Miika Lahti

JÄRJESTELMÄKEHITTÄJÄN OSAAMINEN PROJEKTIN ONNISTUMISEN NÄKÖKULMASTA

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2019

TIIVISTELMÄ

Lahti, Miika
Järjestelmäkehittäjän osaaminen projektin onnistumisen näkökulmasta
Jyväskylä: Jyväskylän yliopisto, 2019, 26 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaajat: Luoma, Eetu & Taipalus, Toni

Ohjelmistoprojektien onnistuminen ei ole itsestään selvä asia. On tyypillistä, että projektien aikataulu, laatu tai resurssit eivät vastaa suunniteltuja arvoja. Järjestelmäkehittäjän rooli nähdään merkittävänä ohjelmistoprojektin onnistumisen kannalta. Tässä kandidaatintutkielmassa tarkastellaan järjestelmäkehittäjän osaamisen roolia projektin onnistumisen näkökulmasta ja pyrittiin löytämään tärkeimpiä osaamisalueita, jotka auttavat projekteja onnistumaan. Tutkielma toteutetaan kirjallisuuskatsauksena ja sen tärkeimpiä löydöksiä ovat järjestelmäkehittäjän sosiaalisten taitojen merkitys ja liiketoiminta-alueen tuntemus. Sosiaaliset taidot ovat varsin tarpeellisia, jotta järjestelmäkehittäjä saa hyödynnettyä tietoa tehokkaasti ympäriltään. Kehittäjä, jolla on vahva osaaminen liiketoiminta-alueesta ei kuitenkaan tarvitse yhtä paljon yhteistyötä liiketoiminta-alueen asiantuntijoiden kanssa. Heikomman liiketoiminta-alueen tuntemuksen omaava järjestelmäkehittäjä ei kuitenkaan välttämättä kasvata projektin epäonnistumisen riskiä, sillä kehittäjä pystyy haalimaan tarvittavaa tietoa ympäriltään juuri sosiaalisten taitojen avulla. Tämä löydös tukee olemassa olevaa tietoa järjestelmäkehittäjän viestintätaitojen merkityksestä ja voi ohjata jatkotutkimusta sen osalta.

Asiasanat: järjestelmäkehittäjä, sovelluskehittäjä, osaaminen, kompetenssi, ohjelmistoprojekti, projektin onnistuminen, tiedon integraatio

ABSTRACT

Lahti, Miika
System developer competence in project success
Jyväskylä: University of Jyväskylä, 2019, 26 p.
Information Systems, Bachelor's Thesis
Supervisors: Luoma, Eetu & Taipalus, Toni

Software project success in not a matter of course. It is typical that the schedule, quality, or resources of the project does not match with the planned. The role of the system developer is seen as a significant for the success of the project. This bachelor's thesis examines system developer's competence's role in project success' point-of-view and sought to find the most important knowledge areas that help projects to thrive. This thesis was conducted as a literature review and its key findings was the importance of social skills and domain knowledge of the system developer. Social skills are necessary for the developer in order to utilize efficiently the information that is spread around. However, system developer with strong domain knowledge will not need as much collaboration with other domain experts. A developer with weaker domain knowledge does not necessarily increase the risk of the project failure, because with strong communication skills developer can scrape together the spread vital domain information for the project with socials skills. These findings support existing knowledge of the importance of system developer communication skills and can guide further research in this regard.

Keywords: system developer, software developer, knowledge, competence, software project, project success, knowledge integration

KUVIOT

KUVIO 1 Osaamispyramidi 10
KUVIO 2 Ohjelmistoprojektin elinkaari 15

SISÄLLYS

TIIVISTELMÄ
ABSTRACT
KUVIOT

1 JOHDANTO 6

2 JÄRJESTELMÄKEHITTÄJÄN OSAAMINEN JA TYÖKALUT 8
2.1 Osaaminen 8
2.2 Järjestelmäkehittäjän osaaminen 11
2.3 Järjestelmäkehittäjän työkalut 13

3 PROJEKTI JA SEN ONNISTUMINEN 14
3.1 Ohjelmistoprojekti 14
3.2 Onnistunut ohjelmistoprojekti 16

4 JÄRJESTELMÄKEHITTÄJÄN ROOLI PROJEKTIN ONNISTUMISESSA 19
4.1 Järjestelmäkehittäjän osaaminen projektissa 19
4.2 Järjestelmäkehittäjän motivaatio 20
4.3 Yhteistyö asiakkaan kanssa 20

5 YHTEENVETO 22

LÄHTEET 24

1 JOHDANTO

Ohjelmistoprojektin onnistuminen ei ole koskaan itsestäänselvyys, ja epäonnistuessaan projekti voi aiheuttaa yrityksille resurssien tuhlaantumista. (El Emam & Koru, 2008; Agarwal & Rathod, 2006) Yksi keskeinen menestystekijä ohjelmistoprojektille on järjestelmäkehittäjä, ja siksi onkin tärkeää selvittää, mitkä ja millä tavoin järjestelmäkehittäjän taidot vaikuttavat ja edesauttavat ohjelmistoprojektin onnistumista.

Tässä tutkielmassa käsitellään, mitä taitoja järjestelmäkehittäjällä on, ja mitkä niistä liittyvät olennaisesti ohjelmistoprojektin onnistumiseen. Järjestelmäkehittäjällä tarkoitetaan tässä tutkielmassa henkilöä, jonka tehtäviin kuuluu ohjelmistojen laatiminen (ohjelmoiminen) sekä niiden testaaminen. Järjestelmäkehittäjästä voidaan käyttää myös nimityksiä sovelluskehittäjä ja ohjelmoija. Järjestelmäkehittäjästä puhuttaessa tarkoitetaan usein suurempiin järjestelmiin keskittyvää kehitystä. (MOT, 2016)

Tutkielma toteutetaan kirjallisuuskatsauksena, jonka avulla pyritään löytämään linkki projektin onnistumista tutkivien tutkimusten ja ohjelmistokehittäjän osaamista tarkastelevien tutkimusten välille. Tietojärjestelmäkehittäjän osaamista projektin onnistumisen näkökulmasta on tärkeää tutkia, sillä tuottamalla tietoa mahdollisista keinoista välttyä projektien epäonnistumisilta yritykset saavat eväitä projektien onnistuneeseen toteuttamiseen ja ammattilaisten valintaan.

Tutkimuksen tärkein löydös on järjestelmäkehittäjän liiketoiminta-alueen tietämys ja sen vaikutus projektin onnistumiseen. Järjestelmäkehittäjä, jolla on vahva liiketoiminta-alueen osaaminen, ei tarvitse yhtä paljon yhteistyötä järjestelmän loppukäyttäjän liiketoiminta-alueen asiantuntijan kanssa. Heikomman osaamisen omaava järjestelmäkehittäjä ei kuitenkaan välttämättä tarkoita suurempaa projektin epäonnistumisen riskiä, jos omaa vahvat sosiaaliset taidot, sillä kehittäjä pystyy haalimaan tarvittavaa tietoa ympäriltään. Tämä löydös tukee olemassa olevaa tietoa järjestelmäkehittäjän viestintätaitojen merkityksestä ja voi ohjata jatkotutkimusta sen osalta.

Seuraavassa luvussa käsitellään, mitä on osaaminen ja miten se ilmenee järjestelmäkehittäjän työssä. Tämän lisäksi luvussa esitellään järjestelmäkehittä-

jän käytössä olevia työkaluja. Luvussa 3 käsitellään projektia, mikä se on ja miten ohjelmistoprojekti määritellään onnistuneeksi. Luvussa 4 yhdistetään osaaminen, onnistunut ohjelmistoprojekti ja se, minkälainen rooli yksittäisellä järjestelmäkehittäjällä on projektin onnistumisen näkökulmasta. Viimeisessä luvussa esitellään yhteenveto tutkielmasta ja esitellään tutkimustulokset ja johtopäätökset.

2 JÄRJESTELMÄKEHITTÄJÄN OSAAMINEN JA TYÖKALUT

Tässä luvussa tarkastellaan, mitä on osaaminen ja miten se ilmenee järjestelmäkehittäjän työssä. Tämän lisäksi luvussa esitellään järjestelmäkehittäjän käytössä olevia työkaluja.

