Loppuraportti

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	13.5.2004	Ensimmäinen versio

Sisältö

1 Johdanto

Toteutimme Ohjelmistotuotantoprojekti-kurssin puitteissa TTK-91-konekielisimulaattorin. Tietokoneen toiminta -kurssilla opetuksen tukena käytetty Koksi-simulaattori oli 13 vuoden palvelun jälkeen lopulta tulossa tiensä päähän, joten keväällä 2004 perustettiin yksi, mutta suunnattoman suosion tarkistuksen jälkeen sittenkin kaksi ryhmää toteuttamaan uusi ja entistä ehompi simulaattori. Sovelluksen tarkoituksena oli korvata Koksi-simulaattori opetuksen apuvälineenä.

Asiakas oli ehtinyt 13 vuoden aikana kehittää muutaman idean simulaattorin parantamiseksi. Uuden sovelluksen tulisi puhua myös englantia, jotta se voitaisiin ottaa kansainväliseen käyttöön. Muutama käyttöliittymällinen puute tulisi korjata, yksi konekielikomento lisätä ja sokerina pohjalla toteuttaa TTK-91-koneen toimintaa matalammalla tasolla havainnollistava suoritussyklin animointi. Itse konekieleen ei sinänsä suuremmin puututtaisi, mutta Koksin “muistidumppien” sijaan uuden ohjelman tulisi tallentaa “binääritiedostoja”, joissa käännetty ohjelma näkyisi kokonaislukuina data-alueineen ja symbolitaulu lisättynä tiedoston loppuun.

Asiakas toivoi erityisesti, että silloin, kun TTK-91-koneen speksi ei tarkkuudeltaan riittäisi, seurattaisiin Koksin toimintaa. Auvo Häkkinen ei ollut järin halukas näyttämään Koksin koodia, joten tämä toiminta tuli selvittää yrityksen ja erehdyksen kautta tai asiakkaalta kysymällä.

Projekti alkaa olla ohi. Kun tässä viimeistelemme dokumentaatiota ja valmistelemme levityspakettia, lienee hyvä pysähtyä pohtimaan, mitä puolen vuoden aikana tuli tehtyä. Ja koska menneisyydestä on hyvä oppia, katsotaan, mitä tekisimme nyt toisin. Lopuksi erittelemme toteutetun ohjelmiston puutteita ja listaamme testaajilta, asiakkaalta ja ryhmältä itseltään tulleita jatkokehitysajatuksia.

2 Puoli vuotta tiivistettynä

Projekti aloitettiin maanantaina 19.1. Ensimmäisessä kokouksessa puhuttiin byrokratiaa, harjoitettiin alustavaa tutustumista ja jaettiin vastuualueet. Projektipäällikkö valikoitui suuremmitta hampaiden kiristelyittä yli-innokkuutensa perusteella. Työvaiheille valittiin kullekin oma vastuuhenkilönsä, suunnitteluun kaksi. Seuraavassa kokouksessa päätettiin seurata tuttua ja turvallista vesiputousmallia “hyvin spesifioidun” aiheen nimissä. Aikatauluksi valittiin ns. 4-2-4-malli, jossa vaatimusanalyysille ja suunnittelulle varataan 4 viikkoa, toteutukseen 2 ja testaukseen 4. Pian tämän jälkeen toteutettavan ohjelmiston kooksi arvioitiin 3000-5500 laskutavasta riippuen (alkuperäinen Koksi mahtui 3600 riviin). Käytännössä lopullinen ohjelmisto vei noin 13 000 riviä, josta 8000 rivin ylimäärä ei aivan selity kattavalla kommentoinnilla. Tästä lähes 5000 riviä (GUI*- ja Animator-luokat) on käyttöliittymää ja sen logiikkaa; loppuosa on käyttöliittymäriippumattomampaa. Tännehän toki tultiin tekemään kunnon korviketta Koksille, ei helppojen opintoviikkojen toivossa.

