Taikuus - joka koodin riesa (Miska Hiltunen 4.11.2006)

Walt Disneyn ***Fantasia***-piirretyssä Mikki Hiiri joutuu ongelmiin innostuessaan käyttämään taikuutta apunaan osaamatta hallita sitä kunnolla. Samoin kuin Mikki meinaa hukkua taikuudella aiheuttamaansa tulvaan, myös monet ohjelmistoammattilaiset ovat vaarassa hukkua liiallisen taikuuden aiheuttamaan koodin monimutkaisuuteen. Joka sellaiseen koodiin tarttuu, se koodiin hukkuu.

Kovakoodatut numerot tuovat koodiin hämmästyttävää taikuutta. Niin hienolta kuin tuo kuulostaakin taikuus ja hämmästyminen eivät ole tavoiteltavia asioita koodissa, jota pitäisi muokata, laajentaa tai ymmärtää. Numerot, kuten ‘100’, ‘42’ ja ‘0xFF’, ovat välttämättömiä koodin toiminnallisuuden kannalta mutta ne eivät välitä tarpeeksi informaatiota. Juuri puuttuva informaatio onkin eniten ylläpitoa haittaava asia taikanumeroiden kohdalla.

Ensinnäkin kovakoodattu numero ei kerro mistä se tulee. Miksi juuri ‘100’, miksei ‘99’? Ylläpitäjä, jonka tehtävänä on kenties muuttaa annettua arvoa, joutuu selvittämään mistä arvo tulee nähdäkseen onko se oikein. Paremmin kommunikoiva koodi nimeäisi tietyn arvon sen mukaan mistä standardista tai päätöksestä arvo johtuu. Esimerkiksi absoluuttista nollapistettä kuvaavan arvon voisi C-kielessä nimetä:

```
#define ABSOLUTE_ZERO -273
```

, jolloin koodissa olisi kaikkialla selkeä arvon nimi sen arvoituksellisen arvon sijasta.

Toiseksi kovakoodatut arvot saattavat olla suhteessa muihin arvoihin tai tietorakenteisiin. Esimerkiksi taulukon koon määrittävä arvo on tiukasti liitoksissa taulukon läpikäyvän for-silmukan indeksin raja-arvoihin. Jos sekä taulukon koko että silmukan indeksi on kovakoodattu, on muutoksella normaalia suurempi todennäköisyys aiheuttaa virhe, vaikka lähtötilanne olisikin täysin virheetön. Käytännössä ylläpitäjä muuttaa vain toista arvoista, esimerkiksi taulukon kokoa, ja unohtaa muuttaa silmukan indeksin raja-arvoja. Seurauksena voi olla taulukon osittainen läpikäyminen tai dataviittaukset taulukon ulkopuolelle. Riippuen testauksen tehokkuudesta tehdyt virheet saadaan kiinni nopeasti (tiimi itse löytää), liian myöhään (asiakas huomaa ja valittaa) tai ei koskaan (järjestelmä toimii “oudosti” silloin tällöin, asiakas harmistuu mutta asialle ei voida sen tarkemmin tehdä mitään. Vika voi olla missä hyvänsä.)

Puuttuvat riippuvuudet haittaavat ylläpitoa korottaen virheen mahdollisuutta. Riippuvuuksien selventäminen ei välttämättä ole lainkaan hankalaa. Esimerkiksi kahden toisistaan riippuvan arvon määrittely voisi mennä seuraavasti:

```
#define WEEKDAYS 7
#define LAST_DAY_INDEX (WEEKDAYS - 1)
```

Mahdollinen muutos ohjelmassa voisi liittyä esimerkiksi siihen, että vain työpäivät tarvitsisi ottaa huomioon. Viikonpäivien lukumäärä tuskin muusta syystä muuttuu, vaikka tunteehan historia mm. pari uutta kuukauttakin, jotka ovat sotkeneet monien kielten kuukausinumeroinnin (‘December’-sanahan tarkoittaa kymmenettä kuukautta, mutta Rooman keisarien kunniaksi lisätyt ‘July’ (Julius) ja ‘August’ (Augustus) tekivät siitä kahdennentoista). Ylläpitäjän tarvitsee muuttaa vain yhtä arvoa (‘7’) ja kaikki tarvittavat arvot muuttuvat automaattisesti oikein, eikä yksikään ylimääräinen muutu.

Qualiteers / Miska Hiltunen
Saladin-Schmitt-Strasse 43 ● 44789 Bochum ● GERMANY ● tel. +49-234-3382359 ● fax. +49-234-3382361
info@qualiteers.com ● www.qualiteers.com

Jos ylläpitäjän tulisi muuttaa arvo ‘72’ arvoksi ‘70’, saattaa tulla mieleen suoraviivainen automaattinen korvaustoiminto, joka varmaankin kaikissa koodinkäsittelyohjelmistoissa on. “Sea rch and replace”-toimintoon syötetään muutettava arvo (‘72’) ja korvattava arvo (‘70’). Samantien kaikki alkuperäiset arvot on korvattu uudella arvolla, jopa koko projektissa. Sen jälkeenhän tarvitsee vain kääntää ja lisätä muutettu tiedosto versionhallintaan. Eihän näin pienen muutoksen takia sentään tarvitse testejä ajaa, eihän? Kaikkihan on kunnossa.

Paitsi, jos koodissa on käytetty arvoa ‘72’ useaan eri tarkoitukseen. Tällöin vain yhteen tarkoitukseen käytettyä arvoa piti muuttaa, kaikki muut muutokset ovat virheitä. Aikaa ei paljon mennyt, mutta virheitä saattoi tulla tehtyä useita. Automaattisen korvaustoiminnon käyttäminen on siis vähintään erittäin vaarallista, ellei jopa täysin mahdotonta.

