Matti Välimäki

Improvisointi ohjelmistokehityksen työvälineenä

Tietotekniikan kandidaatintutkielma

23. helmikuuta 2016

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Matti Välimäki

Yhteystiedot: matti.j.valimaki@jyu.fi

Työn nimi: Improvisointi ohjelmistokehityksen työvälineenä

Title in English: Improvisation in Software Development

Työ: Kandidaatintutkielma

Sivumäärä: 16+0

Tiivistelmä: Tietojärjestelmien kehittämistä pidetään yleisesti hyvin organisoituna ja suunnitelmallisena tapahtumana. Tämä ei kuitenkaan aina pidä paikkaansa, sillä kehittäminen ei aina tapahdu hallinnoidussa ja tiukasti organisoidussa ympäristössä. Etenkin pienten ja epävirallisten järjestelmien kehittämisessä improvisoinnilla on suurehko rooli.

Avainsanat: ohjelmistokehitys, improvisointi, bricolage

Abstract: Software development is seen as an organised process. However not all software is developed in a tightly managed and organised environment. Improvisation often plays a role in the development, especially in the case of small and informal systems.

Keywords: software development, improvisation, bricolage

Sisältö

1 Johdanto

Perinteisesti tietojärjestelmien kehittämisessä on keskitytty malleihin ja tiukkaan organisaatioon ja pidetty koko kehitysprosessia loogisena ja suoraviivaisena. Tämä näkemys kuitenkin jättää huomiotta kaikki projektit, jotka eivät välttämättä ole virallisia tai kaavamaisia, vaan toteutetaan muilla keinoin (Bansler & Havn 2003, 2004).

Improvisaatiota pidetään helposti sekavana ja vieraana käytänteenä, jota käytetään lähinnä taiteissa. Ohjelmistokehityksessä sillä on kuitenkin oma roolinsa muiden kehityskäytänteiden rinnalla.

Tässä raportissa esitellään, mitä tarkoitetaan improvisoinnilla tietojärjestelmien kehittämisympäristössä sekä pyritään selvittämään, miten improvisointia käytetään ja voidaan käyttää tietojärjestelmien kehittämisessä. Käydään läpi erilaisia kehitysmenetelmiä ja miten improvisaatio toimii niiden kanssa. Tarkastellaan myös miltä improvisaatio näyttää koko kehitystiimin ja erityisesti projektipäällikön näkökulmasta. Siinä sivussa tutkitaan millaista tutkimusta aiheesta on jo tehty.

Ohjelmistokehityksen nopeasti muuttuvassa maailmassa usein voittajia ovat ne, jotka nopeiten pystyvät reagoimaan. Kenties improvisoinnista olisi tällaisessa ympäristössä apua.

2 Improvisointi

Tietojärjestelmäkehityksessä improvisoinnilla viitataan muun muassa nopeaan reagointiin odottomattomissa tilanteissa. Eräs määritelmä on lainattu jazz-musiikista, jossa säveltäminen ja esittäminen tapahtuvat samanaikaisesti (Bansler & Havn 2003, 2004). Bansler & Havn (2003, 2004) on myös määritellyt improvisoinnin tarkemmin neljällä väitteellä:

1. Improvisointi on *tarkoituksellista*. Tämä tarkoittaa, että se on seurausta organisaation ja/tai jonkun sen jäsenten tahallisista yrityksistä.
2. Improvisointi on *ennalta-arvaamatonta*. Se tapahtuu reaktiona odottamattomaan ja toimii ilman aikaisempaa suunnitelmaa tai metodeja.
3. Improvisointi *tapahtuu toiminnan aikana* eli organisaation jäsenet eivät pysähdy pohtimaan ongelmaa tai yllättävää mahdollisuutta ja miettimään suunnitelmaa. Sen sijaan he reagoivat suoraan ja voivat tulkita toimintansa soveliaisuutta vain jälkikäteen, eivätkä pohtia sitä etukäteen, kuten perinteisessä suunnittelussa.
4. Improvisointi olettaa, että on olemassa jonkinlainen valmis suunnitelma, resurssi, työkalu tai muu vastaava viitekehys, josta variaatioita voidaan luoda.

