Tarkistusmenetelmien eroista **(Miska Hiltunen 6.11.2006)**

Ohjelmistokoodia voidaan tarkastaa useilla erilaisilla tavoilla. Menetelmien erot eivät ole pelkästään niiden toteutustavoissa, vaan myös eri menetelmien tuottamissa tuloksissa on eroja sekä määrässä että laadussa. Seuraavassa vertailen muutamaa menetelmää toisiinsa.

Jos tarkistuksen kohteena ovat pelkästään koodin muutokset, kutsun menetelmää lyhenteellä ‘j-c’ eli “*just changes*”, tai *pelkät muutokset*-tarkistus. Apuna tarkistuksessa voi olla tiedostojen vertailuun tarkoitettu ohjelma, jonka avulla näytölle saadaan näkyviin pelkästään muutetut, lisätyt ja poistetut koodirivit. Niitä tarkastelemalla koitetaan sitten selvittää, tulevatko muutokset aiheuttamaan mahdollisesti ongelmia. Vielä yksinkertaisemmassa muodossa ohjelmistosuunnittelija pyytää kollegaansa tarkistamaan muuttamansa rivit ilman vanhaa versiota. Pyyntö saatetaan jopa asettaa johdattelevasti “eihän näissä muutoksissani ole mitään vikaa, eihän?” Johdattelevuus ja suhteellisen lyhyt tarkistusaika johtavat helposti siihen, ettei muutoksista mitään suoranaista vikaa löydy. Tietyn koodin liittyminen toiminnallisuuteen ja koodin muuttamisen kaikkien seuraamusten selvittäminen nopeasti on vaikeaa, ja kynnys valittaa muotoseikoista on melko korkea tällaisessa psykologisesti latautuneessa tilanteessa. Tarkistus ei vie paljon aikaa, muttei huomautuksiakaan juuri löydy.

Ammattikirjallisuudessa on kuvattu *walkthrough*-tyyppinen koodin tarkistusmenetelmä, josta käytän seuraavassa ‘*WT*’-lyhennettä. *WT*-menetelmässä koodattu materiaali, esimerkiksi uusi moduli, lähetetään etukäteen tarkistajille, jotka toivottavasti valmistautuvat käymällä koodin läpi kukin omalla tavallaan. Koodin kirjoittaja kutsuu tarkistajat kokoukseen, jossa esittelee koodin käyttäen esimerkiksi videotykkiä koodin heijastamiseksi seinälle. Funktio funktiolta kirjoittaja kertoo, mitä koodin tulisi tehdä ja miten se sen oletettavasti tekee. Tarkistajat voivat esittää kommentteja, huomautuksia ja kritiikkiä. Selkeä vastakkainasettelu - kirjoittaja yksin jännittävässä esitystilanteessa ryhmän edessä ja suurehko joukko kärkkymässä jokaista virhettä lauman suojissa - aiheuttaa helposti psykologisen paineen, joka johtaa tilanteen kärjistymiseen. Kirjoittaja-esittäjä asettuu puolustuskannalle ja pyrkii selittämään tarkistajien huomautusten syitä. “Tiedän, että minun olisi pitänyt tehdä tuossa huolellisemmin, mutta..”, “oli vain niin kova kiire,...” “joo, sen jätin tehtäväksi ensi viikolla”. Sen sijaan, että kirjoittaja ottaisi huomautukset vastaan rakentavana kritiikkinä, jonka avulla hän saisi koodistaan vielä parempaa, hän saattaa käsittää huomautukset itsensä arvosteluna ja kokea tulevansa hyökätyksi. Koodin parantaminen jää sivuseikaksi ja omien kasvojen säilyttäminen nousee tärkeimmäksi asiaksi. Seurauksena ovat pitkähköt aikaa hukkaavat keskustelut, ja kiristynyt tunnelma osallistujien välillä. Kynnys “hyökätä” kasvaa, mikä osaltaan vähentää parannusehdotusten määrää. Raportoitujen huomautusten määrä jää pieneksi, mutta kokous vaatii helposti melko paljon aikaa. Kahdeksanhenkinen tunnin kokoushan käyttää efektiivisesti työpäivän verran työtunteja organisaatiolta.