2.1 Osaaminen

Osaamisesta, varsinkin yksilötasoisesta, puhutaan usein monenlaisilla eri käsitteillä kuten pätevyys, kyvykkyys, kompetenssit tai ammattitaito (Viitala, 2008, s. 113). Näistäkin yksinään kompetenssille on useita erilaisia määritelmiä (Hanhinen, 2010). Garavanin ja McGuiren (2001) tekemässä tutkimuksessa linjataan, että kompetenssit viittaavat yksittäisen työtehtävän suorittamisen valmiuksiin. Kyvykkyyden he taas linjaavat tarkoittavan valmiuksia laajemmalla mittakaavalla, eli esimerkiksi minkälaisia kommunikointi-, johtamis-, päätöksentekokykyjä yksilöllä on. Viitalan mukaan (2008, s. 113) kompetenssi ja ammattitaito ovat sama asia; hyvä ammattitaito koostuu tiedoista, taidoista, valmiuksista sekä asenteista. Ammattitaito määritellään työtehtävien hyvään hallintaan, jossa työntekijä on kyvykäs suoriutumaan tehtävästään itsenäisesti ja on vastuussa omasta tuloksestaan. (Viitala, 2008)

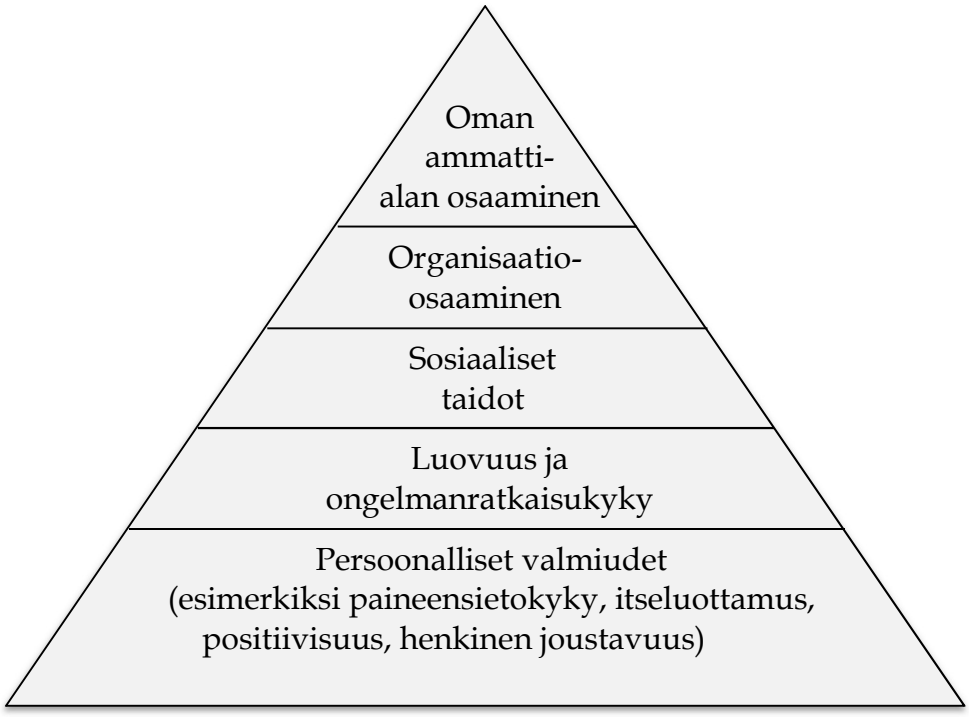
Yritystasolla osaamisesta puhutaan usein yleisellä tasolla, mutta yksilötasolla osaamiseen liittyen tulee ottaa huomioon, mitä kriteereitä yksilön pätevyyden tulisi täyttää. Viitala (2008, s.109) vertaa yksilöiden henkilökohtaista osaamista varantoon, jonka rajoissa yrityksen toimintamallit, prosessit, rakenteet ja muu organisaation sisällä näkyvä tietämys kehittyvät (Viitala, 2008, s. 109-110). Yrityksen perustehtävän ja strategioiden toteutumiseksi tulisi hyödyntää kaikkien jäsenten osaamista. Viitala (2008, s. 109) näkeekin osaamisen yrityksen perustana.

Viitala (2008, s. 109-110) käyttää kirjassaan esimerkkinä moninaisesta osaamisesta yrityksen sisällä sadan hengen yritystä, jossa kullakin työntekijällä on oma työtehtävä, jota harvoin kukaan muu kuin itse työntekijä voi täysin hallita. Kullakin työntekijällä yrityksessä on osaamista, jonka siirto toiselle työntekijälle nopeasti on usein mahdotonta. Tämä korostuu varsinkin aloilla, joilla työntekijän työ vaatii paljon tietotaitoa ja työtehtävät eivät ole toisteisia ja mekaanisia. Tästä esimerkkinä järjestelmäkehittäjä, jonka takana voi olla paljon työtehtävään vaadittavaa osaamista verrattuna postin lajittelijaan. Työntekijällä tulee siis olla osaamista tietyllä tasolla, jotta yrityksen toiminnan tasokkuus on riittävää. (Viitala, 2008, s. 110.)

Osaamisen tarkastelu muuttuu vaikeammaksi siirryttäessä yleisen tason osaamisnimikkeistä yksilön osaamisen tasolle. Viitala (2008, s. 110) käyttää esimerkkinä johtamisosaamista. Yrityksen tasolla se voi olla yksi strateginen osaamisalue, mutta yksilötasolla johtamis-osaaminen voi koostua monista erilaisista valmiuksista, joita ei välttämättä opita koulutuksen kautta. (Viitala, 2008, s. 110.)

Suppeisiin ja staattisiin työnkuviin verrattuna monimutkaistuvassa työssä teoreettinen tieto voidaan nähdä tarpeellisempana. Viitala (2008, s. 111) luettelee kirjassaan seuraavat kyvyt monimutkaistuvan työn näkökulmasta tarpeellisiksi: ongelmien ratkaisukyvyt, tavoitteiden asettamiskyvyt, suunnittelukyvyt, arviointikyvyt sekä kyvyt löytää erilaisia keinoja ongelmien ratkaisemiseksi. Viitala (2008, s. 111) viittaa kirjassaan Hackerin (1987) tekemään tutkimukseen, jossa yli keskiarvosuorituksia tehneet työntekijät osoittivat parempaa kykyä ennakoida ja suunnitella toimintaansa sekä nähdä työnsä osana kokonaiskuvaa. Tämä johti mahdollisuuteen joustaa ja ennakoida työssä sekä tukeutua teoreettiseen tietoon ongelmatilanteissa. Tämän lisäksi se johti myös mahdollisuuteen pyrkiä ymmärtämään paremmin ongelmien taustalla olevia tekijöitä ja tätä kautta luoda uusia ratkaisumalleja ongelmiin. (Viitala, 2008.)

Kompetenssin määritelmästä eroten ammattitaitoa määritellään koostuvan erilaisista kvalifikaatioista (eng. *qualification*), jotka ovat osaamisen osa-alueita. Tätä voidaan kuvata esimerkiksi kuvion 1 mukaisesti osaamispyramidilla. Mitä lähempänä pyramidin pohjaa kvalifikaatio on, sitä enemmän se edustaa henkilön persoonallisuutta. Mitä lähempänä pyramidin huippua kvalifikaatio on, sitä enemmän se taas liittyy yksittäiseen työtehtävään. (Viitala, 2008, s. 116.)