Projektille määriteltiin 10.5. valmistumiseen tähtäävä aikataulu, jossa taisimmekin pysyä

jonkin aikaa. Projektille tehtiin riskianalyysi. Totesimme, että Javan toteutus vaihtelee hieman eri alustoilla varsinkin käyttöliittymän suhteen, ja että käyttöliittymät voivat toimia jokseenkin hitaasti. Alustojen vaihtelu ei varsinaisesti toteutunut, tosin Macintoshilla kosmeettisia poikkeuksia tuntuu heitettävän enemmän – itse sovellus kuitenkin loppujen lopuksi näyttää toimivan. Hitaus sen sijaan toteutui, mikä aiheutti testausdokumentissakin kuvaillun poikkeaman itse asetetusta tavoitteestamme käynnistyä porkkala-koneilla alta viiden sekunnin. Ennalta ehkäisy auttoi moneen riskiin, mutta jäimme esimerkiksi rinnakkaisryhmän kanssa hieman epäyhteensopivaksi sisäisen lukuesityksen suhteen; Koksi käytti kahta esitystä, ja kumpikin ryhmä valitsi yhden ja pysyi siinä. Tämä senkin jälkeen, kun etenkin projektipäälliköt olivat viettäneet tuntikaupalla aikaa yhteisen eAssari-rajapinnan määrittämisessä.

Kaksi hieman odottamatonta aikasyöppöä ilmaantui sotkemaan aikatauluamme. Vaatimukset olivat sinänsä tarkasti määritellyt, mutta käytännössä “toimi kuten Koksi” oli helpommin sanottu kuin tehty. “Reverse engineering” vei aikaa ja poltti hermoja, kun kaikki oudot rajatapaukset piti kaivaa esiin. Toiseksi käyttöliittymän 5000 koodirivistä enemmistö oli lähinnä yhden ihmisen vastuualuetta; sitä ei osattu jakaa suunnittelussa sen erillisempiin osiin, eikä varsinaisesti kyllä suunnitellakaan kovin yksityiskohtaisesti. Jos projekti aloitettaisiin nyt uudelleen, ainakin projektipäällikkö viettäisi varmaankin aluksi aikaa GUI-pikakurssilla ihmettelemässä eri toteutusmahdollisuuksia, jotta Swing-työkalut olisivat jo viimeistään suunnitteluvaiheessa valmiina hyppysissä. Koodimäärän ennalta ymmärtäminen ajaisi sekin todennäköisesti hieman korjattuun työnjakoon, vaikka se sitten tarkoittaisi sitä että kaksi ihmistä tekisi saman luokan kahta eri osaa.

Projektin aikataulua korjattiin seurantakokouksissa. Se kulki lopulta melko tasaisesti viikon jäljessä, ja testausvaihe oli osittain toteutuksen viilausta (käyttöliittymä ja etenkin animaatio eivät kai koskaan sinänsä “valmistu”, mutta toteutusaikataulussa ne eivät kyllä ehtineet edes lähelle; pari muutakin suurempaa luokkaa valmistui jälkijättöisesti). Säikeitä ei suunniteltu mukaan, vaan ne todettiin toteutusvaiheen loppupuolella välttämättömiksi ja lisättiin jokseenkin lennossa – tästä seuranneet ongelmat ovatkin sitten puutelistalla. Hieman ennen testausvaiheen alkua ryhmä päätti käyttää yksikkö- ja osin integrointitestaukseen JUnit-apuohjelmaa. Järjestelmällinen testaus ei ollut ryhmälle juurikaan ennestään tuttua. Vaikka testauksemme jäljiltä emme välttämättä suosittelisikaan Titokonetta minkään avaruussukkulan kriittiseksi osaksi, olemme oppineet melkoisesti testauksen järjestämisestä. Mm. projektipäällikkö osaisi kenties nyt, jos aloittaisi projektin alusta, tehdä testinsä niin, että niitä voisi käyttää pienin muutoksin myös tynkäluokkien, ’stubien’, poistamisen jälkeen. Tyngistä riippuvaiset testit menivät jokseenkin hukkaan, kun niitä ei voinut enää integraatio- ja regressiotestauksessa käyttää.