Jos halutaan olla todella tarkkoja, voidaan tarkistukseen ottaa esimerkiksi vaatimusmäärittely. Vaatimusmäärittelyn vertaaminen sen pohjalta tehtyyn koodiin on hyvinkin hyödyllistä. Tällaisen ristiintarkistuksen ideana on varmistaa, että kaikki halutut ominaisuudet on toteutettu, mitään ylimääräistä tai turhaa ei ole toteutettu ja kaikki toteutettu on toteutettu oikein. Löytyvät epätarkkuudet tai ristiriitaisuudet ovat merkkejä mahdollisesti kalliistakin virheistä. Mitä aikaisemmin kalliit virheet voi välttää, sen enemmän rahaa säästetään.

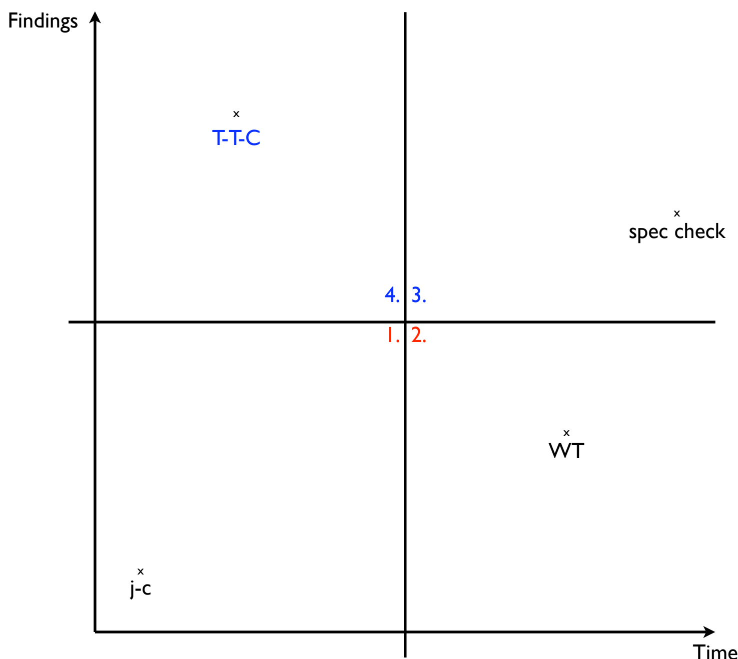
Qualiteers / Miska Hiltunen
Saladin-Schmitt-Strasse 43 ● 44789 Bochum ● GERMANY ● tel. +49-234-3382359 ● fax. +49-234-3382361
info@qualiteers.com ● www.qualiteers.com

Joissain organisaatioissa saatetaankin vaatimusmäärittelydokumentaatiolle tehdä erittäin tarkka lopputarkistus ennen siihen perustuvien työvaiheiden aloittamista. Lopputarkistuksessa varmistetaan että kaikki haluttu on listattu, mitään ylimääräistä ei ole listattu ja listatut asiat on listattu oikein ja ymmärrettävästi. Vaatimusmäärittelyn huolellinen tarkistaminen saattaa vaatia isonkin joukon eri osa-alueiden asiantuntijoita, jotka tietävät jo etukäteen paljon kyseisestä projektista. Löydösten kokoaminen saatetaan järjestää kokouksen muodossa, mikä omalta osaltaan lisää jo muutenkin suurta ajankulutusta.

Kutsun sekä pelkän vaatimusmäärittelyn tarkistamista että vaatimusmäärittelyn ristiintarkistamista koodin kanssa nimellä ‘spec check’ eli *vaatimusten tarkistus*. Tyypillistä on joko löydösten suuri määrä tai niiden suhteellisen suuri merkitys eli kalleus virheinä ja *WT*-menetelmääkin suurempi ajankäyttö.

Tick-the-Code-katselmointi (‘T-T-C’) tuottaa selvästi enemmän löydöksiä kuin ‘j-c’-menetelmä. Löydösten suuri määrä ja erityisesti niiden relevanttius perustuu käytettyihin selkeisiin sääntöihin, jotka pohjautuvat hyviin ohjelmointiperiaatteisiin. Lisäksi itse tarkistuksen kurinalaisuus auttaa tarkistajia löytämään haluamiansa asioita tehokkaasti. Keskittyminen yhteen sääntöön kerrallaan tehostaa löytötarkkuutta. Kun vielä lakataan tuijottamasta pelkkää toiminnallisuutta ja hyväksytään päätavoitteeksi koodin yksinkertaistaminen, löydösten määrä kasvaa huimasti.