KUVIO 1 Osaamispyramidi (Viitalan (2008) mukaan)

Osaamispyramidin lisäksi kvalifikaatioista on muitakin jaotteluita. Esimerkiksi Hanhinen (2010) on jaotellut kvalifikaatiot kolmeen osaan:

1. Tuotannolliset kvalifikaatiot
2. Normatiiviset kvalifikaatiot
3. Innovatiiviset kvalifikaatiot

Tuotannollisista kvalifikaatioista puhuttaessa tarkoitetaan teknisiä työhön liittyviä tietoja ja taitoja. Nämä taidot ovat tarpeen työtehtävän suorittamiseksi ja ne vaikuttavat suoraan yrityksen tuotantoon. Normatiivisista kvalifikaatioista puhuttaessa tarkoitetaan yksilön henkilökohtaisia ominaisuuksia, joita voidaan edellyttää työskentelyssä. Tällaisia ovat esimerkiksi työhön mukautuminen, joustavuus, työhön sitoutuminen, motivaatio ja sosiaalinen kyvykkyys. Innovatiivisia kvalifikaatioita käsiteltäessä voidaan käyttää myös nimitystä kehittävät kvalifikaatiot. Niillä tarkoitetaan muun muassa henkilön kykyä kehittää itseään ja työprosessia sekä kykyä analysoida ja ratkaista ongelmia. (Hanhinen, 2010.)

Sosiaaliset taidot voivat kuitenkin olla ratkaisevimmassa roolissa huolimatta siitä, missä työtehtävässä henkilö on. Muun muassa Viitala (2008, s. 112) perustelee tätä sillä, ettei juuri ole olemassa työtä, joka on täysin riippumatonta muista ihmisistä. Hän kuvaileekin vuorovaikutustaitoja teknisen osaamisen voiteluaineeksi. Osaamisen arviointi ei aina välttämättä anna tarpeeksi painoarvoa yksilön sosiaalisille taidoille. Sosiaalisten taitojen todistamisellekaan ei ole välttämättä edes mahdollisuuksia. (Viitala, 2008, s. 112.)

2.2 Järjestelmäkehittäjän osaaminen

Riittävän osaamisen saaminen ja säilyttäminen voi olla nykyään yhä vaikeampaa alati muutoksessa olevan ympäristön takia (Viitala, 2008, s.112). Tämä korostuu erityisesti ohjelmistoalalla, sillä käytössä olevat teknologiat kehittyvät jatkuvasti ja uusia teknologioita tulee nopealla syklillä järjestelmäkehittäjien käyttöön. Tällaisessa jatkuvasti muutoksessa olevassa työskentely-ympäristössä kokeneemmatkin asiantuntijat voivat usein kohdata työssään tilanteita, joissa virheiden tekeminen on todennäköistä (Viitala, 2008, s.112). Työntekijöiden osaaminen voidaankin nähdä kriittisenä tekijänä organisaation kilpailullisen edun saavuttamiseksi, erityisesti kun kilpailu voi tapahtua globaalilla tasolla. (Rashid ym., 2019)

Järjestelmäkehittäjän tehtäviin kuuluu luotettavien ohjelmien tuottaminen tunnettujen piirteiden mukaisesti sekä ainutlaatuisten teknisten ongelmien ratkonta. Tehtävien toteuttamiseksi järjestelmäkehittäjältä vaaditaan asianmukaista osaamista niin taitojen, koulutuksen kuin ammatillisen kokemuksen pohjalta (Bourque & Fairley, 2014). Bourquen ja Fairleyn (2014) mukaan järjestelmäkehittäjän osaaminen koostuu kolmesta osa-alueesta: ammattitaidosta, ryhmädynamiikasta ja viestintätaidoista. Ammattitaito käsittää kyvyt toteuttaa annettuja tehtäviä. Tästä voi olla todisteena erilaiset sertifikaatit. Kehittäjän tulee myös tuottaa koodia yhteisten sääntöjen mukaan ja tuottaa tekemästään työstä selvää dokumentaatiota. Ammattitaitoon lasketaan myös kehittäjän kyky analysoida eri ratkaisuvaihtoehtoja ongelmaa ratkottaessa. Analysointiprosessissa kehittäjän tulee ymmärtää mahdolliset vaihtoehtoiskustannukset ja pystyä valitsemaan paras tapa ratkaista ongelma näiden puitteissa. (Bourque & Fairley, 2014.)

Ryhmädynamiikka käsittää kehittäjän kyvyt työskennellä tiimissä, sillä iso osa järjestelmistä toteutetaan tiimityönä. Puhuttaessa vuorovaikutuksesta tiimitoverien ja asiakkaiden kanssa kehittäjän tietämys ryhmädynamiikasta ja psykologiasta voidaan nähdä etuna. Useat vaikeat tekniset ongelmat ovat myös helpompi selättää yhteistyöllä. Vuorovaikutustaidot käsittävät kehittäjän kyvyt viestiä hyvin niin suullisesti kuin kirjallisesti. Kehittäjän täytyy pystyä viestimään tehokkaasti niin tiiminsä sisällä kuin eri sidosryhmien kanssa. Viestintä on tärkeässä roolissa, kun kehittäjän tulee analysoida eri ratkaisuvaihtoehtoja. Vastavuoroinen ymmärrys järjestelmän määrittelyssä sidosryhmien välillä voi olla myös avaintekijänä onnistumiselle. (Bourque & Fairley, 2014.)

Havelka ja Merhout (2009) ovat toteuttaneet kenttätutkimuksen, jonka kautta he pyrkivät luomaan teoreettisen mallin IT-alan ammattilaisten toivotuista taidoista, tietämyksistä ja kyvyistä. Tutkimuksen tuloksena Havelka ja Merhout (2009) jakavat osaamisen neljään pääkategoriaan: henkilökohtaisiin piirteisiin, ammattitaitoon, liiketoiminnan tietämykseen ja tekniseen tietämykseen. Henkilökohtaisiin piirteisiin luetaan muun muassa henkilön asenne, luotettavuus sekä kyky toteuttaa annettuja tehtäviä. Tämän kategorian ominaisuudet eivät liity suoraan IT-alaan osaamiseen, vaan ovat ennemmin mahdollistavia tekijöitä. Ammattitaitoon taas luetaan yksilön piirteitä kuten ongelmanrat-

kaisukyky, organisointikyky, vuorovaikutustaidot ja tiimityöskentelytaidot. Nämä ovat taitoja, joita ammattilaisilta odotetaan työelämässä. Jos järjestelmää kehitetään tiimissä, on oletettavaa, että myös tiimin jäseniltä odotetaan taitoa työskennellä yhdessä. Liiketoiminnan tietämys sen sijaan sisältää henkilön kyvyt ymmärtää liiketoiminnan perusperiaatteet ja tuntea organisaation rakenne. Siihen sisältyy muun muassa ymmärrys siitä, mikä on yrityksen ydinliiketoiminta ja mitä sijoitetun pääoman tuottoaste (ROI) tarkoittaa. Neljänneksi pääkategoriaksi Havelka ja Merhout (2009) listaavat teknisen tietämyksen. Tekninen tietämys sisältää tietotekniikan, informaatioteknologian tai tietojärjestelmien hallinnan aihealueet. Siihen kuuluu muiden muassa yksilön tietämys kehitysmetodeista ja -prosesseista, järjestelmän arkkitehtuurista ja infrastruktuurista, ohjelmoinnista ja siinä käytettävistä työkaluista, sekä myös erikoisala, johon luetaan trendissä olevien teknologioiden erikoistietämys (Havelka & Merhout, 2009).

Surakan ja Malmin (2004) tutkimuksessa on listattu ominaisuuksia, joita alan työntekijät pitävät tärkeinä ja ammattitaitoa ilmentävinä. Tärkeimpänä ja tilastollisesti merkittävänä ammattitaitoa ilmentävänä kykynä tutkimuksessa nousi esille kyky nähdä kaikki mahdolliset vaihtoehtoisratkaisut suoraan lähdekoodista. Muina tärkeinä kykyinä vastaajat pitivät kykyä huomata kahden erilaisen koodirivin tekevän saman asian ja siitä mahdollisesti aiheutuvat ongelmat. Tämän lisäksi tärkeänä pidettiin myös kykyä hahmottaa, miten järjestelmä tulee toimimaan jo ennen kuin sitä on edes toteutettu. Tutkimukseen vastasi 11 alalla pitkään työskennellyttä henkilöä, joiden sijoittuminen työelämässä oli jakautunut kolmeen eri ryhmään. Suurin osa (45 %) työskenteli kehittäjinä. Kaksi muuta ryhmää olivat samankokoiset (27 %), ja ne koostuivat tutkijoista ja päällikkötehtävissä työskentelevistä henkilöistä. (Surakka & Malmi, 2004.)