Ryhmän ilmapiiri on ollut hyvä, ja ryhmätyöskentely on ollut käyntiin päästyään varsin tehokasta. Alussa ryhmätyö oli tehotonta kuuden ihmisen erillään toimimista, jota synkronoitiin kokouksissa. Laadukkaampaan ryhmätoimintaan pääseminen ei ole itsestäänselvyys, mutta ryhmä ylsi alkulämmittelyjen jälkeen ajoittain varsin hyvään yhteistoimintaan. Alkukankeus vain jatkui jokseenkin pitkälle: vaatimusmäärittely jäi enimmäkseen yhden hengen kirjoitustyöksi, kunnes varsin tyrmäävä vaatimusdokumentin FTR-palaute kurssin vastuuhenkilöltä yhdisti ryhmän. Suunnittelussa sama kaava toistui osin, nyt kuitenkin lähinnä siksi, että yhteisen APIn suunnittelu ja varsin kookkaan järjestelmän ra-

kennemuutokset keskittivät tietoa parille ryhmäläiselle, jolloin muiden oli hankala pysyä perillä siitä, mikä tämän päivän arkkitehtuuriversio olikaan, saati kommentoida sitä. Yhdessä tekemiselle ei tahdonnut liietä aikaa ennen kuin sitä opittiin varaamaan kokousten ulkopuolelta; vaikka sähköpostiviestintä oli monessa asiassa riittävää, samassa huoneessa suunnittelu ja koodaaminen vaikutti olevan lopulta huomattavasti tehokkaampaa. Tässä saattaa auttaa myös helpommin havaittava ryhmäpaine, joka tuntuu tehoavan varsin hyvin opiskelijoihin yleisestikin.

3 Puutteita ja jatkokehityksen paikkoja

Testidokumentin liitteenä on ryhmän ulkopuolisten Titokoneen testaajien lausunto siitä, mitä kehitysehdotuksia heillä on antaa. Joitakin niistä on toistettu tässä. Numerointi ei osoita varsinaisesti kriittisyys- tai suosituimmuusjärjestystä, vaan mahdollisen jatkokäsittelyn helpottamiseksi.

3.1 Puutteita

1. Suorituksen lopuksi uudelleenladattaessa koodialuettaan muuttanutta ohjelmaa käyttöliittymän koodinäkymä ei päivity oikein. Itse tietorakenteissa tieto on ajantasaista.

2. Koodia voi muokata sovelluksessa. Joskus usean tiedoston käsittely saman session aikana ja joidenkin muokkaus on johtanut lähdekooditiedostojen ylikirjoittumiseen. Emme saaneet paikallistettua ongelmaa. Todennäköisesti “tallenna”-napin mahdollisen lisäämisen yhteydessä tallennusrutiinit kannattaa tarkistaa ja varmistaa, etteivät säikeet aiheuta ns. *race condition* -ongelmia.

3. Joissakin mm. Compiler-virheissä on hieman harhaanjohtava virheilmoitus:
   muuttuja dc -999999999999999
   muuttuja dc 999999999999999
   ja vastaavat komennot DS:llä ja EQUlla valittavat “invalid opcode”, perässä joko dc, ds tai equ. Kun Integer.parseInt() saa liian suuren luvun merkkijonona, se käyttäytyy samalla tavalla kuin saadessaan satunnaisen merkkijonon. Vastaavasti komennot
   stdin def
   stdout def
   aiheuttavat virheilmoituksen “invalid opcode def”. Näitä voisi tarkentaa.

4. Titokoneen binääriesitys eroaa Koksin muistidumpin binääriosasta. Koksi käytti prosessorissaan 2:n komplementtia myös addr-osan esityksessä, mutta lisäsi dumpissa addr-osaan etumerkkibitin. Päätimme käyttää kaikkialla 2:n komplementtia. Rinnakkaisryhmä puolestaan valitsi kuuleman mukaan päinvastoin, joten positiivisista luvuista koostuneella esimerkillä spesifioitu binääriformaattikaan ei ole negatiivisten lukujen suhteen yhteensopiva ryhmien välillä. Suosittelisimme yhden

esitystavan valitsemista ja siinä pysymistä; koska Javan sisäinen esitys on kahden komplementti, tämä yksinkertaistaisi bittioperaatioita suorittimella. Asiakkaan kanssa kannattanee pitää kunnon palaveri asiasta.

5. Animatorin nopeussäätimen “slow” ja “fast” Animatorissa eivät käänny. Ne lisätään GUI-luokassa, ja käännöksen tukeminen vaati päivityksen lisäystä; ongelma huomattiin vasta koodin jäädytyksen jälkeen.

