sisältö, on luonnollista, että kaikkia TRAKLA-järjestelmän ominaisuuksia voidaan käyttää hyväksi WWW-sivujen kautta. Tätä tarkoitusta varten sivutuotteena syntyi joukko WWW-lomakkeita, joilla hoidetaan kurssille ilmoittautuminen, tehtävien uusintatilaukset, pistetietojen kyselyt jne. Koko WWW-sivukokonaisuudesta käytetään yleisnimitystä WWW-TRAKLA. WWW-TRAKLA voidaan näinollen nähdä kirjallisuustutkimuksen alussa esitetyn TAO-määritelmän mukaisena tietokoneavusteisena oppimisympäristönä.

Ympäristö löytyy kurssin kotisivulta [46]. Tehtävien ratkomiseksi opiskelija tarvitsee selainohjelman, joka osaa suorittaa Java-ohjelmasia. Muilta osin ympäristön käyttö onnistuu aivan tavallisella selaimella. Oppimisympäristön kehitystyö on kesken, mutta tällä hetkellä sen tukemia ominaisuuksia ovat mm.

- tehtävien ratkominen ja vastausten palautus,
- mallitehtävät,
- erilaisten tilausten ja kyselyiden suorittaminen,
- linkit ulkoisiin resursseihin sekä
- mahdollisuus käyttää ympäristöä opettajan havainnollistamis- ja esitysvälineenä.

Esimerkiksi kurssin assistentti voi käyttää järjestelmää harjoituksissa ratkaisemalla WWW-TRAKLAn avulla TRAKLA-järjestelmän tukemat mallitehtävät. Jatkossa järjestelmää on tarkoitus kehittää siten, että WWW:n hypermediaominaisuuksia käytetään hyväksi entistä paremmin.

6.4.1 WWW-TRAKLA ja turvallisuus

Turvallisuusominaisuuksiensa puolesta WWW-TRAKLA vastaa lähes täysin TRAKLA-järjestelmää. Viestinvälitys on hoidettu muutamalla CGI-scriptillä, jotka käyttävät sähköpostia. CGI-scriptien huolellisella suunnittelulla ja toteutuksella niistä saadaan turvallisia. Pitäytymällä viestinvälityksessä WWW:ltä TRAKLAlle päin sähköpostissa, ei muita eroavaisuuksia ole. Itse asiassa TRAKLA-järjestelmä ei edes tiedä, tuleeko viesti WWW-TRAKLAlta vai suoraan opiskelijan lähettämänä sähköpostina.

Tehtävien lataaminen TRAKLA-järjestelmästä selainohjelmaan TraklaEditiä varten on hoidettu myös CGI-scriptillä ja erillisella räätälöintiohjelmalla. Koska kyseinen ohjelma ainoastaan lukee yleisiä TRAKLA-järjestelmän tiedostoja, ei tästäkään aiheudu uusia turvallisuusriskejä.

7. Työn toteutus

Tässä luvussa kuvataan toteutetun järjestelmän pääpiirteet.

7.1 Lähtökohta

Työn toteutusosan lähtökohtana oli graafisen, vuorovaikutteisen, oliopohjaisen, laajennettavan, parametrisoidun ja laiteriippumattoman käyttöliittymän suunnittelu ja toteutus TRAKLA-järjestelmälle. Javan komponenttikirjasto tarjosi osan graafisista olioista valmiina ja osa jouduttiin toteuttamaan itse. Lähtökohtana oli toteuttaa sellainen joukko uusia olioita, joilla mahdollisimman moni TRAKLA-järjestelmän harjoitustehtävistä olisi esitettävissä graafisesti. Vuorovaikutteisuus oli tarkoitus toteuttaa WWW-sivuilla ajettavalla Java-ohjelmasella, joka kommunikoi sekä käyttäjän että TRAKLAn kanssa. Laiteriippumattomuus oli tarkoitus saavuttaa Java-yhteensopivilla selainohjelmilla, joita oli tarjolla mm. Unix-, Windows- ja Macintosh -käyttöjärjestelmille. Laajennettavuus pyrittiin toteuttamaan sopivalla luokkahierarkialla, jossa uusien olioiden ja ominaisuuksien lisäys olisi helppoa. Lisäksi parametrisoimalla kaikki käyttöliittymän toiminnot pyrittiin joustavaan järjestelmään, jossa uusien tehtävien laatiminen ja vanhojen muuttaminen olisi helppoa, ilman Java-ohjelmasen uudelleenkääntämistä.

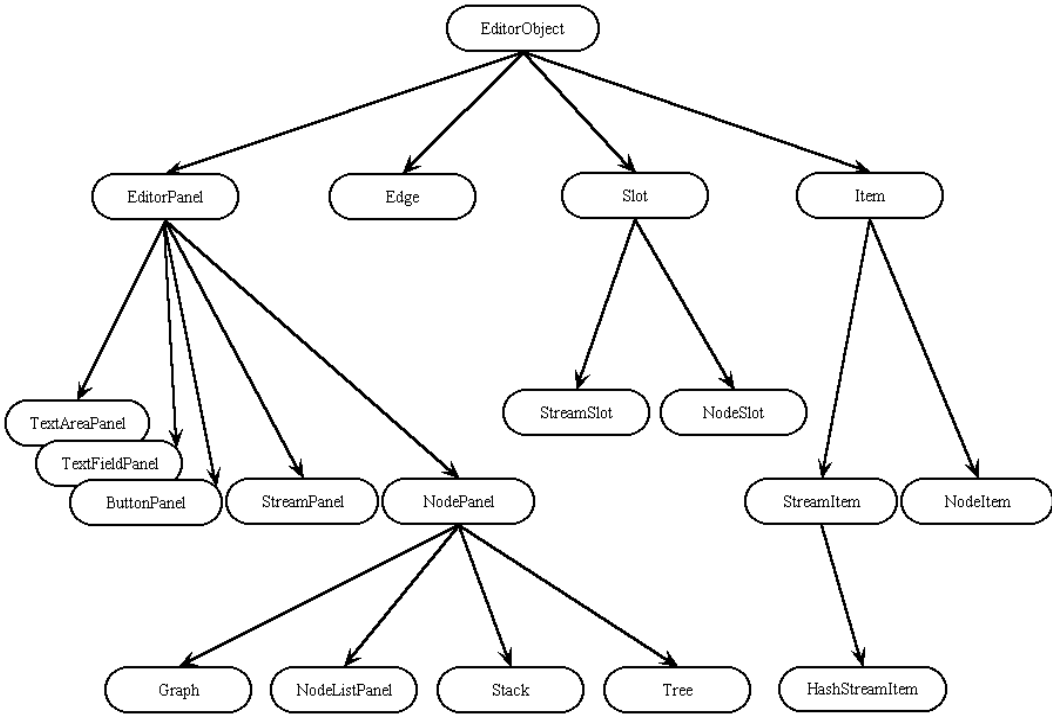
7.2 Toteutus

Työn toteutus alkoi sopivien graafisten olioiden suunnittelulla. Javan peruskomponenttikirjasto (Core API) tarjosi valmiit oliot tekstikentille (TextField), tekstialueille (TextArea), painikkeille (Button) sekä valintalistoille (Choice). Näiden avulla oli mahdollista toteuttaa minkä tahansa tekstimuodossa palautettavan tehtävän vaatimat toiminnot. Graafisesti esitettäville tehtäville tuli tämän lisäksi suunnitella omat olionsa. Ensimmäisessä vaiheessa toteutettiin paneeli-tyyppiset oliot tietovuolle (Stream), puulle (Tree), pinolle (Stack) sekä verkolle (Graph). Edellä esitetyistä olioista eli entiteeteistä on käytetty paneeli-nimitystä, koska ne koostuvat pienemmistä olioista nimeltä lovi (Slot) ja polku (Edge). Ne ovat siis tässä yhteydessä alustoja pienemmille olioille ja määrittelevät näiden olioiden sijoittelun toisiinsa nähden. Tässä lovi-olio on yleisnimitys esimerkiksi puissa käytetyille solmuille (NodeSlot) sekä tietovuo'issa käytetyille paikoille (StreamSlot). Paneeli, lovi- ja polkuentiteettien lisäksi tarvitaan varsinaisia liikuteltavia olioita edustava alkio-entiteetti (Item).

Edellämainituista luokista (olioista) suurin osa on määritelty abstrakteiksi eli varsinaiset instantioitavat luokat ovat näistä perittyjä aliluokkia. Esimerkiksi Item-luokasta on erikseen peritty luokat StreamItem (tietovuon alkio, joka on implementoitu neliönä) ja NodeItem (puun alkio, joka on implementoitu ympyränä). Vastaavasti Slot-luokasta on peritty jo edellämainitut paneeleiden paikkoja vastaavat luokat StreamSlot ja NodeSlot. Puhuttaessa abtraktin luokan oliosta, tarkoitetaan jonkin aliluokan instantiaatiota yleisesti.

Kuvassa 9 on esitetty luokkahierarkia kokonaisuudessaan. Kaikki oliot periytyvät abstraktista luokasta EditorObject, jossa on määritelty kaikille olioille yhteiset ken-

tät, kuten sijainti alustalla, olion koko, olion nimi jne. Lisäksi on määritelty joukko yhteisiä metodeja, kuten esimerkiksi olion liikuttelun salliminen tai estäminen (enable, disable) ja olion koon palauttava metodi size. Kaikkien EditorObject-luokasta perittyjen instanssiluokkien tulee määritellä metodi paint, jolla kyseinen olio piirretään alustalle.



Kuva 9. TraklaEdit-ohjelman luokkahierarkia.