Sovelluskehitys sisältää usein todella monimutkaisia tehtäviä ja onkin hyvin osaamisintensiivistä. Suoriutuakseen kehitystehtävistä kehittäjällä tulee olla tietämystä monelta eri toimialueelta. Kehittäjän tulee muun muassa tuntea ja ymmärtää sovelluksen kattama liiketoiminta-alue. Sen lisäksi kehittäjän tulee ymmärtää kehittämänsä järjestelmän arkkitehtuuri, joka kuvaa, miten järjestelmän eri palaset toimivat keskenään. Näiden lisäksi kehittäjällä täytyy olla tietämystä eri kehitysmetodologioista, tarvittavista algoritmeista, ohjelmointikielestä sekä kehitysympäristöistä, joita tarvitaan ohjelmointitehtävissä. Koska ongelmanratkontakyky on olennainen osa kehittäjän työtä, tulee ongelmaa ratkaistaessa kehittäjän ensiksi ymmärtää ongelma ja sen syyt. Tämän jälkeen kehittäjän tehtävänä on suunnitella ongelmalle ratkaisu ja ohjelmoida se. Kehittäjän ongelmanratkontaprosessissa tarvitaan yhdistelmä edellä mainittuja osaamisalueita. (Rashid ym., 2019.)

Kehittäjän tieto voidaan Rashidin ja hänen kollegoidensa (2019) mukaan jakaa yleiseen ja hiljaiseen tietoon. Avoin tieto on formaalia ja dokumentoitua, johon kaikki pääsevät helposti käsiksi ja joka löytyykin usein kirjoitetussa muodossa. (Rashid ym., 2019) Avointa tietoa voisi olla esimerkiksi järjestelmän dokumentoitu arkkitehtuuri ja liitokset. Myös järjestelmän koodi on avointa tietoa. Hiljainen tieto taas koostuu Rashidin ja hänen kollegoidensa (2019) mu-

kaan henkilöiden taidoista, jotka on opittu heidän henkilökohtaisten kyvykkyyksien kautta ilman dokumentointia. Hiljainen tieto syntyy yksilöiden välisen vuorovaikutuksen seurauksena ja on usein juuri liiketoiminta-alueen tietämystä. Jos yrityksessä on poistumaa, on tärkeää, ettei poistuville henkilöille jää hiljaista tietoa, sillä muuten se voi aiheuttaa organisaatiossa osaamisvajetta. (Rashid ym., 2019.)

2.3 Järjestelmäkehittäjän työkalut

Järjestelmäkehittäjän päätyökaluihin kuuluu tietokoneella käytettävä ohjelmointiympäristö (eng. *integrated development environment,* jäljempänä IDE), joka sisältää kehitystoimintoja järjestelmän rakentamista varten. IDE:n tarkoitus on tehostaa kehittäjän tuottavuutta integroimalla samaan ohjelmaan muun muassa tiedoston muokkaamisen, hallinnan, niiden validoinnin ja kääntämisen sekä ohjelman ajon ja debug-ajon (ohjelman ajo manuaalisesti rivi kerrallaan). Osaamalla tehokkaasti hyödyntää ohjelmointiympäristöjä, kehittäjä pystyy nopeuttamaan koodin tuottamista ja varmistamaan paremmin sen laatua. Ennen kehittyneempiä IDE-ohjelmia koodia kirjoitettiin pelkillä tekstieditoreilla, joka aiheutti ongelmia, sillä niissä ei välttämättä ollut vielä integroitua validointia. (Kerrigan ym., 2007.)

Kehittäjien yleisiin työkaluihin kuuluu myös graafisten käyttöliittymien (eng. *graphical user interface*, jäljempänä GUI) rakennustyökalut, joilla kehittäjät voivat tehokkaasti luoda ja ylläpitää sovelluksen lähdekoodia niin, että he näkevät tekemänsä muutokset suoraan työkalussa. Monet kattavat IDE-ohjelmat sisältävät integroidun GUI:n. GUI sisältää kehitystä nopeuttavia aputoimintoja. Sillä pystyy muun muassa luomaan uusia näkymiä helposti *raahaa ja pudota* -periaatteella graafisia komponentteja visuaalisesti ikkunaan. (Bourque & Fairley, 2014.)

Yksikkötestaus kuuluu myös olennaisesti kehitystyöhön. Se testaa ohjelmiston toiminnallisuuksia irrallaan muusta järjestelmästä. (Bourque & Fairley, 2014) Yksikkötestauksella voidaan varmistaa järjestelmän toiminnallisuus, mistä on apua, kun järjestelmän koodia esimerkiksi refaktoroidaan. (Runeson, 2006) Refaktoroinnilla tarkoitetaan järjestelmän lähdekoodin muuttamista ilman, että itse koodin tuottama lopputulos muuttuu. Prosessissa pyritään parantamaan koodin rakennetta ja yksinkertaistamaan sitä. Tällä pyritään minimoimaan bugien ilmeneminen. (Fowler ym., 1999)

Top IDE Index (2018) -listaus seuraa Google Trendsin kautta, mitä IDE:jä ihmiset syöttävät eniten Googlen hakukoneeseen. Neljä suosituinta olivat vuoden 2018 syyskuussa Visual Studio, Eclipse, Android Studio ja NetBeans. Eri IDE:t ovat usein teknologiakohtaisia. Esimerkiksi Visual Studiolla kehitetään Visual Basic, C#, F#, JavaScript, Python ja C++ kielillä. (About Visual Studio 2017, 2018) Javaa sen sijaan voidaan ohjelmoida Eclipsellä, joka tukee myös C ja C++ kieliä (Eclipse IDE, 2018).

3 PROJEKTI JA SEN ONNISTUMINEN

Tässä luvussa käsitellään projektia, mikä se on ja miten ohjelmistoprojekti määritellään onnistuneeksi.

