T-76.5613 Software Testing and Quality Assurance, 2011
Individual essay 5

Ohjelmiston laatu

Mikko Koski
mikko.koski@aalto.fi
66467F

19.12.2011

1 Mikko rakastaa Kattua!

2 Mikä on riittävän pahaa?

Kuten johdantokappaleessa mainittiin, vain täydellinen laatu on riittävä. Toisaalta, emme pysty saavuttamaan täydellistä laatua. Haluamme kuitenkin tuottaa ohjelmistoja, joten johonkin on vedettävä raja. Tätä ongelmaa pyrkii ratkaisemaan riittävän hyvän laadun (engl. good enough quality) käsite.

Riittävän hyvän laadun viitekehys pyrkii antamaan strukturoidun tavan vastata kysymykseen mikä on riittävän hyvä, kun täydellisyyteen ei kaikesta huolimatta päästä. Viitekehys antaa eri näkökulmia laatuun, joiden perusteella voidaan tehdä argumentoituja päätöksiä, onko laatu riittävän hyvää vai pitäisikö sitä parantaa. **?**

Olennainen osa riittävän hyvän laadun viitekehystä on tunnustaa ohjelmistotuotannon laadun kannalta muun muassa seuraavat ongelmalliset tosiasiat:

- Ohjelmistotuotannossa täytyy pärjätä kompleksisessa ympäristössä, joka on täynnä epävarmuustekijöitä ja rajoituksia.
- Kaikella on hintansa, ja se mitä haluamme ei välttämättä ole sitä, mihin meillä on varaa.
- Laatu on tilanneriippuvaista ja subjektiivista. Erinomaisuuden saavuttamiseksi on tehtävä kompromisseja.

Mielestäni viitekehys vaikuttaa todella toimivalta. Luulen, että moni ohjelmoija ja projektimanageri tekee vastaavanlaista ajattelutyötä alitajuisesti päivittäin. Koitamme mielessämme miettiä vastauksia seuraaviin kysymyksiin: Mikä virhe on kaikkein kriittisin?

Minkä virheen korjaan seuraavaksi? Onko kyseinen virhe niin kriittinen, että sitä kannattaa korjata?

Riittävän hyvän käsite on saanut osakseen myös kritiikkiä (**?**). **?** toteaa, että eräs syy on käsitteen ymmärtäminen väärin. Näin varmasti onkin, mutta uskon, että osasyy on myös ammattiylpeys. Väitän, että ohjelmistoalalla työskentelee maailman älykkäimpiä insinöörejä. Siitä huolimatta ohjelmistoissa on lukuisia virheitä. Riittävän hyvän laadun viitekehys vaatii meitä hyväksymään tämän tosiasian. Se voi olla vaikea paikka ja myös isku ammattiylpeydelle.

3 Mitä on oikeasti ohjelmiston laatu?

Laatu ohjelmistoalalla on määritelty virallisesta muutamallakin eri tavalla. IEEE määrittelee laadun seuraavalla tavalla: 1) Määrä, jolla systeemi, komponentti tai prosessi täyttää määritetyt vaatimukset. 2) Määrä, jolla systeemi, komponentti tai prosessi vastaa asiakkaan tai käyttäjän tarpeita tai odotuksia. Gerald M. Weinberg sen sijaan määrittelee laadun seuraavasti: Laatu on tuotettu arvo jollekin henkilölle (/joillekin henkilöille). (**?**)

Molemmissa laatumääritelmissä on otettu huomioon loppukäyttäjä, mikä mielestäni on hyvä asia. Erityisesti Weinbergin määritelmä osuu kokonaisvaltaisuudessaan naulan kantaan.

Kaikilla tuotetuilla ohjelmilla on aina olemassa jonkinnäköinen suora tai epäsuora vaikutus ohjelmiston käyttäjään. Tästä syystä olen sitä mieltä, että ohjelmiston arvo on sen tuottama vaikutus ohjelmiston käyttäjälle. Ainoa tapa, jolla käyttäjä pystyy arvioimaan ohjelmiston laatua ja täten sen tuottamaa arvoa käyttäjälle, on käyttäjän ohjelmiston käytöstä saama käyttökokemus, eli ulkoinen laatu.

