Oliokielet ja oliokeskeinen lähestymistapa

Markku Sakkinen

Yleistä

Oliolähestymistapaan on laitoksessamme kiinnitetty viime vuosina varsin paljon huomiota. Sitä on lisätty useiden kurssien sisältöön läpäisyperiaatteella, ja opetusohjelmaamme on tuotu uusiksi cl-tason vakiokursseiksi "oliokeskeinen tietojärjestelmien rakentaminen" sekä "olio-ohjelmointi". Kaikille pääaineopiskelijoillemme on jompi kumpi näistä pakollinen, suuntautumisesta riippuen.

Muutamat laitoksemme tutkimushankkeet ovat jo useiden vuosien ajan käyttäneet oliokeskeisiä menetelmiä ja välineitä intensiivisesti. kuten niiden kuvauksista varmasti ilmenee. Useita laitoksen opettajia ja tutkijoita (minäkin alkuvaiheessa) oli mukana myös vuosina 1992 - 1993 JOO-hankkeessa (Jyväskylä Object Oriented), jossa edistettiin oliomenetelmien käyttöä jyväskyläläisissä atk-yrityksissä ja niiden yhteistyötä näissä asioissa.

Oliokeskeinen lähestymistapa selvästi itse tutkimuksen pääkohteena on Jyväskylän yliopistossa toistaiseksi ollut (aloittamisaikajärjestyksessä) minulla, tohtoriopiskelija Antero Taivalsaarella ja professori Juhani Iivarilla. Taivalsaari siirtyi väitöskirjansa valmistuttua Nokia Oy:n tutkimuskeskukseen vuoden 1994 alussa ja Iivari Oulun yliopistoon vuoden 1996 alussa. Koska tässä kirjoituksessa on kiinnostavaa keskittyä laitoksen juuri nyt ja lähitulevaisuudessa aktiiviseen tutkimukseen, jätän näiden tutkijoiden työn lyhyelle maininnalle. Kirjoituksen loppuosassa keskityn omien tutkimusteni esittelyyn.

Minä olen tietääkseni Suomen ensimmäisiä varsinaisia olio-ohjelmoinnin tutkijoita. Selvästi varhaisempi pioneeri on ainakin Juha Vihavainen (Helsingin yliopisto), joka toteutti oman Mode-kielensä ensimmäisen version jo vuonna 1984. Myös jotkut tekoälyn tutkijat ovat jo pitkään harrastaneet olio-ohjelmointia.

Antero Taivalsaari oli erityisen kiinnostunut ns. prototyyppipohjaisista oliokielistä; hän suunnitteli ja toteutti oman, Forth-pohjaisen kielen nimeltä Kevo. Jo väitöskirjassaan hän laajensi aluettaan, kuten sen nimikin ilmaisee (suomennettuna): "Kriittinen näkemys periytymisestä ja uudelleenkäytettävyydestä olio-ohjelmoinnissa". Tämä työ sai Tietotekniikan Tutkimussäätiön vuoden 1994 väitöskirjapalkinnon. Sen lyhyt suomenkielinen tiivistelmä on tutkimusesittelynä Tietojenkäsittelytiedelehden numerossa 5 (tammikuu 1994).

Juhani Iivarin pitkäaikainen analyysi- ja suunnittelumenetelmien tutkimus lienee monien Sytykkeen lukijoiden tiedossa. Viime vuosina se on kohdistunut erityisesti oliokeskeisiin menetelmiin. Suhteellisen lyhyenä Jyväskylän-kautenaankin Iivari ennätti julkaista useita kirjoituksia tästä aihepiiristä.

C++:n ongelmat

Vuoden 1986 lopulla Heikki Laitinen (nyttemmin VTT:ssä) ja minä huomasimme, että AT&T:n Bellin laboratorioissa oli kehitetty C:hen perustuva uusi ohjelmointikieli C++, jossa oli muun muassa "oliosuuntautuneita" ominaisuuksia.