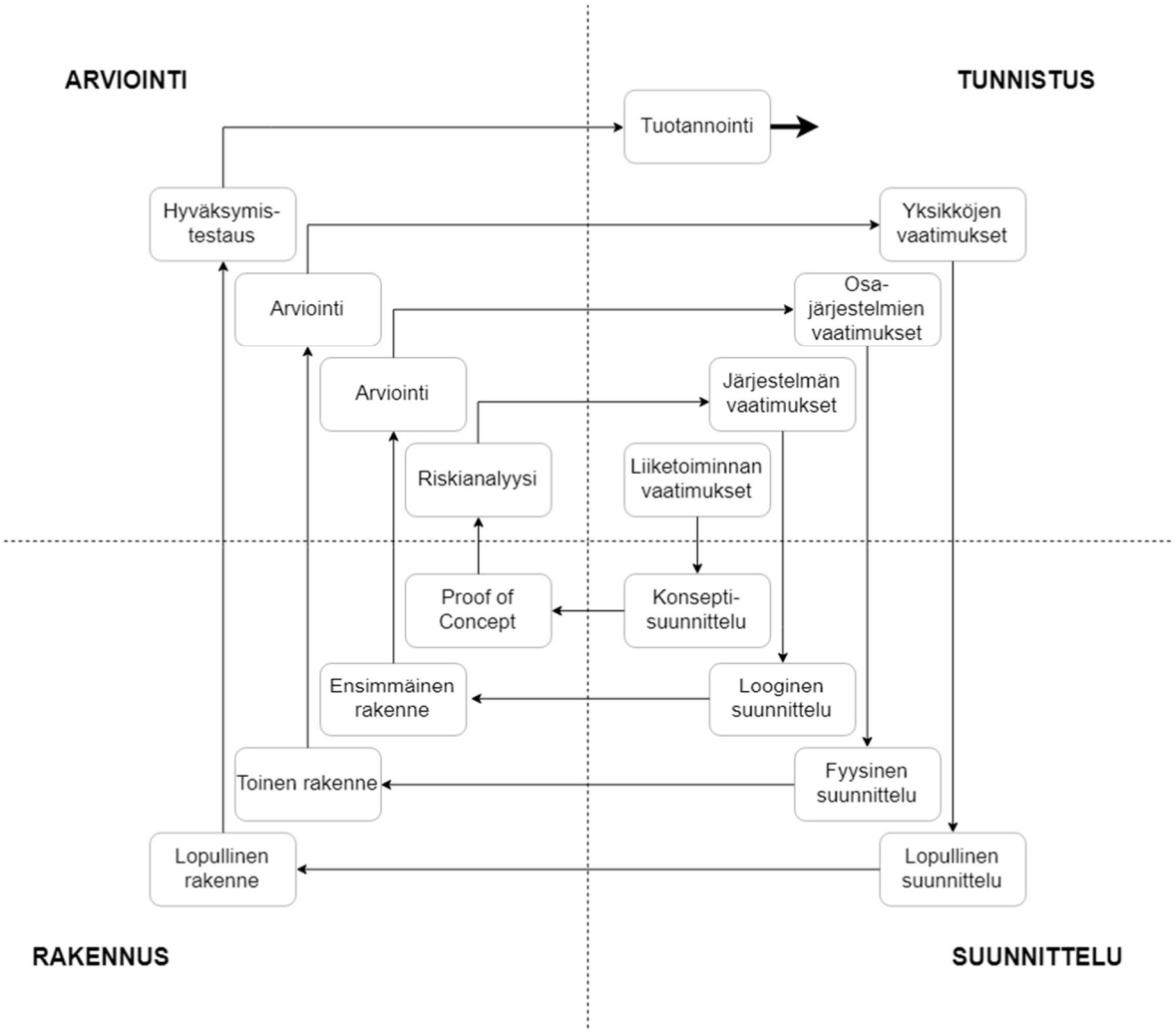
3.1 Ohjelmistoprojekti

Erilaisia projektin määritelmiä on lukuisia. Tässä tutkielmassa käytetään PMBOK:n (Project Management Body of Knowledge) mukaista määritelmää, jossa on pyritty tekemään määritelmästä yleispätevä. Projekti käsitetään aina kertaluontoisena yrityksenä tuottaa ainutkertainen tuote, palvelu tai tulos ja sillä on aina selkeä alku ja loppu. (PMI, 2000)

Projektin alussa perustetaan projektiorganisaatio, joka puretaan projektin päätyttyä. Projektiorganisaatio koostuu projektitiimistä sekä sidosryhmistä. Projektitiimiin kuuluvat projektipäällikkö, johtoryhmä sekä muut projektitiimin jäsenet. Projektipäällikön tehtäviin kuuluu projektin hallinnollinen työ ja hänellä on vastuu projekti tavoitteiden saavuttamisessa. Projektitiimin jäsenet ovat mukana tekemässä tuotetta, mutta eivät välttämättä osallistu projektin hallinnollisissa asioissa. Sidosryhmät taas koostuvat henkilöistä tai organisaatioista, jotka osallistuvat aktiivisesti projektin toteutukseen tai joilla on vaikutusta projektin suorittamisen kannalta. (PMI, 2000.)

Projekti suoritetaan ihmisten toimesta ja sitä ohjaa sille määrätyt resurssit. Näitä resursseja ovat tyypillisesti budjetti ja aikataulu. Budjettiin sisältyy muun muassa työntekijöiden tuottamat kustannukset. Projektin koko voi vaihdella aina yhdestä henkilöstä jopa satoihin. Projekti nähdään loppuneeksi silloin, kun sen tavoitteet ovat täyttyneet, sekä silloin, kun projekti on keskeytetty, koska sen tavoitteita ei ole pystytty täyttämään. Esimerkkejä projekteista voi olla uuden tuotteen kehitys, muutoksen toteuttaminen organisaatiossa tai uuden rakennuksen rakentaminen. (PMI, 2000) Projektin tuotoksena syntyvä tuote, palvelu tai tuotos on aina yksilöllinen. Vaikka projekti nähdään kertaluontoisena ja

tilapäisenä rupeamana päästä asetettuun tavoitteeseen, se ei välttämättä tarkoita, että projektit ovat aina lyhytaikaisia. (PMI, 2000.)



KUVIO 2 Ohjelmistoprojektin elinkaari (Muenchin (1994) mukaan)

PMBOK:ssa (2000) esitetään esimerkki tyypillisestä ohjelmistoprojektin elinkaaresta, joka usein toteutetaan sykleittäin, joissa kussakin tapahtuu tunnistusta, suunnittelua, rakennusta ja arviointia. Ensimmäinen sykli käsittää hankkeenmäärittelyjen ja mahdollisuuksien kartoituksen konseptoinnin. Se alkaa liiketoiminnan tarpeiden tunnistamisella ja jatkuu konseptin suunnittelulla. Kun konsepti on suunniteltu, alkaa konseptin toimivuuden todistaminen (*eng*. *Proof of Concept*, PoC). Ensimmäisen syklin lopuksi arvioidaan riskit. Toinen sykli käsittää ensimmäisen konkreettisen rakennusvaiheen. Se alkaa järjestelmän vaatimusten tunnistamisella ja jatkuu loogisella suunnittelulla, jossa järjestellään loogiset riippuvuussuhteet keskenään. Tämän jälkeen alkaa ensimmäisen rakenteen (eng. *build*) toteutus. Ensimmäisen rakenteen pohjalta tehdään arviointia tuotoksesta ja suositukset seuraavaa sykliä varten. Kolmannen syklin tavoitteena on toteuttaa toinen rakenne, jota varten sykli käynnistyy järjestelmän

eri osien vaatimusmäärittelyllä. Tämän jälkeen toteutetaan fyysinen määrittely, jossa loogisesta määrittelystä poiketen suunnitellaan, minkälaista infrastruktuuria järjestelmä vaatii toimiakseen. Lopulta toteutetaan toinen rakenne, jonka lopuksi taas arvioidaan tuotosta. Neljäs sykli alkaa koko järjestelmän kattavien vaatimusten määrittelyllä ja lopullisen mallin toteutuksella, joiden jälkeen alkaa lopullisen rakenteen toteutus. Kun toteutus on valmis, tehdään hyväksymistestaus, jossa tarkistetaan, että tuote vastaa määrittelyjä ja toimii oikein. Kun tuote hyväksytään, voidaan tuotetta alkaa siirtämään tuotantoon. (PMI, 2000.) Tämä elinkaarimalli on kuvattu kuviossa 2, joka on mukaelma PMBOK:ssa (2000) esitetyssä Muenchenin (1994) elinkaarimallista.

1990-luvun loppupuolelta on yleistynyt, että ohjelmistoprojektit toteutetaan niin kutsutuilla ketterillä (eng. *agile*) menetelmillä (Parsons ym., 2007). Ketteriä menetelmiä käyttämällä pyritään vastaamaan nopeammin liiketoiminnan tarpeisiin tekemällä ohjelmistokehitysprosessista nopeampi ja kevyempi, joka pystyy paremmin vastaamaan muutostilanteisiin. Ketterien menetelmien perusperiaatteet voidaan kiteyttää neljään arvoon; arvostukseen, sykleittäinen toimiminen, yhteistyö asiakkaan kanssa sekä muutoksiin reagoiminen. Ensimmäinen on sovelluskehittäjien ja kehitystiimin arvostus ja hyvän ryhmähengen luomisen tärkeys yli prosessien ja työkalujen. (Abrahamsson ym., 2002.) Ketteriä metodologioita käyttävä kehitystiimi näkee toiminnassaan yksittäisen kehittäjän taitojen merkityksen tärkeänä tekijänä projektin onnistumiseksi (Cockburn & Highsmith, 2001). Toinen on toimivan ja testatun ohjelmiston tuottaminen nopeasti sykleittäin verrattuna kaiken kattavaan dokumentaation. Kolmas on asiakkaan ja kehittäjien työskentely yhdessä yhteisen maalin saavuttamiseksi verrattuna sopimuksien kautta toimivaan neuvottelutyyliseen yhteistyöhön. Neljäs ketterien menetelmien perusarvo on muutokseen reagoiminen. (Abrahamsson ym., 2002.) Yleisimpiä ketteriä menetelmiä on eXtreme Programming (XP), Scrum, Crystal Family of Methodologies (CM) ja Dynamic System Development Method (DSDM). Kaikki nämä perustuvat neljään ketterän kehityksen arvoon tarkoituksenaan vastata muutoksiin nopeasti, tuottamaan korkealaatuista tuotosta, tukemaan toimivaa ja aktiivista vuorovaikutusta eri sidosryhmien välillä, ja näiden kautta tyydyttää asiakasta. (Hamed & Abushama, 2013.)