Siinä missä perinteinen, systemaattinen menetelmä on hallittu ja kontrolloitu prosessi, improvisointi on satunnainen ja opportunistinen (Truex, Baskerville & Travis 2000). Systemaattinen menetelmä siis seuraa tiettyä ”polkua”, jonka päässä on valmis tietojärjestelmä, kun taas improvisoiva menetelmä vastaa paremmin todellisuuden muuttuvaa ja ennakoimatonta luonnetta. Tällöin virheisiin pystytään reagoimaan ja ne pystytään korjaamaan joustavasti.

Systemaattinen menetelmä on osaltaan hyvin lineearinen ja peräkkäinen, kun taas improvisoivassa menetelmässä prosesseissa on samanaikaisuutta, päällekkäisyyttä ja aukkoja (Truex ym. 2000). Ensimmäisessä odotetaan, että kehitys koostuu tietyistä tehtävistä, jotka voidaan suorittaa peräkkäisessä järjestyksessä. Jälkimmäisessä sen sijaan reagoidaan tarpeisiin sitä mukaan, kun niitä ilmenee, jolloin tehtäviä suorite-

taan usein jonkin verran samanaikaisesti ja päällekkäisesti.

Improvisoiva menetelmä ilmenee ainoastaan ainutkertaisissa ja yksittäisiä tapauksia koskevissa muodoissa, kun systemaattinen menetelmä on toistettavissa ja yleistettävissä oleva (Truex ym. 2000). Improvisoinnissa jokainen tapaus on ainutkertainen ja siten siihen tulee reagoida omalla tavallaan. Myös jokainen asiakas on erilainen ja haluaa eri asioita, joten samat keinot eivät välttämättä toimi ja kaikkiin kehitysvaiheisiin pitää pystyä palaamaan milloin vain. Systemaattisessa tavassa on monia toistettavia menetelmiä, joista voidaan valita toimivin. Menetelmästä poikkeaminen on kuitenkin vaarallista, sillä se johtaa helposti virheisiin. Menetelmä myös varmistaa, että kaikki tarvittavat vaiheet tulee suoritetettua, joten niihin ei tarvitse jälkikäteen palata.

Systemaattinen menetelmä on myös rationaalinen, päättäväinen ja tavoitehakuinen ja improvisaatio taas vaatii neuvottelua, kompromisseja ja on oikukasta (Truex ym. 2000). Ensimmäisessä tavoitteet ovat tärkeitä ja niihin pyritään tietyin ennalta määrätyin keinoin. Prosessissa määritellään tavoitteet, keinot ja vaiheet ja ne toteutetaan määrätyssä järjestyksessä. Jälkimmäisessä tavoitteet ovat laajoja ja tarkentuvat vasta kehityksen yhteydessä mahdollisesti yrityksen ja erehdyksen kautta. Tavoitteen tarkentumiseen vaikuttavat voimakkaasti myös sosiaaliset tekijät.

Myös improvisoidussa tietojärjestelmänkehittämisessä tarvitaan jonkinlainen edes epävirallinen toimintamalli, joka määrittelee oman ainutkertaisen tapansa ratkaista kehitysongelma. Usein osa toimintamallista on lainattu systemaattisilta menetelmiltä, osa muualta (kuten taiteesta, kirjallisuudesta tai arkkitehtuurista) ja osa on keksitty itse juuri tätä nimenomaista tarvetta varten (Truex ym. 2000).

Bansler & Havn (2004) määrittelee kolme pääseikkaa, jotka erottavat improvisoivan lähestymistavan perinteisistä kehitysmalleista: ymmärtäminen (engl. *sensemaking*), improvisaatio ja bricolage. Näitä pääseikkoja selitetään tarkemmin seuraavissa kappaleissa.

Koska uudet teknologiat ovat epäselviä ja siten helposti monilla ristiriitaisilla tavoilla tulkittavissa, teknisten järjestelmien kehitys ja käyttö vaatii jatkuvaa uudel-

leenymmärtämistä. Ymmärtäminen tässä mielessä onkin pohjimmiltaan aktiivinen prosessi, jossa opitaan tekemällä. Prototyyppaaminen ja iterointi ovat esimerkkejä ymmärtämisestä manipulaationa eli jonkin luomisena, jotta sitä voisi hahmottaa ja siten käyttää (Bansler & Havn 2004).

