Teksti: Janne Korhonen

Kirjoittaja Janne Korhonen toimii Business Development Managerina Fujitsu Services Oy:ssä. Hän työskentelee tällä hetkellä terveydenhuollon sovellusliiketoiminnan parissa.

janne.korhonen @fi.fujitsu.com

Tuottavuuden unohdettu ulottuvuus

Luovaa työtä

Muistan urani alkuajoilta Juhan. Juha ei varsinaisesti ollut Java-ohjelmoija, kuten meidän nelihenkinen tiimimme toimiston samassa siivessä. Jo pitkän uran c-ohjelmoijana tehnyt Juha oli kuitenkin tutustunut Java-kieleen töidensä ohessa. Juhan taidot paljastuivat, kun tiimillämme oli vaikeuksia saada erästä projektia aikataulussa valmiiksi. Juha auttoi tekemällä omien töidensä ohessa kirjaston, jonka avulla saimme projektin valmiiksi muutamassa päivässä. Juhan tuottavuus jätti meidät sanattomaksi. Käytännössä Juha olisi toteuttanut koko projektin samassa ajassa kuin meidän neljän hengen tiimimme. Juhan tuottavuus oli siis ainakin viisinkertainen meihin verrattuna.

Tällaiset yksilöiden väliset tuottavuuserot ovat tuttuja kaikille ohjelmistoja työkseen tehneille. Useiden tutkimusten mukaan saman kokemuksen omaavien ammattilaisten väliset tuottavuuserot ovat tyypillisesti vähintään kymmenkertaisia. Gerald Weinberg nosti tämän ohjelmistotuotannon inhimillisen ulottuvuuden esille jo vuonna 1971 ilmestyneessä kirjassaan The Psychology of Computer Programming. Viime vuosikymmeninä tämä havainto on kuitenkin jäänyt taustalle, ja ohjelmistokehityksen tuottavuutta on pyritty parantamaan lähinnä prosesseja kehittämällä.

Perinteisen prosessiajattelun näkökulmasta ohjelmiston kehittäminen jakaantuu yleensä suunnittelu- ja toteutusvaiheisiin. Tämä kahtiajako juontaa juurensa perinteisestä insinöörityöstä. Perinteisessä teollisuudessa insinöörit tekevät suunnitelmat, joiden mukaan tuotanto toistaa itseään. Ohjelmistotuotanto sen sijaan on alusta loppuun luovaa suunnittelutyötä. Sen päämääränä on monimutkaisten abstraktien käsitemallien ja prosessien kuvaaminen tietokoneelle täsmällisessä muodossa. Täsmällisyys erottaa ohjelmistosuunnittelun monesta muusta luovasta työstä. Tietokone ei ymmärrä kuin täsmällisiä ohjeita. Usein kuulee puhuttavan, ettei ohjelmoijia enää tarvita, kun tietokoneet oppivat ymmärtämään luonnollista kieltä. Tämä päättely on virheellinen. Vaikka tietokone ymmärtäisi luonnollista kieltä, olisi jonkun kuvattava sille täsmällisesti, kuinka toimia missäkin poikkeustilanteessa. Ohjelmistotuotanto ei siis sisällä itseään toistavaa vaihetta, jota voisi optimoida taylorismin oppien mukaan.

Ketterät menetelmät ovat uudelleen nostaneet esiin ohjelmistokehityksen luovana työnä. Ne korostavat yksilöiden merkitystä ohjelmistotyön tuottavuuden osana. Harvoin ohjelmistosuunnittelua kuitenkaan tarkastellaan yksilön tuottavuuden näkökulmasta. Tunnettu menestysresepti kuuluukin: ”Some people deliver software some don’t. Hire people who do.”

Hullu töitä tekee

Veljeni aloittaessa ensimmäisessä kesätyössään tehtaan linjalla hän oli innoissaan tarjotusta urakkapalkasta. Mikäli hän taittelisi tietyn määrän aihioita, palkan päälle tulisi reilu bonus. Ensimmäisen päivän jälkeen veljeni olikin ylittänyt tavoitteet reilusti ja odotti vähintäänkin suurta kiitosta. Kiitosten sijaan hän joutui työkavereidensa puhutteluun. Uuden kesätyöntekijän ei ollut soveliasta olla liian tuottelias. Samalla tahdilla jatkaessaan hän nostaisi kaikilta vaadittavan suorituksen tasoa.