7.2.1 Entiteetit

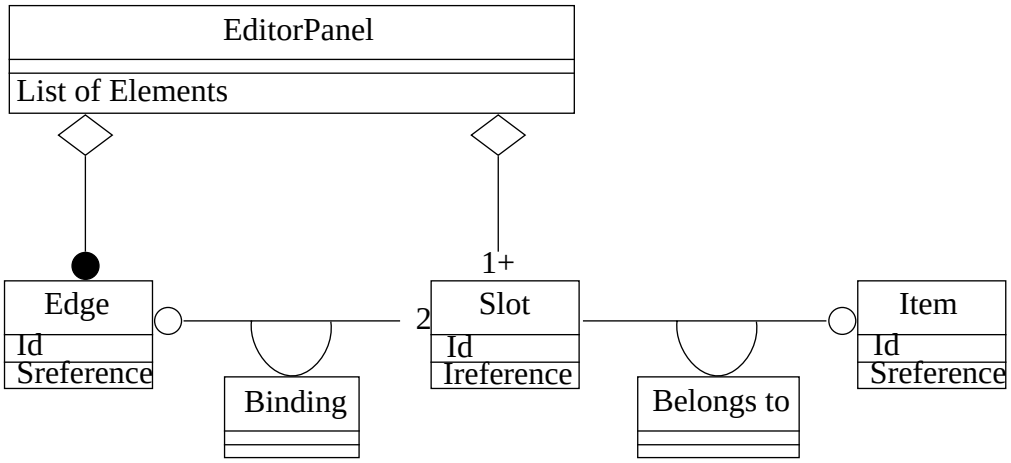
EditorObject-luokasta peritään neljä eri entiteettiä. Näistä paneelit (EditorPanel-luokka), lovet (Slot-luokka) ja alkiot (Item-luokka) ovat abstrakteja luokkia. Niistä on edelleen peritty varsinaiset instantioitavat luokat, joille on siis implementoitu myös kuvausta vastaava ulkoasu (metodi paint). Polku (Edge-luokka) ei ole abstrakti, eikä sillä ole myöskään aliluokkia.

Erityyppisiä paneeleita on viisi. NodePanel-luokan oliot edustavat paneeleita, joissa esiintyy solmuja (Node). Solmujen ulkoasu on ympyrä. Esimerkiksi luokka Tree on peritty NodePanel-luokasta ja sen implementaatio on puurakenne, jossa lovina käytetään solmuja (NodeSlot). StreamPanel-luokan paneelit puolestaan ovat olioita, joissa lovina on paikkoja (StreamSlot), joiden ulkoasu on neliö. Näiden kahden paneeliluokan lisäksi on määritelty luokat TextAreaPanel, TextFieldPanel ja ButtonPanel. Nämä luokat on tehty yhtenäisyyden vuoksi ja ne käyttävät hyväkseen Javan komponenttikirjaston vastaavia olioita TextArea, TextField ja Button.

Kuvassa 10 on esitetty entiteetit ja niiden väliset suhteet luokkakaaviona [15]. Kuten aikaisemmin jo mainittiin, paneelit voivat *koostua* (koostesuhde) lovista ja poluista. Esimerkiksi verkko (Graph-luokka) on olio, joka koostuu sekä solmu-tyyppisistä lovista (NodeSlot) että niitä yhdistävistä kaarista eli poluista (Edge). Kaari on olio,

joka *yhdistää* (binding) kaksi lovea toisiinsa. Jokainen lovi voi *sisältää* (belongs to) jonkin alkion. Tyypillisesti esimerkiksi solmu-tyyppisissä lovissa on solmualkioita (NodeItem). Vaikka erityyppisten alkioiden ja lovien sekoittaminen (esimerkiksi solmualkio (ympyrä) paikka-lovessa (neliö)) ei olekaan mielekästä, sitä ei ole mitenkään erikseen estetty.

Edellä kuvatut assosiaatiot on toteutettu kunkin luokan sisäisillä viittauksilla (Sreference, Ireference) vastapuolen olioihin.



Kuva 10. TraklaEdit-ohjelman entiteettien välinen luokkakaavio.

Paneeleiden, lovien ja alkioiden ulkoasua voidaan muuttaa myös erilaisilla attribuuteilla. Selkeyden vuoksi attribuutteja ei ole piirretty kuviin, koska niitä on melko paljon. Kattava lista ulkoisista attribuuteista on taulukossa 1. Esimerkkinä mainittakoon StreamPanel-luokasta attribuutti hashstream, jolla voidaan määritellä paneelin alustuksessa saamat alkiot soveltumaan erityisesti hajautusrakenteisiin. Tällöin alustettavista alkioista luodaan StreamItem-luokan olioiden sijasta tästä luokasta perityn HashStreamItem-luokan olioita, joiden ulkoasussa näytetään myös alkion numeerinen hajautusavain. Toinen esimerkki attribuuteista on lovien listarakenteet. Lovista voidaan attribuutin avulla tehdä listoja, jolloin yhteen ja samaan loveen voidaan siirtää useampia alkioita ja ne talletetaan listaksi. Kyseinen attribuutti on toteutettu erillisellä ListObject-luokalla, jota käytetään myös muualla lähdekielisessä ohjelmassa. Attribuutteihin voidaan vaikuttaa ohjelmasen saamissa parametreissä.

7.2.2 Parametrisointi ja toiminnallisuus

Java-ohjelmasilla ei ole varsinaista pääohjelmaa. Tästä syystä johtuu myös nimitys ohjelmanen, koska ne eivät ole itsenäisiä ohjelmia, vaan tarvitsevat erityisen alustan toimintaympäristökseen. Tällainen alusta voi olla esimerkiksi selainohjelman ikkuna tai erillinen Java-virtuaalikone. Jokaisessa Java-ohjelmasessa tulee olla komponenttikirjaston Applet-luokasta peritty luokka, josta alusta luo instanssin ja kutsuu kyseisen olion Init-metodia. TraklaEditissä tämä luokka on nimeltään TraklaEditor.

TraklaEditor-luokan Init-metodissa jäsennetään ohjelmasen saamat parametrit. Parametrisoinnin etuna on, että erilaisia tehtäviä voidaan koostaa kirjoittamalla sopi-

via HTML-tiedostoja. Näin yhtä ja samaa ohjelmasta voidaan käyttää kaikkien editoreiden pohjana, eikä jokaiselle tehtävälle tarvitse olla omaa ohjelmasta. Parametreissä määritellään editorin sisältämät graafiset oliot, niiden ominaisuudet sekä editorin toimintaan liittyvät attribuutit. Taulukossa 1 on esitetty parametrit, lyhyt kuvaus siitä mitä kullakin parametrillä voidaan määritellä sekä luettelo attribuuteista, joihin parametrillä voidaan vaikuttaa.

Parametrisoinnin huonona puolena voidaan pitää sitä, että koska ohjelmanen on yksi kokonaisuus, yksittäistä tehtävää varten joudutaan verkon yli lataamaan myös sellaisia ohjelman luokkia, joita ei tarvita. Ohjelmanen pyrkii kuitenkin käyttämään hyväksi komponenttikirjastoja, joiden ansiosta ohjelmasen koko on varsin pieni (kirjoitushetkellä n. 60 kilotavua). Tällöin latausaika on ongelma ainoastaan hitailla verkkoyhteyksillä. Tällöinkin on syytä huomata, että tyypillisesti ohjelmanen ladataan käyttökertaa kohden vain kerran. Ts. jos opiskelija palauttaa saman yhteyden aikana useita tehtäviä, ohjelmanen ladataan verkon yli ainoastaan ensimmäisen tehtävän yhteydessä. Tämä on mahdollista, koska selainohjelmilla on mahdollisuus käyttää käteismuistia. Jälkimmäisten tehtävien yhteydessä ohjelmanen voidaan ladata verkon sijasta nopeasti käteismuistista.

Tulevaisuudessa ohjelmasen koko tulee kuitenkin kasvamaan ja näin ollen ilman lisätoimenpiteitä latausaikakin pitenisi. Tällä hetkellä kuitenkin uusimmat selainohjelmat tukevat jo luokkien pakkaamista. Tällöin ohjelmasen latausaika pienenee olennaisesti, koska se saadaan pieneen kokoon tehokkaiden pakkausmenetelmien ansioista ja useiden tiedostojen (luokkien) sijasta voidaan siirtää ainoastaan yksi pakattu tiedosto. Ominaisuuden käyttöönotto ei kuitenkaan tällä hetkellä ole mahdollista, koska se sulkisi vanhempien selainohjelmaversioiden käytön pois. On kuitenkin odotettavaa, että tämä tekniikka voidaan ottaa käyttöön varsin pian vanhempien selainohjelmaversioiden poistuttua käytöstä.

Taulukko 1: Parametrien kuvaukset ja vaikutukset attribuutteihin.

Parametri	Kuvaus	Attribuutit
objects	Editoriin kuuluvat paneelit	label, hashstream, space
output	Vastaukseen kuuluvat paneelit	dstr, addStr, outputStateList
order	Lisämääre vastauksen ulkoasuun	order
posturl	CGI-scriptin URL-osoite	posturl
encode	Merkistön muunnos	
realname	Käyttäjän nimi	realname
exercise	Tehtävän numero	execise
userid	Käyttäjän opiskelijanumero	userid
courseid	Kurssin tunniste	courseid
solution	Varattu myöhempää käyttöä varten	

Taulukko 1: Parametrien kuvaukset ja vaikutukset attribuutteihin.

Parametri	Kuvaus	Attribuutit
rename	Olion uudelleennimeäminen	label
itemsize	Alkion koko	itemsize
show	Paneelin lovien indeksien näyttäminen	showableSlots
disable	Olion liikuttamisen estäminen	disabled
listarray	Loven listarakenteen salliminen	listarray
shadowed	Loven kiinnittäminen alkioon. Alkion nimi näytetään lovessa himmennettynä.	shadowed

Jäsentäjän lisäksi Init-metodissa määritellään editorin painikkeiden ja valintalistojen (reset, export ja size) nimet ja toiminta. Graafisille olioille luodaan oma alusta nimeltään GraphPanel (tätä ei tule sekoittaa graafiseen verkko-olioon Graph), johon varsinaiset graafiset oliot (nekin siis paneeleita) sijoitetaan. Luokka määrittelee metodit, joilla näitä olioita voidaan lisätä, poistaa ja siirtää alustalla. Alkioiden siirtelyt graafisten paneeliolioiden välillä rekisteröidään. Tällainen siirto tai siirtymä on tyypillisesti käyttäjän suorittama toiminto, jossa esimerkiksi alkio siirtyy lovesta toiseen tai käyttäjä merkitsee hiiren painikkeella verkon särmän. Toiminto on toteutettu Animated-rajapinnalla (interface) ja sitä vastaavalla AnimatedImpl-luokan implementaatiolla. Alkion move-metodi kutsuu Animated-rajapinnan metodia, joka tallentaa tietorakenteisiinsa tiedot siitä, mistä alkiosta oli kyse sekä mistä ja mihin loveen alkio siirtyi. Siirtojen rekisteröinti mahdollistaa käyttäjän tekemien siirtojen selaamisen eteen- ja taaksepäin, sekä lopullisen vastauksen generoimisen siirto kerrallaan. Käyttäjä voi itse selata siirtojaan GraphPanel-alustassa määritellyillä painikkeilla alkuun tai loppuun (begin, end), sekä siirto kerrallaan eteen- ja taaksepäin (forward, backward). TRAKLA-ohjelmistolle lähetettävä vastaus generoidaan, kun käyttäjä on painanut export-painiketta. Tällöin siirrot käydään läpi yksi kerrallaan ja vastaukseen sisällytetään alkioiden tilat graafisissa paneeleissa parametreissä määritellyllä tavalla. Tyypillisesti parametreissä määritellään jokin paneeli, jonka tiloista vastaus muodostuu. Vastaus voi olla esimerkiksi jonkin paneelin kaikki tilat, vain lopputila tai jotkut käyttäjän erikseen vastaukseen sisällyttämät tilat. Viimeksi mainitussa voidaan käyttää apuna esimerkiksi ylimääräistä painiketta, jolla käyttäjä voi määritellä kunkin palautettavan tilan.

