Vaatimusmäärittelydokumentti

Koski-ryhmä

Helsinki 18.5.2004

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Olli Alm
Seppo Hätönen
Sini Ruohomaa
Antti Takalahti
Sampo Yrjänäinen
Arto Åkerlund

Asiakas

Teemu Kerola

Johtoryhmä

Raine Kauppinen (ohjaaja)
Juha Taina (vastuuhenkilö)
Turjo Tuohiniemi (vastuuhenkilö)

Kotisivu

`http://www.cs.helsinki.fi/group/koski`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	9.2.2004	Formal Technical Reviewissä tarkasteltava verio
1.1	16.2.2004	FTR:n pohjalta korjattu versio.

Sisältö

2 Trainer2-rajapintaluokat

3 Levy-I/O

1 Johdanto

Koski-projekti on osa Helsingin yliopiston Tietojenkäsittelytieteen laitoksen kurssia 581260 Ohjelmistotuotantoprojekti. Projekti toteutetaan keväällä 2004 projektisuunnitelman mukaisesti. (Projektisuunnitelman saa pyytämällä sitä ryhmän sähköpostiosoitteesta `ohtuk04-koski-list@cs.helsinki.fi`.) Tämä dokumentti on tarkoitettu projektiryhmän ja asiakkaiden väliseksi sopimukseksi ja kattaa kokonaisuudessaan toteutettavan ohjelmiston. Erityisesti projektiryhmä ei sitoudu toteuttamaan sellaisia vaatimuksia, joita ei ole kirjattu jäädytettyyn ja kaikkien osapuolten hyväksymään vaatimusmäärittelydokumenttiin.

Koski-projektin tavoitteena on kehittää konekielisimulaattori Helsingin yliopiston Tietojenkäsittelytieteen laitoksen kurssin 581305 Tietokoneen toiminta tarpeita varten. Projektiryhmään kuuluvat Olli Alm, Seppo Hätönen, Sini Ruohomaa, Antti Takalahti, Sampo Yrjänäinen ja Arto Åkerlund. Ohjaajana toimii Raine Kauppinen, vastuuhenkilöt ovat Juha Taina ja Turjo Tuohiniemi. Projektin asiakas on Tietojenkäsittelytieteen laitoksen puolesta Teemu Kerola. Projektin tuloksena syntyvä ohjelmisto julkaistaan GNU Lesser General Public lisenssin alla. (Lisää tietoa lisenssistä osoitteessa `http://www.gnu.org/copyleft/lesser.html`.)

Dokumentin alussa on kuvattu asiakkaan projektille asettamat vaatimukset. Vaatimuksia on analysoitu, jonka jälkeen on kuvattu toteutettava ohjelmisto pääpiirteineen.

1.1 Termit

tito tarkoittaa tässä dokumentissa Helsingin yliopiston Tietojenkäsittelytieteen laitoksen kurssia 581305 - Tietokoneen toiminta.

trainer2 tarkoittaa Harri Laineen kuvailemaa, myöhemmin toteutettavaa, ohjelmaa, joka mahdollistaa laskuharjoitusten automaattisen tarkistamisen. Ellei erikseen mainita, niin ohjelmistolla tarkoitetaan ohjelmistoa ilman trainer2-käyttöliittymää.

API-termi tarkoittaa tässä dokumentissa trainer2-ohjelmaa varten toteutettavaa rajapintaa.

.k91-termi tarkoittaa symbolisessa konekielimuodossa olevaa TTK-91-ohjelmaa.

.b91-termillä tarkoitetaan käännettyä, binäärimuotoista TTK-91-ohjelmaa.

2 Asiakkaan vaatimukset

Tämä luku kuvaa asiakkaan ohjelmistolle asettamat vaatimukset niiltä osin kun asiakas on ne spesifioinut. Osa vaatimuksista on kuvattu yksityiskohtaisemmin, mutta tämän kappaleen tarkoituksena ei ole kuvata ohjelmiston tarkkaa toimintaa, vaan esitellä asiakkaan vaatimukset. Tässä luvussa esiteltyjä vaatimuksia on analysoitu luvussa 4.

2.1 Yleiskuva

Projektin tavoitteena on tehdä TTK-91-koneen simulaattori, joka korvaa vanhan Koksi-simulaattorin tito-kurssin apuvälineenä.

Simulaattorin tavoitteena on perehdyttää opiskelijat konekieleen siten, että he pääsevät kirjoittamaan konekieltä ja näkevät, mitä kone tekee kunkin käskyn seurauksena. Tämä tarkoittaa sitä, että jokaisen konekäskyn jälkeen ohjelma näyttää koneen tilan rekistereiden ja muistin osalta käyttäjän niin halutessa.

Lisäksi ohjelman tulee mahdollistaa myöhemmin toteutettavan laskuharjoitustehtävien automaattisen tarkistamisen mahdollistavan ohjelman toteutuksen tarjoamalla rajapinnan, jonka avulla ohjelmia voi kääntää ja suorittaa. Rajapintaa käyttävästä ohjelmasta käytetään tämän jälkeen trainer2-nimeä.

Vaatimusten numerointi noudattaa asiakkaan antamaa listaa, joka löytyy osoitteesta `http://www.cs.helsinki.fi/u/kerola/ohtu/ttk91_simu.html`.

Vaatimukset on priorisoitu siten, että perustavoitteita ovat kaikki, lukuunottamatta kohtia V18 ja V19. Lisätavoitteina on käyttöliittymän toteuttaminen suomeksi ja dokumentoinnin ja käyttöohjeen suomentaminen. Lisätavoitteet on priorisoitu siten, että ensin toteutetaan suomennos, toisena graafinen suoritussyklin animointi ja kolmantena laiteajurin toiminnan animointi.

Perustavoitteet toteutetaan siten, että ensin tehdään perustoiminnallisuus, eli ohjelmien kääntäminen ja emulointi. Trainer2-liittymä toteutetaan vasta kun perustoiminnallisuus on tehty, minkä jälkeen käsitellään syvällisemmin vikasietoisuutta. Lisätavoitteita toteutetaan mahdollisuuksien mukaan.

2.2 Toiminnalliset tavoitteet

Ohjelmiston päätavoite on, että sen tulee soveltua hyvin tito-kurssin tarpeisiin. Käyttäjinä ohjelmistolla tulee olemaan opiskelijoita ensimmäisestä vuosikurssista alkaen. (**V1**)

Ohjelmiston tulee suorittaa kaikki vanhat, jo olemassa olevat TTK-91-ohjelmat. (**V2**)

Simulaattorin operaatiokoodien tulee noudattaa vanhaa semantiikkaa niin, että se on sama kuin vanhassa Pascal-ohjelmassa, sekä Auvo Häkkisen opetusmonisteessa kuvailtu (Auvo Häkkinen, Tietokoneen toiminta, opetusmoniste D390). Kuitenkin niin, että tärkeintä on noudattaa Pascal-ohjelman toiminnallisuutta mikäli nämä kaksi ovat ristiriidassa. (**V3**)

Ohjelmiston tulee toimia laitoksen Windows-, ja Linux-ympäristöissä ja tukea erityisesti Windowsin malleja XP ja 2000. Ohjelmiston tulee myös olla yhteensopiva vanhojen, jo olemassaolevien Koksi-ohjelmien kanssa sekä tukea Windowsin ja Linuxin erilaisia rivinvaihtoja käyttäviä ohjelmia. (**V4**)

