Is Design Dead?

Referaatti Martin Fowlerin artikkelista (http://martinfowler.com/articles/designDead.html)

Martin Fowlerin mukaan suunnittelu ei suinkaan ole kuollut; sen käyttö on vain XP:n tapaisten uusien toteutus- ja ajattelutapojen myötä muuttunut olennaisesti. Fowlerin mukaan on kaksi yleistä tapaa suunnitella: evoluutiona kasvata suunnittelu, joka johtaa yleensä katastrofiin, kun sovelluksia suunnitellaan toteutuksen yhteydessä ja vain silloin. Suunnittelun tehtävänä on edesauttaa tekemään sovelluksia joiden rakennetta on helppo muuttaa. Evolutionaarinen ad-hoc suunnittelu toimii yleensä tätä vastaan.

Suunniteltu suunnittelu (heh) muistuttaa perinteisempää insinöörisuunnittelua ja sitä on käytetty pimeältä 70-luvulta alkaen. Sovellus suunnitellaan ensin abstraktilla tasolla esimerkiksi UML:n avulla ja suuret suuntaviivat luodaan etukäteen.

Suunnittelijat tapaavat olla kokeimpia toteuttajia, mutta yleensä suunnittelu vie kaiken ajan, ja nopeasti kehittyvän alan uusimmat muutokset tuppaavat jäämään kokematta - ja siten huomioimatta. Toteuttajat saattavat hyvinkin olla näistä muutoksista selvillä, jolloin sovelluksen tekninen suunnittelu voi vaikuttaa aikansa eläneeltä, mikä on omiaan nostamaan muutosvastarintaa sitä kohtaan. Lisäksi toteuttajat, etenkin ylläpitovaiheessa, valitsevat yleensä helppoja ja suunnittelemattomia ratkaisuja. Kun paine ja kiire ovat kovia, ihminen tapaa valita yksinkertaisimman tien ulos tilanteesta. Ad-hoc suunnittelu tekee tästä helpompaa - ja pidemmän päälle katastrofaalista.

Muuttuvat vaatimukset ovat kuitenkin suurin ongelma. On hyvin vaikea suunnitella sovelluksia niin että mahdollisesti muuttuvat vaatimukset pystytään eristämään muusta toteutuksesta niin, että muutos pysyisi paikallisena. Mutta mikäli näin ei tehdä, muuttuvilla vaatimuksilla voi olla koko sovelluksenlaajuisia vaikutuksia, mikä taas aiheuttaa ennakoimatonta työtä, aikataulupaineita, jne.

Externe Programming on erinomaisen haavoittuvainen evoluutiona kasvavalle suunnittelulle, mille Fowler esittää vastavoimana testauksen (oletettavasti TDD:n), jatkuvan integraation ja refaktoroinnin. Myös suunnittelun suunnittelun pitäisi olla mahdollista. Tärkeintä, ja vaikeinta, on löytää kohta jossa XP:n vapaus, suunnittelu ja nämä välineet ovat tasapainossa.

Fowler korostaa toteutuksen yksinkertaisuutta, eli sitä että kulloinkin on toteutettava vain sillä hetkellä tarpeellinen osa sovellusta - eikä esimerkiksi toiminnallisuutta jonka tietää tulevan myöhemmin vastaan. Vaikka toiminnallisuuden lisääminen olisi triviaalia, ei voida olla varmoja ettei sitä jouduta muuttamaan myöhemmin (esimerkiksi refaktoroinnin vuoksi), jolloin muutostyöllä on hintansa. Jos toteuttaminen on triviaalia, toiminnallisuus voidaan yhtä hyvin toteuttaa vasta kun sitä todella tarvitaan.

Yksinkertaisuus tarkoittaa tässä yhteydessä neljää asiaa; kattavaa ja toimivaa testausta, toteutusta joka paljastaa mihin sillä pyritään, koodin kertautumattomuutta ja mahdollisimman vähäistä määrää luokkia tai metodeja. Paljastavuus on näistä vaikeinta saavuttaa, koska koodin

pitäisi olla luettavaa ja ymmärrettävää niin noviisille kuin kokeneellekin kehittäjälle.

Fowler sivuaa myös design pattereneita ja tiivistää: niiden oppimiseen kannattaa käyttää aikaa, niitä kannattaa soveltaa vasta kun toteutus on riittävän kypsää, ensin kannattaa soveltaa kevyesti - eikä pidä pelätä niiden poistamista mikäli lopputulos ei ole haluttu.

Arkkitehtuurin osalta Fowler suosittelee ainakin jonkinlaisen arkkitehtuurin luonnostelua, mikäli sovelluksesta tiedetään alussa tarpeeksi. UML:ää ja mallintamista on syytä käyttää keskeisempien asioiden kuvaamiseen mikäli tekijöillä on halua niiden käyttöön. Kaikkea ei kuitenkaan pidä yrittää kuvata. Sovellusten dokumentointiin kuvaukset käyvät vain mikäli ne ovat helposti ylläpidettäviä, helposti löydettäviä ja ihmiset todella käyttävät niitä. Jos kuvauksia ei käytetä, niillä voi heittää vesilintua.

Arkkitehdin roolin - sikäli kun sellaista edes pitäisi olla olemassa - Fowler näkee mentorin roolina, jossa autetaan tuoreempia kehittäjiä kasvamaan ja ottamaan vastuuta suunnittelustaan. Arkkitehdin roolissa painottavat opettaminen ja ryhmätyötaidot. Yleensä arkkitehdin vastuulle jää suunnittelun ja toteutuksen laadusta huolehtiminen.

Suunnittelun puuttumisen - tai huonon suunnittelun - huomaa yleensä siitä että teknisen ihmiset valittavat muutoksen tekemisen vaikeudesta. Muutoksen yhteydessä vanhaa koodia yleensä heitetään pois; mikäli näin ei tapahdu, ei välttämättä tapahdu refaktorointia, mikä voi olla ongelma.

Lopuksi Fowler toteaa että suunnittelu ei ole kuolluttua, se on vain muuttunut. XP:n myötä suunnittelutaitoja ovat muun muassa:

- Halu pitää toteutus yksinkertaisena ja siistinä.
- Taito ja halu tehdä refaktorointeja jotta toteutusta voidaan tarvittaessa parantaa.
- Hyvä ymmärrys design patterneista.
- Suunnittelu tulevaisuuden muutoksia silmällä pitäen.
- Kommunikointitaidot, kyky välittää suunnittelun tulos muille.