3.2 Onnistunut ohjelmistoprojekti

Ohjelmistoalalla ohjelmistoprojektin onnistuminen ei ole itsestäänselvyys ja se koetaankin suhteellisen harvinaisena asiana (Agarwal & Rathod, 2006). Moni asiaa tutkiva julkaisu viittaa Standish Groupin (1995) praktiseen CHAOS-raporttiin, jossa esitetään vain 16 %:n ohjelmistoprojekteista valmistuvan aikataulussa pysyen budjetissaan. (Tsui ym., 2016; Agarwal & Rathod, 2006; Berntsson-Svensson & Aurum, 2006) Vaikka raportin esittämien lukujen ja mittariston oikeellisuus on kyseenalaistettu Eveleensin ja Verhoefin (2009) julkaisussa, voi

raportin esittämät luvut kertoa ohjelmistoprojektien ongelmallisuudesta. Ohjelmistoprojektien huonon onnistumisasteen vuoksi aihe on ollut tutkijoiden sekä myös työelämän kiinnostuksen kohteena jo vuosikausien ajan (Svensson & Aurum, 2006).

Agarwalin ja Rathodin (2006) mukaan onnistuminen riippuu usein siitä, millä tavoin se määritellään. Klassinen määritelmä projektin onnistumiselle on se, miten projekti on pysynyt budjetissaan, aikataulussaan ja onko se vastannut liiketoiminnan tavoitteisiin (Berntsson-Svensson & Aurum, 2006). PMBOK:ssa (2009) projektin onnistumiselle määritellään neljä tekijää: tuotteen sekä projektin laatu, aikataulussa pysyminen, budjetissa pysyminen, sekä asiakkaan tyytyväisyysaste. Agarwal ja Rathod (2006) määrittelevät ohjelmistoprojektin onnistuneen, kun se on tuottanut tuotteen, jonka laatu vastaa ennalta hyväksyttyjä ehtoja annetuissa aika- ja resurssimääreissä. Neljä tärkeintä projektin onnistumiseen johtavaa tekijää on Tsuin ym. (2016) mukaan loppukäyttäjän osallistuminen projektiin, johdon tuki, selkeä vaatimusmäärittely ja kunnon suunnittelu.

Vaikka projekti täyttäisikin onnistuneen projektin kriteerit, voi se silti olla epäonnistunut. Tämän vuoksi projektin onnistumista tuleekin määritellä projektin hallinnan ja projektin tuotteen onnistumisen summana. Berntsson-Svensson ja Aurum (2006) ovatkin löytäneet monesta tutkimuksesta suhteen näiden kahden tekijän välillä. Projektin lopputuote voi silti olla onnistunut, vaikka projektin aikataulu ja budjetti eivät ole pitäneet - ja toisin päin. Tällainen tilanne syntyy esimerkiksi silloin, kun projektin lopputuotetta ei otetakaan käyttöön johtuen muuttuneesta liiketoiminnan tilanteesta. (Berntsson-Svensson & Aurum, 2006.)

Ohjelmistoprojektin onnistuminen on usein myös alasta riippuvainen. Berntsson-Svenssonin ja Aurumin (2006) tekemässä tutkimuksessa selvitettiin kolmelta eri IT-alan osa-alueelta, mitkä olivat kolme tärkeintä tekijää projektin onnistumiselle ja projektin tuotteen onnistumiselle. Finanssialalla kolmeksi tärkeimmäksi tekijäksi nousivat asiakkaan osallistaminen projektiin, sitoutunut rahoittaja ja hyvä vaatimusten määrittely. Tuotteen osalta tärkeimpiä tekijöitä olivat asiakkaan tyytyväisyys, tuotteen hyvä laatu, ja että tuote tyydyttää ylemmän johdon. Konsultointialalla projektin tärkeimmiksi tekijöiksi onnistumisen kannalta valikoitui hyvä projektipäällikkö, asiakkaan tarpeiden ymmärtäminen sekä hyvin määritelty viestintä sidosryhmien välillä. Tuotteen osalta konsultointialalla tärkeimpiä tekijöitä olivat asiakkaan tyytyväisyys, tuotteen toimivuus ja toimittajasta koituva taloudellinen hyöty. (Berntsson-Svensson & Aurum, 2006.) Televiestintäalalla tärkeimmiksi projektin onnistumisen tekijöiksi Berntsson-Svensson ja Aurum (2006) esittivät hyvät suhteet henkilöstön kanssa, asiakkaan tarpeiden ymmärtäminen sekä hyvät ja tarkat vaatimukset. Tuotteen osalta televiestintäalalla tärkeimpiä tekijöitä olivat asiakkaan tyytyväisyys, asiakkaan palaaminen ja tuotteen toimivuus (Berntsson-Svensson & Aurum, 2006).

Koska iso osa ohjelmistoprojekteista toteutetaan ketteriä kehitysmetodologioita noudattamalla, ovat Chow ja Cao (2008) toteuttaneet tutkimuksen, jossa

selvitettiin ketterien ohjelmistoprojektien onnistumisen kannalta kriittisiä tekijöitä. Selviä tekijöitä tutkimuksessa löytyi kuusi kappaletta:

1. Toimitusstrategia
2. Ketterä ohjelmistotekniikka
3. Tiimin kyvykkyydet
4. Projektinhallinnan prosessi
5. Tiimin ympäristö
6. Asiakkaan osallistuminen

Ensimmäinen tekijä, toimitusstrategia, merkkaa sovelluksen ja sen osien toimitusta asiakkaalle. Toimituksen tulisi olla säännöllistä ja tärkeimmät ominaisuudet tulisi toimittaa ennen muita. Toisessa tekijässä eli ketterässä ohjelmistotekniikassa ohjelmoinnin tulisi noudattaa tarkkaan määriteltyä kehitysstandardia, pyrkiä yksinkertaiseen suunnitelmaan ja optimoida dokumentoinnin määrää. Kolmas tekijä, tiimin kyvykkyydet, kertoo kehitystiimin valmiuksista vastata haasteisiin. Tiimin jäsenillä tulisi olla korkea osaaminen ja motivaatio, ja tiimin päälliköiden tulisi pystyä toimimaan ketterästi ja vastaamaan nopeasti mahdollisiin muutoksiin. Neljännen tekijän, projektin hallinnan prosessin, tulisi olla ketterän toiminnan mukainen ja sen pitäisi pystyä monitoroimaan projektia hyvin ja tehostaa vuorovaikutusta projektin sisällä. Viidennen tekijän, tiimin ympäristön, tulisi ruokkia yhteistyötä ja itseohjautuvuutta, eikä tiimi saisi olla liian iso. Kuudes kriittinen tekijä ketterän ohjelmistoprojektin onnistumisessa on asiakkaan osallistuminen projektiin. Asiakasta tulisi osallistaa täysivaltaisena vahvasti projektiin, sekä asiakkaan kanssa tulisi pitää hyvät suhteet. (Chow & Cao, 2008.)

4 JÄRJESTELMÄKEHITTÄJÄN ROOLI PROJEKTIN ONNISTUMISESSA

Tässä luvussa yhdistetään osaaminen, onnistunut ohjelmistoprojekti ja se, minkälainen rooli yksittäisellä järjestelmäkehittäjällä on projektin onnistumisen näkökulmasta.