6. DS-komento alustaa varaamansa muistin nolliksi. Olisi varmaankin opettavaisempaa, että muistissa näkyisivät vanhat arvot.

7. Animatorin rekisterinäkymä ei nollaudu koneen muistin tyhjetessä, vaan vasta kun uusi suoritus aloitetaan.

8. Joissakin koneissa on havaittu, että käyttöliittymä ei välillä piirry uudelleen. Tämä saattaa johtua prosessoritehosta, tai olla Swing-bugi. Emme löytäneet sille selvää syytä.

3.2 Jatkokehitysehdotuksia

Testaajien tekemiä käyttöliittymän jatkokehitysehdotuksia on myös testausdokumentin liitteenä. Totesimme, että vaikka lopun viilauksesta on vaikea päästää irti, jossakin vaiheessa se pitää kuitenkin tehdä ja jäädyttää koodi. Tämän jälkeen saadut ajatukset ja aiemminkin kehitetyt ideat, joiden toteuttamiseen ei aika riittänyt, on kirjattu tähän. Ehkä ne voivat toimia tavoitekeskustelun pohjana seuraavalle ryhmälle.

1. Käännösasetuksia on toivottu ajoasetusten tapaan pikanapeiksi työkalupalkkiin.

2. Stop-napin voisi poistaa käytöstä kun suoritus on jo päättynyt virheeseen, ja tehdä selkeämmäksi että jatka-napin painalluksella päästään lataamaan sovellus uudelleen.

3. “Uudelleenavaus”-nappi voisi olla hyödyksi koodinäkymään pääsemiseksi esimerkiksi kesken ajon. Lisäksi mikäli parempia muokkausmahdollisuuksia lähdetään tukemaan (lähinnä uusien rivien lisäämismahdollisuutta), “uusi” ja “tallenna” -napit lienevät tarpeen. Myös “avaa animaatioikkuna” -nappia on ehdotettu; nyt itse suljetun animaatioikkunan saa uudelleen auki säätämällä animaation ensin pois päältä ja sitten takaisin päälle. Toinen mahdollisuus olisi tulkita ikkunan sulkeminen animaation pois päältä asettamiseksi.

4. Näppäinsyötekentän toivottiin ilmestyvän jotenkin myös animaatioikkunaan. Lisäksi toivottiin kohdistimen automaattista siirtymistä syötekenttään.

5. Valittua muistin kokoa, kieltä ja stdin/stdout-asetuksia on toivottu näkyvämmiksi. Asetusdialogit voisi ehkä yhdistää yhdeksi isoksi. Samoin avatun tiedoston nimi toivottiin näkyvälle paikalle.

6. Komentojen kaksoispistejaoteltu esitys tuntuu vetoavan käyttäjistöön. Ehkä tälle voisi tehdä asetuksen, jolla esimerkiksi voisi valita, mitä esitysmuotoja missäkin tilassa (ja kenties erikseen koodi- ja data-alueella) näytetään – kenties järjestyksineen. Eräs testaaja toivoi erillistä “disassemble”-komentoa demonstroimaan sitä, ettei binäärin mukana tosiaan tule symbolista muotoa, vaan että se voidaan päätellä haluttaessa koodin pohjalta. Tästäkin lienee parasta keskustella asiakkaan kanssa.

7. Asetusvalikossa ei erään testaajan mukaan välttämättä tarvitsisi “set” “configure” ym. lisäsanoja, kun ne ovat jo valmiiksi “options”-valikon alla. Valikon kohdat, joista avautuu dialogi, voisivat kaivata yhtenäisesti “...”-merkintöjä kertomaan, että niistä aukeaa dialogi.

8. Kun suoritus päättyy virheeseen, virheen aiheuttanut rivi voitaisiin kenties korostaa esimerkiksi punaisella. Suorituksen onnistunutta loppumista voisi samoin ilmaista jotenkin näkyvämmin.

9. Virheilmoituksia käännöksen ja suorituksen aikana voisi yhtenäistää; mm. nyt käännösvirheet alkavat sanomalla, että kyseessä on käännösvirhe, kun taas ajonaikaiset virheet eivät tätä mainitse.

