

Ylläpidettävyyden Ylistys

Uusin tapa katsoa **Tick-the-Code** -menetelmää on jakaa maailma neljään osaan kahden akselin avulla: toisaalta toiminnallisuus-ylläpidettävyyden akseli ja toisaalta syntaksi-semanttisuus-akseli.

On olemassa ulkoisia vikoja, jotka näkyvät toiminnallisuudessa ja joita tyypillisesti kutsutaan bugeiksi. Toisaalta lienee kaikille selvää, etteivät kaikkia koodin toteutuksia ole luotu tasa-arvoisiksi, vaikka kaikki niistä toteuttaisivatkin saman toiminnallisuuden. Koodeissa on sisäisiä eroja, joskus dramaattisiakin. Sisäiset epätäydellisyydet, jotka haittaavat jatkokehitystä, ja joita voitaisiin yhtä hyvin kutsua virheiksi kuin bugejakin, ovat ylläpidettävyyden ongelmia.

Tarkistuksessa voidaan joko etsiä yksinkertaisesti jotain ohjelmointikielen elementtiä (vaikkapa avainsanaa 'return'). Tällaiset syntaksitarkistukset voitaisiin yleensä helposti automatisoida. Toisaalta voidaan pureutua koodin merkityksiin, esimerkiksi funktioiden nimiin ja verrata niitä niiden sisältöön ja merkitä virheiksi kaikki ristiriidat. Merkitysten selvittäminen vaatii semanttista ymmärtämistä ja lienee täysin mahdollista automatisoida.

Jos yhdistetään ylläolevat nelikentäksi, nähdään mitä erilaiset työkalut tekevät ja mitä kuvasta puuttuu.

	Toiminnallisuus	Ylläpidettävyyden akseli
Syntaksitarkistus	staattiset analysointit	
Merkitystarkistus	testaus	???

Taulukko 1. Mihin työkalut ja menetelmät on tarkoitettu?

Taulukon 1. nelikentässä nähdään, että testauksella pyritään etsimään toiminnallisuuden väärinymmärryksiä (järjestelmätestauksessa käytetään testitapauksia, jotka on luotu vaatimusmäärittelyä tulkitsemalla eli ymmärtämällä) ja että staattiset analysointit (lint, Klocwork, Polyspace) pystyvät teoriassakin korkeintaan kattamaan syntaksitarkistuksen alueen (käytännössä edes näin ei ole). Kukaan eikä mikään ei tarkista, että ylläpidettävyyden koodissa olisi hyvällä tasolla, niin että koodi olisi ymmärrettävää, se sisältäisi kaiken tarpeellisen (hyvät kommentit), ei mitään turhaa, eikä luottaisi sokkona esimerkiksi muuttujensa arvoihin.

Jos tehdään toinen nelikenttä (Taulukko 2), johon laitetaan **Tick-the-Code** -menetelmän säännöt ja se, mitä sääntöjen tarkistaminen vaatii, nähdään selvästi mihin menetelmällä pyritään.

	Toiminnallisuus	Ylläpidettävyyden akseli
Syntaksitarkistus	8 (20%)	13 (32.5%)
Merkitystarkistus	4 (10%)	15 (37.5%)

Taulukko 2. Tick-the-Code -menetelmän säännöt tarkistavat pääasiassa ylläpidettävyyttä (28 tavalla).

Kun satsaa Tick-the-Code -menetelmään, satsaa ylläpidettävyyteen. 70 prosenttia tikkaussäännöistä pyrkii paljastamaan sisäisiä ristiriitoja ja siten parantamaan ylläpidettävyyttä, jolla on kiistatta vaikutusta myös ulkoisten virheiden määrään.