Improvisaatio käsittelee ennalta-arvaamatonta ja siten toimii ilman etukäteistä suunnitelmaa, pohjapiirrustusta tai menetelmää. Sitä voidaan pitää lyhyen aikavälin oppimisena, jossa tosiaikainen kokemus ohjaa toimintaa sen tapahtuessa. Improvisaatiota tapahtuu sekä yksilö- että ryhmätasolla. Tällöin kehitysprosessia ohjaa ennemminkin tarkkaavaisuus ja tulkinta kuin tarkoitus ja päätökset. Improvisoidessa pyritään ymmärtämään yllättäviä mahdollisuuksia ja ilmeneviä rajoitteita. Kehittäessä joudutaan usein reagoimaan epätahallisiin vaikutuksiin, joten ei voida sanoa kehittämisen olevan koskaan täysin hallinnassa. Tämän seurauksena kehittäjät joutuvat jatkuvasti määrittämään uudelleen, mitä kulloinkin tapahtuu ja mitä on silloin saavutettavissa. Nämä uudet määritelmät — eivät niinkään alkuperäiset päätökset — ohjaavat toimintaa. Koska ainoastaan jo tapahtuneisiin asioihin voidaan reagoida, suunnitelmallisuuden sijaan kehitysprosessin keskiöön nousee tarkkaavaisuus (Bansler & Havn 2004).

”Uudet järjestelmät luodaan, joskus kirjaimellisesti, vanhojen järjestelmien raunioille” (Lanzara 1999, s.346, kirjoittajan suomentamana). Bricolage on olemassa olevien resurssien ja taitojen käyttäminen annetun tehtävän suorittamiseksi. Tähän sisältyy myös aikaisemmin luodut tietojärjestelmät tai niiden osat. Improvisaatio lisää bricolagen todennäköisyyttä, sillä uusien resurssien hankkimiseen on vähemmän aikaa. Bricolagea voi tapahtua silti myös improvisaation ulkopuolella (Bansler & Havn 2004).

3 Improvisointi työelämän projekteissa

Sovelluskehitysalan tutkijat ja johtajat yleisesti jakavat näkemyksen, että menestys riippuu siitä, miten hyvin alan tulevia muutoksia pystytään ennustamaan ja miten hyvät selviytymisstrategiat niitä varten tehdään. Ennustaminen kuitenkin onnistuu lähinnä yksinkertaisissa järjestelmissä, jollainen sovelluskehitysala ei ole, joten tulevaa ei voida tietää eikä siten myöskään hallita (Dybå 2000). Ohjelmistokehityksessä tarvitaan tietenkin kurinalaisuutta, mutta tiukka kurinalaisuus ja aikaisessa vaiheessa lukkoon lyödyt vaatimukset saattavat aiheuttaa projektin kariutumisen joustamattomuuteen (Magni, Provera & Proserpio 2006). Tarvitaankin tasapainoa kurinalaisuuden ja vapaan luovuuden välille — ilman kuria vallitsee kaaos, mutta tiukka kuri tappaa luovuuden (Dybå 2000).

Yrityksissä ollaan taipuvaisia nojaamaan vanhaan tietoon ja kokemuksiin ja näistä luotuihin rutiineihin. Tällöin oppiminen vaikeutuu. Vanhoihin oppeihin luottaminen voi johtaa myös siihen, että improvisaatiota ei saa käyttää. Tämä taas saattaa johtaa epäluovaan työympäristöön ja ideoiden syntymättömyyteen (Magni ym. 2006). Kuitenkin suurissa ohjelmistoyrityksissä toiminta muuttuu kankeammin, kun pitäydytään siinä, miten on ennenkin tehty ja noudatetaan jotain ennaltamäärättyä strategiaa ja suunnitelmaa. Pienissä yrityksissä on helpompi tehdä nopeita muutoksia ja reagoida tilanteisiin joustavammin. Improvisaatio sopii paremmin pienille kuin suurille yrityksille (Dybå 2000).

Eräässä case-tutkimuksessa todettiin, että improvisoitua tietojärjestelmäkehitystä tehtäessä etusijalla on tavoite, eikä niinkään suunnitelma. Resursseja ja töitä ei organisoida alun suunnitelman mukaan, vaan tavoitehakuisesti kuhunkin tilanteeseen heijastaen. Suunnitelmaa voidaan myös muuttaa tilanteen mukaan. Tällaisessa tilanteessa johtajien ja kehittäjien tuleekin luoda hyvin selkeä tavoite, johon tähdätä (Madsen, Kautz & Vidgren 2006).