Ohjelmiston tulee koostua useammasta palikasta niin, että ohjelman valitsin, kääntäjä, lataaja ja suoritin on toteutettu omana palasenaan. Kääntäjän tulee lukea .k91-muotoisia merkkijonoja ja palauttaa .b91-muotoisia merkkijonoja. Simulaattorin käyttöliittymä on toteutettava englanniksi, mutta niin, että uusien kielten lisääminen on helppoa. Myös toinen toteutettava käyttöliittymä toteutetaan englanniksi. (**V5**)

Simulaattorissa tulee olla graafinen suorituksenaikainen suorituksen emulointi. Ohjelman tulee siis näyttää jokaisen käskyn jälkeen rekistereiden ja muistin sisältö. Tämä ei kuitenkaan koske trainer2-käyttöliittymää, vaan ohjelman varsinaista käyttöliittymää. (**V6**)

Muisti on esitettävä niin, että koodi- ja datasegmentit on voitava erottaa toisistaan helposti. Tämä koskee siis ohjelman varsinaista käyttöliittymää, eikä trainer2-käyttöliittymää, jossa muistia ei esitetä lainkaan. (**V7**)

Ohjelmassa ei tarvitse olla mahdollisuutta ohjelman editoimiseen, vaan riittää, että käyttäjä voi valita ohjelmia levyltä ja editointi tapahtuu jossain muualla. (**V8**)

Simulaattori on dokumentoitava englanniksi. Tämä koskeen TTK-91-koneen määrittelyä, käyttöohjetta, sekä muita liitteitä kuten esimerkiksi sallittuja operaatiokoodeja. Lisäksi käyttöohje tulee toteuttaa niin, että se on saatavilla sekä ohjelman sisäisenä että www-linkitettävänä kokonaisuutena. (**V9**)

Simulointikoodin on oltava helppolukuista. Tämä tarkoittaa sitä, että koodissa käytetyt tunnukset ovat englanninkielisiä, kuvaavia sanoja. Myös koodin kommentointi tulee tehdä englanniksi ja kommentoinnin helpottaa koodin lukemista niin, että koodia voidaan esitellä tito-kurssin luennolla. (**V10**)

Ohjelmaa on voitava käyttää automaattisesti tarkistettavien harjoitustöiden yhteydessä. Tämä tarkoittaa sitä, että varaudutaan toteutuksessa, tässä dokumentissa myöhemmin kuvattavin toimenpitein, tukemaan myöhemmin toteutettavaa trainer2-ohjelmaa. Erityisesti kääntäjä ja simulaattori tulee toteuttaa niin, että syötteet ja tulosteet ovat merkkijonomuotoisia. (**V11**)

Ohjelman tulee toimia oikein myös itseään muuttavia TTK-91-ohjelmia simuloitaessa. Tämä tarkoittaa sitä, että mikäli koodisegmenttiä manipuloidaan LOAD- tai STORE-käskyllä, tulee muistin näytön tukea tätä mahdollisuutta, eikä muistin esitys saa poiketa muista riveistä. (**V12**)

Muistin graafinen esitys on toteutettava niin, että konekäskyn numeerinen arvo on oltava näkyvissä symbolisen konekäskymuodon rinnalla. Mahdollisesti kolmantena esityksenä on myös hex- tai binäärimuoto. (**V13**)

TTK-91-ohjelmaa valittaessa on käytössä oltava koko hakemistopuu, eikä vain ohjelman oma hakemisto. (**V14**)

Simulaattorissa on oltava mahdollista vaihtaa muistin kokoa, käyttöliittymän kieltä sekä levyä simuloivia luku- ja kirjoitustiedostoja. (**V15**)

TTK-91-kieleen on lisättävä uusi konekäsky, SHRA, eli aritmeettinen siirto oikealle niin, että etumerkki säilyy. (**V16**)

Simulaattorin tulee simuloida levyoperaatioita kahdella erillisellä tiedostolla. Lukeminen tapahtuu IN-käskyllä ja kirjoittaminen OUT-käskyllä. (**V17**)

Simulaattoriin toteutetaan graafinen käskyn suoritussyklin animointi niin, että kuvataan koneen toiminta suoritettavalla käskyllä rekisteritasolla, mutta väylän toimintaa simuloimatta. Näytetään siis esimerkiksi kuinka STORE R1, =100 siirtää rekisterin yksi arvon muistiin MAR:ia ja MBR:ää käyttäen. (**V18**)

Simulaattori animoi laiteajurin toimintaa laiterekisteritasolla. (**V19**)

2.3 Laadulliset tavoitteet

Siirrettävyys toiseen ohjelmistoympäristöön/käyttöjärjestelmään: Asiakas haluaa mahdollisimman helpon siirrettävyyden siten, että ohjelma toimii sekä Windows- että Linux-ympäristössä. (**V20**)

Osien uudelleenkäytettävyys muissa järjestelmissä ja yhteiskäyttöisyys: Asiakas haluaa, että käyttöliittymän voi vaihtaa trainer2-yhteensopivaksi ja että ohjelmaa voi käyttää verkon yli. Vaihtoehtoinen käyttöliittymä voisi esimerkiksi olla HTML-sivu, jonka kaavakkeista käyttäjän syötteet keräävä ohjelma välittää ne ydinjärjestelmälle, joka koostuu tässä projektissa toteutettavista komponenteista. Uuden toiminnallisuuden lisääminen ei siis vaatisi muuta kuin käyttöliittymän toteuttamisen uusien vaatimusten mukaiseksi. (**V21**)

Ylläpidettävyys: Asiakas vaatii TTK-91-koneen määrittelystä, simulaattorin käytöstä sekä operaatiokoodeista dokumenttia englannin kielellä. Lisäksi asiakas vaatii kattavaa koodin kommentointia. (**V22**)

Joustavuus: Asiakas vaatii, että muistin kokoa on voitava muuttaa, kieli on vaihdettavissa ja suorittamisen optiot ovat helposti muutettavissa niin, että voidaan suorittaa komento kerrallaan, tai koko koodi pysähtyen ainoastaan mikäli ohjelma tarvitsee syötettä näppäimistöltä, tai ohjelma pääsee loppuun asti. Tämä koskee ohjelman varsinaista käyttöliittymää, eikä trainer2-rajapinnan toteuttavaa käyttöliittymää. (**V23**)

Testattavuus: Asiakas ei esitä vaatimuksia ohjelmiston testattavuudelle. Asiakas kuitenkin vaatii ohjelmistolta virheetöntä toimintaa. (**V24**)

Oikeellisuus: Asiakas vaatii, että toiminnalliset vaatimukset 1-17, sekä kaikki laadulliset tavoitteet toteutetaan. Ohjelma, joka toteuttaa mainitut tavoitteet, kattaa asiakkaan ohjelmistolle esittämät vaatimukset. (**V25**)

Luotettavuus: Asiakaan luotettavuudelle asettamat vaatimukset sisältyvät implisiittisisti muihin vaatimuksiin. (**V26**)

Tehokkuus: Asiakas ei aseta tarkkoja vaatimuksia ohjelman tehokkuudelle. Päätavoite ei ole toimia mahdollisimman nopeasti, vaan mahdollisimman oikein. (**V27**)

Itsesuojelukyky: Ohjelma on toteutettava niin, ettei suoritukseen pääse vaikuttamaan ohjelman ulkopuolelta. Javan luokat on siis suljettava mahdollisimman tehokkaasti niin, et-

tei ohjelman suoritusta pääse sotkemaan käyttämättä ohjelmiston tarjoamaa rajapintaa. (**V28**)

Käytettävyys: Asiakas vaatii, että ohjelma on riittävän helppokäyttöinen käytettäväksi tito-kurssilla. (**V29**)

3 Simulaattorin tarjoamat palvelut

Tässä luvussa on kuvattu, edellisessä luvussa kuvatuista vaatimuksista johdetut, simulaattorin käyttäjälle tarjoamat palvelut.