Myös merkkivakiot (‘b’) ja merkkijonot (“ERROR: this should never happen!!!”) sisältävät taikuutta. Niiden kohdalla kyse on usein erikielisten versioiden tekemisestä eli lokalisoinnista. Lokalisointi on huomattavasti vaivattomampaa, jos tekstitieto on erotettu muusta koodista selkeästi. Toisaalta on syytä muistaa, että esimerkiksi suomenkielisten debug-tekstien käyttäminen rajoittaa mahdollisen ylläpitäjän kansallisuuden hyvinkin tiukasti. Jos asiakas haluaa itse ylläpitää koodia, on suomenkielisiä kommentteja ja merkkijonoja käyttävää koodia mahdotonta myydä Suomen rajojen ulkopuolelle. Englanninkielestä poikkeaminen rajoittaa koodin markkinoita turhasti.

Laskurityyppisille numeroille on lähes mahdotonta keksiä parempaa merkintätapaa kuin suoran numeron käyttäminen. On silti syytä olla tarkkana juuri toisistaan riippuvien numeroiden merkinnässä. Kuten edellä mainittiin, raja-arvoille kannattaa usein antaa kuvaavat nimet. Ensimmäinen ja viimeinen arvo ovat tärkeitä kohtia, joihin tarvitsee kiinnittää erityistä huomiota. Virheitä tapahtuu eniten juuri arvoalueiden reuna-alueilla.

Tick-the-Code-katselmoinnissa tarkistajan tehtävänä on merkitä kaikki ‘MAGIC’-sääntöä rikkovat asiat. Näihin kuuluvat myös luvut ‘0’ ja ‘1’ vaikka niitä käytettäisiin vain for-silmukoissa tai alustamaan paljonkin muuttujia. On erittäin todennäköistä, ettei koodin kirjoittaja eli sen ylläpitäjä tee merkatuille asioille mitään, mutta se onkin hänen oikeutensa. Mikään ei kuitenkaan estä tarkistajaa merkkaamasta niitä, jolloin mahdollisuus niidenkin muuttamiselle koodin korjausvaiheessa on olemassa.

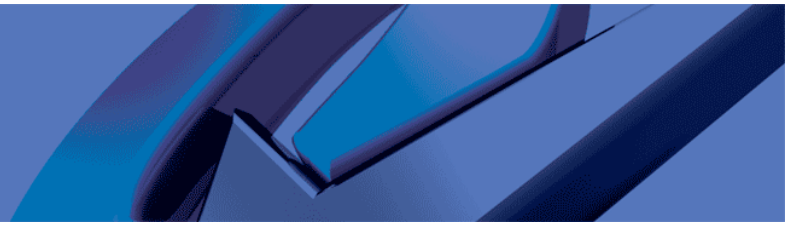
Taikuus ei poistu huonolla nimeämisellä. Esimerkiksi

```
#define ZERO 0
```

vain pahentaa tilannetta. Ylläpidossahan saattaa käydä niin, että tuota arvoa pitää muuttaa, jolloin

```
#define ZERO 2
```

saa koodin toimimaan täysin oikein, mutta koodin ymmärtäminen on erittäin hankalaa. Kaikkialla missä lukee ‘ZERO’ tarkoitetaankin arvoa ‘2’. Tässä esimerkissä ilmenee toiminnallisuuden ja sisäisen laadun ero selvästi. Vaikka ohjelma toimisi täysin oikein, saattaa sen muuttaminen silti olla täysin perusteltua.



Loppukevennys

Arthur C. Clarke sanoi joskus jotakuinkin näin: "*Mikä tahansa riittävän edistynyt teknologia vaikuttaa taikuudelta.*"

Uusia taikuutta käsitteleviä elokuvia ovat "***The Illusionist***" ja "***Prestige***".

Neil Gaimanin kirjoittama sarjakuva "***Books of Magic***" käsittelee taikuutta hienosti. Sen päähenkilö Tim Hunter muistuttaa hämmästyttävästi **J.K.Rowling**in luomaa, paljon kuuluisampaa virkaveljeään ***Harry Potter***ia, vaikka Tim kehitettiin jo vuosia aikaisemmin.

Susanna Clarke kirjoitti esikoisromaaninaan tiiliskiven, josta tuli todellinen myyntimenestys. "***Jonathan Strange & Mr.Norrell***" on suomennettukin. Se käsittelee Napoleonin aikaista Eurooppaa, josta oikea taikuus on lähes kadonnut, kunnes kirjan nimihahmot päättävät palauttaa sen oikeaan kukoistukseensa. Kirjailijalta on ilmestynyt juuri novellikokoelma samaan aiheeseen liittyen. Nimeltään se on "***The Ladies of Grace Adieu***", enkä ole sitä vielä lukenut.

Taikuus on aina kiehtonut ihmisiä. Taikuuden näennäinen yksinkertaisuus ja kaikkivoipaisuus hämmästyttää. Parhaat ohjelmistot ovat juuri sellaisia. Jos haluatte pelata taikuuden kanssa, tehkää ohjelmistanne taianomaisia pitämällä taikuus poissa itse koodista.

Qualiteers / Miska Hiltunen
Saladin-Schmitt-Strasse 43 ● 44789 Bochum ● GERMANY ● tel. +49-234-3382359 ● fax. +49-234-3382361
info@qualiteers.com ● www.qualiteers.com