Koska kielen kääntäjän ensimmäinen versio oli saatavissa varsin muodolliseen hintaan, asia vaikutti tutustumisen arvoiselta. Aloin harrastaa läheisesti Airi Salmisen dokumenttimalliin ja dokumenttitietokantoihin liittyvää tutkimusta, ja pidin C++:aa mahdollisena toteutusvälineenä.

Kokeiltuani C++:aa jonkin verran kävi niin, että väline muuttuikin tutkimuskohteeksi. Kielessä oli monia myönteisesti mielenkiintoisia ominaisuuksia mutta myös paljon huonoja puolia ja ongelmia. Ennen kaikkea liian vahva sitoutuminen C:hen, joka on nykypäivän näkökulmasta varsin alhaisen tason kieli, tuntui olevan ristiriidassa oliokeskeisyyden vaatimusten kanssa. Näistä tarkasteluista syntyi artikkeli "On the darker side of C++", joka hyväksyttiin toiseen, Oslossa 1988 pidettyyn ECOOP-konferenssiin (European Conference on Object-Oriented Programming).

En ole suhtautunut C++:aan yksisilmäisen kielteisesti. Artikkelissa "Comments on the "Law of Demeter and C++" (ACM SIGPLAN Notices, joulukuu 1988) pyrin täsmentämään noiden olio-ohjelmoinnin tyylisääntöjen (jotka ovat sittemmin tulleet varsin yleisesti tunnetuiksi) soveltamista C++:n erikoisominaisuuksiin. Kolmannen ECOOP:in (Nottingham 1989) esitelmässä "Disciplined inheritance" taas totesin C++:n periytymisen periaatteet monia muita kieliä paremmiksi.

Vuosien varrella C++ on muuttunut paljon. Vuonna 1991 vaikutti siltä, että useita asioita oli korjattu ja lisätty siihen tapaan, kuin olin toivonutkin (esim. ECOOP'88-esitelmässäni).

Toisaalta jotkin perusvirheet, kuten taulukoiden ja osoittimien käsittely, olivat selvästi jääneet pysyviksi, ja olin aikaa myöten havainnut myös sellaisia ongelmia, joita en vielä 1988 ollut huomannut. Kieleen lisätyissä uusissa ominaisuuksissa oli myös uusia pulmia, ja lisäksi C++ oli jo paisunut hyvin laajaksi, mutkikkaaksi ja vaikeasti omaksuttavaksi kieleksi. Näitä näkökohtia esitteli artikkeli "The darker side of C++ revisited" (Structured Programming 4/1992).

Periytymisestäkin löysin vuoden 1989 jälkeen yhä enemmän ongelmia. Niiden selvittely tuntui aiheelliselta tehdä erillisessä artikkelissa, "A critique of the inheritance principles of C++" (Computing Systems, talvi 1992). Siinä on tarkat ehdotukset, miten muutamat, varsinkin moniperintää koskevat seikat pitäisi määritellä paremmin ja johdonmukaisemmin. Valitettavasti C++:n entiset periaatteet ovat jääneet voimaan korjaamattomina. Siksi neuvoni C++-ohjelmoijille on edelleen se, että ainakin ns. haarautuvaa moniperintää (jossa saman luokan kahdella yliluokalla on yhteinen esivanhempi) pitää kaikin keinoin välttää!

Niinkin viattoman tuntuinen asia kuin olioiden alustaminen osoittautui yllättävän hankalaksi moniperinnän yhteydessä. Suurelta osalta ongelmat ovat kuitenkin väistämättömiä eivätkä johdu C++:n suunnittelussa tehdyistä ratkaisuista. Niistä kirjoitin artikkelin "How should virtual bases be initialized (and finalized)?" (C++ Report, maalis-huhtikuu 1993), jossa ehdotin myös muutamia parannuksia C++:n periaatteisiin.

