Suhteellinen koodikirnu ohjelmiston virheherkkyyden analysoimisessa

Kasper Hirvikoski

Referaatti
Helsingin Yliopisto
Tietojenkäsittelytieteen laitos

Helsinki, 17. helmikuuta 2013

Sisältö

1 Johdanto

Ohjelmistot kehittyvät elinkaarensa aikana muun muassa uusien vaatimusten, optimisaatioiden, tietoturvaparannusten ja virhekorjausten johdosta. Nagappan ja Ball esittävät artikkelissaan "Use of relative code churn measures to predict system defect density" [NB05] suhteellisen koodikirnu-tekniikan järjestelmän virhetiheyden ennakoimiseen. Koodikirnu (code churn) mittaa ja ilmaisee määrällisesti ohjelmiston komponentteihin kohdistuvia muutoksia tietyn ajanjakson aikana. Nagappan ja Ball tuovat esille joukon suhteellisia koodikirnu-mittayksiköitä, jotka he rinnastavat muihin muuttujiin kuten komponenttien kokoon tai muokkauksen ajalliseen pituuteen. Käyttäen apuna tilastollisia regressiomalleja, he osoittavat suhteellisten koodikirnu-mittojen kyvyn havaita järjestelmän virhetiheyden paremmin kuin ehdottomien mittojen. Väittämien tueksi he suorittivat tapaustutkimuksen, jonka kohteena oli Windows Server 2003. Samalla he osoittavat, että relatiivinen koodikirnu pystyy paikallistamaan virheherkät komponentit 89 % tarkkuudella.

2 Ohjelmiston laadun arviointi koodikirnulla

Kehitysvaiheessa olevan ohjelmiston laadun varmistaminen on hankalaa. Ohjelmiston testaamisen ja käytännössä havaittujen virheiden välillä on usein suuri kuilu. Virheiden määrää ei yleensä pystytä laskemaan luotettavasti ennen kuin tuote on valmis ja julkaistu asiakkaalle. Tässä piilee kuitenkin ongelman ydin: virheiden korjaaminen kehityksen lopussa on usein erittäin kallista. On siis selvää, että laadun varmistaminen ja mahdollisten ongelmakohtien havaitseminen mahdollisimman aikaisessa vaiheessa hyödyttää kehitystyötä suuresti.

Nagappan ja Ball esittävät ohjelmistojen virhetiheyden arvioimiseen ratkaisuksi koodikirnua. Se mittaa ohjelmiston komponenttien ohjelmakoodiin kohdistuvien muutosten määrää tietyn ajanjakson aikana. Muutosten määrä on helposti saatavilla ohjelmiston versionhallintajärjestelmien muutoshistoriasta. Useimmat versionhallintajärjestelmät vertailevat lähdekooditiedostojen historiaa ja laskevat automaattisesti koodiin kohdistuvia muutoksia. Nämä muutokset ilmentävät kuinka monta riviä tiedostoon on ohjelmoijan toimesta lisätty, poistettu tai muutettu sitten viimeiseen versiohistoriaan tallennetun version. Nämä muutokset muodostavat koodikirnun pohjan.

Nagappan ja Ball esittävät joukon suhteellisia koodikirnu-mittoja virhetiheyden havaitsemiseen. Mitat ovat normalisoituja arvoja koodikirnun aikana saaduista tuloksista. Normalisoinnilla niistä on pyritty poistamaan mahdolliset häiriötekijät. Näitä mittoja on muun muassa yhteenlaskettujen koodirivien määrä, tiedostojen muutokset ja tiedostojen määrä. Tutkimukset ovat osoittaneet, että ehdottomat mittayksiköt, kuten pelkkä koodirivien summa, ovat huonoja ohjelmiston laadullisia ennusteita. Yleisesti ottaen oh-

jelmiston kehitysprosessia mittaavien yksiköiden on havaittu olevan parempia osoittimia vikojen määrästä kuin pelkkää koodia arvioivat tekijät.

Ohjelmistoa kehitettäessä sen komponenttien monimutkaisuus muuttuu. Monimutkaisuuden kasvun suhde on hyvä mittari virheherkkyyden kasvulle. Koodikirnu-mittojen on havaittu korreloivan selvästi ohjelmistoista tehtyjen vikailmoitusten kanssa. Mittojen välillä on havaittavissa myös keskinäisiä suhteita, joita voidaan mallintaa verkkoina. Yksinään kyseiset mittarit eivät välttämättä tuota toivottua tulosta. Näin ollen mittoja verrataan keskenään mahdollisten ristiriitaisuuksien havaitsemiseksi. On kuitenkin selvää, että johtopäätöksiin päätyminen on hankalaa empiirissä tutkimuksissa, koska prosessien taustalla on usein laajoja kontekstisidonnaisia tekijöitä.

