Atte Komppa

Matti Virkkunen

Työkalu testauksen työmäärän arviointiin Salesforce-projekteille

Tietotekniikan pro gradu -tutkielma

22. helmikuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijät: Atte Komppa ja Matti Virkkunen

Yhteystiedot: atte.komppa@student.jyu.fi ja
matti.e.virkkunen@student.jyu.fi

Ohjaajat: Tommi Mikkonen ja Samuli Pekkola

Työn nimi: Työkalu testauksen työmäärän arviointiin Salesforce-projekteille

Title in English: Tool for test effort estimation for Salesforce projects

Työ: Pro gradu -tutkielma

Opintosuunta: Tietotekniikka ja Tietojärjestelmätiede

Sivumäärä: 61+0

Tiivistelmä: Työmääräarvioiden antaminen on usein haastava prosessi ja ilman käytössä olevia malleja tai standardeja johtaa usein määrittelijän omaan parhaaseen arvaukseen. Ohjelmistotuotannossa tavoitellaan nykyään mahdollisimman nopeasti tuloksia. Kehitettyjä toiminnallisuuksia ei tulisi ottaa käyttöön ilman kunnollista testausta. Tämän Pro Gradu -tutkielman tarkoituksena on selvittää, miten Salesforce-projektin testauksen työmääräarviointia voidaan helpottaa ja automatisoida työkalulla. Työkalun logiikka mukailee vahvasti toisessa tutkielmassa toteutettua testausstrategiaa, josta tämä tutkielman aihe on saatu jatkotutkimusaiheena. Lisäksi pyrimme parantamaan testausstrategian jalkauttamista kohdeorganisaatiossa sekä mahdollistamaan yhtenäisempiä käytäntöjä ja arvioita testauksen työmäärille.

Avainsanat: Asiakastietojärjestelmä, Testaus, Työmääräarvio

Abstract: Providing effort estimates is often a challenging process, and without available models or standards, it often leads to the assessor's best guess. In software production, there is a current emphasis on achieving results as quickly as possible. Developed functionalities should not be implemented without proper testing. The purpose of this Master's thesis is to investigate how the effort estimation for testing in a Salesforce project can be facilitated and simplified with the help of a tool. The tool's logic closely follows a testing strategy implemented in another thesis, from which this thesis topic has been derived for further

research. Also we aim to improve the implementation of the testing strategy in the target organization and enable more consistent practices and estimates for testing workloads.

Keywords: Customer Relationship Management, Testing, Effort estimation

Termiluettelo

Apex	Salesforcen kehittämä vahvasti tyypitetty oliosuuntautunut ohjelmointikieli palvelinpuolen automaatioiden ja mukautettujen rajapintojen kehitystä varten
COCOMO	Constructive Cost Model, malli ohjelmistoprojektien kustannusten arviointiin.
CRM	Customer Relationship Management eli asiakastietojärjestelmä, joka auttaa hallitsemaan asiakastietoja sekä helpottaa asiakassuhteiden kehittämisessä.
DSRM	Design Science Research Method, kuusivaiheinen suunnittelutieteellinen tutkimusmenetelmä
Käyttötapauspisteet	Menetelmä, jolla voidaan arvioida ohjelmistoprojektin kokoa sen toiminnallisten vaatimusten perusteella.
Lightning-sovellus	Salesforcen ominaisuus, jolla voi koota joukon toisiinsa liittyviä toiminnallisuuksia yhteiseksi kokonaisuudeksi.
Low-code	Ohjelmistonkehitysmenetelmä, jossa edistyneiden ominaisuuksien käyttö saattaa edellyttää ohjelmointitaitoa. Low-code hyödyntää paljon visuaalisia käyttöliittymiä ja valmiita komponentteja.
LWC	Lightning Web Component, Salesforcen kehittämä web-komponentteihin perustuva viitekehys käyttöliittymäkomponenttien kehitystä varten.
No-code	Ohjelmistonkehitysmenetelmä, jossa kehittäminen onnistuu ilman perinteisiä ohjelmointitaitoja. Sen sijaan tyypillisesti käytetään visuaalisia käyttöliittymiä, vedä ja pudota -työkaluja ja valmiita komponentteja sovellusten rakentamiseen.
PaaS	Platform as a Service eli alusta palveluna, joka tarkoittaa palveluntarjontamallia, jossa kehittäjille tarjotaan kehitysalusta, sisältäen kaikki tarvittavat kehitystyökalut ja -ympäristöt.
Pro-code	Perinteinen ohjelmistonkehitysmenetelmä, jossa toiminnallisuu-

	det toteutetaan ohjelmoimalla.
SaaS	Software as a Service eli ohjelmisto palveluna, joka tarkoittaa pilvessä sijaitsevaa palvelua. Palvelua voidaan hyödyntää suoraan internetin välityksellä ilman erillisien ohjelmien lataamista laitteelle.
Salesforce	Yhdysvaltalainen ohjelmistoyritys, jonka pääasiallinen tuote on CRM-alusta.
Salesforce-objekti	Salesforcen tiedontallennusyksikkö, jonka avulla tieto määritellään ylätasolla.
Salesforce-tietue	Tietyntyyppinen tiedontallennusyksikkö Salesforce objektin alla, johon yksittäisen tietueen tietoja tallennetaan.
Sandbox	Salesforcen kehitys- ja testausympäristö, joka on kopio tuotantoympäristöstä.
SLIM	Software Lifecycle Management, ohjelmistokehityksen työmäärän arviointi- ja projektinhallintatyökalu, jonka on kehittänyt QSM, Inc.
SOQL	Salesforce Object Query Language, SQL-kyselykielen kaltainen kyselykieli Salesforce-datan noutamista varten.
Testipisteanalyysi	Funktiopisteanalyysiin perustuva ja testipisteitä hyödyntävä menetelmä ohjelmistoprojektin testausvaiheen työmäärän arviointiin.

Kuviot

Sisällys

1 Johdanto

Ohjelmistokehitys on monimutkainen tehtävä, jonka kustannuksia on vaikea arvioida. Projektien epäonnistumiseen vaikuttavat erityisesti liian alhainen kustannusarvio ja tiukka aikataulu. Näin käy edelleen, vaikka asia on yrityksissä tiedostettu (Trendowicz 2013). Onnistunut ohjelmistokehitysprosessin kustannusarviointi tuo kuitenkin monia hyötyjä. Se helpottaa tarjousten tekemistä, toteutettavan hankkeen aikataulutusta ja läpivientiä. Liian pieneksi arvioitu budjetti taas johtaa todennäköisesti ohjelmiston laadun heikkenemiseen ja viivästymisiin. (Ali ja Gravino 2019)

Ohjelmistokehitysprosessi sisältää useita erilaisia tehtäväalueita liittyen mm. suunnitteluun, ohjelmointiin, testaukseen sekä tukitoimintoihin, joten eri osa-alueisiin kuluvaa aikaa on vaikea ennustaa. Tarve kehittää luotettavia menetelmiä ja näin parantaa kustannusten arviointia on ilmeinen. (Radliński 2023)

Nykyään ohjelmistot ovat levinneet laajasti moniin eri paikkoihin ja laitteisiin ja ohjelmistojen koko on kasvanut niin suureksi, että ohjelmistotestaus on välttämätöntä. Tämän lisäksi automatisointi ja erilaiset sovellukset leviävät jatkuvasti kriittisempiin järjestelmiin, kuten liikenteeseen ja asuntoihin, minkä myötä järjestelmillä ei ole varaa toimia väärin tai päätyä käsittelemättömiin vikatiloihin (Ammann ja Offutt 2016). Historiassa on olemassa useita esimerkkejä testauksen puutteista johtuneista virheistä ja onnettomuuksista, joissa on menetetty joko kriittisiä tietoja tai pahimmassa tapauksessa ihmishenkiä (Ammann ja Offutt 2016).

Ohjelmistotestaus on prosessi, jossa varmistetaan ohjelman tai ohjelmiston täyttävän etukäteen määritellyt tavoitteet sekä teknisesti että liiketoiminnan kannalta (Quadri ja Farooq 2010). Toisen määritelmän mukaan ohjelmistotestauksen yhtenä tärkeimpänä edellytyksenä on suorittaa ohjelma virheiden löytämiseksi (Singh ja Singh 2012). Virheitä voivat olla esimerkiksi väärät toiminnallisuudet, väärät lopputulemat tai toimintapolut, joiden avulla ohjelmaa tai ohjelmistoa voidaan käyttää väärin.

Testauksella pyritään huomaamaan ongelmat mahdollisimman aikaisessa vaiheessa. Mitä pidemmälle kehitysvaihetta prosessi etenee, sitä kalliimmaksi korjaustyö tulee. Harris (2022) mukaan kustannukset voivat nousta jopa seitsemänkertaiseksi, jos virhe huomataan vasta

tuotannossa, kuin jos kyseinen virhe huomattaisiin testausvaiheessa. Resurssien näkökulmasta onkin viisasta panostaa huolelliseen testaukseen jo kehittämisvaiheessa, erityisesti testausvaiheessa, jotta mahdollisimman suuri osa virheistä saadaan korjattua ennen varsinaista tuotantokäyttöä.

Tämän tutkimuksen tavoitteena on selvittää, miten Salesforce-projektin testauksen työmääräarviointia voidaan helpottaa ja yksinkertaistaa työkalulla. Tutkimus on jatkotutkimusta aikaisemmin toteutettuun tutkimukseen, jossa samalle kohdeorganisaatiolle kehitettiin testausstrategia Salesforce-kehitykseen. Tutkimuksen tutkimusmenetelmänä käytettiin suunnittelutiedettä ja tutkimuksen tuloksena sekä artefaktina kehitettiin tutkimuksessa olleelle kohdeorganisaatiolle työkalu testauksen työmäärän arviointiin.

