Testausdokumentti

Kaapo - Kaavioiden piirto-ohjelma

Helsinki 1.9.2005

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Ilari Heikkinen
Allan Holsti
Tero Kallioinen
Kristian Ovaska
Mikko Paltamaa
Hannu-Pekka Rajaniemi

Asiakas

Inkeri Verkamo

Johtoryhmä

Juha Taina
Sampo Yrjänäinen

Kotisivu

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

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	1.9.2005	Dokumentti valmis
0.1	15.8.2005	Dokumenttirunko luotu

Sisältö

1 Johdanto

Oops on Helsingin yliopiston tietojenkäsittelytieteen laitoksella toteutettava ohjelmistotuotantoprojekti. Kaapo, eli Kaavioiden piirto-ohjelma on geneerinen, käyttäjän tarpeisiin mukautuva piirtotyökalu. Projektiin liittyvä materiaali on saatavissa ryhmän kotisivulta osoitteesta

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

Tämä dokumentti sisältää raportin ohjelman ensimmäisen julkaistun version testauksesta ja tunnetuista virheistä.

2 Yksikkötestaus

Yksikkötestauksessa on käytetty apuna JUnit-ohjelmaa. Testauksen suunnittelusta poiketen kaikille luokille ei ole kirjoitettu testejä eivätkä kirjoittajat ole aina olleet eri henkilöitä kuin luokan koodaajaat. Useimmissa tapauksissa testit on jätetty kirjoittamatta, koska kyseinen luokka ei sovellu yksikkötestattavaksi kuten esimerkiksi käyttöliittymäluokat, grafiikkaluokat sekä kaavioiden ja elementtejen tyyppiluokat. Monet abstraktit luokat on myös jätetty testaamatta koska ne sisältävät vain hyvin triviaaleja ei-abstrakteja metodeita. Tapauksissa joissa abstrakti luokka implementoi testattavia metodeita on avuksi koodattu ei-abstrakti tynkäluokka (stub class). Tynkä perii abstraktin luokan ja toteuttaa triviaalisti sen abstraktit metodit. Abstraktin luokan ei-abstraktit metodit testataan tämän luokan ilmentymän avulla.

Muita puutteita testauksen kattavuudessa ovat export-luokat sekä ConnectionWrapper-luokka, joka on vain osittain testattu. Export-luokat ovat ulkopuolisia komponentteja eivätkä matalan prioriteetin vuoksi valmiiksi koodattuja. ConnectionWrapper luokkan oikea toiminta on selkeintä GUI:n välityksellä ,jota on myös käytetty sen testaamiseen. Niinpä luokan testaus on ollut lähempänä järjestelmätestauksen menetelmää kuin yksikkötestausta.

2.1 Testien listaus

Seuraavassa on listattu järjestelmän luokat sekä niiden testaustiedot.