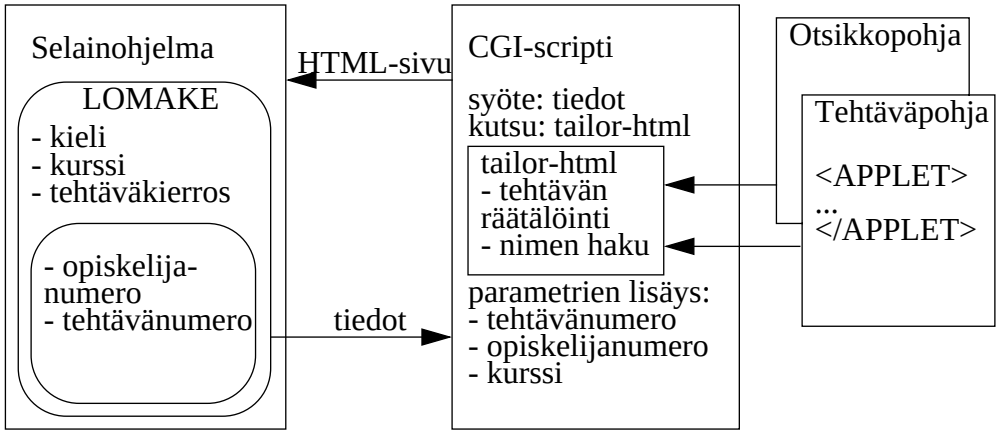
7.2.3 Kommunikaatio ulkomaailman kanssa

Kommunikaatio ulkomaailman kanssa tapahtuu kahdella eri CGI-scriptillä. Ensimmäistä käytetään, kun tehtäväpohja parametreineen luodaan dynaamisesti TraklaEdit-ohjelmaa varten. Toinen on tarkoitettu vastauksen generoimiseen TRAKLA-ohjelmistolle.

Dynaamiset tehtäväpohjat

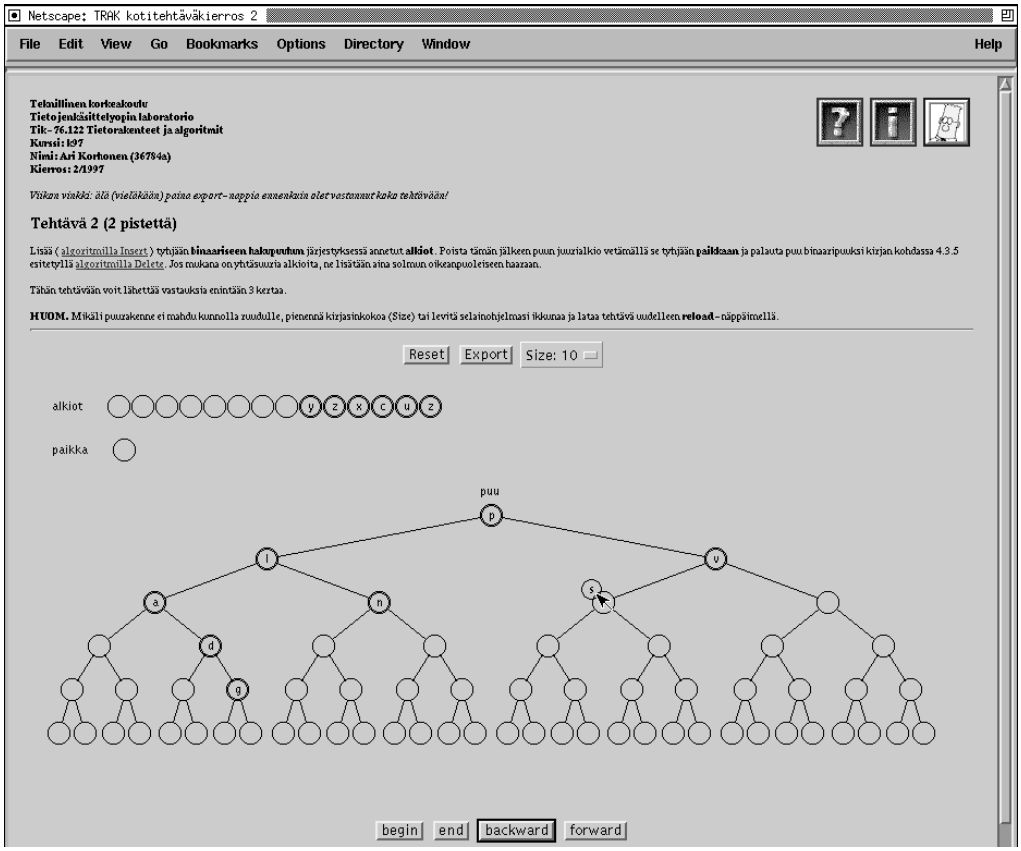
Tehtäväpohjien dynaamisuudella tarkoitetaan sitä, että lopullinen selainohjelmalle välitettävä HTML-tiedosto luodaan ajonaikaisesti. Koska jokainen opiskelija saa henkilökohtaiset tehtävät, on myös vastaava HTML-tiedosto kaikilla erilainen.

Kuvassa 11 on esitetty dynaamisten tehtäväsivujen generointi. Pääsy harjoitustehtäviin on toteutettu kieli-, kurssi- ja tehtäväkierroskohtaisilla WWW-sivuilla. Sivut sisältävät WWW-lomakkeen (Form), johon opiskelija täyttää opiskelijanumeronsa sekä valitsee haluamansa tehtävän numeron. Lomakkeen tiedot lähetetään CGI-scriptille (liite A), joka käynnistää erillisen räätälöintiohjelman (tailor-html). Räätälöintiohjelma hakee annetuilla parametreillä oikeat HTML-pohjat, joista se räätälöi käyttäjäkohtaisia tietoja hyväksikäyttäen henkilökohtaisen tehtäväsivun HTML-muodossa. Pohjia on kaksi, joista ensimmäinen on yleinen otsikkopohja, jossa näkyvät mm. tehtäväkierros, opiskelijan opintokirjan numero sekä nimi. Toinen on tehtäväkohtainen pohja, jossa on varsinainen tehtävä ja sitä vastaavan ohjelmasen määrittelyt. CGI-scripti lisää myös sivulle muutaman parametrin käyttäjän tietojen perusteella. Tämä siksi, että TRAKLAn tehtäväkohtaiset tailor-moduulit eivät osaa palauttaa mm. tehtävän numeroa eivätkä käyttäjän tietoja. Kun tehtävä on räätälöity, CGI-scripti palauttaa sivun selainohjelmalle.



Kuva 11. Dynaamisten tehtäväpohjien generointi

Liitteessä C on esimerkkitehtävän tehtäväpohjat ja niitä vastaava HTML-tiedosto. Räätälöintiä varten tarvittavat makrot ja niiden lavennukset on lihavoitu. CGI-scriptin lisäämät parametrit HTML-tiedostossa on kursivoitu. Kuvassa 12 on vastaava tehtävä selainohjelmalla katsottuna.



Kuva 12. Liitteen C HTML-tiedoston mukainen tehtävän ulkoasu selainohjelmassa.

Esimerkkitehtävässä pyydetään lisäämään tyhjään binääriseen hakupuuhun annetut alkiot. Osa alkioista on jo siirretty hakurakenteeseen ja osa on vielä siirtämättä. Alkiota S ollaan juuri asettamassa oikeaan paikkaan. Tehtävänannossa on lisäksi annettu linkkien avulla viitteet tehtävässä käytettyjen algoritmien lähdekielisiin ohjelmiin, jotka löytyvät myös verkosta. Järjestelmän ohjeisiin pääsee ylälaidassa oikealla olevien kuvakkeiden avulla.

Tehtävän vastauksen palauttaminen

Toinen CGI-scripti (liite B) on tarkoitettu viestinvälitykseen TraklaEditiltä ja WWW-sivuilta TRAKLAlle. Samaa CGI-scriptiä käytetään mm. vastausten palauttamiseen, kurssille ilmoittautumiseen sekä pisteiden ja muiden kyselyjen lähettämiseen.

Export-painikkeen painaminen aiheuttaa vastauksen generoimisen ja lähetyksen. TraklaEditin parametreissa määritellään URL-viittaus vastauksen käsittelevään CGI-scriptiin. Viittaus on annettu liitteen C HTML-tiedostossa ohjelmasen parametrillä **posturl**. Vastauksen lähetyksessä muodostetaan yhteys WWW-palvelimeen, jossa CGI-scripti sijaitsee. Palvelimen tulee olla sama, josta ohjelmanen on ladattu. Tästä syystä palvelimen osoite tulee antaa IP-numerolla, koska selainohjelmien turvallisuusmanagerit eivät luota nimipalvelimeen. Kun yhteys on muodostettu, vastaus lähetetään määrämuodossa, jonka jälkeen scripti lähettää sen sendmail-

ohjelmalla TRAKLA-järjestelmälle. Onnistuneesta lähetyksestä palautetaan CGI-scriptin antama viesti käyttäjälle.

