Referaatti Martin Fowlerin artikkelista ”Is Design Dead?”

Artikkelissaan Fowler lähestyy kysymystä kahden erityylisen ohjelmistosuunnittelun näkökulmasta, yhtenä on perinteinen, ennen implementointia luotava suunnitelma ohjelmistosta ja toisena implementoinnin aikana tapahtuva evolutiivinen suunnittelu. Perinteinen suunnittelu on laajasti omaksuttu käytäntö 70-luvulta lähtien ja perustunut UML-kaavioihin tai sen edeltäjiin. Siinä oletetaan että ohjelmisto pystytään suunnittelemaan aluksi kokonaan ja täydellisesti, jonka jälkeen implementointi voidaan antaa muiden tehtäväksi. Fowler nostaa artikkelissa esiin ongelmia joita tästä on seurannut, suunnittelun aikana on mahdoton ottaa kaikkia tapauksia huomioon. Suunnittelijoilla ei ole välttämättä reaalielämän kokemusta ohjelmointivaiheesta ja ohjelmoinnin aikana törmätään helposti tapauksiin joissa suunnitelmaa pitäisi muuttaa. Mitä myöhemmässä vaiheessa muutostarve tulee ilmi, sitä suurempia kustannuksia siitä seuraa. Artikkelissa mainitaan eksponentiaalinen muutoskäyrä, joka kuvaa tarvittavien muutosten kustannusten nousua projektin edetessä.

Fowler kirjoittaa, ettei evolutiivinen suunnittelukaan yksin takaa hyvää suunnittelua. Helposti törmätään ad-hocin aiheuttamaan sekasotkuun jonka johdosta ohjelmistosta tulee erittäin hankalasti muutettava. Tämä taas näkyy merkittävästi kohonneina kustannuksina. Ketterissä menetelmissä on kuitenkin omaksuttu evolutiivinen suunnittelu ja Fowler esittää erilaisia menetelmiä, kuinka pitää hyvästä suunnittelusta huolta.

Pääosassa on huolellinen testaus sekä jatkuva integrointi. Lisäksi tehdään refaktorointia tarvittaessa. Refaktoroinnilla tarkoitetaan koodin sisäisiä muutoksia, siten että toiminnallisuus säilyy mutta varsinaisia lisäyksiä ei tehdä. Testauksella varmistetaan että refaktorointi ei aiheuta koodiin toimimattomia osia. Muutokset suunnittelussa ja sitä myötä koodin refaktoroinnissa, ovat näin turvallisempia. Jatkuvalla integraatiolla pystytään tahdistamaan ryhmän toiminta ja varmistamaan että ohjelmistosta on koko ajan saatavilla toimiva versio.

Yksinkertaisuuden periaate jonka Fowler tuo esille, on ajatus siitä että tekemällä yksinkertaisen koodin, suunnittelu pysyy myös yksinkertaisena ja muutoksia on myöhemmin helpompi toteuttaa. Aiheellinen kysymys jonka Fowler esittää, on kuinka yksinkertaisuus määritellään. Fowler mainitsee XP manifestit "Do the Simplest Thing that Could Possibly Work" eli pyritään tekemään yksinkertaisin mahdollinen toimiva ratkaisu ja "You Aren't Going to Need It" (tunnetaan nimellä YAGNI). YAGNI on määritelty että “älä tee tänään koodia jota tulet vasta huomenna tarvitsemaan” vapaasti suomennettuna.

Refaktoroinnin vaarana on että yli-suunnitellaan, mutta on kuitenkin vaarattomampaa mieluummin yksinkertaistaa yli tarpeen. Fowlerin mukaan yksinkertaistaminen on tehokkaasti mahdollista kun testaus ja jatkuva integrointi ovat kehityksen osana.

Fowler tuo esille myös suunnittelumallien käytön. Yhden teorian mukaan yksinkertaisuuden noudattaminen johtaa suunnittelumallien käyttöön. Fowlerille suunnittelumallit tarjoavat taustan suunnittelun ymmärrykselle. Fowlerin suositus onkin hankkia tietämystä suunnittelumalleista, jolloin niitä voidaan käyttää työn edetessä tarpeen mukaan.

Mitä sitten tarkoitetaan ohjelmistoarkkitehtuurilla, Fowler kysyy ja tarjoaa ajatuksia nähdä asia uudella tavalla. Hänen mukaansa arkkitehtuuria ei ole syytä muodostaa kovin tarkasti vaan se tarkentuu työn edetessä. Alussa on hyvä muodostaa kuitenkin näkemys siitä miten ohjelmiston eri kerrokset jakautuvat, miten käsitellään tietokantaa ja miten palvelut jakautuvat. Fowler ajattelee näiden alueiden pohjautuvan kokemuksen myötä opittuihin suunnittelumalleihin.Ohjelmistoarkkitehdin rooli on Fowlerin mukaan tarpeeton XP-projekteissa, ryhmän jäsenet vastaavat yhdessä muiden jäsenten kanssa arkkitehtuurin toteutumisesta. Aiemmin ohjelmistoarkkitehdin rooli on ollut erillinen, eikä arkkitehti ole useinkaan osallistunut ohjelmointityöhön. Koska jokainen ryhmän jäsen on vastuussa arkkitehtuurista, tämä asettaa vaatimuksia jäsenten osaamistasoon.

UML-kaavioiden käytöstä hän esittää havaintojaan, osalle ihmisistä ne ovat hyödyllisiä ja toisille taas eivät. UML-kaavioiden roolin, tulisikin hänen mukaansa olla enemmän kommunikoinnin väline. Tarkkoja kuvauksia ei ole syytä tarpeettomasti tehdä, vaan kaavioissa on syytä keskittyä ainoastaan kunkin käyttötarkoituksen vaatimalle tarkkuustasolle. Fowlerin mukaan lähdekoodi on riittävä kokonaisuuden hahmottamiseen. Kaavioita on tarpeen piirtää ymmärtääkseen suunnittelun, mutta ne voidaan sen jälkeen huoletta heittää pois. Usein käytetään pelkkiä CRC-kortteja mutta Fowler käyttää sujuvasti molempia rinnakkain.

Fowler näkee että molemmille suunnittelutavoille on oma paikkansa, tarpeen mukaan on mahdollista tasapainotella näiden kahden välillä. XP-projekteissa hän valitsee mieluummin aina evolutiivisen suunnittelun.

Lähde: Martin Fowler, Is Design Dead?, 2001 – 2004, http://martinfowler.com/articles/designDead.html, viitattu 28.4.2012