Kun edellä mainitut neljä menetelmää sijoitetaan graafisesti menetelmän vaatiman kokonaisajan ja sen tyypillisesti tuottamien löydösten määrän tai painoarvon muodostamaan koordinaatistoon, saadaan aikaan kuvan 1. mukaiset neljä luokkaa.



Kuva 1. Erilaiset tarkistusmenetelmät löydösten määrän ja niiden etsimiseen kuluvan ajan mukaan luokiteltuna.

Luokka 1, jossa on *pelkät muutokset*-tarkistus (‘j-c’) voidaan kuvailla sanoilla nopea muttei kovin hyödyllinen. Vaikkei menetelmään kulukaan paljon aikaa, ei se myöskään paljon auta. Menetelmän hyötysuhde ei ole hyvä. Tarkistettavaa materiaalia on usein niin vähän, ettei kovinkaan monta huomautusta ole mahdollista löytää.

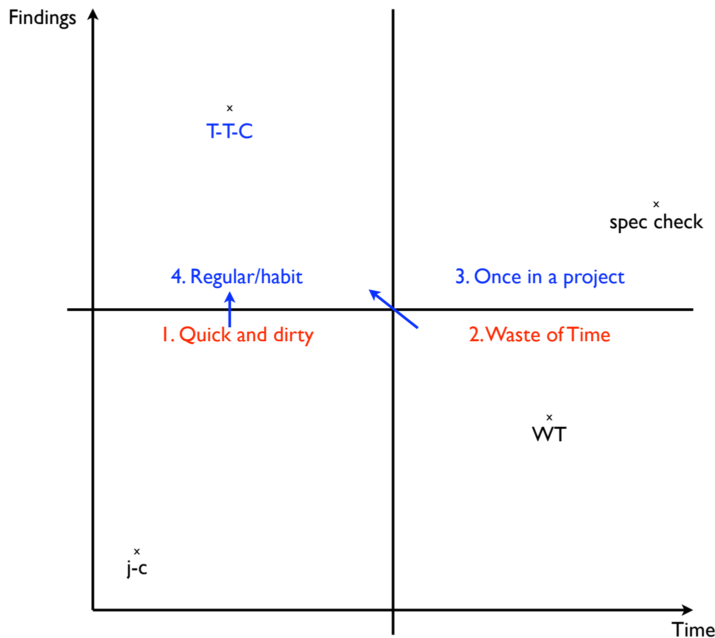
Luokka 2, johon kuuluu *walkthrough*-tarkistus ('WT'), voitaisiin nimetä lyhenteensä mukaisesti ajanhukaksi (Waste of Time). Koska palautetta varten on pidettävä kokous, on selvää, että ajankäyttö on menetelmää määrittävä tekijä. Sen osoittaa jo pelkästään sana 'kokous'. Keskustelukerhomaisuus jättää löydösten määrän pieneksi, mistä johtuu usein jopa menetelmäluokkaa 1 huonompi hyötysuhde. Materiaalia voi olla paljonkin, mutta kommunikoinnin kaistanleveys on niin kapea, ettei aikaa moneen huomautukseen useinkaan ole. Tietoa ei välity riittävän nopeasti.

Luokka 3 'spec check'-menetelmineen on hyötysuhteeltaan parempi kuin kumpikaan aikaisempi luokka johtuen löydösten erittäin hyvästä laadusta. Vaatimusmäärittelystä löytyvien virheiden korjaaminen ennen siihen pohjautuvan työn alkamista säästää helposti paljonkin turhaa työtä. Valitettavasti absoluuttisesti mitattuna tämän luokan ajankäyttö on liian suurta mahdollistaakseen menetelmän käytön useammin kuin kerran projektissa. Menetelmä on erittäin hyödyllinen, mutta sen toistaminen ei ole mahdollista. Kannattavuuden maksimoimiseksi tarkistuksen oikeaa ajankohtaa on syytä miettiä ja suunnitella hyvinkin tarkkaan.