Classes	Javadoc	Tests	Outcome	Comment
filemanager.ComponentNotSupportedException	yes	-	-	Exception class
filemanager.ExportPlugin	yes	-	-	Only abstract methods
filemanager.FileManager	yes	yes	ok	-
filemanager.ObjectSerializer	yes	yes	ok	-
filemanager.ToStringExport	-	-	-	Dummy class
filemanager.EPSExport	-	-	-	Not finished
gui.attributeview.AttributeView	yes	-	-	GUI class
gui.attributeview.AttributeViewJLabel	yes	-	-	GUI class
gui.diagramview.DiagramPanel	yes	-	-	GUI class
gui.diagramview.DiagramView	yes	-	-	GUI class
gui.diagramview.DiagramViewJLabel	yes	-	-	GUI class
gui.diagramview.HotSpot	yes	-	-	GUI class
gui.projectview.ProjectView	yes	-	-	GUI class
gui.projectview.ProjectViewCellRenderer	yes	-	-	GUI class
gui.projectview.ProjectViewJLabel	yes	-	-	GUI class
gui.projectview.ProjectViewJTree	yes	-	-	GUI class
gui.projectview.ProjectViewTreeModel	yes	-	-	GUI class
gui.projectview.ProjectViewTreeNode	yes	-	-	GUI class
gui.KaappoGui	yes	-	-	GUI class
gui.ToolPanel	yes	-	-	GUI class
project.Attribute	yes	yes	ok	-
project.Connection	yes	-	-	Mostly abstract methods
project.ConnectionWrapper	yes	some	ok	-
project.Diagram	yes	yes	ok	Stub test
project.DiagramComponent	yes	-	-	Mostly abstract methods
project.DiagramComponentWrapper	yes	yes	ok	Stub test
project.DiagramWrapper	yes	yes	ok	-
project.Element	yes	-	-	Mostly abstract methods
project.ElementWrapper	yes	yes	ok	-
project.Project	yes	yes	ok	-
project.ProjectComponent	yes	yes	ok	Stub test
project.graphics.AttributePanel	yes	-	-	Stub test
project.graphics.ConnectionGraphics	yes	-	-	GUI class
project.graphics.DCGraphics	yes	yes	ok	Stub test
project.graphics.DirectTextArea	yes	-	-	GUI class
project.graphics.FilledArrowHead	yes	-	-	GUI class
project.graphics.GenericAttributePanel	yes	-	-	GUI class
project.graphics.LineArrowHead	yes	-	-	GUI class
project.graphics.LineHead	yes	-	-	Stub test
project.graphics.SolidGraphics	yes	-	-	GUI class
projectmanager.command.AbstractEdit	Yes	-	-	Abstract class
projectmanager.command.AbstractUndoableEdit	Yes	-	-	Abstract class
projectmanager.command.AddConnectionEdit	Yes	yes	ok	-
projectmanager.command.AddElementEdit	Yes	yes	ok	-
projectmanager.command.AddToSelectionEdit	Yes	yes	ok	-
projectmanager.command.ChangeDiagramNameEdit	Yes	yes	ok	-
projectmanager.command.ChangeProjectNameEdit	Yes	yes	ok	-
projectmanager.command.ChangeVisibleDiagramEdit	Yes	yes	ok	-
projectmanager.command.Edit	Yes	-	-	Interface class
projectmanager.command.EditFailedException	Yes	-	-	Exception class
projectmanager.command.ModifyDComponentAttribute	Yes	yes	ok	-
projectmanager.command.MoveConnectionEditPoint	Yes	yes	ok	-
projectmanager.command.MoveDComponentEdit	Yes	yes	ok	-
projectmanager.command.MoveDiagramUpEdit	Yes	yes	ok	-
projectmanager.command.MoveDiagramDownEdit	Yes	yes	ok	-
projectmanager.command.NewDiagramEdit	Yes	yes	ok	-
projectmanager.command.RemoveDComponentEdit	Yes	yes	ok	-
projectmanager.command.RemoveDiagramEdit	Yes	yes	ok	-
projectmanager.command.ResizeElementEdit	Yes	yes	ok	-
projectmanager.command.RemoveFromSelectionEdit	Yes	yes	ok	-
projectmanager.command.SelectionEdit	Yes	yes	ok	-
projectmanager.ProjectManager	yes	yes	ok	-
projectmanager.Selection	yes	yes	ok	-
types.diagramComponents.Actor	-	-	-	Type class
types.diagramComponents.ActorGrpahics	-	-	-	Type class
types.diagramComponents.Association	-	-	-	Type class
types.diagramComponents.AssociationGraphics	-	-	-	Type class
types.diagramComponents.Dependency	-	-	-	Type class
types.diagramComponents.DependencyGraphics	-	-	-	Type class
types.diagramComponents.Generalization	-	-	-	Type class
types.diagramComponents.GeneralizationGraphics	-	-	-	Type class
types.diagramComponents.Text	-	-	-	Type class
types.diagramComponents.TextGraphics	-	-	-	Type class
types.diagramComponents.UseCase	-	-	-	Type class
types.diagramComponents.UseCaseGraphics	-	-	-	Type class
types.diagrams.DataFlowChart	-	-	-	Type class
types.diagrams.UseCaseDiagram	-	-	-	Type class
types.TypeManager	yes	yes	ok	-
Kaapo	yes	yes	ok	-
Observable	yes	yes	ok	Stub test
Observer	yes	-	-	Interface

3 Integrointitestaus

Suunniteltua integrointitestausta ei viety läpi. Syynä tähän on toteutuksessa käytetty menetelmä, jossa ohjelman eri osia on evoluutiomallin mukaisesti rakennettu ja suunniteltu eteenpäin kokonaisuutena. Rajapintoja ei myöskään toteutettu yksi yhteen suunniteltujen kanssa. Ohjelman osat toimivat hyvin läheisessä yhteistyössä eivätkä ole helposti jaettavissa osajärjestelmiin. Ryhmän yhteinen päätös oli, ettei integrointitestaus ole mielekästä. Rajapintojen oikea toiminta varmennetaan osittain yksikkötesteissä, joissa niitä samoista syistä ei voida sivuttaa, sekä lopullisesti ohjelman järjestelmätesteissä.