Tutkimusongelma muodostettiin aiemman tutkimuksen yhdestä esitetystä jatkotutkimusaiheesta, joka oli "testausstrategian käytettävyyden kehittäminen"(Virtanen ja Lehikoinen 2023). Tutkimusongelmasta käytiin keskustelua aiemman tutkimuksen toteuttaneiden henkilöiden kanssa, jonka pohjalta saatiin näkökulmia erilaisiin ongelmiin. Tämän keskustelun ja monien pohdintojen tuloksena tutkimuksen tutkimusongelmaksi muodostui testauksen työmääräarvioita toteuttava työkalu testausstrategiadokumentin sijaan. Toteutettava työkalu tulee käyttämään pohjanaan testausstrategiassa esitettyjä malleja ja ratkaisuja. Tähän tutkimusongelmaan pyrittiin vastaamaan keskeisten tutkimuskysymysten kautta. Tutkimuksen tutkimuskysymyksiksi määriteltiin seuraavat kysymykset:

1. Miten Salesforce-projektin testauksen työmäärän arviointia voi tukea työkalulla?
2. Millä tavoin työkalu yksinkertaistaa työmäärän arviointia?

Pro gradu -tutkielma etenee seuraavasti. Johdantoluvun jälkeen luvussa 2 esitellään aikaisempaa tutkimusta ohjelmistoprojektien työmäärän sekä testauksen työmäärän arviointimenetelmistä. Luvussa 3 käydään läpi yleisesti, mikä on Salesforce sekä miten Salesforce-ympäristöjä voidaan kehittää ja testata. Luvussa 4 esitellään tutkielmassa käytetty tutkimusmenetelmä ja tarkastellaan, miten tutkimusmenetelmää sovelletaan tässä tutkimuksessa. Luvussa 5 käsitellään tutkimuksessa toteutetun alkuhaastattelun tuloksia sekä käydään läpi, miten tutkimuksessa kehitetty artefakti on toteutettu. Luvussa 6 käydään läpi artefaktin arviointia varten toteutettu loppuhaastattelu ja sen tulokset sekä reflektoidaan toteutettua artefaktia

tulosten perusteella. Luvussa 7 pohditaan tutkimuksen tuloksia, onnistumisia ja puutteita, vastataan tutkimuskysymyksiin sekä esitellään mahdollisia jatkotutkimusaiheita. Yhteenvedossa luvussa 8 käydään läpi tutkimuksen tärkeimmät aiheet tiivistetysti.

2 Kustannusarviointi- ja työmääräarviointimenetelmiä

Tutkimuskirjallisuuden perusteella kustannusarviointimenetelmiä voidaan luokitella eri tavoin. Huang, Li ja Xie (2015) jakavat menetelmät asiantuntija-arvioihin, parametrisiin malleihin sekä koneoppimismenetelmiin. Asiantuntija-arviot perustuvat asiantuntijoiden kokemukseen ja osaamiseen aikaisemmista projekteista. Parametrisissä malleissa kustannusarviointi perustuu matemaattisten mallien hyödyntämiseen. Tekoälyä hyödynnetään koko ajan enemmän, joten Huang, Li ja Xie (2015) ovat katsoneet tarkoituksenmukaiseksi luokitella nämä menetelmät omaan luokkaansa. Tässä työssä keskitytään tarkastelemaan asiantuntija-arvioita ja parametrisiä malleja. Alaluvussa 2.1 käydään kustannus- ja työmääräarviointia yleisellä tasolla. Alaluvussa 2.2 syvennytään asiantuntija-arviointien erilaisiin malleihin ja alaluvussa 2.3 puolestaan syvennytään parametrisiin malleihin. Viimeisessä alaluvussa 2.4 otetaan testauksen näkökulma mukaan työmäärän arviointeja tehdessä.

2.1 Yleisesti kustannusarvioinnista ja työmäärän arvioinnista

Ohjelmistokehitysprosessi sisältää useita erilaisia tehtäväalueita liittyen mm. suunnitteluun, ohjelmointiin, testaukseen sekä tukitoimintoihin, joten eri osa-alueisiin kuluvaa aikaa on vaikea ennustaa. Tarve kehittää luotettavia menetelmiä ja näin parantaa kustannusten arviointia on ilmeinen. (Radliński 2023)

Radliński (2023) mukaan ohjelmistokehitysprosessien kustannusarviointia on tutkittu paljon, mutta ne keskittyvät pääosin prosessin kokonaisarviointiin. Osa-alueiden kustannusten arviointia on tutkittu vähän. Esimerkiksi testaus olisi hänen mukaansa merkittävä ja haastava osa-alue. (Radliński 2023)

Ohjelmistokehitysprosessin kustannusarviota laadittaessa on yleinen käytäntö arvioida testauksen osuus prosenttiosuutena kokonaiskustannuksista. Tutkimusten mukaan prosenttiosuus kuitenkin vaihtelee kymmenestä kuuteenkymmeneen prosenttiin, mikä vaikeuttaa projektista vastaavien henkilöiden työtä arvioida testauksen kustannuksia. (López-Martín 2022). López-Martín (2022) mukaan on julkaistu satoja tutkimuksia vuodesta 1981 koskien ohjelmistokehitysprosessin kustannusten arviointia ja vain 31 tutkimusta siitä, mikä on ohjelmis-

totestauksen osuus kustannuksista.

Kustannusarvion osuvuuden parantamiseksi käytettyihin termeihin tulisi kiinnittää erityistä huomiota. Kustannusarvioiden tarkkuus ei ole juurikaan parantunut, sillä termien käytön vaihtelevuus vaikeuttaa saatujen arviointitulosten tarkempaa tulkintaa ja johtopäätösten tekemistä. Kuitenkin hyvin tehty kustannus- ja työmääräarvio on kymmenen tärkeimmän tekijän joukossa arvioitaessa ohjelmistokehitysprosessin onnistumista. (Grimstad, Jørgensen ja Moløkken-Østvold 2006)

Trendowicz (2013) mainitsee, että ohjelmistokehityksen kustannuksia arvioitaessa tutkimuskirjallisuudesta voi löytää kaksi toteutustapaa: kustannusarviointi ja työmääräarviointi. Kustannus (cost estimation) viittaa palvelun tai tuotteen valmistukseen liittyvään rahalliseen kustannukseen eli sitä mitataan rahayksikköinä. Työmäärä (effort estimation) viittaa palvelun tai tuotteen valmistukseen liittyvään työpanokseen ja mittayksikkö on työtunti tai muu työaikaa mittaava yksikkö. Termien vakiintumattomuuden takia ohjelmistokehityksen tutkimuskirjallisuudessa termiä kustannus käytetään usein termin työmäärä synonyyminä (Trendowicz 2013). Tässä tutkimuksessa käytämme yleisesti termiä työmäärä.

Käytännössä on aina muistettava, että työmääräarviointi ei todennäköisesti kata yleiskustannuksia, kuten suunnittelua, budjetointia ja hinnoittelua. Työmääriä arvioitaessa on myös tärkeä varmistaa, että arvioitu ja todellinen työmäärä ovat vertailukelpoisia. Grimstad, Jørgensen ja Moløkken-Østvold (2006) mainitsevat, että nämä kaksi asiaa voivat tuntua itsestään selviltä asioilta, mutta niitä ei ole useinkaan noudatettu. Työmääräarviointeja hyödynnetään yleisesti projektien suunnittelussa, projektien hinnoittelussa ja budjetoinnissa, joten yritysten näkökulmasta termien selkeys vähentäisi virheitä tulosten tulkinnassa, parantaisi kustannusarvioiden tarkkuutta sekä vahvistaisi kokemukseen pohjautuvaa osaamista. (Grimstad, Jørgensen ja Moløkken-Østvold 2006).

2.2 Asiantuntija-arviointi

Asiantuntija-arviointi ja siihen perustuvat menetelmät ovat yksi yleisimmistä ja vanhimmista ohjelmistokehitysprojektin kustannusten arviointimenetelmistä. Menetelmissä hyödynnetään yhden tai useamman asiantuntijan aikaisempia kokemuksia projekteista. Kokemuksen

lisäksi heillä on sekä osaamista toimialasta että tietoa organisaatiosta, mikä antaa hyvät valmiudet työmääräarvioiden tekemiseen (Molokken ja Jorgensen 2003). Trendowicz ja Jeffery (2014) mukaan asiantuntija-arvioita kannattaa käyttää silloin, kun kvantitatiivista dataa ei ole saatavilla. Esimerkkejä asiantuntija-arvioon perustuvista menetelmistä ovat muun muassa suunnittelupokeri ja Wideband Delphi (Trendowicz ja Jeffery 2014). Suunnittelupokeri ja Wideband Delphi ovat kummatkin ryhmätyöskentelyä hyödyntäviä menetelmiä (Mahnič ja Hovelja 2012).