Vuonna 1992 en vielä ottanut kantaa C++:n geneerisyyteen ('templates') enkä poikkeustenkäsittelyyn, koska ne olivat kielessä uusia ja vielä vakiintumattomia ominaisuuksia. Myöhemmin on ilmennyt, että ainakin C++:n geneerisyys on todella kehnosti määritelty verrattuna moniin varhaisempiin esikuviin (esim. Adaan ja Eiffeliin). Nykyään kieli tuntuu edelleen paisuvan, ja uusien ominaisuuksien vuorovaikutuksia kaikkien entisten ominaisuuksien kanssa on varmasti hyvin vaikea selvittää.

Viime aikoina en ole enää seurannut tarkkaan C++:n uusinta kehitystä. Koska on ilmeistä, että kielen pahimpiakaan perusvirheitä ei enää pystytä korjaamaan, kaikkien yksityiskohtien tutkiminen ei motivoi. Yksi perusvirhe (minun mielestäni) on sentään osittain korjattu viimeaikaisissa standardiehdotuksissa: olion todellisen luokan ajonaikainen tutkiminen on nyt joissakin tapauksissa mahdollista.

C++:n nykyistä vallitsevaa asemaa pidän hyvin masentavana. Siksi on mielestäni myönteistä, että Javan suosio tuntuu nousevan hyvin nopeasti. Vaikka Javakin perustuu C:hen, sen suunnittelussa on rohjettu luopua joistakin pahimmista menneisyyden jäänteistä, ennen kaikkea osoitinaritmetiikasta. Itse kieleen on otettu ainakin alkeellinen rinnakkaisuus (säikeet), mutta toisaalta siitä puuttuu mm. geneerisyys. Mainoslauseita Javan yksinkertaisuudesta on vaikea uskoa, kun uusi referenssikäsikirja (Gosling, Joy ja Steele) on yli 800 sivun paksuinen. Lähiaikoina aion analysoida Javaa yhdessä muutaman opiskelijan kanssa.

Yleisempi periytymisen tutkimus

Oliokeskeisen ohjelmoinnin keskeisissä periaatteissa ja mekanismeissa tuntuu olevan ongelmia mm. siksi, että niitä on ohjelmointikieliä ym. järjestelmiä suunniteltaessa ehkä pidetty intuitiivisesti liian selvinä. Periaatteiden selvittäminen on tärkeää sekä kielten kehittämiseksi että olemassa olevien kielten käyttämiseksi oikealla tavalla.

Ehkä ongelmallisin asia olio-ohjelmoinnissa on periytyminen eli perintä; se on myös tekijä, joka selvimmin erottaa oliokeskeiset kielet useimmista muista nykyaikaisista ohjelmointikielistä. Sillä tuntuu välttämättä olevan useita erilaisia merkityksiä ja käyttötapoja, mutta aivan ilmeisesti kirjallisuudessakin on usein harrastettu perintää virheellisesti. Lisäksi periytymissuhteiden merkitystä ohjelmien rakenteessa ja oliomalleissa korostetaan usein aivan liikaa, esim. koostamissuhteiden kustannuksella.

Varsinkin moniperintä (luokalla voi olla useita välittömiä yliluokkia) aiheuttaa hankalia pulmia, etenkin nimikonflikteja, jotka on eri kielissä yritetty ratkaista monilla eri tavoilla. Jo yllä mainitussa ECOOP'89-artikkelissa tutkin ja vertailin näitä ratkaisuja. Siinä pyrkimykseni oli selittää periytyminen mahdollisimman pitkälle koostamisen avulla.

Moniperinnän tunnetuimman formaalin mallin on esittänyt Luca Cardelli alun perin vuonna 1984, etupäässä funktionaalisia kieliä varten. Muutamat oliokielet, esim. Eiffel, Modula-3 ja O2C, noudattavat ainakin jossakin määrin tätä mallia. Usein on jäänyt huomaamatta, että Cardellin malli perustuu oleellisesti erilaisiin lähtökohtiin kuin oliokielten "valtavirtaus" (Simula, Smalltalk, C++, monet Pascalin laajennukset jne.). Näiden periytymismallia voi kutsua alioliokeskeiseksi, Cardellin mallia taas attribuuttikeskeiseksi.