3.1 TTK-91-ohjelman valinta

- Toiminnon nimi: TTK-91-ohjelman valinta.
- Kuvaus: Käyttäjä valitsee haluamansa .k91- tai .b91-ohjelman. Valinta tapahtuu joko graafisesti käyttämällä esimerkiksi Javan valmista palvelua tiedoston valintaan tai sitten syöttämällä tiedoston nimi tekstikenttään. Mikäli valitaan .b91-muotoinen ohjelma, suoritetaan myös sen lataus.
- Syötteet: Käyttäjää pyydetään valitsemaan haluamansa ohjelma joko antamalla sen nimi tai valitsemalla se graafisesta hakemistonäkymästä.
- Tulosteet: Ohjelma tulostaa käyttäjän näkyviin ohjelman käskylistauksen. Jos ohjelma on kääntämätön, tulostetaan ohjelmakoodi symbolisina konekäskyinä, mutta mikäli ohjelma on jo käännetyssä muodossa, tulostetaan ohjelma määritellyssä binäärimuodossa positiivisina kokonaislukuina sekä symbolisessa muodossa. Mikäli syötteenä saatiin virheellinen .b91-tiedosto, kerrotaan siitä selkeästi, ja ohjelman toiminta keskeytyy.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys: Toiminto koskee vaatimuksia V2, V5, V13 ja V14.

3.2 TTK-91-ohjelman kääntäminen ja lataus

- Toiminnon nimi: TTK-91-ohjelman kääntäminen ja lataus.
- Kuvaus: Ohjelma tarjoaa käyttäjälleen mahdollisuuden kääntää TTK-91-ohjelmia symbolisesta konekielestä käännettyyn .b91-muotoon. Kääntämisen jälkeen ohjelma linkitetään ja ladataan muistiin. Ohjelman tulee olla sellaisessa vaiheessa, että valittu ohjelma on .k91-muodossa.
- Syötteet: Käyttäjä valitsee valikosta Compile-käskyn.

- Tulosteet: Mikäli syötteenä saatu symbolinen konekäskysarja oli kelvollinen TTK-91-ohjelma, tulostetaan se ruudulle käännettynä siten, että käyttäjälle näytetään käännetty ohjelma sekä positiivisina kokonaislukuina että symbolisessa muodossa. Mikäli syötteenä saatu merkkijono ei ole kelvollinen ohjelma, suoritus pysähtyy ensimmäiseen virheelliseen operaatiokoodiin, muistiosoitukseen tms. ja palauttaa käyttäjälle virheilmoituksen kuvaten tapahtuneen virheen selkeästi.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys: Toiminto koskee vaatimuksia V5, V7, V11 ja V13.

3.3 TTK-91-ohjelman suoritus varsinaisessa käyttöliittymässä

- Toiminnon nimi: TTK-91-ohjelman suoritus.
- Kuvaus: Käyttäjä voi suorittaa TTK-91-muotoisia ohjelmia simulaattorilla. Suorittamiseen on useita mahdollisia tapoja, kuten puhdas suoritus ilman keskeytystä, suorittaminen käsky kerrallaan, käsky kerrallaan suoritus kommentoiden sekä animoitu käsky kerrallaan suoritus. TTK-91-ohjelman suorittaminen vaatii, että on valittu suoritettava ohjelma ja että valittu ohjelma on käännetyssä .b91-muodossa.
- Syötteet: Käyttäjä valitsee käyttöliittymän valikosta Run-käskyn. Suorituksen parametrit on valittava ennen suoritusta.
- Tulosteet: Ohjelma tulostaa rekistereiden ja muistin tilan jokaisen suoritetun käskyn jälkeen, minkä lisäksi mahdolliset OUT-käskyllä tapahtuvat tulostukset näytetään selvästi käyttäjälle. Mahdolliset virheet, kuten muistin loppuminen, virheellinen parametri käskyssä tai nollalla jako johtavat suorituksen keskeytymiseen, minkä lisäksi käyttäjälle kerrotaan selkeästi mikä virhe tapahtui.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa muuten, paitsi vaatimuksen V18 osalta, joka toteutetaan mahdollisuuksien mukaan joko tässä tai jossain myöhemmässä projektissa.
- Jäljitettävyys: Toiminto koskee vaatimuksia V2, V5, V6, V12 sekä V18.

3.4 TTK-91-ohjelman kääntäminen trainer2-rajapinnan avulla

- Toiminnon nimi: TTK-91-ohjelman kääntäminen trainer2-rajapinnan avulla.
- Kuvaus: Valittu ohjelma käännetään .b91-muotoon.
- Syötteet: Merkkijonomuotoinen symbolinen konekäskysarja.
- Tulosteet: Trainer2-liittymä tulostaa kääntäjän kommentit tuloste-ikkunaan. Kääntäjän kommentteja ovat mahdolliset virheet käännöksen kuluessa tai onnistuneesta käännöksestä kertova viesti.

- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on mainittu kohdassa V11.

3.5 TTK-91-ohjelman suoritus trainer2-rajapinnan avulla

- Toiminnon nimi: TTK-91-ohjelman suoritus trainer2-rajapinnan avulla.
- Kuvaus: Ohjelmaa suoritetaan trainer2-käyttöliittymällä joko annettu määrä käskyjä tai kokonaan.
- Syötteet: Suoritettava käsky annetaan yhteen tekstikenttään. Toiseen kenttään tulee syöte, mikä annetaan ohjelman pyytäessä syötettä näppäimistöltä ja kolmanteen mahdollinen STDIN-syöte, mikäli ohjelma pyytää syötettä levyltä. Napin ja kentän yhdistelmässä voidaan syöttää suoritettavien komentojen määrälle yläraja.
- Tulosteet: Tulosteet siten, että mahdolliset OUT-käskyllä tulostettavat arvot tulevat merkkijonona ja koneen sisäistä tilaa on mahdollista kysyä koneen pysähdyttyä suoritettuaan joko määritellyn määrän käskyjä tai ohjelman tullessa lopputilaan. Mahdollisessa virhetilanteessa suoritus keskeytyy ja käyttäjälle näytetään virheilmoitus, joka kertoo mikä virhe tapahtui.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on mainittu kohdassa V11.

3.6 Muistin nollaus

- Toiminnon nimi: Muistin nollaus.
- Kuvaus: Käyttäjä haluaa nollata rekistereiden arvot ja muistin.
- Syötteet: Käyttäjä valitsee käyttöliittymän File-valikosta kohdan Erase memory.
- Tulosteet: Tulostetaan koneen muuttunut tila.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on seurausta kohdassa V2 esitetyistä vaatimuksista.

3.7 Käyttöliittymän kielen valinta

- Toiminnon nimi: Käyttöliittymän kielen valinta.
- Kuvaus: Käyttäjä valitsee valikosta käyttöliittymän kielen. Mahdolliset valinnat haetaan levyltä siten, että kielivaihtoehdot päivitetään levyllä olevien kielitiedostojen mukaan.