3.1 Sisäinen laatu ja ulkoinen laatu

ISO 9126 standardi määrittelee laadun neljään osaan: Laatumalli (engl. quality model), sisäiset mittaukset (engl. interal metrics), ulkoiset mittaukset (engl. external metrics) ja käytön laatu (engl. quality in metrics). Edellesin kappaleen perusteella olenkin itse sitä mieltä, että ainoa pätevä laadun mittari on käyttäjän käyttökokemus, joka muodostuu ulkoisesta laadusta. Esimerkiksi koodin huono laatu ei mielestäni välttämättä tarkoita sitä, että ohjelmisto olisi huonolaatuista, jos huonolaatuinen koodi ei heijastu itse ohjelmistoon virheide kautta. Huonolaatuinen tietysti ajaa ohjelmiston virheisiin, mutta näin ei välttämättä ole. Koodi, joka on yhdelle huonolaatuista ja vaikealukuista, voi olla toiselle täysin luonnollista. Niin

kauan kuin huonolaatuinen koodi ei ilmene ohjelmiston virheinä, voidaan ohjelmistoa pitää kaikesta huolimatta hyvälaatuisena.

Vastaavasti esimeriksi koodin epäoptimaalisuus on hyvä esimerki huonosta koodin laadusta, joka ei välttämättä näy käyttäjälle mitenkään. Jos epäoptimaalinen koodi on ohjelmiston osassa, jota ajetaan vain harvoin, ei käyttäjä välttämättä edes huomaa kyseistä heikkoa laatua. Jos epäoptimaalinen koodi sen sijaan on ohjelmiston osassa, jota ajetaan jatkuvasti, esimerkiksi aina kun hiirtä liikutetaan, heijastuu se käyttäjälle helposti ohjelmiston hitautena ja täten huonona käyttökokemuksena.

On myös hyvin vaikea määritellä mikä on huonolaatuista koodia. Käsitys siitä, mikä on hyvää ja mikä huonoa koodia on muuttunut ohjelmoinnin historian aikana monesti, ja tulee jatkossakin muuttumaan. Hyvänä esimerkkinä tästä on Java-kieli. Vielä muutamia vuosia sitten Javan tyypillisiä piirteitä, kuten vahvaa tyypistystä ja informaation piilotusta kapseloinnin avulla pidettiin hyvänä ja ainoana hyväksyttävänä ohjelmointitapana. Muun muassa **?** artikkelissaan sivuaa tätä käsitystä. Siitä huolimatta nykypäivän suosituimmat ohjelmointikielet kuten Ruby ja JavaScript toimivat täysin päinvastoin. Ne ovat löysästi tyypitettyjä ja niiden olioiden sisältämää informaatiota pystyy manipuloimaan hyvin vapaasti. Näiden kielien myötä käsitys siitä, mikä on hyvää ja mikä huonoa, on ainakin vahvan tyypityksen osalta muuttuneet.

4 Laadun aikakaudet

Ohjelmisto tuotanto on nuorena alana käynyt läpi suuria muutoksia olemassa olo aikanansa. Nämä muutokset ovat heijastuneet myös ohjelmistojen laatuun eri aikakausina.

4.1 Historia

Artikkelissaan **?** käy läpi ohjelmistotuotannon viisi vuosikymmentä ja tuo esille mielenkiintoisia seikkoja, jotka ovat olleet syynä ohjelmiston laadun vaihteluun. Hän muun muassa esittää, että 1950-luvulla ohjelmistoalan alkuaikoina ohjelmistojen laatu oli huomattavasti nykyistä parempaa, koska ohjelmistoja tuottivat vain harvat ja älykkäät ihmiset. 1960-luvulla tietokoneiden yleistyminen aiheutti sen, että vähemmän älykkäät ihmiset tuottivat ohjelmistoja, mikä taas johti laadun heikkenemiseen. Toisaalta 1960-luvulla kääntäjiä oli harvassa, mikä johti siihen, että yksikin virhe vaati suuren määrän korjaustyötä. Tästä johtuen virheitä pyrittiin välttämään kaikin keinoin, muun muassa katselmointien avulla. 1970-luvulla kään-

täjien yleistyminen aiheutti laiskuutta ohjelmoijissa. Koodin laatu huononi ja tältä vuosikymmeneltä saimme taakaksi valtavan määrän vanhentunutta koodia (engl. legacy code), joka on tänä päivänäkin käytössä monissa suurissa ohjelmistoissa. (**?**)