4.1 Järjestelmäkehittäjän osaaminen projektissa

Tietojärjestelmiä koskevat tutkimukset sekä alalla työskentelevät henkilöt viittaavat suurimman osan järjestelmäprojektien epäonnistumisista johtuvan kehitystiimin puutteellisesta tietoresurssista (Tesch ym., 2009). Jos ohjelmistotuote epäonnistuu, epäonnistuu silloin myös projekti. Tuotteen epäonnistumiseen voi vaikuttaa järjestelmässä olevat virheet eli bugit. Tsui kollegoineen (2016) erittelee tutkimuksessaan bugien syitä ja niiden syiden osuutta keskimääräisesti. Määrittelyvirheistä johtuvien bugien osuus oli 12,5 %, suunnitteluvirheistä johtuvien 24,17 %, koodivirheistä johtuvien 38,33 %, dokumentoinnista johtuvien 13,33 % ja huonoista korjauksista johtuvien osuus 11,67 %. Näistä välittömästi kehittäjän alle kuuluvat koodivirheet ja huonot korjaukset. Tsuin ja hänen kollegoidensa (2016) esittämässä listauksessa nämä ovat yhteensä 51,66 %. Eli yli puolet ohjelmistotuotteen bugeista voi johtua suoraan kehittäjästä.

Tesch kollegoineen (2009) viittaavat julkaisussaan Cheneyn ja Lyonsin (1980) julkaisuun, jonka mukaan ohjelmistoprojektin käytössä olevat metodit ja työkalut eivät ole niin merkittäviä tekijöitä projektin onnistumisen kannalta kuin projektin henkilöstön osaaminen. Yksittäisen kehittäjän kohdalla tehokkainta on, jos hänellä on mahdollisimman laaja tietämys liiketoimintalogiikasta. Jos järjestelmää kuitenkin tehdään tiimissä, nousevat esille tiimin jäsenten erilaisuus ja tiimin yhteistyötaidot. (Tesch ym., 2009.) Vaikka kehittäjän taidot ovat olennainen osa kehitystyön onnistumista, Tesch kollegoineen (2009) eivät löytäneet yhdistävää tekijää projektin onnistumisen ja järjestelmäkehittäjän taitojen välillä. Sen sijaan tiimityöskentelyllä on suuri merkitys projektin onnistumisen

kannalta, vaikka tiimin jäsenillä ei olisikaan täydellistä tietämystä liiketoimintalogiikasta (Tesch ym., 2009).

4.2 Järjestelmäkehittäjän motivaatio

Motivaatio antaa virtaa ja suuntaa ihmistä käyttäytymään tietyllä tavalla. Motivaation puute taas päinvastoin vähentää ihmisen halua käyttäytyä tietyllä tavalla. (Deci & Ryan, 2008) Motivaatio tarkoittaa järjestelmäkehittäjän kohdalla kehittäjän halukkuutta panostaa kehittämäänsä järjestelmään. Järjestelmäkehittäjän korkea motivaatio heijastuu myös ohjelmistoprojektin tuottavuuteen, tehdyn työn laatuun, sekä myös projektin onnistumiseen (Beecham ym., 2008.)

Beecham kollegoineen (2008) toteuttivat kattavan kirjallisuuskatsauksen motivaatiosta ohjelmistokehityksessä. Sen mukaan kuusi yleisintä tunnistettua vaikutusta ohjelmistokehittäjän huonosta motivaatiosta ovat irtisanoutuminen, projektin aikataulun venyminen, tuottavuuden laskeminen, budjetin ylittyminen, poissaolot sekä projektin epäonnistuminen. Beecham ym. (2008) löytivät tutkimuksessaan 62 eri lähteestä 21 eri kehittäjää motivoivaa tekijää. Näistä tekijöistä he erityisesti mainitsivat kirjallisuudessa eniten eteen tulleet tekijät: selvät tavoitteet, henkilökohtaiset mieltymykset, tehtävien tarkoituksen ymmärtäminen ja työtyytyväisyyden saavuttaminen (Beecham, ym., 2008). Järjestelmäkehittäjän motivaation ylläpidolla voidaan siis nähdä olevan kytkös projektin suorituskykyyn. Pelkästään järjestelmäkehittäjän osaaminen ei riitä.

Motivaation hakeminen sovelluskehitystehtävissä voi kääntyä itseään vastaan. On yleistä, että kehittäjät alkavat itse parantelemaan ohjelmiston ominaisuuksia täyttääkseen tarvettaan toteuttaa luovuutta työssään. Tämän tarpeen vuoksi kehittäjien koetaankin usein lankeavan ominaisuuksien viilailuun, mikä sen sijaan vaikuttaa projektin aikatauluun negatiivisesti. (Agarwal & Rathod, 2006.)

4.3 Yhteistyö asiakkaan kanssa

Loppukäyttäjän ja kehittäjän välinen yhteistyö on järjestelmäkehitysprojektissa hyvin tärkeää, sillä loppukäyttäjällä on laaja tietämys liiketoiminnan logiikasta, kun taas kehittäjällä on tekninen osaaminen. Kun nämä kaksi tekijää saadaan lähelle toisiaan, projektilla on paremmat mahdollisuudet onnistua. Tilaa, jossa kehittäjä ja loppukäyttäjä jakavat toistensa tietämystä keskenään, kutsutaan tiedon integraatioksi (eng. *knowledge integration*). (Tesch ym., 2009.)

Tiedon integraation tasosta kertoo se, kuinka paljon yhteistä tietoa näillä kahdella osapuolella on, ja kuinka toimivaa osapuolten välinen viestintä on (Tesch ym., 2009). Korkealla tasolla toimiva tiedon integraatio voi minimoida

projektien viivästymistä, sillä se lisää projektin ennustettavuutta. Tämän saavuttamiseen vaikuttaa kaksi tekijää. Ensimmäinen tekijä on järjestelmäkehittäjän mahdollisuus saada vaihdettua tietoa sisäisesti yrityksessä, jossa hän työskentelee. Toinen tekijä on järjestelmäkehittäjän pääsy ulkoiseen osaamiseen. (Mitchell, 2006.) Tämä tarkoittaa sitä, että kehittäjän tulisi päästä käsiksi sekä työtovereidensa tietämykseen että loppukäyttäjän tai lopputuotteen liiketoiminta-alueen asiantuntijan tietoon, jotta mahdollisilta aukoilta tietämyksessä vältyttäisiin. Tilanteessa, jossa järjestelmäkehittäjällä on hyvin vahva tietämys liiketoiminta-alueesta, ei tiedon vaihdon merkittävyys ole niin suuri (Tesch ym., 2009, s. 663).

5 YHTEENVETO

Osaamista voidaan jaotella työhön liittyviin, henkilökohtaisiin ominaisuuksiin liittyviin sekä valmiuksiin liittyviin osaamisiin. Näistä henkilökohtaisiin ominaisuuksiin lukeutuvat sosiaaliset taidot nousivat kirjallisuudessa vahvasti esille, sillä lähes jokainen työ vaatii jonkin kaltaista vuorovaikutusta muiden ihmisten kanssa.

Tarkennettaessa osaamista ohjelmistokehittäjän tasolle tarkentuu myös työhön liittyvä osaaminen. Järjestelmäkehittäjän osaaminen koostuu henkilökohtaisista piirteistä, ammattitaidosta, liiketoiminnan tietämyksestä sekä teknisistä taidoista. Henkilökohtaisiin piirteisiin lukeutuvat muun muassa sosiaaliset taidot, ammattitaitoon liittyvä ongelmanratkontakyky, liiketoiminnan tietämykseen liittyvä kehitettävän järjestelmän ympärillä olevan liiketoiminnan tuntemus sekä teknisiin taitoihin kuten erilaisten ohjelmointikielien ja -ympäristöjen osaaminen. Koska kehitystyö vaatii ongelmanratkontakykyä, sitä pidetään kirjallisuudessa eniten ammattitaitoa ilmentävänä taitona. Teknisiä taitoja määrittelee myös järjestelmäkehittäjän käyttämät työkalut, jotka ovat usein teknologiaspesifejä, kuten tietylle ohjelmointikielelle tarkoitetut ohjelmointiympäristöt. Ohjelmointiympäristöt karsivat ennen yleisiä kirjoitusvirheitä koodissa niiden integroidun validoinnin ansiosta.