- Syötteet: Syötteenä on kieli, joka on valittu graafisesti valikosta.
- Tulosteet: Käyttöliittymän jokainen tekstikenttä korvataan uuden kielen mukaisella merkkijonolla. Ohjelman suoritukseen toiminnolla ei ole vaikutusta.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on kohdassa V5 mainitun toiminnon osa.

3.8 Käytettävien levytiedostojen valinta

- Toiminnon nimi: Käytettävien levytiedostojen valinta.
- Kuvaus: Käyttäjä haluaa vaihtaa käytettäviä STDIN- ja STDOUT-tiedostoja.
- Syötteet: Käyttäjä valitsee käyttöliittymän valikosta Options-kohdasta Configure filesystem -kohdan, josta valitaan Choose STDIN tai Choose STDOUT.
- Tulosteet: Käyttäjän ruudulle tulostuu dialogi-ikkuna, jonka avulla valitaan haluttu tiedosto. Dialogin muotoa ei ole kiinnitetty, vaan se voi olla joko graafinen kuvaus hakemistopuusta tai tekstikenttä, johon käyttäjä syöttää käytettävän tiedoston nimen.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on kohdassa V17 mainitun toiminnon osa.

3.9 Muistin koon asettaminen

- Toiminnon nimi: Muistin koon asettaminen.
- Kuvaus: Käyttäjä valitsee käyttöliittymän valikoista simuloitavan koneen muistin koon. Valinta johtaa mahdollisesti käynnissä olleen ohjelman käsittelyn keskeytymiseen, sillä muistin koon mahdollinen pienentäminen voi johtaa siihen, että osa muistissa olleesta tiedosta katoaa. Toiminnon suuresta vaikutuksesta johtuen toiminnon toteuttaminen on varmistettava käyttäjältä erillisellä varmistus-popupilla, jos ohjelma oli valittu.
- Syötteet: Käyttäjä valitsee graafisesti muistin kooksi jonkin kahden potenssin väliltä 9-16.
- Tulosteet: Muuttunut tila kuvataan käyttöliittymässä siten, että muistin koko on päivittynyt uuden arvon mukaiseksi. Ellei ohjelma ole alkutilassa, palataan siihen.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on kohdassa V15 mainitun toiminnon osa.

3.10 Suoritustavan valinta

- Toiminnon nimi: Suoritustavan valinta.
- Kuvaus: Käyttäjä valitsee käyttöliittymän valikoista simulointitavan: Suoritetaanko yksi käsky kerrallaan, käsky kerrallaan ja kommentoiden, käsky kerrallaan animoiden vai koko ohjelma pysähtyen ainoastaan virhetilanteessa tai pyydettäessä syötettä näppäimistöltä.
- Syötteet: Käyttäjä valitsee haluamansa suoritustavan valikosta.
- Tulosteet: Valikko päivitetään näyttämään valittua suoritustapaa.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on kohdassa V5 mainitun toiminnon osa.

3.11 Käyttöhjeen katselu

- Toiminnon nimi: Käyttöohjeen katselu.
- Kuvaus: Käyttäjä haluaa nähdä käyttöohjeen.
- Syötteet: Käyttäjä valitsee käyttöliittymän Help-valikosta Manual-kohdan.
- Tulosteet: Tulostetaan käyttöohje omassa ikkunassaan.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toiminto on kohdassa V9 mainitun toiminnon osa.

3.12 Simulaattorista poistuminen

- Toiminnon nimi: Simulaattorista poistuminen.
- Kuvaus: Käyttäjä haluaa sulkea ohjelman.
- Syötteet: Käyttäjä valitsee käyttöliittymän valikosta Exit.
- Tulosteet: Suljetaan kaikki ohjelman käyttämät ikkunat.
- Prioriteetti: Toiminto toteutetaan tämän projektin puitteissa.
- Jäljitettävyys. Toimintoa ei ole erikseen mainittu asiakkaan vaatimuksissa.

4 Vaatimusten analysointi

Tässä luvussa on analysoitu luvussa 2 esitettyjä asiakkaan vaatimuksia. Osa vaatimuksista on hyväksytty sellaisenaan, mutta mahdolliset ristiriidat, ja poikkeamat on esitelty tässä luvussa. Myös laadullisia vaatimuksia on analysoitu, ja osa varsin triviaaleista vaatimuksista saa selvemmän muodon tässä luvussa. Tämä koskee sellaisia laadullisia vaatimuksia, joista asiakas ei ole maininnut, mutta joista ryhmä halusi kommentoida.

4.1 Toiminnalliset tavoitteet

V1: Ohjelmiston käyttäminen tito-kurssilla asettaa vaatimuksia erityisesti käyttöliittymälle. Käyttöliittymän täytyy olla niin selkeä, että ohjelmistoa voidaan käyttää helposti heti ensimmäisestä kerasta lähtien. Voidaan kuitenkin olettaa, että rekisterit, muisti ja CPU ovat tuttuja käsitteitä, eikä ohjelmiston tehtävä ole opettaa prosessorin perusosia, vaan simuloida koneen toimintaa.

V2: Vaatimus vanhojen, olemassaolevien TTK-91-ohjelmien suorittamisesta tarkoittaa sitä, että simulaattori toimii kuten aiempi Pascal-versio. Myös käytettävän tiedostomuodon tulee mahdollistaa vanhojen ohjelmien käytön sellaisenaan. Tästä seuraa se, että käytettävän tiedostomuodon tulee noudattaa samaa formaattia kuin aiemmassa simulaattorissa. Vaatimuksesta 4 seuraa kuitenkin, ettei tiedostomuoto voi olla sama.

V3: Vaatimus tarkoittaa sitä, että toteutuksen tulee seurata ensin Koksi-simulaattoria ja vasta sitten Häkkisen monisteessa mainittua standardia. Tästä aiheutunee hieman hankaluuksia testauksessa, mikäli koneesta ei saada kunnon dokumentointia.

V4: Vaatimus tarkoittaa sitä, että tiedostomuodon tulee tukea sekä \r\n, \r että \n-merkkejä rivinvaihdossa. Vaaditaan myös toiminnan yhteneväisyyttä käytettäviltä komponenteilta Windowsin ja Linuxin virtuaalikoneilla. Tämä asettaa erityisesti vaatimuksia käyttöliittymälle, jonka suunnittelussa on otettava huomioon mahdollinen toiminnan erilaisuus eri käyttöjärjestelmien välillä.

V5: Vaatimus on laaja, ja ensimmäinen, ohjelman valintaa koskeva vaatimus tarkoittaa sitä, ettei tiedoston valintaa pidä kutsua load-operaatioksi, eikä suomeksi lataamiseksi. Kääntäjää koskevaa vaatimusta levyltä lukemisesta ja levylle kirjoittamisesta ei tulkita ehdottomaksi vaatimukseksi, sillä ainakin trainer2-rajapinnassa ei liene järkevää tallettaa .b91-tiedostoa levylle. Mahdollisuus tallettamiseen on kuitenkin oltava kääntäjässä. Myös syötteen muoto on .k91-tiedostomuotoa noudattava merkkijono, eikä tiedosto. Vaatimus lataajan toteutuksesta omana komponenttinaan juontaa juurensa luultavimmin siitä, että ohjelman valinnassa on mahdollista valita käännetty ohjelma, jolloin lataaaminen tehdään valinnan jälkeen, ja koska kääntämisen jälkeen ladataan myös, lienee järkevää käyttää samaa koodia molemmissa tapauksissa.