Jos improvisoiva lähestymistapa syrjäyttää systemaattisen, jätetään kuitenkin huomiotta kaikki systemaattisella metodilla onnistuneet suuren luokan ohjelmistonke-

hitysprojektit. Suuren luokan projektit käsittävät satoja kehittäjiä ja useita kehitysryhmiä. Tällaiset projektit tarvitsevatkin vähintään sopimuksen siitä, mistä metodista kehittäminen aloitetaan, jotta on yhteinen lähtökohta (Truex ym. 2000).

3.1 Ketteryys

Kehitysmenetelmien ketteryys (eng. *agile*) tarkoittaa perinteisten menetelmien raskauden mahdollisimman suurta karsimista, jotta pystyttäisiin nopeasti reagoimaan muuttuviin ympäristöihin, vaatimuksiin, aikatauluihin ja niin edelleen (Dybå & Dingsøyr 2008). Tällaiset kehitysmenenetelmät tukevat lyhyempikestoisia projekteja ja improvisaatiota pysyäkseen reagoimaan monimutkaisiin, nopealiikkeisiin ja kilpailullisiin markkinoihin (Pries-Heje, Baskerville, Ramesh & Levine 2009).

Vaikka perinteiset ja ketterät menetelmät on totuttu erottamaan jyrkästi toisistaan, on niillä myös yhteisiä piirteitä. Tuotteen joustavuus, funktionaalisten tarpeiden ymmärtäminen, muutokseen reagoiminen ja kokemuksesta oppiminen ovat kaikki piirteitä, joita esiintyy sekä ketterissä että perinteisissä menetelmissä. Nämä menetelmät kulkevatkin usein rinta rinnan, sillä back-end -kehitys käyttää yhä usein perinteisiä menetelmiä, kun taas front-end -kehitys suosii ketteriä menetelmiä. Siinä missä perinteinen kehitys keskittyy vakauteen, täsmällisyyteen ja kuriin, ketterä kehitys keskittyy joustavuuteen ja muutoksiin reagoimiseen. Ketterät menetelmät sopivatkin — erityisesti samansijaintisille — pienille kehitystiimeille, joiden tarpeet ovat emergenttejä ja myös jatkuvassa muutoksessa oleviin ympäristöihin (Pries-Heje ym. 2009).

Järjestelmien kehittäminen ei ole koskaan siistiä ja käytännöllistä ja kehitysmenetelmien arvoa on yliarvioitu (Bansler & Havn 2004). Uusia tuotteita kehittävät tiimit kohtaavat usein epäjärjesteltyjä, odottamattomia ja epärutiininomaisia tilanteita (Magni, Maruping, Hoegl & Proserpio 2013). Järjestelmiä kehitetään monimutkaisemmassa, vähemmän vakaassa ja huonommin ymmärretyssä maailmassa kuin kehittäjät ymmärtävätkään, jossa kehittäjillä on parhaimmillaankin vain vaatimatonta vaikutusvaltaa (Bansler & Havn 2004).

Odottamattomat muutokset kehitysympäristössä vaativat tiimiltä usein nopeita reaktioita, jotta tehtävät ovat silti suoritettavissa. Tällaiset odottamattomat tilanteet saattavat saattaa olemassaolevat rutiinit ja varasuunnitelmat käyttökelvottomiksi ja tiimien kyky improvisoida onkin ratkaisevaa aikakriittisten tehtävien onnistuneesti suorittamisessa. Vaikka projektin vaatimukset onkin usein määritelty etukäteen, projektin onnistuminen riippuu kehittäjien kyvystä täyttää ilmeneviä tarpeita ja räätälöintitoiveita sekä kyvystä integroida kehittäjien ja loppukäyttäjien uusia ideoita (Magni ym. 2013).

3.2 Kehitystiimit

Tietojärjestelmäkehitysprojektit nojaavat heuristisiin prosesseihin ilman selkeää ja tunnistettavaa reittiä ratkaisuun, vaativat suurta määrää suunnittelematonta toimintaa kehitystiimiltä ja pakottavat tiimin jatkuvasti luovaan ongelmanratkaisuun. Kehitysprojektit ovat perimmiltään luovia, sillä ne sisältävät uusien ideoiden ja ratkaisujen luomista ja arviointia bisnesongelmien ratkaisemiseksi (Magni ym. 2013). Tiimitasolla improvisaatio on yhteistoimintaa ja määritellään luovaksi ja spontaaniksi toiminnaksi saavuttaakseen tiimin tavoite uudella tavalla. Koska improvisaatio on yhteistoimintaa, se on enemmän kuin yksittäisten kehittäjien improvisoinnin summa. Spontaanina toimintana tiimi-improvisaatio on äkillistä, harkitsematonta ja suunnittelematonta toimintaa käyttäen hyväksi välittömästi käsillä olevia resursseja ongelmanratkaisuun (Magni ym. 2013).