4 Järjestelmätestaus

Järjestelmätestauksessa ohjelman toimintaa testattiin suunnitteludokumentissa määriteltyjen laajennettujen käyttötapausten perusteella. Testauksessa huomioitiin ohjelman rajoitukset, joita syntyi suunnittelun ja toteutuksen välisistä eroista. Testit vietiin läpi niiltä osin kuin ohjelman toteutus sen salli.

4.1 Testaus ja tulokset

Seuraavassa esitellään jokaisesta käyttötaupauksen testaus ja tulokset.

LKT1 Kaavion luonti

Kaavion luonnissa käyttäjä valitsee aluksi haluamansa kaaviotyypin ja sen jälkeen luo kaavion valitsemalla ”add”-painikkeen. Järjestys eroaa suunnitellusta ja näin ollen testauksessa ei voitu toimia käyttötapauksessa kuvatulla tavalla. Käyttötapauksessa pitäisi testata myös mm. luokkakaaviolla, mutta ohjelman tämän hetkisessä versiossa ei ole toteutettuna luokkakaaviotyyppiä, joten sitä ei voida testata. Ohjelman toiminta oli moitteetonta huomioiden käyttötapauksen ja ohjelman toiminnan välisen eron.

LKT2 Elementin lisääminen

Elementin lisääminen tapahtuu käyttötapauksessa kuvatulla tavalla (kts. Suunnitteludokumentti). Ainoa poikkeus käyttötapauksen läpivientiin on se että piirtoalustalla ei ole maksimiarvoa vaan se kasvaa tarpeen mukaan. Eli jos elementti lisätään tämän hetkisen piirtoalustan oikeaan alakulmaan niin piirtoalusta kasvaa elementin tarvitsemalla tilalla.

LKT3 Elementin poistaminen

Ohjelman toiminta kaikissa muissa tilanteissa paitsi ’elementti useassa kaaviossa’ on suunnitteludokumentissa kuvatun päätöstaulun mukaista. Tilannetta elementti on useassa kaaviossa ei voida testata, koska ohjelma ei tällä hetkellä tue kyseistä toimintoa.

LKT4 Elementin koon muuttaminen

Koon muutos toimii odotetulla tavalla kaissa tilanteissa paitsi elementin koon muuttuessa maksimikoon yli. Tilannnetta ei voida testata koska ei ole käytettävissä elementtiä, jolla olisi maksimikoko ja sen kokoa voisi muuttaa.

LKT5 Elementin liikuttaminen

Toimii odotetusti kaikissa tilanteissa.

LKT6 Elementin attribuuttien arvojen muuttaminen

Ohjelma toimii yhdellä elementillä oikein stringien tapauksessa, inttejä ei pääse testaamaan puuttuvien attribuuttien takia. Elementti ei voi esiintyä tällä hetkellä useissa kaavioissa, joten kyseistä tapausta ei päästä testaamaan. Monen elementin ollessa valittuna attribuutteja ei voi muokata.

LKT7 Yhteyden lisääminen

Ohjelma toimii kuvatulla tavalla kun molemmissa päissä on yhteys, alkupää on olemassa, välissä on elementti ja ei elementtejä. Ohjelma ei tue syntaksin tarkistusta, eikä yhteyksillä ole sääntöjä. Mahdoton toteuttaa on myös toiminto, jossa alkupäätä ei määritellä mutta loppupää annetaan, koska ohjelma piirtää yhteyden heti jos ensimmäinen klikkaus ei osu elementtiin. Toisaalta toiminto voidaan suorittaa kun ensin lisätään yhteys, joka ei ole kiinni missään ja sen loppupää vedetään haluttuun elementtiin.

LKT8 Yhteyden poistaminen

Kaikki käyttötapaukset voidaan suorittaa onnistuneesti.

LKT9 Yhteyden uudelleen asemointi

Kaikki käyttötapaukset voidaan suorittaa onnistuneesti.

LKT10 Yhteyden attribuuttien arvojen muuttaminen

String tyyppisillä arvoilla kaikki ok. Huomioitavaa on että yhteys ei voi esiintyä useissa kaavioissa eikä int arvoisia attribuutteja ole.