Tästä eteenpäin vastauksen tarkastaminen sekä siitä saatavat kuittaukset tulevat kuten sähköpostipohjaisessakin järjestelmässä. TRAKLA kuittaa saamansa postin lähettämällä siitä viestin käyttäjän postilaatikkoon. Kun tehtävä on tarkastettu, lähetetään siitä pistetiedot samoin sähköpostilla.

7.2.4 Ohjelman laajennettavuus

Ohjelman laajennettavuus on pyritty pitämään mahdollisimman helppona. Ohjelmaa voidaan laajentaa joko lisäämällä uusia ominaisuuksia olemassa oleville olioille tai luomalla kokonaan uusia olioita. Ensimmäinen vaihtoehto sisältää käytännössä uuden attribuutin, sitä vastaavan parametrisoinnin sekä toiminnallisuuden lisäämisen lähdekieliseen ohjelmaan. Attribuutit ja parametrisointi tulisi suunnitella mahdollisimman yleisiksi, vaikka varsinaista attribuutin mukaista toiminnallisuutta ei kaikille olioille toteutettaisikaan. Näin toiminnallisuus voidaan toteuttaa myöhemmin ilman uuden parametrin lisäämistä. Lisäksi on syytä huomata, että attribuutti saattaa olla jo toteutettu, vaikka siihen ei parametreillä pystyisikään vaikuttamaan. Tällainen attribuutti on esimerkiksi olion piilottaminen (hide) alustalla, jota eräät paneelit käyttävät hyväkseen. Parametriä ei kuitenkaan ole toteutettu, koska sille ei ole ollut TRAKLAn tehtävissä yleisempää käyttöä.

Toinen vaihtoehto ohjelman laajentamiseksi on uusien olioiden luominen. Ensimmäisessä vaiheessa toteutettujen staattisten olioiden lisäksi jatkossa on tarkoitus toteuttaa joitakin dynaamisia rakenteita. Tässä yhteydessä staattisilla olioilla tarkoitetaan rakenteita, joissa olion koko ei muutu. Esimerkiksi binääripuu on toteutettu rakenteena, jossa parametreissä määritellään puun solmujen lukumäärä vakiolla, eikä puu voi kasvaa rakenteen manipuloinnin seurauksena. Esimerkkinä dynaamisesta rakenteesta voisi olla binääripuu, jossa solmujen koko ja puun korkeus voisivat kasvaa manipuloinnin seurauksena (kun puuhun lisätään alkioita). Tällöin olemassa oleva luokka Tree voitaisiin joko korvata uudella luokalla tai luoda kokonaan uusi luokka rinnalle.

Yksinkertaisin tapa luoda uusia luokkia on periä luokka jostain sopivasta vanhasta luokasta. Tällöin uudelle luokalle saadaan valmiiksi joukko ominaisuuksia, joita ei välttämättä tarvitse toteuttaa uudelleen. Mitä ominaisuuksia luokka perii, riippuu siitä miten matalalla luokkahierarkiassa kantaluokka on.

7.3 Johtopäätökset

Oliopohjaisuutensa ansiosta Java tarjoaa erittäin hyvät puitteet ohjelmistojen suunnitteluun ja toteutukseen. Monipuolisten valmiiden komponenttikirjastojen ansiosta ohjelmointi on nopeaa ja oikein suunniteltuna ohjelmista tulee turvallisia ja helposti laajennettavia. Lisäksi komponenttikirjastojen käyttö tekee ohjelmista helppokäyttöisiä, koska Java yhdistettynä WWW:n kanssa tarjoaa ympäristön, jossa saavutettavuus ja päivitettävyys voidaan taata ilman ylimääräistä työtä.

Koska komponenttikirjastot tarjoavat koko joukon valmiita graafisia komponentteja, on Java-ohjelmasilla helppo toteuttaa erittäin visuaalisia ja vuorovaikutteisia ohjelmistoja. Samalla ohjelmille voidaan rakentaa yhdenmukainen ja looginen ulkoasu muihin samankaltaisiin ohjelmiin nähden, jolloin ohjelmien käytettävyys paranee.

TraklaEditin parametrisointi oli looginen seuraus Java-ohjelmasten määrittelytavasta ja Javan oliopohjaisuudesta. Koska itse ohjelmanen ja sen saamat parametrit määritellään HTML-tiedostossa oli parametrisointi valintana luonteva. Aikaisempi Macintosh-versio TraklaEditistä lähtikin liikkeelle TRAKLAn ominaisuudesta, jossa jokaiselle eri tehtävälle on oma erillinen räätäli/tarkastaja-ohjelmapari. Tällöin jokaiselle tehtävälle luotiin vastaavasti oma editorinsa. Parametrisointi osoittautui kuitenkin tässä tapauksessa luontevammaksi vaihtoehdoksi. Samalla sillä voitiin saavuttaa TraklaEditiin sellaisia ominaisuuksia, kuten

- parempi päivitettävyys, koska uusi tehtävä voidaan määritellä suoraan HTML-tiedostolla,
- monikäyttöisyys, koska TraklaEditin ei tarvitse välttämättä olla sidottu TRAKLA-järjestelmään, vaan se voi toimia myös itsenäisenä opetusohjelmana sekä
- ohjelman helpompi hallittavuus, koska editorin yksittäisiä ominaisuuksia ei tarvitse päivittää kuin yhteen paikkaan.

Tehtävien generointi dynaamisesti osoittautui myös erittäin onnistuneeksi vaihtoehdoksi. Tätä varten tarvittiin ainoastaan yksi CGI-scripti sekä oma erillinen yleinen räätälöintiohjelma. Dynaamisilla tehtäväpohjilla saavutettiin mm. seuraavia etuja, verrattuna siihen, että jokaiselle opiskelijalle olisi generoitu jokainen tehtävä valmiiksi tiedostoon.

- Tehtäviä voi päivittää helposti jopa kesken tehtäväkierroksen, koska muutokset tehdään vain yhteen paikkaan.
- Päivitykset näkyvät opiskelijoille ilman lisätoimenpiteitä, koska tehtävä ladataan joka kerta verkosta.
- Tehtäviä ei tarvitse generoida jokaiselle opiskelijalle erikseen.
- Tehtäviä varten ei tarvita ylimääräistä levytilaa jokaista opiskelijaa kohden.

Jatkossa ohjelmaa on tarkoitus kehittää uusien tietorakennekomponenttien lisäksi myös siten, että vastausten lähetys TRAKLA-järjestelmälle toteutettaisiin suoraan esim. CGI-scriptillä, eikä sähköpostilla kuten nykyisin.

Vaikka ohjelmoijan kannalta Java on turvallinen ja miellyttävä ohjelmointiympäristö ja vaikka työn toteutusta voidaankin pitää onnistuneena, ei ongelmilta vältytty. Ensisijaisesti ongelmat aiheutuivat siitä, että eri selainohjelmissa Java-virtuaalikoneet oli toteutettu huonosti tai Java-ohjelmaset eivät lupauksista huolimatta kaikissa ympäristöissä toimineet lainkaan. Käytännössä ainoa toimiva selainohjelma oli Netscape Navigator. Muilla selainohjelmilla ongelmia aiheutti se, että ne eivät tukenee kaikkia HTML-kielen piirteitä. Esimerkiksi Java-ohjelmaselle määritelty ikkuna voidaan Netscape Navigatorissa määritellä suhteellisesti (esim. 80% ruudun leveydestä), jolloin levittämällä selainohjelman ikkunaa, saadaan ohjelmaselle

enemmän tilaa käyttöön. Muissa selainohjelmissa tehtävät, joissa ikkuna oli määritelty suhteellisesti, eivät kaikilla selainohjelman versioilla toimineet. Tällainen määrittely ei ole kaikissa suhteissa standardin mukainen, joten sen puuttumista ei kuitenkaan voidan pitää varsinaisena selainohjelman virheenä. Varsinainen puute on kuitenkin joidenkin selainonhjelmien puutteellinen HTML-kielen META-komentojen toteutus. Esimerkiksi Macintoshin Internet Explorerilla tehtäviä ei voinut ratkoa, koska tehtäväpohjien eräs keskeinen määrittely on HTML-tiedoston suhteellisen sijainti (BASE-address). Koska tehtäväsivut toteutettiin dynaamisesti, syntyi tarve määritellä tehtäväpohjan sijainti uudelleen, jotta Java-ohjelmanen voitiin paikallistaa oikeasta paikasta. Selainohjelmat, jotka eivät META-komentoja tunne, eivät tällöin löydä määriteltyä Java-ohjelmasta lainkaan.

Myös Netscpape Navigatorin kanssa ilmeni ongelmia. Joissain laitearkkitehtuureissa Netscapen virtuaalikoneet toimivat virheellisesti. Muutamissa tapauksissa nämä virhetoiminnot pystyttiin kiertämään, mutta osa virheistä oli sellaisia, että kaikkia tehtäviä ei pystynyt kyseisellä selainohjelmaversiolla ratkaisemaan.

Edellä kerrotusta seuraa, että laiteriippumattomuus saavutettiin vain osittain. Lisäksi eräs keskeisistä Java-ohjelmoinnin hyvistä puolista menetettiin: toteutettua ohjelmaa jouduttiin testaamaan useilla eri laitearkkitehtuureilla ja ympäristöillä, jotta se saatiin toimimaan edes niissä ympäristöissä, jotka olivat opiskelijoiden kannalta tärkeimmät.

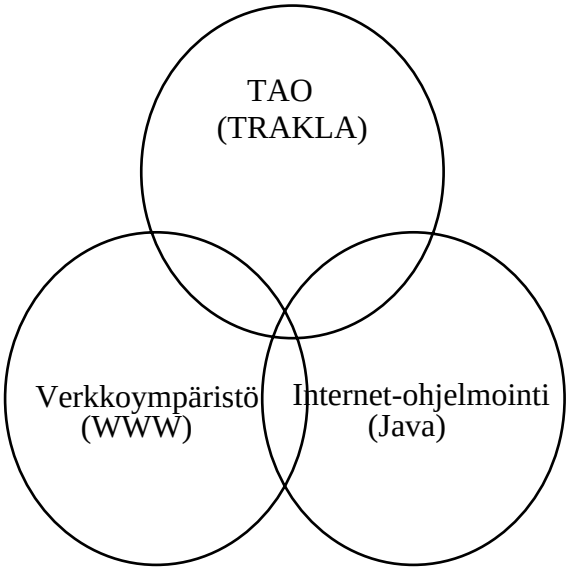
Suurin syy edellä mainittuihin virheisiin on varmaankin siinä, että Java on tuotteena kohtuullisen uusi ja tuki sille on tullut selainohjelmiin vasta äskettäin. Tulevaisuudessa lienee lupa odottaa parempia ja toimivampia Java virtuaalikoneita, jolloin näistä ongelmista on mahdollisuus päästä eroon. Sitä ennen on tyydyttävä niiden ympäristöjen käyttöön, joissa Java-ohjelmaset toimivat oikein.