10. Peruskäyttöliittymässä tulisi näyttää SP:n ja FP:n lisäksi niiden toiset nimet R6 ja R7, samoin animaatioikkunassa SP ja FP. Tilarekisterin voisi lisätä muodossa tai toisessa myös peruskäyttöliittymään. Lisäksi sen vielä hieman tarkempi spesifikaatio voisi olla hyväksi, esimerkiks rekisterin sisällön kokonaislukuna esittämisen helpottamiseksi.

11. G E L -bittien merkitseminen ja MBR/MAR-tekstien kauniimpi sijoittelu Animatorissa voisi olla eduksi. Testaajat ovat ehdottaneet myös tilarekisterin lisäämistä peruskäyttöliittymään, samoin kuin sen laajentamista kattamaan useampia mm. Tietokoneen toiminnan kolmosluennon kalvoilla käsiteltyjä tilabittejä. Tiedot saa tällä hetkellä lähinnä RunInfoista; ehkä tilarekisterille voisi lisätä kunnolisen tuen itse Processor-luokkaankin.

12. Animaatiota yleensä voisi ehtiessään tarkentaa. Eri laitteet voisi kenties erotella, ja mm. “kontrollilangat” lisätä. Lisäksi SVC-kutsu ei tällä hetkellä tee animaatiossamme mitään, koska sillä voidaan tehdä useita eri asioita. Yleisestikin asiakkaan kanssa ei selvitetty yksityiskohtaisesti sitä, miten käyttöjärjestelmän osuus komennossa tapahtuisi; KJ on varsin piilossa Koksissa, mikä kariutti jo laiteajurin animointitoivomuksenkin.

13. Käyttöohje ja tiedot sovelluksesta näytetään nyt dialogissa, joka näyttää hieman selaimelta muttei kuitenkaan ole sellainen. Esimerkiksi paluunäppäin puuttuu, vaikka se voisi olla tarpeen. “Oikea selain” voisi olla vielä parempi käyttöohjeen näyttämiseen, mikäli mahdollista. Toisaalta ehkä pitäisi myös harkita näkymän palauttamista alkutilaan joka kerta, kun ohje/about-ikkuna avataan, jotta pääsy takaisin lähtökohtaan mahdollistuu. Nyt jos about-ikkunassa klikkaa linkkiä ulospäin, sieltä ei enää pääse takaisin edes avaamalla ikkunan uudelleen; toisaalta käyttöohjetta voisi olla mukava lukea eteenpäin siitä kohdasta, mihin viimeksi jäi...

14. Translations_en-luokka on paketin mukana lähinnä siksi, että laitoksen Java-versio ei vielä nähtävästi suostu vaihtamaan kieltä, jos se tarkoittaa oletuskäännöstiedostoon putoamista. Javan päivittyessä tämä voidaan toivon mukaan poistaa. Sitä ei kannata päivittää; aivan sama tieto on jo itse käännöspyynnössä.

15. Kun asetukset tallennetaan tällä hetkellä, ne menevät Hashtablen palauttamaan epämääräiseen järjestykseen. Mikäli käy ilmi, että käyttäjät muokkaavat asetustiedostoja mielellään, tiedoston alkuperäinen, esimerkiksi kommentoitu muoto olisi kenties parasta säästää niin pitkälti kuin mahdollista.

16. Jos kieleen tehty STDIN/STDOUT-laajennos ottaa tulta, SVC =READ- ja SVC =WRITE -komentoja voisi tarkentaa siten, että myös laitenumero välitetään pinossa.

4 Muita terveisiä jatkokehittäjille

Rinnakkaisryhmän kanssa kiireessä sovittu yhteinen API voi sinänsä riittää eAssariin liittämiseen, mutta jos esimerkiksi otatte staattisen Translator-luokan käännöksenvaihdosmetodit käyttöön, saatte eAssarin puhumaan tarvittaessa suomea myös virheilmoitustensa suhteen. Emme erityisesti suosittele APIn pakkokäyttöä, ellei se näytä tosiaan riittävän kaikkiin tarpeisiin. APIn suunnittelu alkoi kunnianhimoisesti, mutta päättyi ehkä hieman kesken kun opetusteknisistä syistä ryhmät haluttiin takaisin erilleen.