Suunnittelupokerissa jokainen kehitystiimin jäsen arvioi itsenäisesti jokaisen käyttäjätarinaan tarvittavan työmääräarvion. Tämän jälkeen jokainen käyttäjätarina käydään läpi ja ryhmän jäsenet paljastavat omat arvionsa samanaikaisesti. Ryhmän jäsenet käyvät keskenään keskustelua tehdyistä arvioista ja tekevät arvion uudelleen. Kaavaa toistetaan, kunnes arviosta on päästy yhteisymmärrykseen. (Mahnič ja Hovelja 2012)

Wideband Delphi on hyvin samankaltainen menetelmä kuin suunnittelupokeri. Sen kehitti Boehm vuonna 1981. Menetelmässä ryhmän jäsenet tekevät arvioinnit anonyymisti, jonka jälkeen he kokoontuvat keskustelemaan eroista arvioinneissa. Tätä toistetaan, kunnes päästään yhteisymmärrykseen. (Mahnič ja Hovelja 2012)

2.2.1 Ylhäältä alas ja alhaalta ylös -menetelmät

Ylhäältä alas -menetelmässä ohjelmistokehitysprojektin työmäärän arviointi tehdään laskemalla yhteissumma kaikille projektin kehitystehtäville ja tuotteille, joita projektissa toteutetaan. Alhaalta ylös -menetelmässä toimitaan päinvastoin. Tällöin projektiin tarvittava työ, erilaisiin aktiviteetteihin ja vielä jokaiselle aktiviteetille yksitellen arvioidaan työmäärä. Kun tätä menetelmää käytetään, täytyy kehitettävä ohjelmisto suunnitella etukäteen ja arvioijan on tiedettävä jo projektin alussa, mistä eri komponenteista se koostuu. Projektin kokonaistyömäärä saadaan laskemalla tehdyt arviot yhteen. Alhaalta ylös -menetelmää voidaan hyödyntää myös projektin myöhemmissä vaiheissa, jos esimerkiksi projektin vaatimukset muuttuvat. Ylhäältä alas -menetelmää voidaan hyödyntää vain projektin alkuvaiheessa. (Trendowicz ja Jeffery 2014; Shekhar ja Kumar 2016)

Tutkimuskirjallisuudessa asiantuntija-arvioita on kuitenkin pidetty ongelmallisina. Grimstad

ja Jørgensen (2007) havainnoivat tutkimuksessaan asiantuntija-arvioiden olevan hyvin epäjohdonmukaisia. Epäjohdonmukaisuus lisääntyi arvioinneissa, kun arvioitavien tehtävien koko ja monimutkaisuus kasvoivat (Grimstad ja Jørgensen 2007). Trendowicz ja Jeffery (2014) havaitsivat, että asiantuntija-arviot perustuvat asiantuntijan omakohtaiseen arviointikykyyn, joten arviointiin liittyy aina jonkin verran epävarmuutta.

2.2.2 Analogiaan perustuva arviointi

Analogiaa hyödyntävässä työmääräarvioinnissa projektissa tarvittavaa työmäärää verrataan aikaisempien vastaavanlaisten projektien toteutuneisiin työmääräarvioihin käyttäen analogista päättelyä. Jotta menetelmä toimisi, vaatii se ainakin yhden, mielellään useamman projektin, jotka ovat vaatimuksiltaan ja ominaisuuksiltaan tarpeeksi lähellä arvioitavaa projektia. (Shekhar ja Kumar 2016). Arviointi voidaan toteuttaa joko koko projektin laajuudella tai yksittäisten ohjelmiston osien laajuudella. Yksittäisiä ohjelmiston osia arvioimalla saatetaan helpommin löytää yhtäläisyyksiä aikaisemmin toteutettuihin projekteihin. (Walkerden ja Jeffery 1999)

Analogiaan perustuvassa arvioinnissa on haasteena löytää aikaisempia projekteja, jotka vastaavat uudessa projektissa kehitettävän ohjelmiston ominaisuuksia ja vaatimuksia. Hyvin samankaltaiselta vaikuttava projekti saattaa osoittautua täysin epärelevantiksi arvioitaessa uuden projektin työmääriä. (Keung 2009). Walkerden ja Jeffery (1999) tekemässä tutkimuksessa huomattiinkin, ettei selkeitä käytäntöjä ole luotu siihen, miten arvioida aikaisempien projektien sopivuutta vertailuaineistoksi.

2.3 Parametriset mallit

Parametriset menetelmät perustuvat projektin matemaattiseen mallinnukseen. Työmääräarvio lasketaan useiden muuttujien muodostamien funktioiden avulla. Muuttujia ovat projektien kustannustekijät, joita ovat tuote-, tietokone-, henkilöstö- ja projektitekijät (Boehm ym. 1995).

2.3.1 COCOMO I ja COCOMO II

Yksi tunnetuimmista parametrisista työmäärän arviointimenetelmistä on Boehmin vuonna 1981 esittelemä Constructive Cost Model -menetelmä, mikä tunnetaan yleisimmin lyhenteellä COCOMO. Se on regressioon perustuva malli ja se kehitettiin 63 ohjelmistoprojektin pohjalta (Singh ja Misra 2012). Mallista on olemassa kolme versiota: perusmalli, keskitason malli ja yksityiskohtainen malli. Kaikki kolme mallia ovat kaavan 2.1 mukaisia. Kaavan muuttuja *KLOC* kuvaa ohjelmiston kokoa. Yleensä ohjelmiston koko on annettu lähdekoodin rivien määränä. Vakiot *a* ja *b* kuvaavat ohjelmistoprojektin muita kustannustekijöitä, jotka riippuvat siitä, onko projekti orgaaninen, puoliksi erillinen (engl. Semi-detached) vai sulautettu. (Shekhar ja Kumar 2016)

$$arvio = a * (KLOC)^b \tag{2.1}$$

Perusmallia voidaan hyödyntää karkean työmäärän arvioimiseen nopeasti, mutta sen tarkkuus voi olla heikko, koska se ei ota huomioon projektin eri kustannustekijöitä. Keskitason mallissa otetaan huomioon myös projektin kustannustekijät. Nämä voidaan jakaa neljään luokkaan: projektin attribuutit, tuotteen attribuutit, henkilöstön attribuutit sekä laitteiston attribuutit. Jokaiseen luokkaan kuuluu joukko avustavia attribuutteja. Keskitason mallia hyödynnetään, kun työmääräarvioista halutaan tarkempia. Yksityiskohtaisen mallin avulla voidaan ottaa huomioon myös projektin eri vaiheiden vaikutus työmääräarviointeihin. (Suri ja Ranjan 2012)

COCOMO:sta jatkokehitettiin paranneltu versio COCOMO II. Se ottaa paremmin huomioon uudistuneet ohjelmiston kehitysprosessit. Kehitysprosessit hyödyntävät uusia ohjelmistokehityksen lähestymistapoja, joita ovat mm. ohjelmistojen uudelleen käyttö ja nopean kehityksen prosessit sekä kaupallisten ohjelmistojen hyödyntäminen.

2.3.2 Putnamin malli

Putnamin malli on myös esimerkki parametrisestä työmäärän arviointimenetelmästä. Ohjelmiston koko on merkittävä kustannustekijä, joten koon mittaukseen on kehitetty erilaisia me-

netelmiä esimerkkinä koodirivien (lines of codes) määrä (Boehm ym. 1995). Putnamin malli on hyvin samanlainen kuin alkuperäinen COCOMO. Molemmat menetelmät hyödyntävät koodirivien määrää kuvaamaan ohjelmiston laajuutta. Kemerer (1987) mukaan koodirivien hyödyntäminen ohjelmiston koon arvioimisessa on saanut kritiikkiä, koska koodirivien määrän ennustamista projektin alkuvaiheessa on pidetty vaikeana (Kemerer 1987). Kaava 2.2 vastaa Putnamin kehittämää mallia. Muuttuja D_0 kuvaa työvoimakertymää. Se voi saada arvoja välillä 8 ja 27. Arvo 8 vastaa uutta kehitettävää ohjelmistoa ja arvo 27 kuvaa uudelleen rakennettavaa ohjelmistoa. Muuttuja E on ympäristötekijä, jolla kuvataan saatavilla olevaa kehityskapasiteettia. Viimeisenä on muuttuja S, mikä vastaa kehitettävän ohjelmiston kokoa koodiriveinä. Putnam kehitti menetelmän käyttämiseen oman ohjelmiston, mikä on nimeltään SLIM (Software lifecycle management). (Shekhar ja Kumar 2016)

$$arvio = \left(D_0^{\frac{4}{7}} \times E^{\frac{9}{7}} \right) \times S^{\frac{9}{7}} \tag{2.2}$$

Tähän asti on perehdytty työmääräarvioiden arviointimenetelmiin yleisesti. Seuraavaksi siirrytään käymään läpi testauksen työmääräarviointia.

2.4 Testauksen työmäärän arviointi

Tärkeä osa ohjelmistokehitysprojektia on ohjelmistotestaus, jonka tarkoituksena on parantaa sekä ylläpitää ohjelmiston laatua ja luotettavuutta. Testauksen työmääräarvioinnilla taas tarkoitetaan tietyn ohjelmistoprojektin testaukseen tarvittavan työmäärän, koon sekä aikataulun arviointia eri menetelmiä hyödyntäen (Jayakumar ja Abran 2013).

Bluemke ja Malanowska (2022) toteavat tekemässään systemaattisessa kirjallisuuskatsauksessa, että tutkimusta testauksen työmääräarvioinnista on viimeisten vuosikymmenien aikana tehty vähän, eikä kirjallisuudesta löydy yhtenäistä linjaa testauksen työmääräarvioinnista. Analyyttisiä malleja esiintyy aikaisemmassa tutkimuksessa vähän ja yleensä testauksen työmääräarviointi on perustunut kokemukseen. Yleisimmät kirjallisuudessa esiintyneet analyyttiset mallit olivat testipisteanalyysi (engl. Test Point Analysis) sekä käyttötapauspisteet (engl. Use Case Points). (Bluemke ja Malanowska 2022)

Testipisteanalyysimenetelmän avulla voidaan arvioida ohjelmistokehitysprojektissa toteutettavan järjestelmätestauksen tai hyväksyntätestauksen työmäärä. Menetelmän esittelivät Veenendaal van ja Dekkers vuonna 1999 ja se perustuu funktiopisteanalyysiin (engl. Function Point Analysis). Jotta arvioinnin tekijä voi hyödyntää testipisteanalyysia, hänen täytyy tietää kolme asiaa, joita ovat testattavan ohjelmiston koko, testausstrategia sekä tuottavuuden taso. Ohjelmiston koolla viitataan ohjelmistolle laskettujen funktiopisteiden määrään. Testausstrategia määrittää laatuvaatimuksen testattavalle ohjelmistolle tai tietojärjestelmälle. Tuottavuuden taso taas määrittää kuinka kauan yhden testipisteen toteuttamiseen menee aikaa. (Veenendaal ja Dekkers 1999)