8. Tulosten tarkastelu

Tässä luvussa tarkastellaan työstä saatuja tuloksia luvussa 2 asetettujen tavoitteiden ja kriteereiden pohjalta.

8.1 Yleistä

Tässä työn pohjana oleva malli tietokoneavusteisen oppimisympäristön toteutukseen WWW-ympäristössä on kuvassa 13. Oppimisympäristö koostuu kolmesta keskeisestä komponentista: TAO, verkkoympäristö sekä Internet-ohjelmointi. Näistä tietokoneavusteiseen opetukseen (TAO) tarkoitettu TRAKLA-järjestelmä oli toteutettu jo aikaisemmin ja oli varsinaisen työn pohjana. Verkkoympäristössä WWW oli työn kannalta keskeisessä asemassa ja Internet-ohjelmointiin tarkoitetuista ohjelmointiympäristöistä toteutusta varten valittiin Java-ohjelmointikieli.



Kuva 13. Malli tietokoneavusteiseksi oppimisympäristöksi WWW:ssä. Työssä toteutettiin tietokoneavusteiseen opetukseen tarkoitettuun TRAKLA-järjestelmään WWW-pohjainen käyttöliittymä Java-ohjelmointikielellä.

8.2 Alkuperäiset tavoitteet

Alkuperäisistä tavoitteista laiteriippumattomuus saavutettiin periaatteellisella tasolla. Kuten jäljempänä esitettävästä palauteesta käy ilmi, ei ongelmilta kuitenkaan täysin vältytty. Vaikka Java-yhteensopivia selainohjelmia olikin tarjolla kaikille käytetyille laitearkkitehtuureille, niiden toiminnassa oli toivomisen varaa. Eräissä ympäristöissä järjestelmää ei saatu toimimaan lainkaan tai se toimi vain osittain. Joissain tapauksissa ongelma voitiin kiertää vaihtamalla selainohjelmaa tai sen versiota. Joissain tapauksissa taas edes ongelman todellinen syy ei selvinnyt. Yhteenvetona voidaan kuitenkin todeta, että löytyi joukko ympäristöjä, joissa järjestelmä

toimi moitteettomasti tai lähes moitteettomasti. Näin ollen kaikille testikäyttäjille voitiin osoittaa ympäristö, jossa järjestelmän käyttö oli mahdollista.

Toinen keskeinen tavoite oli vastaavien sovellusten kartoittamisen, jonka yhteydessä kävi ilmi, että täysin vastaavia sovelluksia ei löytynyt. On kuitenkin huomattava, että kuvan 13 eri komponenteille löytyy kirjallisuudesta hyvinkin runsaasti vertailuaineistoa, kuten kirjallisuustutkimus osoitti. Työn eräänä keskeisenä johtopäätöksenä voidaankin pitää, että tässä työssä esitetty lähestymistapa, jossa eri komponentteja yhdistellään uudella tavalla, on varsin onnistunut ja mielekäs tietokonevusteisten opetusohjelmien suunnittelussa ja toteutuksessa.

Kolman tavoite oli toteuttaa turvallinen järjestelmä. Tämän tyyppisistä järjestelmistä saadaan turvallisia, koska Java tarjoaa turvallisen ohjelmointiympäristön (ks. 5.4) eikä järjestelmä oikein suunniteltuna ja toteutettuna aiheuta uusia turvallisuusaukkoja (ks. 6.2.2, 6.4.1).

Lisäksi kuvan 13 eri komponenttien hyvät puolet voidaan yhdistää ja käyttää hyväksi. Tällaisia hyviä puolia ovat mm. luvuissa 3.3.1 ja 3.3.2 esitetyt opetusohjelmien tuomat edut. Saavutetut edut riippuvat pitkälti kunkin järjestelmän ominaisuuksista. WWW-TRAKLAn kohdalla tällaisia keskeisiä etuja ovat mm. erilaisten rutiininomaisten toimintojen siirtäminen tietokoneavusteiseksi, visuaalisten ja vuorovaikutteisten animaatioiden hyödyntäminen sekä ohjelmiston saavutettavuus, siirrettävyys ja päivitettävyys. Yleisesti voidaan sanoa, että yhdistämällä voidaan saada opetusohjelman ominaisuuksille luvussa 4 esitetyt hypermedian tuomat mahdollisuudet sekä luvussa 5 esitetyt Java-ohjelmointikielen tuomat edut. Yhdistelyllä saavutetaan myös aivan uusia ominaisuuksia. Koska käytössä on verkkoympäristö sekä sopivat ohjelmointityökalut, voidaan kokonaisuus nähdä erittäin sopivana juuri asiakas/palvelin-arkkitehtuureihin. Raskas palvelinsovellus (tässä TAO-järjestelmä TRAKLA) voidaan toteuttaa sopivalla perinteisellä ohjelmointikielellä ja kevyt asiakaspään sovellus puolestaan Javalla.

8.3 Kriteerit

Järjestelmä oli testikäytössä vuoden 1997 kevään Tietorakenteet ja algoritmit -kurssilla [46]. Järjestelmälle asetettuja kriteereitä on seuraavassa tarkasteltu toisaalta kurssin suorittaneiden opiskelijoiden antaman palautteen kautta ja toisaalta vertailemalla järjestelmän ominaisuuksia luvussa 3 esitettyihin esimerkkijärjestelmiin ASA [13] ja Dynalab [7].

Seuraavassa on yhteenveto saadusta palautteesta WWW-TRAKLAn osalta. Kurssille osallistui n. 600 opiskelijaa, joista 476 vastasi palautekyselyyn. Kysely koski koko kurssia, mutta tässä on esitetty vain WWW-TRAKLAa koskeva osuus. Varsinainen kyselystä saatu palaute, jossa näkyy myös kysymyksenasettelu, on liitteessä D. WWW-TRAKLAa oli käyttänyt 37,5% (218 opiskelijaa) vastanneista.

Ylivoimaisesti suurin (57,4%) syy siihen, että opiskelijat **eivät** olleet käyttäneet WWW-TRAKLAa oli se, että heidän mielestään sähköpostilla tehtävien palauttaminen oli helpompaa. Toisaalta lähes 60% näistä opiskelijoista ei edes yrittänyt käyttää järjestelmää hyväkseen! Muita syitä käyttämättä jättämiseen olivat ajan

puute uuden järjestelmän omaksumiseen (39,8%), pelko siitä, että käytöstä olisi ollut ylimääräistä vaivaa (26,4%) sekä se, että graafiselle päätteelle olisi joutunut jonottamaan (16,8%). Huomionarvoista on, että kukaan vastanneista ei antanut syyksi sitä, ettei saanut järjestelmää toimimaan. Samaan aikaan WWW-TRAKLAa **käyttäneistä** opiskelijoista enemmistö piti tehtävien palauttamista WWW-TRAKLAlla helpompana (56,9%) vaihtoehtona kuin sähköpostilla palauttamista. Muista syistä järjestelmän käyttöön sähköpostin sijasta olivat keskeisimpiä järjestelmän helppokäyttöisyys (53,7%), halu tutustua uuteen järjestelmään (45,0%) ja se, että tehtävien ratkaiseminen oli nopeampaa (44,0%).

Saadusta palauteesta voidaan arvioida, että opiskelijoista noin puolet käytti järjestelmää kotoaan käsin. Lähes puolet käytti MS-Windows ympäristöä (45,9%) ja toinen puoli jotakin Unix-ympäristöä. Vain murto-osa oli käyttänyt Macintoshia tai jotakin muuta ympäristöä (2,9%). Selainohjelmista käytettiin eniten Netscape Navigatoria (85,3%).

Yli 90% opiskelijoista ilmoitti, että järjestelmän kanssa ei ollut ongelmia lainkaan tai vain hieman ongelmia. 82,2% sai järjestelmän toimimaan välittömästi ja 14,1% ohjeiden avulla. Vain 2,6%:lla oli suuria ongelmia (5 opiskelijaa). 94,7% piti järjestelmää erittäin tai kohtuullisen helppokäyttöisenä. Tietorakenteiden graafisia toteutuksia piti 93,6% erittäin tai melko loogisina.

Järjestelmän ohjeista voidaan todeta, että 53,1% ei tarvinnut niitä lainkaan ja 30,3% oli sitä mieltä että ne olivat riittävät. 16,6% ei joko löytänyt ohjeita tai piti niitä riittämättöminä.

Kysymys järjestelmän vuorovaikutteisuudesta aiheutti eniten hajontaa. 17,1% oli sitä mieltä että se olisi saanut olla parempi, 40,0% sanoi sen toimivan jotenkuten, 38,6% oli sitä mieltä että se toimi hyvin ja 4,3% piti sitä erinomaisena.

Asetettujen kriteerien pohjalta järjestelmän yleiset kriteerit täyttyvät erinomaisesti. Saadun palautteen pohjalta järjestelmää voidaan pitää toimivana, helppokäyttöisenä ja loogisena. Ohjeita voidaan pitää hyvinä. Opetuksellisista kriteereistä vuorovaikutteisuus toimii tyydyttävästi. Vaikka ohjelmaa voidaankin pitää visuaalisena siltä osin kun se on valmis, ei vuorovaikutteisuus toimi parhaalla mahdollisella tavalla. Tähän voidaan löytää useitakin eri syitä. Ensinnäkään kaikkia TRAKLA-järjestelmän tehtäviä ei ole mahdollista esittää graafisesti johtuen niiden luonteesta. Toiseksi järjestelmä oli testikäytössä eikä kaikkia tehtäviä oltu vielä toteutettu loppuun saakka. Lisäksi arvioitaessa koko WWW-TRAKLAa on huomattava, että kaikkia sen potentiaalisia mahdollisuuksia oppimisympäristönä ei ole vielä täysin käytetty hyväksi. Saadusta palautteesta voidaan lisäksi todeta, että tämän tyyppinen oppimisympäristö ei edes ole kaikille opiskelijoille mielekäs vaihtoehto. Tai ainakaan kaikki eivät koe, että järjestelmällä olisi heille kovinkaan paljon annettavaa.