Tiimin toiminnasta tulee spontaanimpaa, kun suunnittelun ja toteutuksen välinen aika lyhenee. Tällaiset hetken mielijohteesta syntyvät toiminnot ovat kriittisiä tilanteessa, jossa välittömän reagoimisen epäonnistuminen johtaa hukattuun mahdollisuuteen, ongelman pahenemiseen tai hengenvaarallisiin tilanteisiin (Magni ym. 2013). Aikakriittisissä tilanteissa tiimien täytyy käyttää bricolagea. Tietojärjestelmäkehitys on perimmiltään monimutkainen tehtävä, joka sisältää monia riippuvaisuuksia. Useiden kehittäjien täytyy kehittää, testata, muokata ja integroida osia koodista tiukkojen aikataulujen mukaisesti ja kehittäjien välinen vuorovaikutus on kriittistä kehitysprojektin onnistumiselle. Magni ym. (2013) esitti myös kolme hypotee-

sia:

1. Tiimin hajanaisuus vaikuttaa negatiivisesti sen suorituskykyyn.
2. Tiimin improvisointi vaikuttaa positiivisesti sen suorituskykyyn.
3. Tiimin hajanaisuus sääntelee tiimin improvisoinnin hyötyä siten, että hyöty pienenee hajanaisuuden kasvaessa.

Artikkelin mukaan näistä toinen ja kolmas saivat tukea tutkimustuloksista, mutta tiimin hajanaisuus eli kohta 1 ei suoraan merkittävästi vaikuttanut tiimin suorituskykyyn. Ohjelmistokehitys on hyvin osaamiseen keskittyvää ja perimmiltään monimutkaista ja siten hyötyy improvisaatiosta. Monimutkaisissa ja yllättävissä tilanteissa rutiineihin nojaaminen voi olla haitallista. Tiimi-improvisaatiolla on löytynyt olevan positiivinen vaikutus ko. tutkimuksessa. Etenkin projekteissa, joissa olennaiset parametrit on huonosti määritelty, ratkaisuja täytyy räätälöidä tai vaatimukset voivat muuttua, kannattaa suosia samansijaintisia tiimejä, sillä heidän täytyy todennäköisesti improvisoida toimiakseen tehokkaasti. Tämä on vähemmän olennaista, jos määritelmät ovat tarkkoja ja muuttumattomia ja tehtävät ovat erillisiä. Tiimi-improvisaation positiivisten vaikutusten takia samansijaintisten tiimien päälliköiden kannattaakin pyrkiä luomaan ympäristö, joka kannustaa ja tukee improvisointiin (Magni ym. 2013).

3.3 Projektinhallinta

Projektinhallinta on todellisuudessa merkittävästi sekavampaa kuin kirjoissa ja teorioissa. Projektipäälliköt usein vain toimivat ajattelematta erityisiä prosesseja, ennemminkin pyrkien spontaaniin ja intuitiiviseen vastaukseen tilanteeseen senhetkisen tavoitteen saavuttamiseksi (Klein, Biesenthal & Dehlin 2015). Lähteessä Klein ym. (2015) projektinhallinnan improvisoinnin määritelmä esitetään seuraavien lainauksien avulla (kirjoittajan suomentamana, alkuperäinen lähde suluissa):

1. ”Toiminnan käsittäminen sitä mukaa, kun se kehittyy, ammentaen saatavista resursseista” (Cunha, Cunha & Kamoche 1999, s.302).
2. ”Intuitio ohjaa toimintaa spontaanilla tavalla” (Crossan & Sorenti 1997, s.156).

Improvisaatio ei suinkaan ole erillinen projektinhallinnan menetelmä vaan tapa käyttää olemassa olevia teorioita ongelman välittömään ratkaisuun. Täten improvisaatiota tapahtuukin enemmässä tai vähemmässä määrin joka päivä kaikissa projekteissa. Improvisaatio kuvaa spontaania yritystä ratkaista ongelmaa välittömässä tilanteessa, erottamatta ymmärtämistä ja toimintaa. Luovassa työskentelyssä on aina jonkin verran epävarmuutta, joka luonnollisesti johtaa epäonnistumisen mahdollisuuteen. Tällaisissa tilanteissa on tärkeää pystyä nopeasti ja spontaanisti säätämään toimintaansa ja toimintamallejaan (Klein ym. 2015).