Suorittimen sijoittaminen omaksi kokonaisuudekseen vaikuttaa myös järkevältä vaatimukselta, sillä se on kaikkein kriittisin osa. Vaatimus puhtaan käännöksen ja suorituksen käyttöliittymästä on ristiriidassa trainer2-liittymästä annettujen viimeisimpien tietojen mukaan, ja onkin tulkittu niin, että vaatimus tarkoittaa, että simulaattoriin on toteu-

tettava kaksi erillistä käyttöliittymää, jotka on kuvattu myöhemmin tässä dokumentissa. .b91-ohjelmien tiedostomuoto on kuvattu liitteessä 1, jossa on erimerkkiohjelma talletettuna .b91-muotoon. Ohjelman symbolinen muoto löytyy osoitteesta
`http://www.cs.helsinki.fi/u/kerola/ohtu/ttk91_simu.html`.

V6: Vaatimus graafisesta suoritusaikaisesta suorituksen emuloinnista tarkoittaa sitä, että kaikkia käyttäjälle näkyviä osia on päivitettävä jokaisen suoritetun käskyn jälkeen. Käyttäjälle näkyvillä osilla tarkoitetaan rekistereitä, niiltä osin kuin ne on käyttöliittymässä esitetty, sekä muistia.

V7: Vaatimus datan ja koodin erottamisesta muistin näytössä tarkoittaa sitä, että datasegmentti on erotettava omaksi osakseen muistin näytössä. Näkymä on voitava jakaa kahteen osaan, joita kumpaakin voi vierittää erikseen.

V8: Kahdeksannessa kohdassa mainittu editoinnin poisjättäminen tarkoittaa, että aiemmasta poiketen ei ole tarpeen mahdollistaa koodin muokkausta simulaattorissa, vaan käyttäjä muokkaa ohjelmiaan muualla.

V9: Vaatimus englanninkielisistä dokumenteista koskee mainittuja osia. Osien kääntäminen suomeksi on lisätavoite, ja se toteutetaan mahdollisuuksien mukaan.

V10: Vaatimus koodin helppolukuisuudesta tarkoittaa sitä, että koodi on kommentoitava niiltä osin kun se ei ole niin selvää, että se selittäisi itsensä. Tarkkaa määrittelyä hyvin kommentoidusta koodista ei ole esitetty, ja vaatimus onkin enemmän ohje kuin eksakti, jäljitettävä, vaatimus. Koodin on kuitenkin oltava niin luettavaa, että sitä voi esitellä ensimmäisen vuoden opiskelijoille.

V11: Vaatimus mahdollisen trainer2-ohjelmiston tukemisesta tarkoittaa käyttöliittymän korvattavuutta uudella, trainer2-mallisella käyttöliittymällä, joka tarkistaa simulaattorin tulosteesta laskuharjoitustehtävien oikeellisuuden. Vaatimus toteutuu niin, että Teemu Kerola määrittelee rajapinnan ja molemmat ryhmät toteuttavat annetun rajapinnan itsenäisesti. Tämä varmistaa sen, että ryhmät toimivat erillisinä, mutta myös sen, että rajapinta on molemmilla sama. Mahdolliset lisäykset ja kommentit kulkevat Teemu Kerolan kautta. Viimeisin rajapintaehdotelma trainer2-liittymälle on kuvattu liittessä 2.

V12: Vaatimus tarkoittaa, että mikäli suoritettava ohjelma viittaa koodisegmenttiin, toimii muistin visualisointi siten, ettei käskyn suorittaminen sotke muistin esitystä, vaan muutettu rivi tulkitaan konekäskyksi mikäli se on sallittu konekäsky. Jos taas tallennettu arvo ei ole laillinen konekäsky, kirjoitetaan vain tallennettu arvo.

V13: Vaatimus tarkoittaa, että koodisegmentti on kuvattava muistissa sekä positiivisina kokonaislukuina että symbolisina konekäskynä.

V14: Tiedostoa valittaessa on oltava mahdollisuus valita mikä tahansa hakemistopuussa oleva tiedosto, eikä pelkästään tiedostoja simulaattorin omasta hakemistosta.

V15: Optioista toteutetaan ainakin tässä dokumentissa mainitut vaatimukset. Muistin koon täytyy olla jokin kahden kokonaislukupotenssi yhdeksästä kuuteentoista, esimerkiksi pienimmillään muistin koko on 512 sanaa, suurimmillaan 65536 sanaa. Alaraja on vaatimuksen V2 vuoksi yhdeksän ja yläraja on käskyformaatista johtuen kuusitoista. Käskyssä on muistiviitteelle vain kuusitoista bittiä, joten sitä suurempia muisteja ei kannata tukea.

V16: SHRA:n arvo on 27, ja se lisätään käskykantaan tuolla koodilla.

V17: Levy-I/O toteutetaan liitteessä kolme mainitulla tavalla. Tiedostojen vaihtaminen onnistuu myös käyttöliittymän valikosta.

V18: Graafinen käskyn suoritussyklin animointi toteutetaan siten, että rekisterin yksi arvon tallettaminen muistipaikkaan 100 toteutuisi jokseenkin niin, että ensin näytetään miten käsky tulkitaan positiivisesta kokonaisluvusta STORE-käskyksi; seuraavana näytetään mistä tieto siirretään, mitä väylää pitkin se kulkee ja minne se talletetaan. Väyläprotokollia ei animoida muuten kuin näyttämällä mikä väylä on aktiivisena missäkin vaiheessa. CPU:n rekistereiden arvojen vaihtuminen sen sijaan animoidaan näyttämällä mihin väylältä tullut arvo tallettuu.

V19: Animoinnilla pyritään havainnollistamaan korkealla tasolla laiteajurin toimintaa kun palvelupyynnöllä käynnistetään levyoperaatio.

4.2 Laadulliset tavoitteet

V20: Ohjelmisto toteutetaan Java-kielellä sen peruskirjastoja käyttäen ja käyttöliittymään käytetään mahdollisimman ikkunointijärjestelmäriippumatonta Swing-kirjastoa, joten toteutusvirheet poissulkien ohjelman tulisi toimia muutoksitta missä tahansa Java 2 -alustan perusedition, API-version 1.4.2 toteuttavassa järjestelmässä. Ohjelman toiminta kuitenkin testataan vain vaadituissa käyttöjärjestelmissä.

V21: Ohjelmaan kuuluu ulkoinen API, jonka avulla simulaattorin toiminnallisuuteen voidaan päästä käsiksi vaihtoehtoisista käyttöliittymistä. API:n suunnittelussa on otettu huomioon vasta ideointivaiheessa olevan automaattisen harjoitustehtävien tarkastusohjelman, Trainer2:n, todennäköiset tarpeet.

V21: Sovelluksen käyttäjät ovat tyypillisesti tito-kurssin kävijöitä, jolloin he mahdollisesti hakevat sovelluksen itselleen kurssin alussa ja käyttävät sitä kurssin ajan joko kotona tai keskitetysti Helsingin yliopiston Tietojenkäsittelytieteen laitoksen koneilla. Täten mahdollisten päivitysten levitys voidaan keskittää tietylle asiakkaan ylläpitämälle sivustolle, josta sovellus voidaan hakea kotikäyttöön, sekä mahdolliseen laitokselta löytyvän kopion päivitykseen. Keskitymme ylläpidettävyyspyrkimyksissämme uusien komentojen ja sisäisten vakioiden lisäämisen yksinkertaisuuteen sekä virheiden korjaamisen helpottamiseen lähinnä kommentoidun ja itsekommentoivan koodin avulla; lisäksi projektin dokumentaatiolla pyritään helpottamaan uusien komentojen ja vakioiden lisäämistä.