Testipisteanalyysimenetelmä sisältää viisi vaihetta. Ensimmäisessä vaiheessa määritellään ja lasketaan dynaamisten testipisteiden määrä. Dynaamisilla testipisteillä tarkoitetaan testattavia asioita, jotka liittyvät ohjelman toiminnallisuuteen ja suorittamiseen. Toisessa vaiheessa määritellään ja lasketaan staattiset testipisteet. Staattisilla testipisteillä tarkoitetaan testipisteitä, jotka eivät liity koodin suorittamisen. Näitä ovat esimerkiksi koodin vertaisarvioinnit tai koodin staattinen analysointi työkalulla. Kolmannessa vaiheessa dynaamiset ja staattiset testipisteet lasketaan yhteen. Neljännessä vaiheessa lasketaan primääriset testitunnit. Nämä ilmaisevat testaamisen elinkaaren eri työvaiheiden suorittamiseen tarvittavaa työmäärää. Työvaiheita ovat esimerkiksi testaamisen valmistelu, testauksen määrittely, testien suorittaminen sekä testin päättäminen. Viimeisenä saadaan laskettua testaamisen kokonaistyömäärä. Tässä vaiheessa primaarisiin testitunteihin lisätään testaamisen johtamiseen liittyvien aktiviteettien tarvitsema työmäärä. (Veenendaal ja Dekkers 1999)

Toinen kirjallisuudessa eniten esiintyvä menetelmä on käyttötapauspisteet. Menetelmän esitteli Nageswaran vuonna 2001. Käyttötapausmenetelmää käyttämällä määritellään kehitettävän järjestelmän toimintalogiikka käyttäjän määrittelemien ehtojen vallitessa. Käyttötapauspistemenetelmässä jokaiselle käyttötapaukselle ja toimijalle lasketaan määritellyn tyypin ja monimutkaisuuden mukainen painoarvo. Painoarvojen avulla saadaan laskettua yksinkertaisella kaavalla mukautettu määrä käyttötapauspisteitä. Lopuksi käyttötapauspisteet muunnetaan henkilötyötunneiksi, joita tarvitaan testien suunnitteluun, kirjoittamiseen ja suorittamiseen. (Bluemke ja Malanowska 2022)

3 Testaus Salesforce-kehityksessä

Kuten muussakin ohjelmistokehityksessä, myös Salesforcessa on syytä suorittaa testausta. Alaluvussa 3.1 tutustutaan yleisesti Salesforceen, asiakastietojärjestelmään, pilvipalveluun ja Salesforcen kehittämiseen. Alaluvussa 3.2 syvennytään tarkemmin Salesforcen testaukseen liittyviin asioihin.

3.1 Salesforce

Tässä alaluvussa tutustutaan tarkemmin Salesforceen; mikä se on, mitä ominaisuuksia siihen liittyy ja miten sitä kehitetään toimivaksi erilaisiin asiakastarpeisiin. Samassa sivutaan oleellisesti Salesforcea koskeviin käsitteisiin ja tekniikoihin, kuten asiakastietojärjestelmään ja pilvipalveluun.

3.1.1 Asiakastietojärjestelmä

Ennen Salesforceen tarkemmin tutustumista, on tärkeää avata asiakastietojärjestelmä (engl. Customer relationship management, CRM) terminä. CRM on järjestelmä, joka mahdollistaa organisaation keskittymisen suhteisiin ihmisten kanssa, kuten kollegoiden, toimittajien, palvelun käyttäjien ja loppuasiakkaiden välillä, oli sitten kyseessä nykyinen tai tuleva asiakas (Gupta, Verma ja Janjua 2018; Manchar ja Chouhan 2017). Terminä se kuvastaa käytäntöjä, tekniikoita ja teknologioita, joita käytetään asiakassuhteiden ja tietojen ylläpitämiseen (Manchar ja Chouhan 2017). CRM-järjestelmän tarkoituksena on helpottaa myynnin- ja asiakkaiden yhteydenpidon hallintaa, työnkulun prosessien edistämistä ja monia muita prosesseja, joita jokapäiväisessä työssä tulee vastaan (Gupta, Verma ja Janjua 2018). Järjestelmän avulla myyjillä on aina ajantasainen tieto siitä, mitä asiakkaiden kanssa on sovittu aiemmin ja mitä yksittäinen asiakas on aiemmin ostanut. Tämä helpottaa huomattavasti myyntitilanteessa tai jos täytyy selvitellä ongelmaa asiakkaan kanssa sovituista asioista.

3.1.2 Salesforce asiakastietojärjestelmänä

Salesforce on yhdysvaltalainen yritys, joka sai alkunsa pelkästään asiakastietojärjestelmänä. Ennen Salesforcea asiakastietojärjestelmät vaativat isoja investointeja yrityksiltä, kun järjestelmiä kehitettiin jokaiselle yritykselle vastaamaan niiden yksilöllisiä toimintaprosesseja ja tarpeita. Tämän lisäksi järjestelmät edellyttivät omia palvelimia, sovelluksia tai jopa laitteistoja yrityksien tiloissa, joissa asiakastietojärjestelmää pyöritettiin. (Patel ja Chouhan 2016). Tähän malliin tapahtui muutos Salesforcen myötä, kun järjestelmä pyörii kokonaan internetissä pilvipalveluna (Gupta, Verma ja Janjua 2018). Tämä mahdollistaa asiakkaille helpon lähestymistavan järjestelmään olinpaikasta riippumatta (Patel ja Chouhan 2016). Pilvipalveluun tutustutaan seuraavassa alaluvussa 3.1.3.

CRM-järjestelmien välillä on toiminnallisia ja rakenteellisia eroja siinä, kuinka dataa käsitellään. Salesforcen tapauksessa data organisoidaan objekteihin (engl. object) ja tietueisiin (engl. record) (Manchar ja Chouhan 2017). Objekti on ylätason käsite, joka pitää sisällään tietueita. Manchar ja Chouhan (2017) kuvailevat tämän helposti ymmärrettävällä tavalla siten, että objektit ovat laskentataulukon välilehtiä ja tietueet ovat yksittäisiä rivejä kyseisen taulukon välilehdessä. Tietueilla on siis erilaisia tietoja liittyen kyseiseen objektiin, joiden avulla saadaan kerättyä ja tarkasteltua haluttuja arvoja tarvittaviin asioihin liittyen.

Asiaa voidaan pilkkoa vielä pienempiin osioihin, jotta päästään Salesforcen muovautuvuuteen käsiksi. Salesforcessa on vakiona suoraan paketin mukana tulevia objekteja, joita ovat muun muassa tilit, yhteyshenkilöt, liidit, myyntimahdollisuudet, kampanjat, mittaristot ja raportit (Manchar ja Chouhan 2017). Jos nämä objektit eivät riitä liiketoiminnan edellyttämiin tarkoituksiin, voidaan järjestelmään lisätä mukautettuja objekteja, joiden avulla saadaan kerättyä haluttu tieto talteen sille osoitettuun paikkaan.

Objektien alle jäävät vielä kentät, joihin tietoa loppupeleissä kerätään tietueissa. Sekä mukautetuilla että vakio-objekteilla on tietyt vakiokentät, mutta myös mukautetut kentät ovat mahdollisia (Manchar ja Chouhan 2017). Uusien kenttien avulla mahdollistetaan kaiken tarvittavan tiedon kerääminen ja tieto saadaan ryhmiteltyä järkevällä tavalla.

Alkuajoista on tultu paljon eteenpäin eikä Salesforce ole enää pelkkä asiakastietojärjestelmä. Erilaisia tuotteita on nykyisin useita liityen myyntiin, asiakaspalveluun, markkinointiin ja

moneen muuhun yritystoimintaa edistäviin tarpeisiin (Gupta, Verma ja Janjua 2018). Näin mahdollistetaan tilanne, jossa käytetään yhden alustantuottajan tuotteita yrityksen sisäisillä eri tiimeillä, mutta kaikki oleellinen data kulkee tiimien välillä. Mikäli tietoa tarvitaan myös muista järjestelmistä, voidaan Salesforcen ja toisen järjestelmän välille tehdä integraatioita, joiden avulla tieto kulkeutuu järjestelmien välillä. Tähän aiheeseen paneudutaan tarkemmin tulevissa alaluvuissa.

3.1.3 Pilvipalvelu

Salesforcen kaltainen toimintarakenne tunnetaan paremmin termillä Software as a Service (SaaS). SaaS-järjestelmissä infrastruktuuri ja alusta sijaitsevat pilvessä, jonka päällä itse sovellus toimii. Järjestelmän käyttäjien ei tarvitse ladata erikseen mitään sovellusta omille tietokoneilleen, vaan kaikkeen tietoon, dataan ja tietokantoihin pääsee käsiksi internetin kautta, pilvessä sijaitsevien palvelimien välityksellä (Patel ja Chouhan 2016; Gupta, Verma ja Janjua 2018). Käytettävyyttä lisää vapaus päästä järjestelmään millä tahansa laitteella, jossa on internetyhteys. Käyttäjät pääsevät seuraamaan tai tekemään tarvittavia toimenpiteitä mistä vain, milloin vain. Järjestelmän kannalta helpottavana tekijänä on ylläpidettävyys järjestelmässä, kun jokaisella käyttäjällä on aina käytössä sama versio ilman käyttäjän tekemiä päivityksiä (Patel ja Chouhan 2016; Gupta, Verma ja Janjua 2018). Näistä on yhteenvetona vertailutaulukko pilvipalvelun ja tilapalvelimen ohjelmistoratkaisun välillä kuviossa 1. SaaS-järjestelmät hinnoitellaan usein käyttäjämäärän mukaisesti, jolloin lisenssien hinnat pysyvät maltillisina myös pienemmille organisaatioille. Tämä tekee hankinnasta helpompaa ja kannattavampaa (Patel ja Chouhan 2016). Salesforce muutti markkinoita olemalla ensimmäinen CRM-järjestelmä, joka hyödynsi SaaS:a (Manchar ja Chouhan 2017).