4.2 Nykyhetki

1990-luvulla alkoi ohjelmistotuotannossa prosessiajattelun aikakausi (**?**). Tänä päivänä ketterät menetelmät ovat vallitsevia prosesseja ohjelmistoalalla. Kuten monet aikaisemmatkin mullistukset alalla, myös ketterät menetelmät ovat ravistuttaneet ohjelmistojen laatua. Omasta mielestäni ketterät menetelmät ymmärrettiin hyvin pitkään, ellei edelleenkin, täysin väärin. Tämä väärinymmärrys aiheutti holtittomuutta ja sitä kautta laadun heikkenemistä. Muun muassa **?** osoittaa artikkelissaan esimerkillisesti tätä väärinymmärrystä väittäen, että ketterät menetelmät ovat äärimmäisyyteen vietynä täysin kaoottisia prosesseja, jotka ohittavat testaus ja suunnitteluvaiheet.

Ketterien menetelien ongelma on siinä, että ne eivät eksplisiittisesti vaadi tiettyjen käytäntöjen käyttämistä. Tästä syystä on syntynyt väärinkäsitys, että niitä ei myöskään tarvitse toteuttaa. Todellisuudessa ketterät menetelmät kuten esimerkiksi Extreme Programming nimenomaan rohkaisevat laatukäytäntöjen, kuten automaattisen testauksen, pariohjelmoinnin ja katselmointien käyttöön.

4.3 Tulevaisuus

Mielestäni ohjelmistojen laadun paranemiseen on nyt hyvät edellytykset. Ketterien menetelmien yleistymisen myötä uskon, että myös niiden ymmärtämys on kasvanut. Luulen, että alkuaikojen väärinymmärryksistä on opittu ja huomattu, että ketterät menetelmät itsessään eivät takaa laatua ilman hyvien käytäntöjen käyttöönottoa. Tästä syystä luulen esimerkiksi testivetoisen kehityksen suosion kasvavan tulevaisuudessa, mikä antaa hyvän lähtökohdan myös laadun kasvamiselle.

Viime aikoina myös monet menestyvät yritykset ovat ottaneet lähtökohdan, jossa tehdään vähemmän mutta paremmin. Esimerkkejä tällaisista yrityksistä ovat Apple ja Google. Applen tietokoneet eivät useinkaan paperilla pärjää teknisten ominaisuuksiensa ja kyvykkyyksiensä perusteella vastaaville PC-koneille, mutta käyttäjät ovat valinneet Applen niiden ohjelmistojen ja laitteiston paremman laadun johdosta. Vastaavasti Google voitti ihmisten suosion Altavistalta, joka oli hakukoneen lisäksi massiivinen portaali. Siinä missä Altavistan sivuilla oli todella paljon sisältöä, Googlen sivulla oli

vain yksi tekstikenttä, johon käyttäjä kirjoitti hakusanan. Googlen haku tuotti kuitenkin huomattavasti parempilaatuisia hakutuloksia ja tästä syystä Google voitti kilvassa Altavista, ei ominaisuuksien määrän vaan laadun johdosta.

Luulen, että nämä menestystarinat rohkaisevat myös muita yrityksiä ajattelemaan, mikä tuotteessa on oikeasti käyttäjälle tärkeää ja keskittymään juuri sen osan toimivuuteen. Olen vakuuttunut siitä, että lähes kaikissa tapauksissa käyttäjät arvostavat ohjelmistoissa laatua määrän edelle.

5 Yhteenveto

Ohjelmiston laadun dilemmaa ei tulla luultavasti koskaan ratkaisemaan. Ohjelmistoissa on nyt ja tulee jatkossakin olemaan virheitä. Riittävän hyvän laadun viitekehys on auttanut hyväksymään tämän asian ja siirtänyt katseet olennaiseen: täydellisyyteen ei kannata edes pyrkiä, riittävän hyvä on riittävän hyvä.

Ajatusmaailmamme ohjelmistoja kohtaan on kuitenkin muututtava, jotta yleinen ohjelmistojen laatu paranee. Käyttäjät äänestävät jo nyt jaloillaan ja ostavat tuhansien ominaisuuksien kännykän sijaan kännykän, jossa on vain olennaiset ominaisuudet jotka toimivat moitteettomasti. Uskon ja toivon, että Applen ja Googlen kaltaisten yritysten menestys rohkaisee myös muita yrityksiä tekemään vähemmän, mutta paremmin.