V22: Sovelluksen ohjelmoinnissa pyritään ylläpidettävyyden helpottamiseksi seuraamaan hyviä ohjelmointikäytäntöjä, joihin kuuluu mm. luokkien ja metodien jakaminen ymmärrettävyyden kannalta sopivan pieniksi kokonaisuuksiksi sekä luokkien välisen sidosteisuuden minimointi.

V23: Joustavuus toteutetaan vaatimusten kuvaamalla tavalla.

V24: Kunkin metodin parametrien oikeellisuus ja syötteen muoto tulee dokumentoida koodissa, samoin kuin sen tekemät mahdolliset muutokset eri luokkien sisäiseen tilaan.

V25: Oikeellisuuden todentamiseen päästään kattavalla testauksella sekä onnistuneella

vaatimusmäärittelyllä.

V26: Pääpainoisesti pyrimme siihen, että ohjelma toimii oikein oikeellisella syötteellä. Virhetilanteiden sattuessa ohjelma keskeyttää toiminnon, jonka puitteissa virhe tapahtui (esimerkiksi koodin kääntäminen tai käyttöliittymän kielen vaihtaminen), ilmoittaa virheestä ja palautuu sisäisesti toimintoa edeltävään tilaan (esimerkiksi oletus- tai viimeksi valittuun kieleen) tai jää muutettuun tilaan, mikäli virhetilanne johtui käyttäjän simuloimassa ohjelmassa olleesta virheellisyydestä. Esimerkiksi simulointi jää tilaan, jossa seuraavaksi suoritettaisiin virheen aiheuttanutta komentoa seuraava komento.

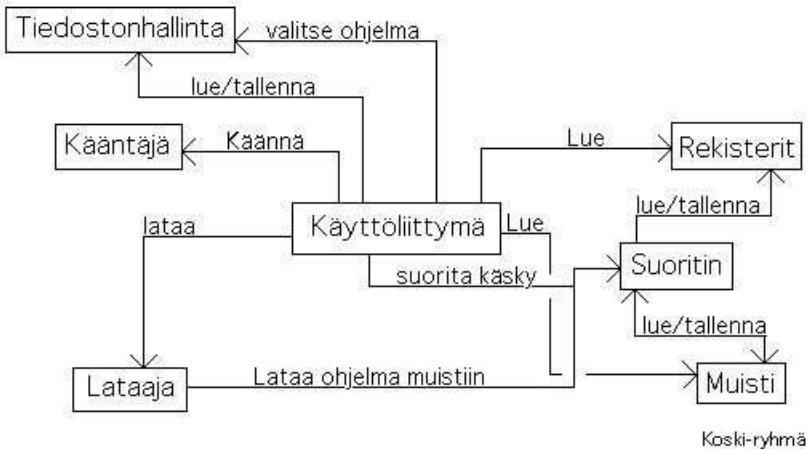
V27: Ohjelmisto käyttää Javan roskien keruun mahdolliset toimintahäiriöt poislukien jokseenkin vakiomäärän muistia käännettävien ohjelmien koon lisäksi, ja sitä tulee voida ajaa 256M muistia sisältävällä CSL-Linux-koneella oletuksena käynnistyvässä ikkunointijärjestelmässä (esimerkiksi Helsingin yliopiston Tietojenkäsittelytieteen laitoksen porkkala-koneet). Lisäksi ohjelmiston tulee tällaisella koneella käynnistyä ja olla valmis ottamaan vastaan käyttäjän syötteitä paikalliselta levyltä ladattuna alle viidessä sekunnissa. Sen ei tällöin tule myöskään käyttää yhden komennon suorittamiseen ja kommentointiin yli sekuntia, ja kommentoimattomassa eräajon kaltaisessa suorituksessa simulaattorin tulee suorittaa vähintään kaksi komentoa sekunnissa. Trainer2-rajapintaa käytettäessä vaatimukset eivät koske mahdollisesta etäkäyttöä verkon yli, vaan verkkokäytöstä seuraavat mahdolliset lisäongelmat jätetään Trainer2-jatkoprojektin huoleksi. Myös trainer2-käyttöliittymä toimii kaikkiaan yhdellä koneella (esimerkiksi www-palvelin), vaikkakin ohjelman syötteet ja tulosteet voivat mahdollisesti kulkea verkon yli.

V28: Sovelluksen itsesuojelukyky jätetään Javan virtuaalikoneen ja allaolevan käyttöjärjestelmän suojausten varaan. Sovellus ajaa sen käynnistäneen käyttäjän tai ohjelman (esimerkiksi www-palvelimen) oikeuksilla. Luokat toteutetaan niin, että niiden sisäistä tilaa voi muuttaa ulkopuolelta vain julkisten metodien kautta, ja esimerkiksi koneen rekistereitä voi muuttaa vain rajapinnan tarjoamilla käskyillä.

V29: Simulaattorin käyttö osana opetusta asettaa tiettyjä erityisvaatimuksia ohjelmistolle. Ensinnäkin ohjelman käyttäjät ovat aina uusia, eikä kurssilla ole vara käyttää paljon aikaa simulaattorin käytön opetteluun. Konekielisestä ohjelmoinnista hullaantuneiden uskotaan siirtyvän suuremmille vesille, eikä heitä varten uhrata kovinkaan paljoa aikaa. Opiskelijoiden ei voida myöskään olettaa omaavan kovinkaan laajaa kokemusta ohjelmoinnista, eikä erityisesti tietokoneen toiminnasta rekisteritasolla. Toiminnan selkeys on selvästi tärkein tavoite käyttöliittymää tehtäessä. Tämä voidaan varmistaa tarpeeksi laajalla käyttöliittymätestauksella, joka toteutetaan niin, että koehenkilöinä on potentiaalisia oikeita käyttäjiä.

5 Ohjelmiston yleisrakenne

Tämä luku kuvaa ohjelmiston tarjoamat palvelut ja antaa kuvan yleisarkkitehtuurista. Ohjelman ulkoiset rajapinnat on kuvattu pääpiirteittäin, mutta varsinainen määrittely jää suunnitteludokumenttiin.



Kuva 1: Ohjelman yleiskuva.

5.1 Arkkitehtuuri

Tässä esitetty kuvaus ei ole luokkakaavio, mutta kuvaa simulaattorin eri osat kiinnittämättä toteutusta niin, että eri osat olisi välttämättä toteutettava omina osinaan. Tarkoitus on kuvata eri osien toiminnallisuus ja kontrollin kulku ohjelman suorituksen aikana. Yleisrakenne ja tärkeimmät palvelut on kuvattu kuvassa 1.

Käyttöliittymä kontrolloi suoritusta, mutta on muuten tyhmä siten, ettei käyttöliittymään kytketä toiminnallisuutta. Käyttöliittymä toimii määritelmän mukaisesti linkkinä käyttäjän ja eri osien välillä kutsuen toiminnallisuutta eri osilta käyttäjän antamien komentojen mukaisesti. Käyttöliittymällä on mahdollisuus pyytää muistin ja rekistereiden arvoja, esimerkiksi lataamisen jälkeen suoraan muistilta ja rekistereiltä.