Salesforcessa on ominaisuuksia, jotka tukevat sen käyttöä alustana (engl. Platform as a Service, PaaS). Gupta, Verma ja Janjua (2018) määrittelevät, että PaaS on tietokoneen kehys, joka sisältää laitteistokomponentit, käyttöjärjestelmän ja joissain tapauksissa käyttöliittymät sekä sovelluskehitystyökalut. Sovelluskehityksen ja kehitetyn ohjelman suorittamiseen tarkoitettujen ympäristöjen lisäksi PaaS tarjoaa myös resursseja, jotka tukevat sovelluksen elinkaaren luomista (Gupta, Verma ja Janjua 2018). Salesforcen PaaS-mahdollisuuksia ovat esimerkiksi Heroku ja Salesforce Platform. Herokun avulla voidaan kehittää verkkoappli-

Funktio	**Tilapalvelin**	**Pilvipalvelu**
Ohjelmisto	Asennetaan jokaiselle tietokoneelle	Toimitetaan internetin välityksellä
Pääsy	Pelkästään tietokoneelta, jossa sovellus on asennettuna	Internetin välityksellä miltä tahansa laitteelta
Päivitykset	Manuaalisesti jokaiselle laitteelle erikseen, monimutkainen prosessi	Automaattisesti palvelun tarjoajan toimesta, käyttäjän ei tarvitse huolehtia päivittämisestä
Versiot	Monia eri versioita, joita täytyy ylläpitää	Yksi kooditietokanta, joka ei vaadi rakenteellista ylläpitämistä
Komponentit	Kaikki tarvittavat komponentit täytyy ostaa itse, hoitaa ylläpito ja hoitaminen	Maksetaan vain tarvittavista ja käytetyistä ominaisuuksista, jotka tulevat palveluntarjoajan toimesta

Kuvio 1. Vertailu tilapalvelimen ja pilvipalvelun välillä (Manchar ja Chouhan 2017)

kaatioita monilla eri ohjelmointikielillä ja sitä käytetään verkkosovelluksien käyttöönottomallina (Patel ja Chouhan 2016). Salesforce Platform on uudistettu nimitys aikaisemmin tunnetusta Force.com:sta. Tämä alusta on mukana kaikessa Salesforceen liittyvässä kehityksessä, oli sitten kyseessä mukautetut koodiratkaisut tai Salesforcen tarjoamien automaa-

tioratkaisujen kehittäminen (Mathew ja Spraetz 2009). Lisäksi Mathew ja Spraetz (2009) toteavat, että Salesforce Platform tarjoaa paremmat mahdollisuudet kolmannen osapuolen kehittäjien, kummpanien sekä asiakkaiden luoda ja ottaa käyttöön sovelluksia. Manchar ja Chouhan (2017) määrittelevät, että tämä alusta mahdollistaa käyttäjille liiketoimintaa edistävien sovelluksien luomisen ja käyttöönoton välittömästi sillä hetkellä, kun niiden halutaan näkyvän. Heidän mukaansa Force.com oli myös maailman ensimmäinen PaaS. Salesforce Platform tarjoaa Salesforcen kehitykseen, testaukseen ja datan luomiseen tai muokkaamiseen tarvittavat ominaisuudet. Mathew ja Spraetz (2009) mukaan alusta mahdollistaa kehittäjille ja kumppaneille sovellusten rakentamisen ja testaamisen ilman huolta ohjelmisto- ja laitteistoinfrastruktuurista.

Pilvipalvelu (engl. cloud computing) on oleellisessa osassa SaaS- ja PaaS-järjestelmiä. Yksiselitteistä käsitteen avausta termille ei ole olemassa, koska jokainen määrittelee pilvipalvelut hieman eri tavalla. Asia voi myös näyttäytyä eri käyttäjäryhmille hieman eri tavalla, jolloin asia monimutkaistuu entisestään (Lin ym. 2009). Pilvipalvelun perusajatuksena on mahdollistaa kaikkialta saavutettavaa laskentaresurssien jakamista, joka vähentää tarvetta käyttää henkilökohtaisten palvelimien ja laitteiden resursseja tallentamiseen ja laskentaan. (Gupta, Verma ja Janjua 2018). Yksinkertaistettuna mallina Gupta, Verma ja Janjua (2018) esittävät asian rajattomana verkkotallennustilana, joka tarjoaa infrastruktuurin, suoritusympäristön ja palvelut vuokrausperiaatteella. Verkkotallennustilassa maksetaan vain siitä mitä ja miten paljon palvelun resursseja käyttää (Gupta, Verma ja Janjua 2018). Tämä antaa käyttäjille käytön ja tarpeen mukaan pääsyn verkossa oleviin yhteisiin skaalautuviin resursseihin, joita voidaan myös säädellä nopeasti ja helposti (Patel ja Chouhan 2016). Patel ja Chouhan (2016) mukaan näitä erilaisia pilvipalvelun tarjoamia resursseja ovat esimerkiksi verkot, palvelimet, tallennustilat, sovellukset ja palvelut.

3.1.4 Salesforcen kehittäminen

Salesforce tarjoaa alustana paljon vakio-ominaisuuksia, mutta usein nämä eivät riitä asiakkaiden erilaisiin liiketoimintamalleihin ja tarpeisiin. Alusta tarjoaa hyvät mahdollisuudet liiketoimintaa edellyttävien ominaisuuksien kehittämiseen ja prosessien automatisointiin “no-code”, “low-code” ja “pro-code”-ratkaisuilla (Harris 2022). “No-code” ja “low-code”-kehi-

tyksillä tarkoitetaan Salesforcen tarjoamien graafisten ohjelmointityökalujen, kuten Flow builderin avulla toteutettavaa kustomointia, jolloin syvempää ohjelmointiosaamista ei vaadita, eikä koodia tarvitse kirjoittaa. Työkalujen toimintaperiaate perustuu “raahaa ja pudota” tai “osoita ja napsauta” -tekniikoihin, joiden avulla pystytään rakentamaan erilaisia toimintapolkuja ja automaatioita. Vaikka työkalut eivät vaadi varsinaista ohjelmointiosaamista tai -kokemusta, vaativat ne silti pientä tietämystä toimintalogiikasta sekä loogisista ehdoista. (Harris 2022)

Salesforcen graafiset ohjelmointityökalut ovat hyvä tapa toteuttaa yksinkertaisempia toimintoja, mutta monimutkaisemmissa ratkaisuissa on parempi siirtyä “pro-code”-ratkaisuihin, eli varsinaiseen ohjelmointiin. Salesforcen kustomointiin käyttäen ohjelmointia liittyy vahvasti Apex, Salesforcen tarjoama Javan kaltainen, vahvasti tyypitetty ja oliosuuntautunut ohjelmointikieli (Patel ja Chouhan 2016; Harris 2022). Apexin avulla voidaan toteuttaa toimintalogiikkaa useisiin käyttöliittymästä toteutettuihin tapahtumiin, kuten nappien painalluksiin tai tietueiden päivityksen yhteyteen (Patel ja Chouhan 2016). Tämänlaisia automaatioita, jotka toimivat tietueisiin tapahtuvien muutoksien yhteydessä, kutsutaan Apex-triggereiksi. Triggerien avulla voidaan toteuttaa automatisoituja toimintoja tietokantaoperaatioiden yhteydessä, joita ovat esimerkiksi lisääminen (insert), päivittäminen (update) ja poistaminen (delete). Muutokset voidaan toteuttaa tapahtuvaksi ennen tai jälkeen tietokantaan tallentamisen, joka mahdollistaa erilaisten skenaarioiden toteuttamiset. (Mathew ja Spraetz 2009)

Salesforcessa tietokantana toimii ympäristössä oleva data ja tietueet. Apexista tietueisiin pääsee käsiksi Salesforce Object Query Languagen (SOQL) avulla, joka on Structured Query Languagen (SQL) kaltainen kieli (Mathew ja Spraetz 2009). Tämän avulla kehittäjät pääsevät käsiksi juuri niihin tarvittaviin objekteihin ja tietueisiin, joihin halutaan tehdä muutoksia tai lisäyksiä. Toisaalta myös muiden järjestelmien yhdistäminen osaksi Salesforcea onnistuu Apexin ja Salesforcen tarjoamien integrointimahdollisuuksien avulla. Dataa voidaan tuoda osaksi Salesforcea suoraan integraation välityksellä, joka helpottaa manuaalista työtä tietueiden lisäyksessä tai päivittämisessä. Integraatioita pystytään toteuttamaan myös toiseen suuntaan, eli Salesforcesta toiseen järjestelmään, jolloin mahdollistetaan järjestelmien ajantasaisuus ja samankaltaisuus datan osalta. (Mathew ja Spraetz 2009)

Käyttöliittymäkomponentteja rakennettaessa Apexin rinnalle tulee Lightning Web Compo-

nent -ohjelmointiviitekehys (LWC), jonka avulla käyttöliittymän ulkoasua ja toiminnallisuutta muovataan HTML:n ja JavaScriptin avulla (Salesforce 2023a). LWC-komponentteihin pystytään upottamaan toiminnallisuuksia, jotka kutsuvat Apex-luokkien metodeja joko hakemaan tarvittavaa tietoa näytettäväksi käyttöliittymässä tai tallentamaan käyttöliittymään annettuja arvoja tietokantaan. Näitä kahta ominaisuutta hyödyntämällä saadaan toteutettua monimutkaisia kustomoituja ratkaisuja helpottamaan loppukäyttäjien työntekoa.