3 Ohjelmiston virheherkkyyteen vaikuttavat mitat

Nagappan ja Ball listaavat seuraavat ehdottomat mitat koodikirnun pohjaksi. Nämä muodostavat suhteellisille mitoille vertailukohdat ohjelmiston virheherkkyyden analysoimisessa. Ehdottomat mitat eivät yksinään tuota luotettavaa tulosta.

Yhteenlaskettu koodirivien määrä, ohjelman uuden version ei-kommentoitujen koodirivien summa kaikkien lähdekooditiedostojen kesken.

Käsiteltyjen koodirivien määrä, ohjelman lähdekoodiin lisättyjen ja muuttuneiden koodirivien summa edelliseen versioon nähden.

Poistettujen koodirivien määrä, ohjelman lähdekoodista poistettujen koodirivien määrä edelliseen versioon nähden.

Tiedostojen määrä, yhden ohjelman kääntämiseen tarvittavien lähdekooditiedostojen määrä.

Muutosten ajanjakso, yhteen tiedostoon kohdistuneiden muutosten ajanjakson pituus.

Muutosten määrä, ohjelman tiedostoihin kohdistuneiden muutosten määrä edelliseen versioon nähden.

Käsiteltyjen tiedostojen määrä, ohjelman käsiteltyjen tiedostojen yhteenlaskettu määrä.

Näiden pohjalta he muodostivat kahdeksan suhteellista koodikirnu-mittaa ja osoittivat, että nämä mitat korreloivat selvästi kohonneeseen virhemäärään koodirivejä kohden. He käyttivät Spearmanin järjestyskorrelaatiokerrointa, joka kuvaa kahden asian keskinäistä vastaavuutta. Analyysissään he havaitsivat suhteellisten mittojen ylivertaisuuden ehdottomiin verrattuna. Empiiristen tutkimusten avulla he havaitsivat seuraavien mittojen soveltuvuuden todellisen virhetiheyden ennakoimiseen.

1. **Käsiteltyjen koodirivien määrä / Yhteenlaskettu koodirivien määrä**

   Suurempi osa käsiteltyjä koodirivejä suhteessa yhteenlaskettuun koodirivien määrän vaikuttaa yksittäisen ohjelman virhetiheyteen.

2. **Poistettujen koodirivien määrä / Yhteenlaskettu koodirivien määrä**

   Suurempi osa poistettuja koodirivejä suhteessa yhteenlaskettuun koodirivien määrään vaikuttaa yksittäisen ohjelman virhetiheyteen. Nagappan ja Ball havaitsivat korkean korrelaation mittojen 1. ja 2. välillä.

3. **Käsiteltyjen tiedostojen määrä / Tiedostojen määrä**

   Suurempi osa käsiteltyjä tiedostoja suhteessa ohjelman rakentavien tiedostojen lukumäärään lisää todennäköisyyttä, että nämä käsitellyt tiedostot aiheuttavat uusia vikoja. Esimerkiksi meillä on kaksi ohjelmaa A ja B, jotka molemmat koostuvat 20 lähdekooditiedostosta. A sisältää viisi käsiteltyä tiedostoa ja B kaksi. Todennäköisyys sille, että muutokset ohjelmaan A saattavat aiheuttaa uusia vikoja on siis suurempi.

4. **Muutosten määrä / Käsiteltyjen tiedostojen määrä**

   Mitä suurempi määrä yksittäisiin tiedostoihin on kohdistunut muutoksia, sitä suurempi on todennäköisyys sille, että tämä vaikuttaa kyseisistä lähdekooditiedostoista muodostuvan ohjelman virhetiheyteen. Esimerkiksi jos ohjelman A viittä lähdekooditiedostoa on muutettu 20 kertaa ja ohjelman B viittä tiedostoja on muutettu kymmenen kertaa, todennäköisyys sille, että ohjelma A sisältää uusia vikoja on suurempi.

5. **Muutosten ajanjakso / Tiedostojen määrä**

   Mitä pitempi aika on kulutettu muutoksiin, jotka kohdistuvat pieneen joukkoon tiedostoja, sitä suurempi on todennäköisyys sillä, että nämä tiedostot sisältävät monimutkaisia rakenteita. Monimutkaisuus vaikuttaa koodin helppoon ylläpidettävyyteen ja näin ollen lisää näiden tiedostojen aiheuttamaa virhetiheyttä.

6. **Käsiteltyjen ja poistettujen koodirivien määrä / Muutosten ajanjakso**

   Käsiteltyjen ja poistettujen koodirivien määrä suhteessa muutosten ajanjaksoon mittaa muutoksen määrää, jota pelkkä muutosten ajanjakso ei yksinään ilmaise. Tätä mittaa tulee verrata mittaan 5. Oletuksena on, että mitä suurempi määrä käsiteltyjä ja poistettuja koodirivejä on, sitä pitempi muutosten ajanjakson tulisi olla. Tämä taas vaikuttaa ohjelman virhetiheyteen.