Järjestelmän ominaisuuskriteereitä oli asetettu kaksi: saatavuus ja päivitettävyys. Vertailtaessa järjestelmää luvun 3.3.3 esimerkkijärjestelmiin voidaan todeta, että saatavuus on WWW-TRAKLAn tyyppisissä ratkaisuissa huomattavasti parempi. Tätä puoltaa myös saatu palaute. Kaikilla opiskelijoilla oli mahdollisuus päästä käyttämään järjestelmää verkon kautta. Huomattava määrä opiskelijoista pystyi käyttämään järjestelmää myös kotoaan käsin. Tämä osaltaan helpottaa koulun rajal-

lisista resursseista johtuvaa jonottamista graafisille päätteille. Tilanne ei kuitenkaan ole aivan ongelmaton: siirtyminen kokonaan WWW-TRAKLAn käyttöön ilman, että sähköpostipohjainen palauttaminen olisi mahdollista, vaatisi huomattavasti enemmän resursseja myös koulun puolelta. Tällä hetkellä laitteistokapasiteetti vaikuttaa riittämättömältä, jos massakursseilla tehtävien ratkaisemiseen olisi kaikkien opiskelijoiden osalta vaatimuksena pääsy graafiselle päätteelle. Päivitettävyydessä järjestelmä on myös huomattavasti joustavampi kuin vertailujärjestelmät. Uusi versio käyttöliittymästä voidaan jakaa kaikille laitearkkitehtuureille yhdellä käännöksellä ja sen toimittamisella verkkoon. Uusi versio on välittömästi kaikkien käytössä, koska käyttöliittymä ladataan joka käyttökerran yhteydessä verkosta.

Kokonaisuutena järjestelmää voidaan pitää varsin onnistuneena. On kuitenkin syytä korostaa sitä, että järjestelmän täysimittainen hyväksikäyttö vaatii vielä paljon jatkokehitystyötä. Lisäksi jatkossa on syytä pohtia myös sitä, kuinka järjestelmän vaatimien resurssien, kuten graafisten työasemien ja päätteiden, saatavuus voidaan järjestää.

9. Yhteenveto

Tässä diplomityössä suunniteltiin ja toteutettiin WWW-pohjainen oppimisympäristö ja tätä varten graafinen käyttöliittymä tietokoneavusteiselle opetusohjelmalle TRAKLA. Toteutettu oppimisympäristö voidaan nähdä kolmesta eri komponentista (TAO, verkko ja Internet-ohjelmointi) yhdistelemällä aikaan saatuna kokonaisuutena. Näistä työn varsinainen toteutusosa sisälsi graafisen käyttöliittymän toteutuksen Java-ohjelmointikielellä (Internet-ohjelmointi).

Työlle asetetuista tavoitteista laiteriippumattomuus saavutettiin periaatteellisella tasolla. Useille eri laitearkkitehtuureille löytyi ympäristöjä, joissa järjestelmän täysipainoinen käyttö oli mahdollista. Selainohjelmien vikojen ja puutteiden vuoksi ei käyttöliittymän käyttö kuitenkaan ollut mahdollista aivan kaikissa ympäristöissä.

Työn kirjallisuustutkimuksen osalta tavoitteet saavutettiin osittain. Täysin vastaavantyyppisiä sovelluksia ei löytynyt. Tämä oli toisaalta oletettavaakin, koska kyseessä on aivan uuden tyyppinen lähestymistapa tietokoneavusteisten oppimisympäristöjen toteutukseen ja Java-ohjelmointikieli on vasta viime vuosina noussut esiin eräänä merkittävänä Internet-ohjelmointiin tarkoitettuna ohjelmointikielenä. Kirjallisuudesta voidaan kuitenkin löytää lukuisia viitteitä vastaavanlaisiin sovelluksiin, jos toteutettua kokonaisuutta tarkastellaan komponenteittain.

Kolmas keskeinen tavoite työlle oli tietoturvallisuuskysymysten ratkaiseminen. Tämän tyyppisissä ohjelmistoprojekteissa tietoturvakysymykset tulee ratkaista useiden eri kohderyhmien osalta. Osoittautui, että valittu periaateratkaisu on tietoturvallisuuden kannalta hyvä. Toteutusvaiheessa tietoturvaan tulee kiinnittää huomiota, mutta riittävällä huolellisuudella toteutetusta järjestelmästä saadaan toimiva, luotettava ja turvallinen.

Työn onnistumista pyrittiin mittaamaan myös asettamalla toteutetulle käyttöliittymälle joukko kriteereitä, joiden täyttymistä tarkasteltiin saatujen kokemusten perusteella. Järjestelmä oli koekäytössä keväällä 1997 ja saadun palautteen perusteella työtä voidaan pitää varsin onnistuneena. Koekäyttöön osallistui yli 200 opiskelijaa, joiden palaute järjestelmästä oli myönteistä.

Kokonaisuutena nähtynä työtä voidaan pitää onnistuneena. Tulevaisuudessa ongelmat myös yksityiskohtien osalta tulevat varmastikin poistumaan, koska kehitys ja panostus tietotekniikan alueella kohdistuu tällä hetkellä juuri kaikkeen verkossa tapahtuvaan. Tätä silmällä pitäen saatuja tuloksia voidaan pitää erittäin rohkaisevina.

9.1 Työn jatkokehitysnäkymiä

Työtä voidaan pitää siinäkin mielessä onnistuneena, että vaikka peruslähtökohtana olikin graafisen käyttöliittymän toteuttaminen TRAKLA-järjestelmälle, ei toteutetun käyttöliittymän mahdollisuudet rajoitu pelkästään tähän. Toteutettu järjestelmä mahdollistaa myös muunlaisen käytön, kuten esimerkiksi käytön opettajan työvälineenä tietorakenteiden ja algoritmien havainnollistamiseen.

Vaikka jatkossa työn ensisijainen kehityssuunta tulee varmaankin olemaan sen toiminnallisuuden lisääminen kattamaan kaikki TRAKLA-järjestelmän tehtävät, voidaa jo nyt nähdä useita muitakin jatkokehityssuuntia. Kehitystyö edellä mainitun havainnollistamisvälineen suuntaan on yksi niistä. Järjestelmä voitaisiin nähdä myös eräänä hypermedian keinona toteuttaa tietorakenteita ja algoritmeja koskevia verkkojulkaisuja. Tällöin käyttöliittymä voisi olla osa julkaisua, jossa se ottaa perinteisen julkaisun kuvan tai kuvasarjan paikan. Pelkän esimerkkikuvan sijasta lukijalle tarjottaisiin vuorovaikutteinen animaatio! Tällöin voitaisiin jo puhua todellisesta hypermedian keinoin toteutetusta hypermediajulkaisusta, jollaiseen perinteisen julkaisutekniikan keinoin ei ole mahdollista päästä.

10. Kirjallisuusviitteet

[1] anon., “Opetusohjelma 1995-1996”. Teknillinen korkeakoulu, opintotoimisto, Espoo, 1996.

[2] Arnold K., Gosling J., “The Java Programming Language”. Addison-Wesley, Reading, Massachusetts, 1996. 333 s. ISBN 0-201-63455-4.

[3] Bajpai J. S., Salter R. M., “HtX reference manual.”. Technical report, Oberlin College Computer Science Program, November 1995.

[4] Benford S., Burke E., Foxley E., “A System to Teach Programming in a Quality Controlled Environment”. The Software Quality Journal, Vol. 2, 177-197, 1993.

[5] Berners-Lee, T., Masinter L., McCahill, M., “Uniform Resource Locators (URL)”. RFC 1738, December 1994.

[6] Blum H., “Internet Connection for Web Access: An Example for Performance Modeling”. SIGCSE Bulletin, Vol. 28(2), 62-64, June 1996.

[7] Boroni C. M., Eneboe T. J., Goosey F. W., Ross J.A., Ross R. J., “Dancing with Dynalab”. SIGCSE Bulletin, Vol. 28(1), 135-139, March1996.

[8] Budd T.A., Rajeev K.P., “Never Mind the Paradigm, What About Multiparadigm Languages?”. SIGCSE Bulletin, Vol. 27(2), 25-30, June 1995.

[9] Carlson D., Guzdial M., Colleen K., Viren S., Statsko J., “WWW Interactive Learning Environments For Computer Science Education”. SIGCSE Bulletin, Vol. 28(1), 290-294, March 1996.

[10] Connelly C., Biermann A. W., Pennock D., Wu P., “Home-Study Software: Flexible, Interactive, And Distributed Software For Independent Study”. SIGCSE Bulletin, Vol. 28(1), 63-67, March 1996.

[11] Dawson-Howe K.M., “Automatic Submission and Administration of Programming Assignments”. SIGCSE Bulletin, Vol. 28(2),40-52, June 1996.

[12] Goldberg M., “CALOS: An Experiment With Computer-Aided Learning For Operating Systems”. SIGCSE Bulletin, Vol. 28(1), 175-179, March 1996.

[13] Guimaràes M. A., de Lucena C. J., Cavalcanti M. R., “Experience Using The ASA Algorithm Teaching System”. SIGCSE Bulletin, Vol. 26(4), 45-49, December 1994.

[14] Güvenir H.A., “An Object-Oriented Tutoring System for Teaching Sets”. SIGCSE Bulletin, Vol. 27(3), 39-46, September 1995.

[15] Haikala I., Märijärvi J., “Ohjelmistotuotanto”. Suomen Atk-kustannus Oy, Espoo, 1997. 357 s. ISBN 951-762-497-2.

[16] Hietaniemi J., Hyvönen J., Juvonen E., Nopanen J., “TRAKLA-järjestelmäkäsikirja (TRAKLA System Manual)”. Helsinki University of Technology, Laboratory of Information Processing Science, Espoo, 1991.

[17] Hyvönen J., Malmi L., “TRAKLA A System for Teaching Algorithms Using Email and a Graphical Editor”. Teknillinen korkeakoulu, Tietojenkäsittelytekniikan laitoksen julkaisu TKO-B100, Espoo, 1993.