LKT11 Murtoviivan kulkureitin muuttaminen

Käyttötapaukset toimivat oikein ohjelman sallimissa rajoissa, eli yhdellä muokkauspisteellä. Ohjelmassa ei ole käytettävissä murtoviivaa, jossa olisi kaksi muokkauspistettä. Murtoviivan toimintaan liittyy ratkaisemattomia ongelmia (kts. 5 ja Toteutusdokumentti).

LKT12 Tekstin lisääminen kaavioon

Kaikissa tuetuissa kaavioissa tekstin lisääminen onnistuu ilman ongelmia. Ongelmia aiheuttaa kentän tyhjänä olo, tällöin jos valinta on poistunut alueelta, niin tekstin kirjoituskohdan uudelleen valinta on vaikeaa.

LKT13 Ryhmän valitseminen

Kaikki toimivat oikein, paitsi että yhteysryhmiä voi liikutella ja valinta tapahtuu ctrl pohjassa. Toisaalta yhteysryhmien liikuttelussa voi tulla erikoisia tilanteita jos yhteys on tavallaan kiinni toisessa yhtydessä.

LKT14 Alueen rajaaminen

Valinta toimii kaikissa tilanteissa oikein.

LKT15 Undo

Toimii ainakin perustilanteissa oikein

LKT16 Redo

Toimii ainakin perustilanteissa oikein

LKT17 Tallennus vanhaan tiedostoon

Toimii kaikissa tilanteissa.

LKT18 Tallennus uuteen tiedostoon

Toimii kaikissa tilanteissa.

LKT19 Tiedoston avaus

Ohjelma toimii odotetusti kaikissa tilanteissa. Avaamisessa on huomioitava ettei ohjelmaan voi välttämättä avata aiemmin tallennettua kaaviota, jos tallennetun kaavion ja ohjelman versio eivät ole yhteensopivat.

5 Tunnetut virheet

Tässä luvussa on lueteltu ohjelmiston tunnetut virheet. Joidenkin virheiden korjaaminen onnistuu parhaiten muuttamalla arkkitehtuuria tai rajapintoja; näiden virheiden kohdalla on korjausehdotus.

5.1 Elementin koon muuttaminen ja yhteydet

Kun elementin (esim. Use Case) koko muuttuu, siinä olevien yhteyksien päätepisteet eivät pysy synkroonissa (eivät liiku).

Ongelma liittyy siihen, että yhteyksien päätepisteet ovat absoluuttisia pikselikoordinaatteja, ja näiden muuttaminen ei ole suoraviivaista kun elementin koko ja muoto muuttuvat. Hyvä ratkaisu olisi ilmaista päätepisteet jonkinlaisina suhteellisina koordinaatteina; tällöin yhteyspisteen päivittäminen olisi helpompaa. Esim. ympyrä-elementillä suhteellinen päätepiste $[0..1]$ voisi olla tietty paikka ympyrän kehällä (0=suoraan ylös, 0.5=suoraan alas, jne).

Suhteellisista päätepisteistä olisi myös se hyöty, että esim. laatikkomallisen elementin sivun keskelle lisätty yhteys pysyy sivun keskellä (yhtä kaukana kulmista), vaikka laatikon koko muuttuu.

5.2 Katkoviivayhteyden piirto

Jos viivan tyyppi on katkoviiva (*DASHED*), `LineArrowHead`-tyypin nuolet piirtyvät normaalilla viivalla. Use Case -kaavion yhteys Dependency on tällainen yhteystyyppi. Nuolenpään yhteysviivan mukainen osa tulisi piirtää katkoviivana.

5.3 Murtoviivayhteyden muokkaus

Kun murtoviivayhteyden polun segmenttiä muokataan tarkistaa ohjelma segmentin suunnan vertailemalla sen alku- ja loppupistettä. Mikäli nämä ovat samat tutkii ohjelma viereisen segmentin pisteitä. Koodi on tältä osin puutteellinen sillä jos viereisen segmentin alku- ja loppupiste ovat samat niin ohjelma ei jatka tarkastusta seuraavaan segementtiin. Lopputuloksena on että jos kaksi viereistä segementtiä ovat molemmat nollan pituisia tulkitsee ohjelma niiden sunnan väärin puolet ajasta. Tilanne olisi mahdollista korjata muokkaamalla olemassa olevaa algoritmia niin että se tarkistaa niin monta yhteyden segmenttiä kuin on tarpeen. Toinen mahdollisuus olisi säilyttää ConnectionWrapper luokassa tieto yhteyden ensimmäisen segmentin suunnasta josta olisi helppo selvittää halutun segmentin suunta sen indexin perusteella.