7. **Käsiteltyjen koodirivien määrä / Poistettujen koodirivien määrä**

   Ohjelmiston kehitys ei koostu pelkästään vikojen korjaamisesta vaan myös uuden kehittämisestä. Uusien ominaisuuksien kehittämisessä käsiteltyjen koodirivien määrä on suhteessa suurempi kuin poistettujen koodirivien määrä. Suuri arvo tälle mitalle ilmaisee uutta kehitystä. Saatua arvoa verrataan mittoihin 1. ja 2., jotka yksinään eivät ennakoi uutta kehitystä.

8. **Käsiteltyjen ja poistettujen koodirivien määrä / Muutosten määrä**

   Mitä suurempi muutoksen laajuus on suhteessa muutosten määrään, sitä suurempi virhetiheys on. Mitta 8. toimii verrokkina mitoille 3. – 6. Suhteessa mittoihin 3. ja 4., mitta 8. ilmaisee todellisen muutoksen määrää. Se kompensoi sitä tietoa, että yksittäisiä tiedostoja ei käsitellä toistuvasti pienten korjausten takia. Suhteessa mittoihin 5. ja 6.,mitä suurempi käsiteltyjen ja poistettujen koodirivien määrä on käsittelyä kohden, sitä pitempi muutosten ajanjakso tarvitaan ja sitä enemmän muutoksia kohdistuu esimerkiksi jokaista viikkoa kohden. Muussa tapauksessa suuri määrä muutoksia on saattanut kohdistua hyvin lyhyeen ajanjaksoon, joka ennakoi suurempaa virhetiheyttä.

4 Johtopäätökset koodikirnusta

Nagappan ja Ball havaitsivat, että koodi joka muuttuu useasti ennen julkaisua on selvästi virheherkempää kuin koodi, joka muuttuu vähemmän saman ajanjakson aikana. He tutkivat kahden ohjelmistojulkaisun Windows Server 2003 ja Windows Server 2003 Service Pack 1 pohjalta saatuja tuloksia. Julkaisuista analysoitiin 44,97 miljoonaa riviä koodia, joka muodostui 96 189 lähdekooditiedostosta. Niistä käännettiin 2 465 yksittäistä ohjelmaa.

He päätyivät tutkimuksessaan neljään johtopäätökseen:

1. Suhteellisten koodikirnu-mittojen nousua seuraa ohjelmiston virheherkkyyden kasvu.

2. Suhteelliset mitat ovat parempia laadullisia arvioijia kuin ehdottomat mitat.

3. Suhteellinen koodikirnu on tehokas tapa arvioida ohjelmiston virheherkkyyttä.

4. Suhteellinen koodikirnu pystyy havaitsemaan virheherkän ja toimivan komponentin toisistaan.

5 Koodikirnun pätevyyteen vaikuttavia tekijöitä

Nagappan ja Ball toteavat, että mittausvirheet vaikuttavat arvion luomiseen. Ongelma ei ole kuitenkaan suuri, sillä versionhallintajärjestelmät hoitavat automaattisesti analyysiin vaadittavat lähtöarvot. Koodikirnu vaatii kuitenkin myös ohjelmiston kehittäjältä hyviä käytäntöjä. Jos kehittäjä on tehnyt useita muutoksia rekisteröimättä niitä versionhallintajärjestelmän historiaan, osa muutoksista jää näkemättä. Kehittäjän toimista riippuen myös muutosten ajanjakson pituus voi selvästi pidentyä, jos muutoksia ei hyväksytä tarpeeksi aikaisin versionhallintajärjestelmään. Mittojen vertaaminen keskenään lieventää tästä johtuvia poikkeamia.

He toteavat myös, että tapaustutkimuksen pätevyyteen voidaan nähdä vaikuttavan myös sen tosiasian, että tutkimuksessa analysoitiin vain yhtä ohjelmistojärjestelmää. Tämä ohjelmistojärjestelmä kuitenkin koostuu lukuisista komponenteista ja suuresta määrästä koodia. Analyysi on siis itsessään erittäin kattava.

6 Yhteenveto

Nagappan ja Ball esittävät, että suhteelliset koodikirnu-metriikat ovat erinomaisia ja tehokkaita ennustajia ohjelmiston virhetiheyden arviointiin ja täten ohjelmiston laadulliseen varmentamiseen. He todistivat, että suhteellisten koodikirnu mittojen nousua seuraa ohjelmiston virheherkkyyden kasvu. He havaitsivat myös, että suhteelliset mitat ovat selvästi parempia laadullisia arvioijia kuin ehdottomat mitat itsessään.

7 Lähteet

[NB05] Nagappan, Nachiappan ja Thomas Ball: *Use of relative code churn measures to predict system defect density.* Teoksessa *Proceedings of the 27th international conference on Software engineering*, ICSE '05, sivut 284 – 292, New York, NY, USA, 2005. ACM, ISBN 1-58113-963-2. `http://doi.acm.org/10.1145/1062455.1062514`.