[18] Jones R. P., Ruehr F., Salter R., “Web-Based Laboratories In The Introductory Curriculum Enhance Formal Methods”. SIGCSE Bulletin, Vol. 28(1), 160-164, March 1996.

[19] Lee P. M., Sullivan W. G., “Developing and Implementing Interactive Multimedia in Ecucation”. IEEE Transactions on Education, Vol 39(3), 430-435, August 1996.

[20] Levy S.P., “Computer Language Usage in CS1: Survey Results”. SIGCSE Bulletin, Vol. 27(3), 21-26, September 1995.

[21] Lifländer V-P., “Tietokoneavusteisen opetuksen kehittäminen”. Lisensiaattityö, Helsingin teknillinen korkeakoulu, tietotekniikan osasto, Espoo, 1989.

[22] Marcy W. M., “Implementation Issues in SIMPLE Learning Environment”. IEEE Transactions on Education, Vol 39(3), 423-429, August 1996.

[23] Mengel S. A., “The Need for Hypertext Instructional Design Methodology”. IEEE Transactions on Education, Vol 39(3), 357-380, August 1996.

[24] Oakley, II B., “A Virtual Classroom Approach to Teaching Circuit Analysis”. IEEE Transactions on Education, Vol 39(3), 287-296, August 1996.

[25] Reek K. A., “A Software Infrastructure To Support Introductory Computer Science Courses”. SIGCSE Bulletin, Vol. 28(1), 125-129, March 1996.

[26] Reek K. A., “The TRY System, or How to Avoid Testing Student Programs”. SIGCSE Bulletin, Vol. 21(1), 112-116, February 1989.

[27] Shaffer C. A., Lenwood S. H., “Using The Swan Data Structure Visualization System For Computer Science Education”. SIGCSE Bulletin, Vol. 28(1), 140-144, March 1996.

[28] Simeonov S., Schneider G. M., “MSIM: An Improved Microcode Simulator”. SIGCSE Bulletin, Vol. 27(2), 13-17, June 1995.

[29] Stasko J.T., Hayes D., “XTANGO Algorithm Animation Designer’s Package”. College of Computing, Georgia Institute of Technology, Atlanta, October 1992.

[30] Stasko J.T., Kramer E., “A Methodology for Building Application-Specific Visualizations of Parallel Programs". Graphics, Visualization, and Usability Center,

Georgia Institute of Technology, Number GIT-GVU-92-10, Atlanta, GA, June 1992.

[31] Sun C-T., “Experiencing CORAL: Design and Implementation of Distant Cooperative Learning”. IEEE Transactions on Education, Vol 39(3), 357-366, August 1996.

[32] Zanconi M., Moroni N., Señas P., “An Educational Project in Computer Science for Primary and High School”. SIGCSE Bulletin, Vol. 27(3), 27-33, September 1995.

10.1 WWW-viitteet

[33] anon., “Microsoft Business Centre“. “**http://www.microsoft.com/uk/business/magazine/activex.htm**”.

[34] Buzbee J., “I think we have a security hole here”. “**http://www.nyx.net/~jbuzbee/filehole.html**”

[35] Buzbee J., “The Hostile Mail Applet Page”. “**http://www.nyx.net/~jbuzbee/filehole.html**”

[36] Chappell D., “Component Software Meets the Web: Java Applets vs. ActiveX Controls”. **“http://www.chappellassoc.com/JavaActX.htm”**.

[37] anon., “CMC Information Sources”. “**http://www.december.com/cmc/info/**”.

[38] anon., “December Communications Home Page“. “**http://www.december.com/**”.

[39] anon., “CSC JAVA Sorting Animation Page“. “**http://www.cs.brockport.edu/cs/javasort.html**“.

[40] anon., “EarthWeb Home Page”. “**http://www.earthweb.com/**”.

[41] anon., “The Official Directory For Java“. “**http://www.gamelan.com/index.shtml**”.

[42] Ottmann T., “Authoring on the fly”. “**ftp://ftp.informatik.uni-freiburg.de/pub/AOF**”.

[43] anon., “Perl CGI Programming FAQ”. “**ftp://ftp.funet.fi/pub/languages/perl/CPAN/doc/FAQs/cgi/perl-CGI-faq.html**”.

[44] anon. “Safe-cgi.txt”. “**http://www.go2net.com/people/paulp/CGI-security/safe-cgi.txt**”.

[45] Ierardi D., Ta-Wei L., “Interactive animations of several types of binary search trees”. “**http://langevin.usc.edu/BST/**”.

[46] anon., “Tietorakenteet ja algoritmit -kurssin kotisivu”. “**http://www.niksula.cs.hut.fi/~tik76122**”.

[47] anon., “The Common Gateway Interface”. “**http://hoohoo.ncsa.uiuc.edu/cgi/**”.

[48] anon., “The WWW Security FAQ”. “**http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html**”.

[49] anon., “JavaSoft Home Page”. “**http://www.javasoft.com/**”.

[50] anon., “What is Java?”. “**http://www.javasoft.com/nav/whatis/**”.

[51] anon., “The MathMol Hypermedia Textbook“. “**http://www.nyu.edu/pages/mathmol/textbook/**”.

[52] Strauss J., Foss J., Kinzie M., “The Interactive Frog Dissection”. “**http://teach.virginia.edu/go/frog**”.

LIITE A: CGI-scripti tehtäväpohjien dynaamiseen räätälöintiin.

```
#!/p/bin/perl -T

# ------------------------------------------------------------
  ----
# tailor.pl, by Ari archie Korhonen (ar-
  chie@hutcs.cs.hut.fi).
#
# Last updated: 24-01-1997
#
# ------------------------------------------------------------
  ----

# Print out a content-type for HTTP/1.0 compatibility
print “Content-type: text/html\n\n”;

# Get the input
read(STDIN, $buffer, $ENV{‘CONTENT_LENGTH’});

# Split the name-value pairs
@pairs = split(/&/, $buffer);

foreach $pair (@pairs)
{
    ($name, $value) = split(/=/, $pair);

    # Un-Webify plus signs and %-encoding
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack(“C”,
  hex($1))/eg;

    # Uncomment for debugging purposes
    # print “Setting $name to $value<P>”;

    $FORM{$name} = $value;
}

# Construct variables
$path = ‘/u/projects/trakla/bin/’;
$tailor = ‘/u/projects/trakla/bin/tailor-html’;
$cid = $FORM{‘cid’};
$uid = $FORM{‘uid’};
$round = $FORM{‘round’};
$name =  $FORM{‘name’};
$email =  $FORM{‘email’};
$exercise = $FORM{‘exercise’};
$lang = $FORM{‘language’};

chdir $path;
open(TAILOR,”-|”) || exec $tailor,$cid,$uid,$na-
  me,$email,$round,$exercise,$lang;

   $[ = 1;                # set array base to 1
```

```
   $, = ‘ ‘;               # set output field separator
   $\ = “\n”;              # set output record separator

   while (<TAILOR>) {
      chop;       # strip record separator
      @Fld = split(‘ ‘, $_, 9999);
      if ($Fld[1] eq ‘</APPLET>’) {
         print “<param name = courseid value = $cid>”;
         print “<param name = userid value = $uid>”;
   print “<param name = realname value = $name>”;
   print “<param name = email value = $email>”;
        print “<param name = exercise value = $round.$exer-
  cise>”;
      }
      print $_;
   }

close TAILOR;
```

LIITE B: CGI-scripti kommunikaatioon WWW:ltä TRAKLAlle.

```
#!/p/bin/perl -- -*-perl-*-

# ------------------------------------------------------------
  ----
# Form-mail.pl, by Ari Korhonen (Ari.korhonen@hut.fi)
#
# Last updated: January 16, 1997
#
# Form-mail provides a mechanism by which users of a World-
# Wide Web browser may submit comments to the TRAKLA.
# It should be compatible with any CGI-compatible HTTP ser-
  ver.
#
# ------------------------------------------------------------
  ----

# Define fairly-constants

# This should match the mail program of the system.
$mailprog = '/usr/lib/sendmail';

# This should be set to the username or alias of TRAKLA.
$recipient = 'trakla@niksula.hut.fi';

# Print out a content-type for HTTP/1.0 compatibility
print "Content-type: text/html\n\n";

# Print a title and initial heading
print "<Head><Title>Kiitos</Title></Head>";
print "<Body><H1>Kiitos</H1>";

# Get the input
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# Split the name-value pairs
@pairs = split(/&/, $buffer);

foreach $pair (@pairs)
{
    ($name, $value) = split(/=/, $pair);

    # Un-Webify plus signs and %-encoding
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",
  hex($1))/eg;

    # Stop people from using subshells to execute commands
   # Not a big deal when using sendmail, but very important
    # when using UCB mail (aka mailx).
    # $value =~ s/~!/ ~!/g;

    # Uncomment for debugging purposes
```

```
    # print “Setting $name to $value<P>”;

    $FORM{$name} = $value;
}

# If the comments are blank, then give a “blank form” res-
  ponse
&blank_response unless $FORM{‘comments’};

# Now send mail to $recipient and add external parameters.

open (MAIL, “|$mailprog $recipient”) || die “Can’t open
  $mailprog!\n”;
print MAIL “From: $FORM{‘REPLY’}\n”;
print MAIL “Reply-to: $FORM{‘REPLY’} ($FORM{‘REALNA-
  ME’})\n”;
print MAIL “Subject: WWW comments (Forms submission)\n\n”;
print MAIL  “-----------------------------------------
  ---------------\n”;
print MAIL “$FORM{‘comments’}\n”;
print MAIL “#CID $FORM{‘CID’}\n”;
print MAIL “#ID $FORM{‘ID’}\n”;
print MAIL “#NAME $FORM{‘REALNAME’}\n” if defined
  $FORM{‘REALNAME’};
print MAIL “#REPLY $FORM{‘REPLY’}\n” if defined $FORM{‘REP-
  LY’};
print MAIL “$FORM{‘answer’}\n”;
print MAIL “$FORM{‘REGISTRATION’}\n”;
print MAIL “#END\n”;
print MAIL “\n-----------------------------------------
  ----------------\n”;
print MAIL “Server protocol: $ENV{‘SERVER_PROTOCOL’}\n”;
print MAIL “Remote host: $ENV{‘REMOTE_HOST’}\n”;
print MAIL “Remote IP address: $ENV{‘REMOTE_ADDR’}\n”;
close (MAIL);

# Make the person feel good for writing to us
print “Viestisi on lähetetty <I>TRAKLA-ohjelmalle</
  I>!<P>”;
# print “Return to <A HREF=\”/\”>home page</A>, if you
  want.<P>”;

# -------------------------------------------------------
  ----
# subroutine blank_response
sub blank_response
{
    print “Viestisi oli tyhjä, joten sitä ei lähetetty. “;
   print “Täytä kaikki lomakkeen kohdat ja yritä uudelleen.
  “;
#    print “return to our <A HREF=\”/\”>home page</A>, if
  you want.<P>”;
    exit;
}
```

LIITE C: Otsikko- ja tehtäväpohja sekävastaava HTML-tiedosto.