Olen jo pitkään uskonut, että myös alioliooperiaatteella voidaan ratkaista moniperinnän ristiriidat ja muut ongelmat johdonmukaisesti ja intuitiivisesti tyydyttävällä tavalla. Tämä työ odottaa kuitenkin vielä viimeistelyään.

Oliotietokannat ja ODMG-93

Oliokeskeisessä lähestymistavassa tietokantojen ja ohjelmien välillä ei yleensä ole leveää kuilua vaan ohjelmointikielten ja tietokantakielten välinen raja on epämääräinen. Oliotietokantahan ei ole passiivinen tietovarasto vaan sisältää (ainakin periaatteessa) myös olioluokkien operaatioita. Oliotietokantoihin liittyvistä asioista minua ovat kiinnostaneet erityisesti tietokantakielet ja oliomallit. Niissä on usein ratkaistu paljon "tavallisia" olio-ohjelmointikieliä paremmin myös sellaisia seikkoja, jotka eivät liity olioiden pysyvyyteen -- esim. koosteolioiden käsittely.

Vuosina 1993 - 1994 olin hiukan yli vuoden Frankfurt am Mainin yliopistossa tutkijana ESPRIT-hankkeessa nimeltä GOODSTEP (General Object-Oriented Database for Software Engineering Processes), professori Roberto Zicarin johtamassa ryhmässä. Projektissa kehitettiin geneerisen ohjelmistotuotantoympäristön prototyyppi, jossa kaikki dokumentit talletetaan hienorakenteisina olioina tietokantaan. Minun osuuteni koski pääasiassa O2C-tietokantakielen staattisen tyyppiturvallisuuden tarkistamista. Siinä esiintyvät ongelmat eivät johdu tietokantaominaisuuksista vaan koskevat myös esim. Eiffel-ohjelmointikieltä.

Lukuvuonna 1995 - 1996 olin vierailevana professorina Linzin yliopistossa Itävallassa, sikäläisen tietojärjestelmien professorin Gerti Kappelin kutsumana. Hänen ryhmänsä tärkeimpiä tutkimusaiheita ovat aktiiviset oliotietokannat, perusmekanismina tavanomaiset ECA-säännöt (event - condition - action). Osallistuin tämän mallin kahden uuden kehityssuunnan tutkimiseen. Vanhempi niistä on sääntömallien (rule patterns) rakentaminen: aktiivisen tietokannan kaavan määritteleminen yksittäisten sääntöjen avulla on työlästä ja virhealtista. Tästä aiheesta meillä oli esitelmä "From rules to rule patterns" CAiSE'96-konferenssissa Heraklionissa.

Uusin tutkimussuunta, jonka osalta yhteistyöni Linzin ryhmän kanssa jatkuu edelleen, on monivanhempaiset sisäkkäiset trans_aktiot. Tällä idealla pyritään ensi sijassa selventämään monimutkaisten (useista alkeistapahtumista koostuvien) tapahtumien (liipaisimien) käsittelyä, mutta se voi olla käyttökelpoinen myös muiden kuin sääntöjen liipaisemien trans_aktioiden yhteydessä. Aiheesta on pidetty esitelmät "Multi-parent subtransactions: covering the transactional needs of composite events" ATMA-työpajassa (Goa 1996) sekä "A transaction model for handling composite events" ADBIS'96-symposiossa Moskovassa.

Vuoden 1995 lopulla innostuin tutkimaan ODMG:n (Object Data Management Group) julkaisemaa oliotietokantastandardia ODMG-93, josta ilmestyi jo kolmas versio (1.2) kirjana. Jonkinlaista yhteensopivuutta takaava standardi on erittäin tärkeä sekä oliokeskeisten tietokannan hallintajärjestelmien valmistajille että niiden käyttäjille. Valitettavasti standardi on vielä teknisesti kovin ristiriitainen ja puutteellinen.