Improvisaatio on jyrkän teknisen lähestymistavan vastakohta, sillä se pakottaa katsomaan kunkinhetkistä tilannetta tarkasti ja ajatuksella pyrkiessään ymmärtämään ja hienosäätämään monimutkaisia ja epäselviä kokemuksia sitä mukaa, kun niitä syntyy (Klein ym. 2015). Klein ym. (2015) määrittelee improvisaatiolle projektinhallinnan näkökulmasta neljä eri astetta:

1. Projektipäälliköt käyttävät taitojaan ja kokemustaan tehdäkseen lieviä hienosäätöjä, missä tarpeen. Lievää improvisaatiota tapahtuu aina, sillä ei ole olemassa sosiaalista järjestelmää, joka antaisi täyden vallan hallita toimintaa.
2. Pyritään käyttämään olemassaolevia resursseja käytännöllisellä tavalla. Tämä tunnetaan myös nimellä bricolage.
3. Uusien työkalujen lisääminen organisaatioon tiettyjen ongelmien ratkaisemiseksi. Vaikka monissa tilanteissa olemassa olevilla resursseilla ja työkaluilla voikin pärjätä (bricolage), joskus niistä täytyy luopua. Tämän tason improvisaatio vaatii avoimuutta uusille ja erilaisille toimintatavoille kuitenkin pitäen organisaation perustukset vakaina täten mahdollistaen päivittäisen tehokkaan toiminnan.
4. Radikaalimpi muutos alkuperäisiin suunnitelmiin. Pohjimmiltaan muuttaa osia alkuperäisestä rakenteesta ja siten toiminnasta tai korvaa osia uusilla, joskus jopa täysin alkuperäisestä suunnitelmasta irrallisilla, osilla. Tällaiset projektinhallintatoimet ovat hyvin ongelmanratkaisuohjautuvaisia ja siten pyrkivät toimittamaan projektille tyydyttävän lopputuloksen.

Nämä improvisaation asteet kuvaavat projektipäälliköiden mahdollisuuksia hyödyntää ja muokata työkalujaan. Ne eivät ole toisiaan — eivätkä perinteisiä projektinhallintamenetelmiä — poissulkevia (Klein ym. 2015).

Projektien lisääntynyt monimutkaisuus yhdistettynä projektien korkeaan epäonnistumisprosenttiin kertoo, että perinteiset projektinhallintateoriat eivät onnistu tuottamaan tyydyttäviä tuloksia. Siten ajattelussa ja metateorioissa tarvitaan muutosta, joka ei erottaisi ajattelua ja toimintaa, mutta kannustaisi useampien koulukuntien yhteiskäyttöä. Improvisationaalinen lähestymistapa näkee työkalujen moninaisuuden yksittäisten työkalujen joukon sijaan työkalupakkina. Projektipäällikön tulee tietää, milloin käyttää kutakin työkalua ja miten, mutta myös milloin hylätä jokin työkalu. Suunnitelman ei koskaan tulisi olla kiinteä ja lopullinen vaan lähtökohta pisteestä toiseen. Improvisaation käsite tukee tätä käytännönläheistä ajatusmallia luomalla joustavampia käyttäytymismalleja ja mahdollistaen uusia tapoja saavuttaa tavoitteita (Klein ym. 2015).

Improvisaatio ei ole itsessään positiivista tai negatiivista vaan ennemminkin väistämätön osa projektityöskentelyä ja siten jo olemassa nykyisissä käytänteissä. Improvisaatio tuo järjestelmälle sinnikkyyttä (eng. *resilience*) — kykyä palautua ongelmatilanteista — sillä se kannustaa ennakointiin, tarkkaavaisuuteen ja reagointikykyyn. Sinnikkyys mahdollistaa reaktiivisten toimintamallien lisäksi ennakoivaa oppimista ja kasvua keskittymällä olemassaolevien ja ilmenevien haasteiden kohtaamiseen. Täten improvisointi sinnikkäissä järjestelmissä omaksuu toiminnallisen ajattelun ajatusmallin, joka lyhentää suunnittelun ja toteutuksen välistä aikaa (Klein ym. 2015).