Luokka 4 sisältää **Tick-the-Code**-tarkistuksen, joka parhaiten soveltuu säännöllisesti toistettavaksi rutiiniksi. Tähän luokkaan kuuluvista tarkistuksista kannattaa tehdä ohjelmistokehitykseen olennaisena osana kuuluvia tapoja, joita ilman ei voisi kuvitellakaan päästävänsä koodia itseltään eteenpäin. Hyötysuhteeltaan tällainen menetelmä soveltuu toistettavaksi esimerkiksi viikottain. Tarvittava absoluuttinen aika on pieni ja silti saatava löydösten määrä riittävän suuri tekemään menetelmän käytöstä tehokasta ja hyödyllistä.

Luokan 4 kurinalaisten tarkistusmenetelmien lisäetuna ovat löydösten määrän vertailukelpoisuus. Muissa luokissa löydösten määrä vaihtelee suuresti annettujen ohjeiden, niiden noudattamisen, valmistautumiseen käytetyn ajan ja yksilöiden löytämistarkkuuden ja raportoimishalukkuuden mukaan. Löydösten määrällä ei ole minkäänlaista korrelaatiota tarkistetun koodin kanssa. Ainakin **Tick-the-Code**-katselmoinnissa erot löydösmäärissä johtuvat lähes pelkästään tarkistetusta lähdekoodista. Standardin mukainen tarkistusmenetelmä mahdollistaa vertailukelpoiset tulokset. Vertailukelpoisuus puolestaan antaa monenlaisia työkaluja laatuosaston käyttöön.

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



Kuva 2. Tarkistusmenetelmäluokat ja niiden mahdolliset positiiviset kehityssuunnat.

Kuvassa 2. olevat kaksi nuolta kuvaavat organisaatioiden kehitysmahdollisuuksia. Organisaatiot, joissa vallalla oleva tarkistusmenetelmä on joko *pelkät muutokset*-tyyppinen tai *WT*-tarkistus, hyötyvät suuresti siirtyessään käyttämään säännöllisesti esimerkiksi **Tick-the-Code**-katselmointia. Parhaan hyödyn tuottava yhdistelmä lienee oikealla hetkellä projektissa hyvin suunniteltu ja toteutettu spec-check-tarkistus yhdistettynä mahdollisimman usein tehtyihin **Tick-the-Code** -tarkistuksiin. **Tick-the-Code** -tarkistuksin kannattaa kattaa mahdollisimman paljon koodia säännöllisesti koko projektin aikana. Itse asiassa se soveltuu käytettäväksi säännöllisesti koko uran ajan.

Esitetty tarkistusmenetelmien vertailu on karkea, eikä pyrikään äärimmäiseen tarkkuuteen. Karkea jaottelu saattaa olla hyödyllinen osoittaessaan parannusmahdollisuuksia tai helpottaessaan nykytilanteen analysointia organisaatiossa. Jo pelkästään oman tarkistusluokan selvittäminen saattaa auttaa organisaatiota sen pyrkimyksissä parantaa laatuaan.

Käytetty tarkistusmenetelmä asettaa ylärajan hyötysuhteelleen. Näin on erityisesti, jos käytetty tarkistusmenetelmä on ‘alarivissä’ eli luokassa 1 tai 2. Löydöksiä ei kerta kaikkiaan ole mahdollista löytää määräänsä enempää. Käytännössä viimeistään ajan loppuminen rajoittaa löydösten määrää. Luokan 4 menetelmää rajoittaa usein pelkästään kirjoittamisnopeus. Ei ole ennenkuulumatonta, että tarkistajat löytävät ja merkitsevät useita löydöksiä minuutissa.

Tarkistusmenetelmissä ja niiden tuloksissa on suuria eroja. On aina parempi tarkistaa kuin jättää tarkistamatta. Tarkistaminen on laadunvalvonnan kannalta paras vaihe löytää vastatekemiään virheitä. Mitä tuoreempi virhe, sitä paremmin siitä voi oppia. Mitä rutiininomaisempaa tarkistus on, sitä suuremmalla todennäköisyydellä se tulee tehtyä myös äärimmäisen kiireen uhatessa, eli juuri silloin kun sitä eniten tarvitaan. Ohjelmistosuunnittelijat ovat liian fiksuja ollakseen tajuamatta milloin tarkistus tehdään vain sen vuoksi että se täytyy tehdä ja milloin tarkistuksesta oikeasti on hyötyä. Tämän vuoksi kaikkien organisaatioiden tulisi pyrkiä parasta hyötysuhdetta kohti. Hyvän tarkistusmenetelmän käyttö motivoi suunnittelijoita entistä parempiin suorituksiin.

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