Luovaa työtä tehdessä aika kuluu ongelmien ratkomiseen. Näitä ongelmia on mahdoton ennakoida, ja ulkopuolisen on vaikeaa seurata työn etenemistä. Tekijällä on tällöin paljon valtaa. Tätä valtaa on helppo käyttää. Kun motivaatio alkaa laskea, on helppoa laskea tuottavuutta. Usein tämä tapahtuu tiedostamatta. Useat tutkimukset ovat osoittaneet, että yksilöiden motivaatio on tärkein tuottavuuden mittari luovaa työtä tehdessä. Lyhyellä tähtäimellä motivoituneet ihmiset tekevät työnsä tehokkaammin ja pitkällä tähtäimellä he kehittyvät ammatillisesti nopeammin. Tämän seurauksena yksilöiden väliset tuottavuuserot kasvavat pitkällä tähtäimellä jopa kymmenkertaisiksi.

Steve McConnell esittelee ohjelmistokehittäjien motivaatiosta tehtyä tutkimusta kirjassaan Rapid Development. Tutkimuksen mukaan ohjelmistosuunnittelijoiden kolme tärkeintä motivaation lähdettä ovat asetettujen tavoitteiden saavuttaminen, mahdollisuus inhimilliseen kasvuun ja työ itsessään.

Ohjelmistosuunnittelijat ovat erittäin motivoituneita saavuttamaan asetetut tavoitteet. Parhaiten motivoivat itse asetetut tavoitteet. Tämän vuoksi kannattaa esimerkiksi ottaa suunnittelijat mukaan aikatauluja tekemään. Hyvä tavoite on realistinen ja selkeä, eikä se muutu kesken tavoitejakson. Tavoitteiden tulee olla myös linjassa keskenään.

Tehokkainta on asettaa vain yksi tavoite. Tavoitteet ja tulokset on myös käytävä läpi ennen ja jälkeen tavoitejakson.

Ohjelmistoammattilaisen osaaminen vanhenee nopeasti. Ohjelmistosuunnittelu houkutteleekin ihmisiä, jotka haluavat oppia jatkuvasti uutta. Organisaatio voi hyödyntää tätä motivaation lähdettä yhdistämällä yksilöiden ja yritysten tavoitteet. Motivoivimpia ovat tehtävät, jotka pakottavat työskentelemään osaamisen äärirajoilla. Haasteellisia tehtäviä tarjoava organisaatio myös houkuttelee motivoituneimpia ja parhaita ammattilaisia.

Ohjelmistosuunnittelu itsessään on hyvin palkitsevaa työtä. Esimerkiksi Linus Torvalds on todennut tehneensä Linuxin huvin vuoksi. Ohjelmistosuunnittelu tarjoaa mahdollisuuden toteuttaa vapaasti omaa luovuuttaan. Monimutkaisten ongelmien ratkominen ja mahdollisuus jatkuvaan uuden oppimiseen tekee ohjelmistosuunnittelusta poikkeuksellisen mielenkiintoisen ammatin. Organisaatio voi hyödyntää tämän esimerkiksi vähentämällä pakollisia hallinnollisia tehtäviä, jotta suunnittelija voi keskittyä varsinaiseen työhön.

Richard Hagmanin ja Greg Oldhamin mukaan ihmiset kokevat työnsä motivoivaksi, kun se tuntuu merkitykselliseltä, he tuntevat kantavansa vastuun työnsä tuloksista ja he saavat nähdä työnsä konkreettiset tulokset. Tämän vuoksi on tärkeää, että suunnittelijat toimivat läheisessä yhteistyössä asiakkaiden kanssa. Samalla hiljainen tieto siirtyy asiakkailta suunnittelijoille ja tarvittavan dokumentaation määrä laskee. Kun suunnittelijat ja asiakkaat mieltävät ohjelmistoprojektin yhteiseksi oppimisprosessiksi päästään parhaaseen tulokseen.