5.4 Luokka project.graphics.SolidGraphics

Kun `SolidGraphics`-luokkaa käyttävän elementin (esim. Use Case) kokoa yrittää pienentää vasemmanpuoleisesta tai ylemmästä hot-spotista, mutta elementtiä ei voi pienentää pienemmäksi kuin se jo on, elementti liikkuu joskus oikealle tai alas. Liikkuminen ei tapahdu aina vaan tapahtuu hiirtä nopeasti liikuttamalla.

Bugi liittyy siihen, että `SolidGraphics` määrää `updateDCGraphics`-metodinsa sisällä elementin (minimi)koon eivätkä ulkopuoliset (esim. GUI) pääse siihen varsinaisesti vaikuttamaan. Paras tapa korjata bugi on tehdä rajapinta, joka pyytää elementin pienimmän mahdollisen koon (vrt. Swingin `minimumSize`). Tällöin ulkopuoliset tietävät, mihin kokoon elementin saa pienentää, eivätkä yritä pienentää liian pieneksi. Rajapinta voi tulla esim. `DiagramComponent`-luokalle.

Piirtämiseen liittyvä virhe: kun elementin (esim. Use Case) tekstikenttään lisää tekstiä piirtoalustalla ja elementin koko kasvaa, siniset hot-spot-neliöt piirtyvät väärin (vanhat jäävät piirtoalustalle). Näkymä korjautuu kun elementiltä katoaa fokus.

Ellipsiin ja pyöristettyyn suorakaiteeseen liittyvä virhe: teksti voi joskus piirtyä elementin reunojen yli. `SolidGraphics`-luokassa olisi tarkoitus laskea matemaattisesti sellainen elementin koko, että teksti mahtuu kokonaan reunojen sisään (halutulla marginaalilla). Em. muodoille ei (tietoisesti) ole vielä tehty matemaattisesti tarkkaa tarkistusta.

5.5 Undo/redo

Undo-redu-ketju ei aina toimi täysin oikein. Eräs virhe liittyy kaavion poistamiseen:

1. lisää Use Case -kaavio
2. lisää Data Flow -kaavio
3. poista Data Flow -kaavio klikkaamalla oikealla hiiren napilla kyseisen kaavion päällä

4. valitse Undo
5. valitse Redo

Data Flow -kaavio ei tule entiselle paikalleen projektipuussa (viimeiseksi), vaan ensimmäiseksi.

5.6 Yhteysviivan ja elementin leikkauspisteen laskeminen

Kun kahden elementin (esim. Use Case) välille vedetään yhteys, ohjelma laskee yhteyspisteet siten, että ne ovat molempien elementtien reunalla. Joskus päätepiste ei jää loppuelementin reunalle, vaan menee virheellisesti reunan ”läpi” elementin sisään. Virhe on hyvin harvinainen ja vaikea toistaa.

6 Yhteenveto

Järjestelmätestejen onnistunut läpivienti osoittaa ohjelman toimivan sille esitettyjen vaatimusten mukaisesti. Integrointitestejen puuttuminen ei tarkoita etteikö rajapintoja olisi tullut testattua. Ohjelman rakenteen vuoksi rajapintojen toimivuutta kokeiltiin jo yksikkötestauksessa. Monet yksikkötestauksessa löydetyt virheet ja ongelmat sijaitsivat jossain muualla kuin testattavassa luokassa, koska testit kirjoitettiin jokseenkin satunnaisessa järjestyksessä riippuvuuksien monimutkaisuuden suhteen.

Yksikkötestejen kattavuus selittänee osittain miksi järjestelmätestaus ei löytänyt yhtään uutta virhettä. Yleisesti testeissä löydettiin enemmän puutteita kuin varsinaisia virheitä. Suuri osa virheistä ja omituisesta käyttäytymisestä löydettiin koodaajien omatoimisella ohjelman kokeilulla, siis beta-testauksella. Tämä lienee jokseenkin normaalia monimutkaisia käyttöliittymiä sisältävissä ohjelmissa.

Koodaajien palaute testeistä oli se, että niiden olemassaolosta oli hyötyä. Tieto siitä, että oma koodi testataan toisten toimesta motivoi tarkempaan työhön. Läpi menneet testit toivat myös itsevarmuutta oman koodin pätevyydestä, varsinkin jos luokkaan joutui muista syistä koskemaan vielä testien kirjoittamisen jälkeen.