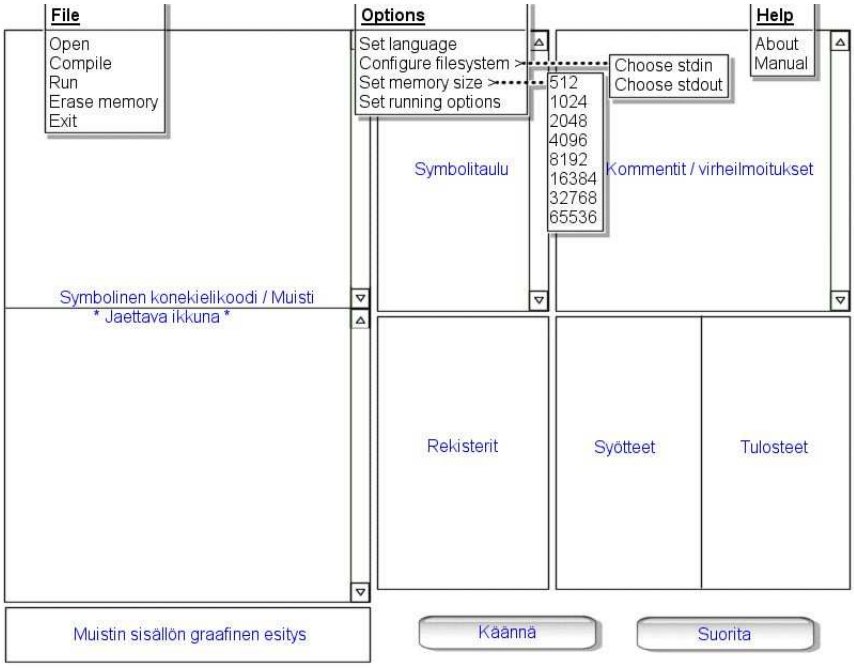
Tiedostonhallinta toteuttaa levyltä lukemisen ja levylle kirjoittamisen ohjelmaa valittaessa, käännettyä ohjelmaa tallennettaessa ja levy-I/O:ta simuloitaessa.

Kääntäjä toimii kahteen suuntaan siten, että se tulkkaa symbolisen konekäskyn positiiviseksi kokonaisluvuksi että positiivisen kokonaisluvun konekäskyksi, mikäli syötteen numerokoodi on tulkittavissa validiksi komennoksi. Tätä palvelua tarvitaan muistin kuvaamiseen, suorittimen toimintaan ja ohjelman kääntämiseen. Kääntäjä palauttaa onnistunessaan palautteena muunnetun käskyn ja virhetilanteessa virhekoodin, joka kertoo mikä virhe tapahtui.

Lataaja vie käännetyn ohjelman muistiin siten, että se kutsuu kääntäjää tarvittaessa kuvaamaan jokaisen käskyn symbolisessa muodossa, ja käskee suorittimen viedä numeerisen arvon muistiin sille kuuluvalle paikalle.

Suoritin toimii siten, että käyttöliittymä pyytää suorittimelta palvelua “suorita konekäsky”. Paluuarvona käyttöliittymä saa tiedon koneen muuttuneesta tilasta. Tarkka formaatti on vielä määrittelemättä.

Muisti voidaan toteuttaa joko suorittimen osana tai omana erillisenä osanaan. Muistilla on kaksi toimintoa; annettuun muistipaikkaan viedään tietoa, tai voidaan kysyä parametrinä



Kuva 2: Käyttöliittymä

annetun muistipaikan sisältö.

Rekisterit voidaan toteuttaa, muistin tapaan, joko omana osanaan tai osana suoritinta. Myös toiminnallisuus on samanlainen kuin muistillakin. Rekisteriin voidaan tallettaa arvo ja rekisterin arvoa voidaan kysyä.

Esimerkkitapaus: Käyttäjä valitsee ohjelman, jolloin käyttöliittymä pyytää palvelua tiedostonhallinalta ja esittää saadun ohjelmakoodin käyttäjälle. Käyttäjän pyytäessä kääntämistä antaa käyttöliittymä tiedostonhallinnan palauttaman merkkijonon kääntäjälle. Kääntäjältä saatu palaute annetaan lataajalle, joka vie tiedot muistiin, alustaa rekisterit ja palauttaa kontrollin käyttöliittymälle. Tämän jälkeen käyttäjä pyytää simulaattoria suorittamaan ohjelman, joka tapahtuu niin, että käyttöliittymä pyytää suoritinta suorittamaan käskyn kerrallaan, kuvaamalla koneen toimintaa saadun tulosteen perusteella.

5.2 Käyttöliittymä

Asiakkaan toiveiden mukaan ohjelmaan tulee kaksi erillistä käyttöliittymää. Varsinaista toteutusta ei ole kiinnitetty tässä dokumentissa, mutta asiakkaan vaatimat palvelut voidaan jakaa kahdelle käyttöliittymälle.

Ohjelman varsinainen käyttöliittymä tullaan määrittelemään myöhemmin siten, että se täyttää asiakkaan tässä dokumentissa asetetut vaatimukset. Tulevasta toteutuksesta on esitetty hahmotelma kuvassa 2. Tämä ei kuitenkaan ole varsinainen käyttöliittymäsuunnitelma, vaan on tehty alustavaksi lähtökohdaksi vanhan Koksin pohjalta.

Käyttöliittymän peruspiirteitä on alustavasti hahmoteltu siten, että ikkuna jakautuisi ku-

van 2 mukaisesti kuuteen osaan eli kehykseen. Näiden kehysten sisältö hahmotellaan pääpiirteittäin seuraavassa, mutta niiden toteutukseen ei nyt oteta kantaa. Osa kehyksistä olisi vieritettäviä (merkitty kaavakuvaan), koska suuresta tietomäärästä johtuen kaikki tieto ei mahtuisi näkyville kerralla.

Kehys “symbolinen konekieli/muisti” sisältää ennen käännöstä symbolisen koodin valekäskyineen, jotka käännöksen jälkeen vaihtuisivat symbolisen konekielikoodin ja binäärisen koodin yhdistelmäksi. Käyttöliittymä korostaa suorituksessa olevan konekäskyrivin, mikäli se näytetään muistia kuvaavassa kehyksessä.

Muistin sisällön graafinen esitys on alalaidassa sijaitseva, muistin sisältöä kuvaava palkki, jossa muistin sisällön eri osat koodattaisiin eri väreillä muistin käytön havainnollistamiseksi. Esimerkiksi koodiosa voisi olla vihreällä ja pino-osa keltaisella.

Symbolitaulussa näytetään käännöksen jälkeen symboliset vakiot ja niiden arvot.

Kommentit/virheilmoitukset -kehykseen tulostetaan sekä käännöksen ja suorituksen aikana havaitut virheet että koodin kommentoinnit.

Rekisterit-kehyksessä näytetään rekisterien sisältö. Myös tilarekisteri näytetään osittain.

Syötteet- ja Tulosteet-kehyksiin tulevat näppäimistöltä tuleva ja näytölle menevä syöte.