Ohjelmistoprojekti on kertaluontoinen tiimityöhön perustuva rupeama, jonka tarkoituksena on tuottaa ohjelmisto. Ohjelmistot toteutetaan tyypillisesti sykleittäin, jossa vaatimusten tunnistamista, toteutuksen suunnittelua, toteutusta sekä arviointia toistetaan niin pitkään, että lopullinen tuote on valmis tuotantoon ottoon. Projektin onnistumiselle on monia mittareita, mutta yleisesti sen tulee täyttää ennalta määritellyt laatu-, aika-, sekä resurssitavoitteet.

Järjestelmäkehittäjä vastaa ohjelmistoprojektissa projektin lopputuotteen toteuttamisesta, joten kehittäjän vaikutus projektin onnistumiseen on huomattava. Kirjallisuudesta ei kuitenkaan tullut ilmi selvää yhteyttä järjestelmäkehittäjän teknisen taidon ja projektin onnistumisen kanssa. Suurin löydös keskittyi kehittäjän liiketoimintaosaamisen ympärille. Tehokas yhteistyö liikealueesta tuntevan tahon kanssa lienee merkittävin kirjallisuudesta noussut projektin onnistumista edistävä tekijä. Yhteistyön avulla mahdollinen aukko tietämyksessä

pystytään paikkaamaan eikä projektissa synny täten turhaa iteraatiota toiminnallisuuksien hiomisen suhteen. Järjestelmäkehittäjä, jolla on vahva osaaminen, ei tarvitse yhtä paljon yhteistyötä järjestelmän loppukäyttäjän liiketoiminta-alueen asiantuntijan kanssa. Heikomman osaamisen omaava järjestelmäkehittäjä ei kuitenkaan välttämättä tarkoita suurempaa riskiä projektin epäonnistumiselle, sillä hyvien sosiaalisten taitojen avulla kehittäjä pystyy haalimaan tarvittavaa tietoa ympäriltään. Tämä löydös tukee olemassa olevaa tietoa järjestelmäkehittäjän viestintätaitojen merkityksestä ja voi ohjata jatkotutkimusta sen osalta. Johtopäätöksenä voisi nähdä, että sosiaaliset taidot ovat järjestelmäkehittäjän olennaisimpana taitoja.

Jatkotutkimuksena aiheesta olisi hyvä selvittää tarkemmin pelkkien teknisten taitojen vaikutusta projektin onnistumiseen. Myös motivaation vaikutus projektitasolla olisi mielenkiintoinen tutkimuskohde, sillä kirjallisuudesta tuli nyt esille, että motivaatio on monisyinen ilmiö. Korkea motivaatio voi mahdollisesti aiheuttaa projektille myös negatiivisia vaikutuksia.

LÄHTEET

About Visual Studio 2017. (2018, 19. Syyskuuta). Haettu https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2017

Abrahamsson, P., Kautz, K., Sieppi, H., & Lappalainen, J. (2002). Improving software developer's competence: Is the Personal Software Process Working? Presented at Workshop on Empirical Software Engineering, 9.12.2002, Rovaniemi, Finland

Agarwal, N., Rathod, U. (2006). Defining 'success' for software projects: An exploratory revelation. *International Journal Of Project Management, 24*, 358-370.

Beecham, S., Baddoo, N., Hall, T., Robinson, H., & Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. *Information and software technology*, 50(9-10), 860-878.

Berntsson-Svensson, R., & Aurum, A. (2006). Successful software project and products: An empirical investigation. *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, 144-153. ACM.

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press.

Cheney, P. H., & Lyons, N. R. (1980). Information systems skill requirements: A survey. *MIS quarterly*, 35-43.

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of systems and software, 81(6)*, 961-971.

Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer, (11)*, 131-133.

Deci, E. L., & Ryan, R. M. (2008). Self-determination theory: A macrotheory of human motivation, development, and health. *Canadian psychology/Psychologie canadienne*, 49(3), 182.

Eclipse IDE. (2018, 19. Syyskuuta). Haettu https://www.eclipse.org/ide/

El Emam, K., & Koru, A. G. (2008). A replicated survey of IT software project failures. *IEEE software, 25*(5), 84-90.

Eveleens, J. L., & Verhoef, C. (2009). The rise and fall of the chaos report figures. *IEEE software, 1*, 30-36.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Garavan, T. N. & McGuire, D. (2001). Competencies and workplace learning: some reflections on the rethoric and the reality. *Journal of Workplace Learning*, 13, 4, 144-163.

Hacker, W. (1987). Human-centered job design in computerized work. In: Future of Work. A Viewpoint of Social Sciences, 124-130. Ed. K. Eklund. Helsinki: Niva.

Hamed, A. M. M., & Abushama, H. (2013). Popular agile approaches in software development: Review and analysis. *Computing, Electrical and Electronics Engineering (ICCEEE), 2013*, 160-166, IEEE.

Havelka, D. & Merhout, J. W. (2009). Toward a theory of iformation technology professional competence. *Journal of Computer Information Systems, 50(2),* 106-116.

Hanhinen, T. (2010). Työelämäosaaminen. Kvalifikaatioiden luokitusjärjestelmän konstruointi. Akateeminen väitöskirja. Tampereen yliopisto.

Kerrigan, M., Mocan, A., Tanler, M., & Fensel, D. (2007, June). The web service modeling toolkit-an integrated development environment for semantic web services. *European Semantic Web Conference*, 789-798. Springer, Berlin, Heidelberg.

Mitchell, V. L. (2006). Knowledge integration and information technology project performance. *MIS Quarterly*, 919-939.

MOT. (2016). Tietotekniikan liiton ATK-sanakirja. Haettu 17.3.2016 osoitteesta: https://mot.kielikone.fi/mot/jyu/netmot.exe?motportal=80

Muench, D. (1994). The Sybase Development Framework. Oakland, CA: Sybase, Inc.

PMI. (2000). Project Management Body of Knowledge (PMBOK® GUIDE 2nd edition). Project Management Institute.

PMI. (2009). Project Management Body of Knowledge (PMBOK® GUIDE 4th edition). Project Management Institute.

Parsons, D., Ryu, H., & Lal, R. (2007). The impact of methods and techniques on outcomes from agile software development projects. *IFIP International Working Conference on Organizational Dynamics of Technology-Based Innovation, June*, 235-249. Springer, Boston, MA.

Rashid, M., Clarke, P. M., & O'Connor, R. V. (2019). A systematic examination of knowledge loss in open source software projects. *International Journal of Information Management, 46*, 104-123.

Runeson, P. (2006). A survey of unit testing practices. *IEEE software, 23(4)*, 22-29.

Surakka, S. & Malmi, L. (2004). Cognitive skills of experienced software developer: Delphi study. *Kolin Kolistelut – Koli Calling 2004*, 37.

Tesch, D., Sobol, M. G., Klein, G., Jiang, J. J. (2009). User and developer common knowledge: Effect on the success of information system development projects. *International Journal of Project Management, 27, 7*, 657-664.

The Standish Group. (1995). *The CHAOS report*. Haettu 25.1.2019 osoitteesta: https://www.csus.edu/indiv/r/rengstorffj/obe152-spring02/articles/standishchaos.pdf

Top IDE Index. (2018). Haettu 19.9.2018 osoitteesta: https://pypl.github.io/IDE.html

Tsui, F., Karam, O., & Bernal, B. (2016). Essentials of software engineering. Jones & Bartlett Learning.

Vartiainen, M., Teikari, V. & Pulkkis, A. (2002). Tavoitteet ja tulkinnat – motivaatio ja palkitseminen työelämässä. *Mikä meitä liikuttaa. Modernin motivaatiopsykologian perusteet*, 188-212. PS-Kustannus. Keuruu

Viitala, R. (2006). Johda osaamista! Osaamisen johtamista teoriasta käytäntöön. Otavan Kirjapaino Oy. Keuruu