Ei ole universaalia ratkaisua projektinhallintaan, joka ylittäisi kaikki muut. Käytänteet, menetelmät ja teoriat ovat vain työkaluja, joita tulee käyttää huolella ja käytännöllisellä viisaudella. Monimutkaiset ympäristöt vaativat kykyä avoimeen luovuuteen, yritykseen ja erehdykseen ja tilannekohtaiseen vanhojen kokemusten muokkaamiseen taitavaksi toiminnaksi sinnikkään lähestymisen saavuttamiseksi. Klein ym. (2015) kannustaakin rutiininomaiseen rutiinien rikkomiseen, joka voidaan myös määritellä improvisaationa.

4 Yhteenveto

Tietojärjestelmäkehittämisen maailma on nopeasti muuttuva. Tällaisessa ympäristössä improvisoinnista voi olla hyötyä etenkin pienille yrityksille, joilla on mahdollisuus mukautua nopeasti erilaisiin tilanteisiin. Suurempien yritysten on vaikeampi mukautua, mutta niillekin voi olla improvisaation ajatusmaailmasta tai mahdollisesti tulevaisuudessa kehitettävistä tekniikoista hyötyä.

Rutiinit ja standardoidut käytänteet voivat olla hyvä lähtökohta, mutta tehokas toiminta vaatii muutakin. Improvisointia kannattaakin pitää vaihtoehtona tietojärjestelmiä kehitettäessä tai ainakin hyväksyä se, että harvoin järjestelmänkehitys seuraa tarkasti ennaltamäärättyä suunnitelmaa. Improvisaation sisäinen paradoksi onkin, että täytyy tehdä virheitä, jotta voi onnistua.

Ymmärtäminen, improvisaatio ja bricolage tarjoavat hyvän lähtökohdan todellisen kehittämisen maailman kuvaamiseen ja sen näyttämiseen, miten rajallisesti metodeista ja suunnitelmista on tosimaailman tilanteissa hyötyä (Bansler & Havn 2004).

Kirjallisuutta

Bansler J. ja Havn E. 2003. *Improvisation in Action: Making Sense of IS Development in Organizations*, Proceedings of the International Workshop on Action in Language, Organisations and Information Systems (ALOIS 2003), s. 51–63

Bansler J. ja Havn E. 2004. *Improvisation in Information Systems Development*, Information Systems Research, s. 631–646.

Cunha M.P., Cunha J.V., Kamoche K. 1999. *Organizational improvisation: what, when, how and why*, International Journal of Management Reviews 1, s. 299—341.

Crossan M., Sorrenti M., 1997. *Making sense of improvisation*, Advances in Strategic Management 14, s. 155—180.

Dybå T. 2000. *Improvisation in Small Software Organizations*, IEEE Software 17(5), s. 82–87.

Dybå T. ja Dingsøyr T. 2008. *Empirical studies of agile software development: A systematic review*, Information and software technology, 50(9), s. 833–859.

Klein L., Biesenthal C. ja Dehlin E. 2015. *Improvisation in project management: A praxeology*, International Journal of Project Management, Volume 33, Issue 2, s. 267–277.

Lanzara G. F. 1999. *Between Transient Constructs and Persistent Structures: Designing Systems in Action*, Journal of Strategic Information Systems (8), s. 331–349.

Magni M., Provera B. ja Proserpio L. 2006. *Individual attitude towards improvisation in information systems development: a multi-level perspective*, ECIS-06 conference.

Magni M., Maruping L. M., Hoegl M. ja Proserpio L. 2013. *Managing the unexpected across space: Improvisation, dispersion, and performance in NPD teams.* Journal of Product Innovation Management, 30(5), s. 1009–1026.

Madsen S., Kautz K. ja Vidgen R. 2006. *A framework for understanding how a unique and local IS development method emerges in practice*, European Journal of Information Systems, 15, s. 225–238.

Pries-Heje J., Baskerville R., Ramesh B. ja Levine L. 2008. *Advances in Information Systems Development: From Discipline and Predictability to Agility and Improvisation*, IFIP International Federation for Information Processing, Volume 274. Advances in Information Systems Research, Education and Practice. David Avison, George

M. Kasper, Barbara Pernici, Isabel Ramos, Dewald Roode (toim.) Boston:Springer, s. 53-–75.

Truex D., Baskerville R. ja Travis J. 2000. *Amethodical systems development: the deferred meaning of systems development methods*, Accounting, Management and Information Technologies, 10, s. 53–79.