Toinen, trainer2-yhteensopiva käyttöliittymä on varsin yksinkertainen ja koostuu neljästä tekstikentästä, joista ensimmäiseen käyttäjä syöttää ohjelmakoodin joko kirjoittamalla tai avaamalla kenttään koodin sisältävän tiedoston. Tiedostossa tai kentässä oleva ohjema suoritetaan toisessa ja kolmannessa kentässä olevilla KBD- ja STDIN-syötteillä. Neljänteen kenttään tulevat ohjelman CRT- ja STDOUT-tulosteet, kun TTK-91-ohjelma suoritettaan annetulla syötteellä. Suorittamiseen liittymässä on nappi, josta ohjelma käynnistyy; tällöin ohjelma käännetään ja suoritetaan. Lisäksi käyttöliittymässä on nappi, jolla ohjelma voidaan vain kääntää, sekä napin ja kentän yhdistelmä, jonka avulla ohjelmaa voidaan suorittaa vain tietty määrä komentoja. Tämä käyttöliittymä toteutetaan, jotta voidaan varmistua trainer2-rajapinnan toteutumisesta niin, että tulevaisuudessa sitä voidaan käyttää laskuharjoitusten automaattiseen tarkastukseen.

5.3 Ulkoiset rajapinnat

Ohjelma tarjoaa ulkoisen rajapinnan ohjelmien suoritukselle siten, että syötteenä annetaan kolme merkkijonoa, joista ensimmäinen on suoritettava symbolisen konekielen ohjelma, toinen on käyttäjän syöte sitä kysyttäessä IN Rj, =KBD-komennolla ja kolmas merkkijono on syöte pyydettäessä syötettä levyltä komennolla IN Rj, =STDIN. Simulaattori suorittaa ohjelmaa joko tietyn käskyjen määrän tai loppuun asti. Mikäli ohjelman suoritus päättyy virheeseen, heitetään poikkeus, josta saadaan virhettä kuvaava virhekoodi. Ohjelman pysähtyessä suoritettuaan laillisen käskyn voidaan rajapinnan kautta selvittää koneen muuttunut tila. Rajapinnan tarkkaa muotoa ei ole lyöty lukkoon, mutta se löytyy osoitteesta `http://www.cs.helsinki.fi/u/kerola/ohtu/ttk91_simu.html`.

Liite 1. Esimerkkiohjelma .b91-muodossa

```
___b91___           (varattu sana, 3 alaviivaa molemmin puolin)
___code___          (varattu sana)
0 9                 (koodialueen alku ja loppu eli init FP)
52428801
18874378
572522503
36175883
287834122
18874379
538968064
36175883
69206016
1891631115
___data___          (varattu sana)
10 11               (data-alueen alku ja loppu eli init SP)
0                   (data-alue)
0
___symboltable___   (varattu sana)
luku 10             (symbolitaulu, vain paikalliset symbolit)
summa 11            (kääntäjän symb. mukana oletusarvoisesti)
___end___           (varattu sana)
```

Liite 2. Trainer2-rajapintaluokat

- CompileSource-rajapintaluokka
  - String getSource() palauttaa lähdekoodin rivinvaihtoineen.
- TTK91Exception
  - int getErrorCode() palauttaa virhekoodin, ks. alla.
- TTK91CompileException extends TTK91Exception
  - Määrittelee virhekoodit SYNTAX_ERROR = 101, SYMBOL_NOT_DEFINED = 102 ja OUT_OF_MEMORY = 103.
- TTK91RuntimeException extends TTK91Exception
  - Määrittelee virhekoodit ADDRESS_OUT_OF_BOUNDS = 201, INVALID_OPCODE = 202, INVALID_DEVICE = 203 (laitenumero ei täsmää tai laitetta ei voi käyttää näin, esimerkiksi lukeminen näytöltä), INVALID_SERVICE = 204 (epäkelpo palvelukoodi), DEVICE_INTERRUPT = 205 (ei käytössä), INTEGER_OVERFLOW = 206, DIVISION_BY_ZERO = 207, BAD_ACCESS = 208 (epäkelpo muistiosoitus), NO_FILE_DATA = 209, NO_KBD_DATA = 210 ja FAILED_WRITE = 211.
- TTK91Application
  - int[] getSerialized() palauttaa sovelluksen koodialueen kokonaislukuarvoina, yksi komento paikkaa kohti.
  - String getStdOut() palauttaa sovelluksen ajon aikana levylle tulostetut luvut, eroteltuna rivinvaihdoin.
  - String getCrt() palauttaa sovelluksen ajon aikana näytölle tulostetut luvut, eroteltuna rivinvaihdoin.
  - void setKbd(String input) asettaa näppäimistöltä luettavan datan, yksi luku per rivi. Tätä käytetään ensisijaisesti, kunnes sovellus voi päättää mitä tehdä kun tämä varanto loppuu. Ratkaisu riippunee käyttöliittymästä.
  - void setStdIn(String input) toimii kuten setKbd, mutta asettaa näppäimistödatan sijaan levyltä luettavan datan.
- TTK91Bridge
  - TTK91Application compile(CompileSource source) kääntää CompileSource-luokan abstrahoiman lähdekoodin ajovalmiiksi sovellukseksi.
  - void run(TTK91Application app, int steps) ajaa annettua sovellusta *steps* askeleen verran ja pysähtyy sitten, vaikkei sovellus olisikaan valmis. Mikäli *steps* on 0, sovellus ajetaan loppuun tai virhetilanteeseen asti.

  - Memory getMemory() palauttaa muistin, jonka tila on ajantasainen palautushetkellä.
  - CPU getCPU() palauttaa olion, jolta voi kysyä eri rekisterien arvoja. Rekisterien tila on ajantasainen palautushetkellä.

- Memory
  - int getLength() returns the size of the memory.
  - int getValue(int memorySlot) returns the integer stored in memory position *memorySlot*.
  - java.util.HashTable getSymbolTable() returns a HashTable containing all the local symbols of the program as keys and their defined values attached to the keys.
  - int[] getMemory() returns the entire contents of the memory as an integer array.
  - int[] getCodeArea() returns the contents of the code area of the memory as an integer array. The code area contains the area where the original commands were stored when the application was loaded.
  - int[] getDataArea() returns the contents of the data area of the memory as an integer array. The data area contains predefined constants and the stack, but eg. not the results of writes to random memory positions unless they happen to be on the data area.

- CPU
  - int getValueOf(int regID) returns the value of the register referred to by *regID*. The acknowledged IDs are CU_TR = 201, CU_IR = 202, CU_PC = 203 (the value of the PC one cycle ago), CU_PC_CURRENT = 204, CU_SR = 205, REG_R0 = 401, REG_R1 = 402, REG_R2 = 403, REG_R3 = 404, REG_R4 = 405, REG_R5 = 406, REG_R6 = 407, REG_R7 = 408, REG_SP = 409 and REG_FP = 410.
  - int getStatus() returns the status of the CPU. The following statuses are possible: STATUS_STILL_RUNNING = 901, STATUS_SVC_SD = 902 (program exited normally by a service call SVC SP, =HALT.), STATUS_EXITED_ABNORMALLY = 903 (during the abnormal exit an exception is thrown as well). When the TTK91Bridge is told to run the program for a certain number of steps, if the program has not exited by some means before the steps run out, the status returned will be STATUS_STILL_RUNNING.

Liite 3. Levy-I/O

- IN Rj,=STDIN
- OUT Rj, =STDOUT
- STDIN on oletusarvoisesti “stdin” nykyhakemistossa. Sitä voidaan vaihtaa käyttöliittymässä. Sama STDOUT:ille.
- Parametreja nykyhakemisto, STDIN, STDOUT voi myös vaihtaa uusilla valekäskyillä:
  - HOMEDIR DEF MyHomeDir
  - STDIN DEF MyInputFileName
  - STDOUT DEF MyOutputFileName