Olen lähettänyt ODMG:lle varsin seikkaperäisiä huomautuksia ja parannusehdotuksia etupäässä standardin oliomallista ja olionmäärittelykielestä (ODL). Konsortio ei kuitenkaan tunnu kovin yhteistyökykyiseltä ulkopuolisten avustajien suuntaan: en ole saanut sieltä mitään todellista palautetta. Sen sijaan yhteistyötä ODMG-93:n analysoinnissa on syntynyt professori Kazimierz Subietan (Puolan tiedeakatemia, viime talvena vielä Kioton yliopistossa) kanssa. Hän on puolestaan keskittynyt ODMG:n kyselykielen (OQL) ruotimiseen. Subietahan on itse viime vuosina kehittänyt erittäin lupaavaa pinopohjaista lähestymistapaa, jolla kyselykieli voidaan luontevasti laajentaa täysipainoiseksi ohjelmointikieleksi.

Olisin hyvin kiinnostunut yhteistyöstä sellaisten suomalaisten tutkijoiden ja yritysten kanssa, jotka analysoivat ODMG-93:a ja yrittävät vaikuttaa sen tulevaan kehitykseen.

Kotimaisia ja kansainvälisiä yhteyksiä

Suomessa minulla on ollut vuosien varrella yhteistyötä edellä mainittujen tutkijoiden lisäksi varsinkin Kai Koskimiehen (Tampereen yliopisto) kanssa. Reino Kurki-Suonion ym. Tampereen teknillisen korkeakoulun tutkijoiden kanssa on syntynyt yksi yhteinen artikkeli heidän DisCo-kielestään. Tutkimusharrastukseni alkuvuosina toimi Suomen Tekoälyseuran käynnistämä oliokerho, jonka kokouksissa käytiin joskus kiinnostavia keskusteluja.

Luonnollisista syistä tutkimuskontaktit ovat olleet etupäässä ulkomaisia. Vahvoja hyödyllisiä yhteyksiä on ollut aikaisemmin mainittujen henkilöiden lisäksi mm. seuraaviin: Karl Lieberherr ja hänen ryhmänsä (Northeastern-yliopisto, Boston), Oscar Nierstrasz (ennen Geneven, nyt Bernin yliopisto) Peter Grogono (Concordia-yliopisto, Montreal), Douglas Lea (New Yorkin osavaltion yliopisto, Oswego), Timothy Budd (Oregonin osavaltion yliopisto), Ole Lehrmann Madsen ja Jörgen Lindskov Knudsen (Århusin yliopisto), John Skaller (Maxtal plc, Sydney), Bent Bruun Kristensen (Ålborgin yliopisto).

ECOOP-konferenssisarjassa olen ollut mukana vuodesta 1988 alkaen, jo useita vuosia myös ohjelmatoimikunnan jäsenenä. Sen nimeä ei pidä käsittää liian kirjaimellisesti kahdessakaan suhteessa. Eurooppalaisuus tarkoittaa vain tapahtumapaikkaa: esitelmiä ja osan_ottajia tulee runsaasti muualtakin maailmasta. Sisältö puolestaan ei rajoitu pelkkään ohjelmointiin vaan käsittää periaatteessa kaikki oliolähestymistavan alueet, mm. analyysin ja suunnittelun.

Viime kesänä ECOOP'96 pidettiin Linzissä; olin aktiivisesti mukana järjestelytoimikunnassa. Ensi vuoden ECOOP'97 saatiin järjestettäväksi Jyväskylässä 9. - 13. 6., ja minä toimin sen järjestelytoimikunnan puheenjohtajana. Tähän konferenssiin toivon runsasta kotimaista osanottoa. Perinteiseen tapaan kahtena ensimmäisenä päivänä on "työpajoja" ja korkeatasoisia käytännönläheisiä tutoriaaleja. Varsinaiset konferenssi-istunnot sijoittuvat kolmeksi viimeiseksi päiväksi. Tietoa konferenssista on parhaiten saatavissa WWW:n kautta:

http://www.ecoop97.jyu.fi.