3.2 Salesforcen testaus

Salesforcen testaaminen sisältää samat peruselementit ja testattavat kohteet, mitä ohjelmistotestaukseen sisältyy yleensäkin. Toimivan alustan perustana on tärkeä saada toteutettua asiakkaille juuri sellaiset ominaisuudet, mitä yritystoiminnan ylläpitäminen vaatii. Tämän saavuttamiseksi on tärkeä tehdä yksikkötestit Apex-luokille, regressio- ja toiminnallisuustestaus ympäristölle sekä lopulta hyväksymistestaus asiakkaan toimesta, ennen muutoksien vientiä tuotantokäyttöön (Mathew ja Spraetz 2009). Tämän alaluvun pääpainot ovat Apex-luokkien testauksessa, johon sisältyy yksikkö-, integraatio- ja järjestelmätestaus sekä Salesforcen vaatimuksissa, jotka liittyvät Apex-luokkien testaamiseen.

Apexiin sisältyy testauksen viitekehys, joka helpottaa automatisoimaan testien suorittamista ja luokkien toiminnallisuuden validoimista (Mathew ja Spraetz 2009). Loppukäyttäjien ei tarvitse huolehtia testien ylläpidosta, koska kaikki Apex-luokat, myös testiluokat ovat alustalla (Mathew ja Spraetz 2009). Mikäli toiminnallisuuteen tulee joitain muutoksia, tulee muutokset hoitaa testiympäristössä. Tuotantoon vietäessä testit ajetaan uudelleen. Tästä johtuen ei tule tilannetta, jossa testit pääsevät vanhenemaan ja ne eivät enää mene läpi tuotannossa. Apex-koodeja ei pysty muokkaamaan suoraan tuotannossa. (Arora ja Gupta 2013)

Salesforcessa on tuotantoympäristön lisäksi tarjolla erilaisia ympäristöjä, joissa kehitys ja testaus suoritetaan ennen tuotantoon vientiä. Kehitysympäristöjä kutsutaan Salesforce-maailmassa sandboxeiksi, joita ovat esimerkiksi Developer, Developer Pro, Partial Copy ja Full Copy -sandboxit (Coxon 2022). Developer ja Developer Pro -sandboxit ovat kehittämiseen tarkoitettuja ympäristöjä, joissa metatiedot kopioidaan suoraan tuotannosta. Developer Pro eroaa Developer -sandboxista vain suuremmalla tallennuskapasiteetilla, joka mahdollistaa parem-

mat edellytykset esimerkiksi integraatiotestaukselle. Partial ja Full Copy -sandboxit kopioivat aiempien lisäksi oikeaa dataa tuotannosta; Partial kopioi vain osan ja Full Copy on täysi kopio tuotantoympäristöstä. Näissä ympäristöissä voidaan suorittaa kokonaisvaltaista testausta järjestelmän toiminnalle oikealla datalla, ilman oikean tiedon häviämisen tai muokkaamisen mahdollisuutta tuotantoympäristössä. (Salesforce, Patel ja Chouhan 2016). Kehittämisen laajuudesta riippuu, kuinka useaa erilaista sandboxia kannattaa tai tarvitsee käyttää kehittämisen ja testauksen aikana. Mikäli Apex-luokkien avulla tehtävää kehitystä toteutetaan, on minimivaatimuksena käyttää Developer-sandboxia. Mikäli kehitystä toteutetaan paljon ja toteutuksista tulee laajoja, niin on hyvä ottaa mukaan Partial tai Full Copy, joissa laajemman testauksen saa suoritettua ennen tuotantokäyttöä. Huomioitavia asioita testauksen aikana ovat aiempien määriteltyjen asioiden lisäksi turvallisuusskenaariot ja tietojen näkyvyysasetukset eri profiileilla, käyttäjäryhmillä ja käyttäjillä. (Arora ja Gupta 2013).

Apex-testiluokat ovat normaaleja Apex-luokkia, jotka määritellään testiluokiksi @isTest-merkinnällä. Testiluokkien yksi suurimmista ominaisuuksista on, ettei testiajojen aikana tapahtuvia muutoksia tallenneta tietokantaan. Tämä ominaisuus varmistaa datan muuttumattomuuden ja eheyden sekä antaa varmuuden testien ajamiseen ympäristöstä huolimatta. (Mathew ja Spraetz 2009)

Apex-testien tulee täyttää tietyt kriteerit ennen tuotantoon viennin mahdollisuutta. Ensimmäisenä oletusarvona on, että testiluokat menevät suoritettaessa läpi ilman virheitä. Toiseksi Apex-luokan tulee saada vähintään 75%:n koodirivikattavuus testiajon aikana. Viimeiseksi kaikkien Apex-luokkaa koskevien triggereiden tulee käynnistyä vähintään kerran, jotta varmistutaan, ettei triggerien toiminta aiheuta suunnittelematonta käytöstä koodin toimintaan. (Mathew ja Spraetz 2009; Harris 2022). Kun edellä olevat kriteerit täyttyvät, voidaan muutokset siirtää sandboxista tuotantoon järjestelmävaatimuksien puitteissa.

Salesforcen rakenne yleisesti pilvipalveluna toimii moniasiakkaisena (multitenant) alustana. Tämän myötä myös Apex-luokat ja -testit ajetaan moniasiakkaisessa ympäristössä. Tämä tuo mukanaan tiettyjä rajoituksia, joilla varmistetaan, ettei yhden organisaation toiminnallisuudet ja ajot vie muiden käyttäjien resursseja. Rajoituksiin kuuluu muun muassa rajoitettu tietokantakyselyiden määrä yhden ajon aikana sekä yleisesti aika, joka kuluu toiminnallisuuden suorittamiseen. Mikäli Apex-luokan tai -testin ajon aikana ajaudutaan näiden rajojen yli,

koko ohjelman suorittaminen keskeytetään ja kaikki tehdyt muutokset palautetaan edelliseen tilaan. (Mathew ja Spraetz 2009)

4 Tutkimusmenetelmä

Aiemmat luvut ovat sisältäneet teoriatietoa tämän tutkimuksen oleellisiin asioihin. Tässä luvussa aletaan paneutua varsinaiseen tutkimusosuuteen, tarkemmin tutkimusmenetelmään. Tutkimuksessa käytetty tutkimusmenetelmä esitellään alaluvussa 4.1 ja tutkimusmenetelmän käyttö tämän tutkimuksen kontekstissa avataan alaluvussa 4.2.

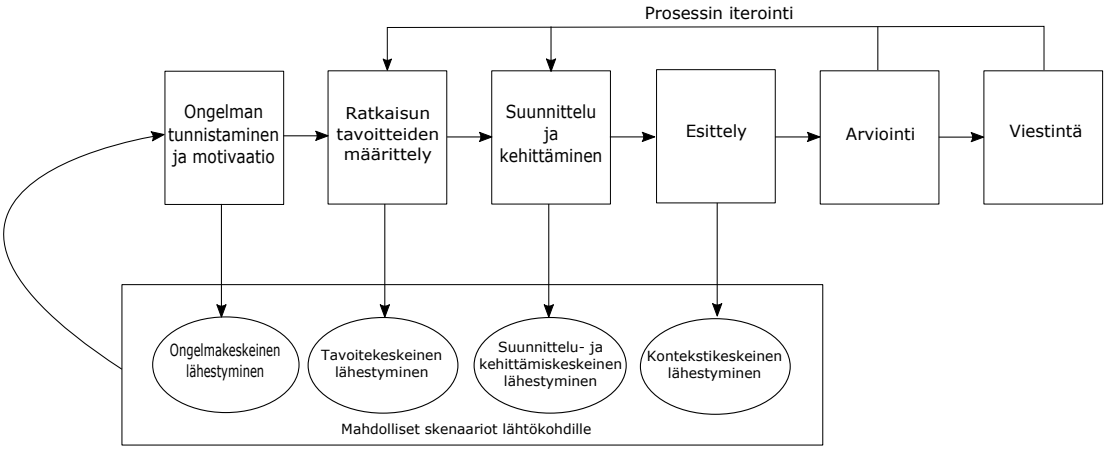
4.1 Tutkimusmenetelmä yleisesti

Tutkimuksessa käytetään tutkimusmenetelmänä suunnittelutiedettä (design science research). Suunnittelutiedettä hyödynnettäessä lopputuloksena syntyy artefakti, jonka tarkoituksena on ratkaista organisaatiossa tunnistettuja ongelmia (Peffers ym. 2007). Peffers ym. (2007) mukaan artefaktiksi kutsutaan tuotosta, joka voi sisältää käsitteitä, malleja, menetelmiä, sosiaalisia innovaatioita tai uusia ominaisuuksia teknisissä, sosiaalisissa tai informatiivisissa resursseissa. Tässä tutkimuksessa suunnittelutiede on sopiva menetelmä, koska toteutamme tutkimuksen yhteydessä testauksen työmääräarviota helpottavan työkalun kohdeorganisaation testausstrategian pohjalta. Toteutettu työkalu toimii tämän tutkimuksen artefaktina.

Kuten edellä kuvailtiin, artefakti on suunnittelutieteen pääroolissa. Artefaktin tulisi olla ongelmaa varten luotu ratkaisu, joka liittyy aiemmin ratkaisemattomaan tärkeään liiketoimintaongelmaan. Artefaktin kehittäminen on etsintäprosessi aiemman teorian ja tiedon perusteella ongelman ratkaisun löytämiseksi, jota lopuksi arvioidaan hyödyllisyyden, laadun ja tehokkuuden perusteella kohdeyleisön näkökulmaan peilaten. (Peffers ym. 2007)

Peffers ym. (2007) esittelevät tutkimuksessaan suunnittelutieteellisen tutkimusmenetelmän (Design Science Research Method, DSRM), jossa prosessi jaotellaan kuuteen eri prosessin vaiheeseen. Prosessin vaiheet ovat nähtävillä kuviossa 2. DSRM on iteratiivinen prosessi, jota jatketaan niin kauan, kunnes haluttu lopputulos on saavutettu. Prosessin eri vaiheet avataan seuraavaksi.