Hiekkaa rattaisiin

Suunnittelijan motivaatio on myös helppo tuhota. Fred Herzberg kutsuu motivaatiota laskevia tekijöitä kahden tekijän motivaatioteoriassaan hygieniatekijöiksi. Tällaisia tekijöitä ovat esimerkiksi tehtäviin soveltumaton työympäristö. Mikäli hygieniatekijät ovat kunnossa, ei niitä parantamalla voi enää nostaa motivaatiota, mutta niiden ollessa puutteellisia motivaatio laskee. Ohjelmistosuunnittelijan motivaatio on helppo viedä esimerkiksi riittämättömällä laitteistolla tai minimalistisella perehdytyksellä.

Epäpätevä johtaminen vie ohjelmistosuunnittelijoiden motivaation. Motivaatio katoaa varsinkin, jos suunnittelijoita pyritään ohjaamaan heidän omalla osaamisalueellaan tai heidän työtään ei arvosteta. Suunnittelijat täytyy ottaa mukaan heitä koskevien päätösten tekoon. Luovan työn tekijälle ei voi kertoa, mitä pitäisi tehdä. Heille täytyy kertoa, mikä on tavoite ja tarjota riittävät resurssit. Jos ymmärtää ohjelmistotuotannon luovana oppimisprosessina, sen johtaminen on helpompaa.

Ohjelmistosuunnittelijat liittävät ammatillisen itsetuntonsa työnsä laatuun, ei sen määrään. Usein ohjelmistoprojektien ongelmat syntyvät, kun vaikea asia sivuutetaan toiminnallisessa määrittelyssä ylimalkaisella kuvauksella ja sen monimutkaisuus tulee esille vasta, kun se pitäisi kuvata täsmällisesti ohjelmointikielellä. Yleensä tässä vaiheessa ei voida enää palata määrittelyn alkuun ja korjata ongelmia käsitemallien monimutkaisissa riippuvuussuhteissa. Kun tyydytään ratkaisemaan ongelma vain poikkeustapauksen ratkaisevalla oikopolulla, menetetään samalla suunnittelijoiden sitoutuminen ja motivaatio. Tätä ongelma voidaan pienentää ketterien menetelmien keinoin iteroimalla ja jatkuvalla refaktoroinnilla. Suunnittelijoiden vastuualueet on hyvä jakaa selkeästi niin, että he voivat huolehtia oman kokonaisuutensa laadusta, vaikka jokin muu osuus laajassa ohjelmistossa ei olisikaan täydellinen. Kokonaisuuksien toimivuutta voidaan valvoa esimerkiksi automatisoidulla testaamisella.

Ohjelmistosuunnittelu on luovaa työtä, jonka tuottavuus perustuu suurelta osin suunnittelijoiden motivaatioon. Tämä inhimillinen ulottuvuus on hyvä pitää mielessään ohjelmistotuotantoprosessien tuottavuutta parannettaessa. Uskonkin, että lähivuosina tullaan kaukoulkoistusten vastapainona näkemään motivoituneiden huippusuunnittelijoiden varaan rakennettujen pienten ohjelmistotalojen lopullinen läpilyönti.

Luettavaa kiinnostuneille:

Alistair Cockburn, Agile Software Development, Addison-Wesley

Frederick P. Brooks, The Mythical Man Month

Gerad M. Weinberg, The Psychology of Computer Programming, Van Nostrand Reinhold Company

Herzberg, F. 1987, 'One More Time: How Do You Motivate Employees?' Harvard Business Review September-October 1987

Jack W. Reeves, Code as Design: Three essays

http://www.developerdotstar.com/mag/articles/reeves_design_main.html

Martin Fowler, The New Methodology

http://www.martinfowler.com/articles/newMethodology.html

Pekka Himanen, Hakkerietiikka ja informaatioajan henki, WSOY

Tom DeMarco & Timothy Lister, Peopleware Productive projects and teams, Dorset House Publishing & Co