Otsikkopohja

```
;$AIHE(0)
<HTML>
<HEAD><TITLE>TRAK kotitehtäväkierros 2</TITLE></HEAD>
<BODY>
<H4>
<a href = “mailto:Ari.Korhonen@hut.fi”>
<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/dilbert.gif”></a>
<a href = “http://www.niksula.cs.hut.fi/~tik76122/TraklaEdit.html”>
<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/toimage.gif”></a>
<a href =“http://www.niksula.cs.hut.fi/~tik76122/TraklaEditOhje.html”>
<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/toquestion.gif”></a>
$TKK()
<BR>
$TKO()
<BR>
$TRAK()
<BR>
Kurssi: $CID()<BR>
Nimi: $NAME() ($ID())<BR>
Kierros: 2/$YEAR()
</H4>
<i>Viikon vinkki: älä (vieläkään) paina export-nappia ennenkuin olet vastannut
koko tehtävään!</i>
</BODY>
</HTML>
```

Tehtäväpohja

```
;$AIHE(32)
;---------------------------------------------------------------------
<HTML>
<base href=”http://www.niksula.cs.hut.fi/~tik76122/”>
<BODY>
<H1>
Tehtävä $NRO() ($PISTEET() pistettä)
</H1>
<P>
Lisää (<a href = “dsaa_c2e/tree.c”>algoritmilla Insert</a>) tyhjään <b>binääriseen hakupuuhun</b> järjestyksessä annetut <b>alkiot</b>. Poista tämän jälkeen puun juurialkio vetämällä se tyhjään <b>paikkaan</b> ja palauta puu binääripuuksi kirjan kohdassa 4.3.5 esitetyllä <a href = “dsaa_c2e/tree.c”>algoritmilla Delete</a>. Jos mukana on yhtäsuuria alkioita, ne lisätään aina solmun oikeanpuoleiseen haaraan.
<P>
Tähän tehtävään voit lähettää vastauksia enintään $YRITYS() kertaa.
<P>
<b>HUOM.</b> Mikäli puurakenne ei mahdu kunnolla ruudulle, pienennä
kirjasinkokoa (Size) tai levitä selainohjelmasi ikkunaa ja lataa tehtävä uudelleen <b>reload</b>-näppäimellä.
<HR>
<APPLET CODE=”tred.TraklaEditor.class” WIDTH=100% HEIGHT=470>
<param name=posturl value=”http://130.233.40.178/CGI-bin/trakla/form-mail”>
<param name=objects value=”nodepanel alkiot;$alkiot()+10,nodepanel paikka;%S(1), tree puu;%S(63)”>
<param name=output value=”puu\paikka;-+a),puu\end;-+b)”>
</APPLET>
<HR>
</BODY>
</HTML>
```

Räätälöity HTML-tiedosto

<HTML>
<HEAD><TITLE>TRAK kotitehtäväkierros 2</TITLE></HEAD>
<BODY>
<H4>

<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/dilbert.gif”>

<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/toimage.gif”>

<IMG ALIGN=RIGHT WIDTH=39 HEIGHT=41
SRC=”http://www.hut.fi/Misc/Images/button/toquestion.gif”>
Teknillinen korkeakoulu

Tietojenkäsittelyopin laboratorio

Tik-76.122 Tietorakenteet ja algoritmit

Kurssi: **k97**
Nimi: **Ari Korhonen (36784a)**
Kierros: 2/**1997**
</H4>
Viikon vinkki: älä (vieläkään) paina export-nappia ennenkuin olet vastannut
koko tehtävään!
</BODY>
</HTML>

<HTML>
<base href=”http://www.niksula.cs.hut.fi/~tik76122/”>
<BODY>

<H1>
Tehtävä **2** (**2** pistettä)
</H1>
<P>
Lisää (algoritmilla Insert) tyhjään binääriseen hakupuuhun järjestyksessä annetut alkiot. Poista tämän jälkeen puun juurialkio vetämällä se tyhjään paikkaan ja palauta puu binääripuuksi kirjan kohdassa 4.3.5 esitetyllä algoritmilla Delete. Jos mukana on yhtäsuuria alkioita, ne lisätään aina solmun oikeanpuoleiseen haaraan.
<P>
Tähän tehtävään voit lähettää vastauksia enintään 3 kertaa.
<P>
HUOM. Mikäli puurakenne ei mahdu kunnolla ruudulle, pienennä
kirjasinkokoa (Size) tai levitä selainohjelmasi ikkunaa ja lataa tehtävä uudelleen reload-näppäimellä.
<HR>
<APPLET CODE=”tred.TraklaEditor.class” WIDTH=100% HEIGHT=470>
<param name=posturl value=”http://130.233.40.178/CGI-bin/trakla/form-mail”>
<param name=objects value=”nodepanel alkiot;**p l n a d g v s y z x c u z**+10,nodepanel paikka;%S(1),tree puu;%S(63)”>
<param name=output value=”puu\paikka;-+a),puu\end;-+b)”>
<param name = courseid value = k97>
<param name = userid value = 36784a>
<param name = realname value = WWW-form>
<param name = email value = user@unknown>
<param name = exercise value = 2.2>
</APPLET>
<HR>
</BODY>
</HTML>

LIITE D: Palautekyselyn tulos kevään 1997 kurssilta Tietorakenteet ja algoritmit.

Kyselyyn vastasi : 476

Traklaeditia ei käyttänyt 364 (62.5 %) vastanneista

21) Eivät käyttäneet TraklaEditia koska:

Ei aikaa tutustua uuteen järj. :	145	(39.8 %)
Ei jaksanut jonottaa graaf. päät. :	61	(16.8 %)
Järjestelmä oli testikäytössä :	55	(15.1 %)
Pitää Javaa turvallisuusriskinä :	5	(1.4 %)
S-postilla palauttaminen helpompaa :	209	(57.4 %)
Ei ollut tarvittavaa laitt./ohjelmaa :	39	(10.7 %)
Ei saanut toimimaan :	0	(0.0 %)
Toimi liian epäluotettavasti :	14	(3.8 %)
Käytöstä olisi ollut ylim. vaivaa :	96	(26.4 %)
Liian monimutkainen :	11	(3.0 %)
Liian yksinkertainen :	0	(0.0 %)
Toimi liian hitaasti :	41	(11.3 %)
muu :	67	(18.4 %)

22) Paljonko vaivaa käytti toimintakuntoon yrittämiseen:

ei yrittänyt :	178	(59.9 %)
olisi toiminut :	50	(16.8 %)
yritti hieman :	64	(21.5 %)
yritti paljon :	5	(1.7 %)

TraklaEditia käytti 218 (37.5 %) vastanneista

23) Käyttivät TraklaEditia koska:

Halusi tutustua uuteen jarj. :	98	(45.0 %)
Järjestelmä oli testikäytossä :	27	(12.4 %)
On kiinnostunut Javasta :	77	(35.3 %)
Palauttaminen helpompaa :	124	(56.9 %)
Ratkominen nopeampaa :	96	(44.0 %)
Järjestelmä helppokäyttöinen :	117	(53.7 %)
Ei aikaa tutustua sposti-palaut. :	26	(11.9 %)
Vastsien lähetys toimi luetettvammin :	24	(11.0 %)
muu :	14	(6.4 %)

Missa ympäristössä köytti:

LK-HP:ssa :	40	(19.3 %)
Alphoissa :	11	(5.3 %)
Niksulassa :	31	(15.0 %)
X-päättellä :	9	(4.3 %)
Linux :	15	(7.2 %)
Windows :	95	(45.9 %)
Mac :	1	(0.5 %)
muu :	5	(2.4 %)

25) Mitä selainta käytti:

Netscape Navigator	186	(85.3 %)
Netscape Communicator	13	(6.0 %)
MS IE	18	(8.3 %)
muu	1	(0.5 %)

26) Ilmenikö TraklaEditin käytössa ongelmia:

ei lainkaan	37	(22.6 %)
hieman ongelmia	111	(67.7 %)
melko paljon	13	(7.9 %)
erittäin paljon	3	(1.8 %)

27) Oliko TraklaEdit riittävän helppokäyttöinen:

Erittäin helppokäyttöinen	71	(42.0 %)
kohtuullisen helppokäyttöinen	89	(52.7 %)
melko vaikeaselkoinen	8	(4.7 %)
erittain vaikeaselkoinen	1	(0.6 %)

28) Tietorakenteiden graafiset toteutukset TraklaEditilla olivat:

erittäin loogisia	33	(21.3 %)
melko loogisia	112	(72.3 %)
melko epäloogisia	8	(5.2 %)
erittäin epäloogisia	2	(1.3 %)

29) Mitä mieltä TraklaEditin on-line ohjeista:

ei tarvinnut	77	(53.1 %)
ei löytänyt	12	(8.3 %)
riittävät	44	(30.3 %)
riittämättömät	12	(8.3 %)

30) Miten TraklaEditin vuorovaikutteisuus toimi:

Olisi saanut olla parempi	24	(17.1 %)
jotenkuten	56	(40.0 %)
hyvin	54	(38.6 %)
erinomaisesti	6	(4.3 %)

31) Paljonko vaivaa näki TraklaEditin käyttökuntoon saattamisessa:

Toimi välittömästi	157	(82.2 %)
Sai toimimaan ohjeilla	27	(14.1 %)
Sai toimimaan kaverin avulla	2	(1.0 %)
Oli suuria ongelmia	5	(2.6 %)

32) Arvosana TraklaEditille:

1	25	(11.5 %)
2	37	(17.0 %)
3	46	(21.1 %)
4	79	(36.2 %)
5	31	(14.2 %)
keskiarvo	3.2	