Vaihe 1: Ongelman tunnistaminen ja motivaatio. Ensimmäisessä vaiheessa määritellään tutkimusongelma ja perustellaan ratkaisun arvo. Artefaktin kehittämisessä käytetään hyväksi



Kuvio 2. DSRM-prosessikaavio (Peffers ym. 2007)

ongelman määrittelyä, joten ongelma kannattaa pilkkoa pienempiin osiin, jotta asian monimutkaisuus saadaan selvitettyä. Tämän avulla ratkaisu on helpommin lähestyttävissä. Ratkaisun arvon perusteleminen motivoi tutkijaa ja tutkimuksen kohdeyleisöä etsimään ratkaisua ja hyväksymään saavutetut lopputulokset. Vaiheen oleellisimpia resursseja ovat tieto sekä ongelman tilasta että sen ratkaisemisen tärkeydestä.

Vaihe 2: Ratkaisun tavoitteiden määrittely. Ongelman määrittelyn ja saatavilla olevien tietojen perusteella päätellään ratkaisun tavoitteet, sekä tutkitaan, mitkä ovat mahdollisia ja toteutettavissa olevia ratkaisuja. Tavoitteet jaotellaan määrällisiin ja laadullisiin. Määrällisiä tavoitteita ovat esimerkiksi termit, joissa ratkaisu olisi parempi kuin nykytilanteessa ja laadullisia tavoitteita ovat esimerkiksi kuvaukset uuden artefaktin odotuksista ratkaista olemassa olevat ongelmat. Tämän vaiheen oleellisimpia resursseja ovat tiedot ongelmien tiloista ja niiden mahdollisista ratkaisuista sekä tehokkuudesta.

Vaihe 3: Suunnittelu ja kehittäminen. Tässä vaiheessa päästään konkreettiseen työskentelyyn

ja luodaan artefakti. Artefakti voi olla käsitteellisesti mikä tahansa asia, jonka suunnitteluun käytetään suuri työpanos. Tähän vaiheeseen kuuluu artefaktin halutun toiminnan ja arkkitehtuurin määrittäminen ja artefaktin varsinainen luominen. Jotta toisesta vaiheesta päästään tähän vaiheeseen, täytyy teoriapohjan olla vahvasti hallussa. Näin pystytään toteuttamaan varsinaista suunnittelua ja kehitystä.

Vaihe 4: Esittely. Esittelyvaiheessa osoitetaan artefaktin käyttöä yhden tai useamman esitetyn ongelman ratkaisuun. Tämän voi tehdä esimerkiksi kokeilemalla tehdyn artefaktin käyttöä, simuloida tilanteita, tehdä tapaustutkimus, todistaa asia tai jollain muulla vastaavalla toiminnalla testata artefaktin varsinaista käyttöä. Tämän vaiheen suorittamiseksi vaaditaan vahvaa tietämystä artefaktista ja sen käytöstä ongelmien ratkaisemiseksi.

Vaihe 5: Arviointi. Toiseksi viimeisessä vaiheessa tarkkaillaan ja mitataan artefaktin kykyjä tukea ongelman ratkaisua sekä verrataan vaiheen 4 tuloksia alkuperäisiin tavoitteisiin. Tavoitteiden saavuttamista voidaan mitata erilaisilla keinoilla riippuen artefaktin luonteesta. Esimerkkejä ovat toiminnallisuuden vertailu vaiheen 2 tavoitteisiin, suorituskykymittaukset erilaisiin kohteisiin, kuten budjetteihin tai tuotettuihin tuotteisiin, tyytyväisyyskyselyiden tulokset, asiakaspalautteet tai simulaatiot. Käsitteellisesti tämä voi sisältää minkä tahansa soveltuvan empiirisen tai loogisen todisteen. Tästä vaiheesta voidaan lähteä jatkamaan iteratiivisesti uutta kierrosta vaiheeseen 3 ja parantaa artefaktia saatujen tuloksien pohjalta tai siirtyä seuraavaan, viimeiseen vaiheeseen.

Vaihe 6: Viestintä. Viimeisessä vaiheessa kommunikoidaan havaitusta ongelmasta ja sen tärkeydestä, käydään läpi artefaktia ja sen hyödyllisyyttä, uutuutta, suunnittelun tarkkuutta ja tehokkuutta kohdeyleisölle. Tämä vaihe edellyttää tuntemusta viestintäkulttuurista tiedeyhteisöille ja muille asiaankuuluville.

Peffers ym. (2007) esittelevät vaiheet tietyssä järjestyksessä, mutta prosessi ei kuitenkaan etene aina peräkkäisessä järjestyksessä ensimmäisestä viimeiseen, vaan prosessi voidaan aloittaa tapauskohtaisesti aiemmista vaiheista. Skenaariona voi olla tilanne, jossa artefakti on ehditty jo aloittaa ennen varsinaista tutkimuksen aloittamista. Tällöin voidaan aloittaa suoraan vaiheesta 3, eli suunnittelusta ja kehittämisestä.

Suunnittelutiedettä voidaan soveltaa moneen eri tutkimusongelmaan, kuten edellä on esitetty.

Lyhyesti tiivistettynä Peffers ym. (2007) toteavat, että mikä tahansa suunnittelun kohde voidaan nähdä suunnittelutieteen valossa, jossa on upotettu ratkaisu ymmärrettyyn tutkimusongelmaan.

4.2 Tutkimusmenetelmä tutkimuksessa

Kuten alaluvusta 4.1 käy ilmi, voidaan tutkimus aloittaa DSRM-prosessin ensimmäisestä vaiheesta tai mistä tahansa myöhemmästä vaiheesta tapauskohtaisesti. Tässä tutkimuksessa DSRM-prosessi on aloitettu osittain vaiheesta 2, mutta pääpaino on ollut heti alusta alkaen vaiheessa 3, varsinaisessa artefaktin suunnittelussa ja kehittämisessä. Koska tutkimuksemme on jatkotutkimusaihe aiemmasta tutkimuksesta, on ongelma tunnistettu etukäteen ja sille on annettu pieni arvio ratkaisun tavoitteista. Ennen artefaktin aloittamista kävimme keskustelua aiemman tutkimuksen tehneiden henkilöiden kanssa siitä, miksi aihe on merkityksellinen ja mitä sen toivotaan saavuttavan. Artefaktin tavoitteena on helpottaa työmääräarvioita tekevien henkilöiden työtaakkaa sekä osaltaan edistää toteutetun testausstrategian käyttöönottoa, kun kaikille projekteille määritellään selkeä työmäärä testaukseen. Tällöin menetelmän jalkauttaminen on helpompaa, kun asialle on varattu konkreettinen aikaikkuna projektin sisällä.

Artefaktin suunnittelu- ja kehittämisvaiheessa toteutimme työkalun, joka yksinkertaistaa testauksen työmääräarvion laskemista. Suunnittelimme työkalulle vaatimuksia, joita sen tulisi täyttää valmiina toteutuksena. Työkalusta täytyisi tulla mahdollisimman yksinkertainen ja helppokäyttöinen käyttöliittymältään, jotta sen käyttäminen onnistuisi kaikilta heti ensimmäisellä kerralla. Tuloksien tulee olla selkeästi esitettynä ja eriteltynä erillisiksi kokonaisuuksiksi sekä antaa kaikkien osa-alueiden yhteenlaskettu summa, eli kokonaistyömääräarvio. Pohjana käytetään toteutettua testausstrategiaa, joten tärkeänä kriteerinä on noudattaa siinä esitettyjä asioita. Viimeisenä vaatimuksena on saada työkalu helposti kaikkien kohdeorganisaation työntekijöiden saataville. Artefaktin varsinaiseen toteutus- ja suunnitteluprosessiin päästään syvällisemmin luvussa 5.

Artefaktin valmistuttua työkalu esiteltiin kohdeorganisaation henkilöstölle ja se julkaistiin kaikkien käytettäväksi Salesforceen. Organisaatiossa työmääräarvioita tekeviä pyydettiin testaamaan työkalu huomioiden sen käytettävyys ja kysymyksien asettelu. Käyttökokemukset

työkalusta ovat välttämättömiä, jotta päästään seuraavaan vaiheeseen, arviointiin.

Arviointivaiheessa pidimme työkalua käyttäneille henkilöille haastattelukierroksen, josta saimme kommentteja ja mielipiteitä työkalun käytöstä ja selkeydestä. Haastatteluista saadun aineiston perusteella pystyimme tarkastelemaan, onko työkalu onnistunut täyttämään sille asetettuja vaatimuksia, joita vaiheessa 2 määriteltiin. Työkalusta tulleiden palautteiden avulla tehdään muutoksia työmääräarvioiden asetteluun ja lisätään infomodaali työkalun alkuun. Varsinaista iteratiivista prosessia tähän ei oteta mukaan, koska muutokset ovat niin pieniä. Suuremmat muutosehdotukset puolestaan ovat niin suuria, ettei niitä ryhdytä toteuttamaan tämän tutkimuksen puitteissa. Näitä kehitysideoita käydään tarkemmin läpi luvussa 6.

Kun palautteet oli saatu ja pienet korjaukset lisätty työkaluun, siirryimme viimeiseen vaiheeseen, viestintään. Tämä tutkimus toimii jalkauttajana tässä roolissa, eikä suurempaa informointia vaadita.

5 Artefaktin tarve ja toteutus

Tutkimuksen artefaktina kohdeorganisaatiolle toteutettiin testauksen työmäärän arviointityökalu. Tämän luvun alaluvussa 5.1 käydään läpi alkutilanteen selvittämiseksi suoritetun alkuhaastattelun empiirinen aineisto. Alaluvussa 5.2 esitellään artefaktin suunnittelu- ja toteutusprosessit kohdeorganisaation Salesforce-ympäristöön. Alkuhaastattelun viittauksista on anonymisoitu viittaukset henkilöihin ja asiakkuuksiin, sillä ne sisältävät kohdeorganisaation liiketoiminnan osalta salassa pidettävää tietoa.

5.1 Alkukyselyn tulokset

Organisaation alkutilanteen ja tutkielman tulosten vertailukohdan selvittämiseksi, tutkielmassa toteutettiin alkuhaastattelu. Haastateltaviksi pyydettiin henkilöitä, jotka ovat pääasiassa hoitaneet työmääräarviointia kohdeorganisaatiossa. Haastattelu suoritettiin puolistrukturoituna haastatteluna. Haastattelua varten oli laadittu ennalta tutkielman kannalta relevantit kysymykset, mutta haastattelun annettiin edetä vapaamuotoisesti ja tarkentavia kysymyksiä esitettiin haastattelun edetessä. Empiirinen aineisto ja keskeisimmät havainnot esitellään kysymyskohtaisesti.

1. Miten työmääräarviointi on toteutettu kohdeorganisaatiossa ja mitä haasteita siinä on ollut?

> *"Minä olen käyttänyt työurani aikana useita erilaisia työmäärän arviointimalleja. Mikään niistä ei ole toiminut kauhean hyvin. Parhaiten toimii asiantuntija-arviot." H1*

> *"...Toinen meistä tekee alustavan min max arvioinnin projektille. Projekti pilkotaan suurempiin toiminnallisiin kokonaisuuksiin. Arvioinnin jälkeen toinen sitten kommentoi tehtyä arviointia..." H1*

Kohdeorganisaatiossa kaksi kokenutta työntekijää ovat hoitaneet pääasiassa työmäärän arviointia. Työntekijöillä on kokemusta monista työmäärän arviointimenetelmistä, mutta mikään niistä ei heidän mielestään ole toiminut kovin hyvin. Heidän kokemuksensa mukaan parhaiten on toiminut asiantuntija-arviointi. Jompikumpi työntekijöistä on tehnyt ensin työ-

määräarviot, jonka jälkeen toinen kommentoi ja antoi palautetta annetuista arvioista. Tehtyjen arviointien onnistumisia seurataan arviointiprosessin parantamiseksi.

> *"...tehtiin työmääräarvio, niin se on mennyt ihan päin honkia, koska uusi tuote, uudet tekijät, jotka eivät ole tehneet sillä tuotteella töitä. Tuotteesta on löydetty bugeja ja puutteellisuuksia..." H1*

> *"...asiakas ei tiedä mitä se on ostanut. Eli jos myydään kiinteähintaista kamaa. Niin, sitten sinne tulee kaikennäköistä vaatimuksia ikkunoiden ja ovien raoista, kun niitä vaatimuksia ei ole niin kuin ollut olemassa..." H2*

> *"...Sitten on datamigraatiot. Se, että missä muodossa asiakkaan data on ja sitten, kuinka monesta lähteestä se data tulee ja kuinka monta Exceliä me joudumme viemään. Yritäpä antaa asiakkaalle etukäteen tarkka arvio siitä, kuinka paljon teidän datamigraationne maksaa..." H1*

> *"...Asiakkaiden on hirveän vaikea ymmärtää sitä, että integraatioon voi mennä 10 päivään, kun toteuttaa sen Salesforce-integraation johonkin toiseen järjestelmään. Ne ovat vielä vähän työläämpiä ja vaikeampia arvioida erityisesti vähän vanhemmat toiminnanohjausjärjestelmät. Ne ovat ihan hirvittäviä himmeleitä ja niihin kun yritetään integroitua, niin työmääräarvion tekeminen on vaikeata..." H1*

Työmäärän arviointi on heidän kokemuksensa mukaan ollut haasteellista, kun arviointia on tehty projektille, missä käytettiin kohdeorganisaatiolle alustavasti tuntematonta Salesforce-tuotetta. Uudessa tuotteessa saattaa projektin aikana ilmetä erikoisuuksia, bugeja tai puutteellisuuksia, joita ei ole dokumentoitu. Haastaviksi koettiin myös asiakkaiden muuttuvat vaatimukset projektille. Asiakkaille ei ole välttämättä selvää, että muutokset projektin vaatimuksissa kesken projektin muuttavat projektin alkuperäistä työmäärää. Integraatiot kolmannen osapuolen järjestelmiin ja palveluihin aiheuttavat myös haasteita. Työmäärien arviointi integraatiossa koettiin hankalaksi, koska integroitavien järjestelmien dokumentaatio tai tuki integraatioille saattaa olla puutteellista. Datamigraatiot olivat myös haastavia. Arviointia tehdessä ei yleensä tiedetä, missä muodossa ja kuinka laadukasta asiakkaan data on.

Haastateltavien vastausten pohjalta voi todeta, että vaikka haastateltavilla oli vankka koke-

mus työmääräarviointien tekemisestä, niin työmäärän arviointi muuttui haasteelliseksi, kun arvioijalta puuttui kokemusta arvioitavasta projektista tai asiakas ei ollut tarjonnut siitä riittävän kattavia tietoja. Tästä voi havaita, että kattavan ja luotettavan asiantuntija-arvioinnin toteuttamiseksi tarvitaan kokemuksen lisäksi tarpeeksi kattavat alkutiedot.

2. Miten testaus on huomioitu työmäärän arvioinnissa?

> *"...arviontipohjassahan on yksi rivi, mutta se on vain ja ainoastaan tavallaan se Apex-testiluokkien kirjoitus..." H2*

> *"...yleensä se lukee joko tuossa arviossa tai sitten siinä tarjouksessa, että asiakas testaa itse. Jos asiakas haluaa, että testataan yhteisessä testisessiossa, niin niihin tarvitsee sitten varata erillisiä aikoja..." H2*

> *"Se tulee vähän sitten sieltä tavallaan perstuntumalla kuitenkin, mutta sen projektin koon mukaan..." H2*

Haastateltavien mukaan työmäärän arvioinnissa testaus oli huomioitu omana osa-alueenaan. Se sisältää Salesforce-alustan pakottamat testit, mutta ei erityistä testaamiseen allokoitua aikaa. Haastateltavat painottivat, että testauksen päävastuu on kuitenkin asiakkaalla, koska he tuntevat hyvin oman liiketoimintaprosessinsa ja huomaavat virheet näin nopeammin. Jos asiakas erikseen haluaa esimerkiksi yhteisiä testisessioita, niin nämä on huomioitu omana kokonaisuutenaan. Testauksen osuus työmääräarvioista lasketaan vasta sitten, kun kaikki projektin toteutuksen työmääräarviot on laskettu.

3. Miten paikkaansa pitäviä arvioinnit ovat olleet?

> *"...suuri ongelma noissa niin on se, että projektit harvoin toteutuvat sellaisena kuin alun perin on sovittu, vaan ne laajenevat ja muuttuvat niin paljon..." H2*

> *"...mitä isompi projekti, sitä vaikeampaa se arviointi on ja sitä isompia todennäköisesti ne heitot ovat..." H1*

> *"...me haluamme meille asiakkaita, joilla on pitkä elinkaari. Asiakkuuden elinkaari on ratkaiseva tekijä..." H1*

Kohdeorganisaatiossa on seurattu vähän työmääräarvioiden paikkansapitävyyttä. Haastateltavat korostivat, että vertailu arvion ja toteuman välillä on ongelmallista, koska usein projektin tarpeet ja vaatimukset muuttuvat projektin edetessä. Arvioitavan projektin koko vaikuttaa vahvasti siihen, kuinka suurella todennäköisyydellä arvio osuu kohdalleen ja kuinka paljon toteutunut työmäärä eroaa arvioidusta työmäärästä. Voidaan olettaa, että projektin koon kasvaessa kaikkia projektin vaatimuksia ei olla osattu huomioida, jolloin ne ovat myös jääneet arvioimatta tai vaatimusten monimutkaisuutta ei ole osattu arvioida oikein. Haastateltavat painottivat myös sitä, että vaikka alkuperäinen arviointi ei osunut kohdalleen, niin tärkeämpää oli asiakkuuden elinkaari.

5.2 Artefaktin suunnittelu

Tutkimuksemme on jatkotutkimusta kohdeorganisaatiolle aikaisemmin toteutettuun testausstrategiaan, joten tämän testausstrategian pohjalta aloitettiin artefaktin suunnittelu. Työmäärän laskentalogiikkaa suunniteltiin yhdessä testaustrategiaa kehittämässä olleiden kohdeorganisaation työntekijöiden kanssa. Artefaktin tulisi noudattaa testaustrategiassa määriteltyä testaustason valintapolkua. Testaustason valintapolku on kuvion 3 mukainen. Artefaktin tulisi myös hyödyntää testauksen työmäärän laskentalogiikassa testaustasoihin määriteltyjä työmääräarvoja. Laskentalogiikan suunnittelun jälkeen määriteltiin tietomalli historiatiedon tallentamista varten. Työmääräarvioille määriteltiin oma Salesforce-objekti sekä kentät käyttäjän antamien vastausten sekä työkalun laskemien työmääräarvioiden tallentamiseen.

Kohdeorganisaation toiveena oli, että työkalu toteutettaisiin Salesforce-teknologioilla. Työkalun tulisi myös tallentaa tehdyt työmääräarviot Salesforceen historiatiedon seuraamista varten. Työkalun käyttöliittymän pitäisi olla intuitiivinen ja mahdollisimman helppokäyttöinen. Näiden vaatimusten pohjalta aloitettiin määrittelemään artefaktille datamallia.

Käyttöliittymä suunniteltiin toteutettavaksi LWC-komponenttina. Tämä mahdollistaa käyttöliittymän kehittämisen juuri kohdeorganisaation toiveiden mukaiseksi. Myös käyttöliittymän jatkokehittäminen tulevaisuudessa on helpompaa verrattuna muihin Salesforcen käyttöliittymän kehittämisessä käytettäviin teknologioihin. Arviointityökalun käyttöliittymän yksinkertaistamiseksi käyttöliittymä suunniteltiin monivaiheiseksi. Tällä tavalla monimutkainen pro-

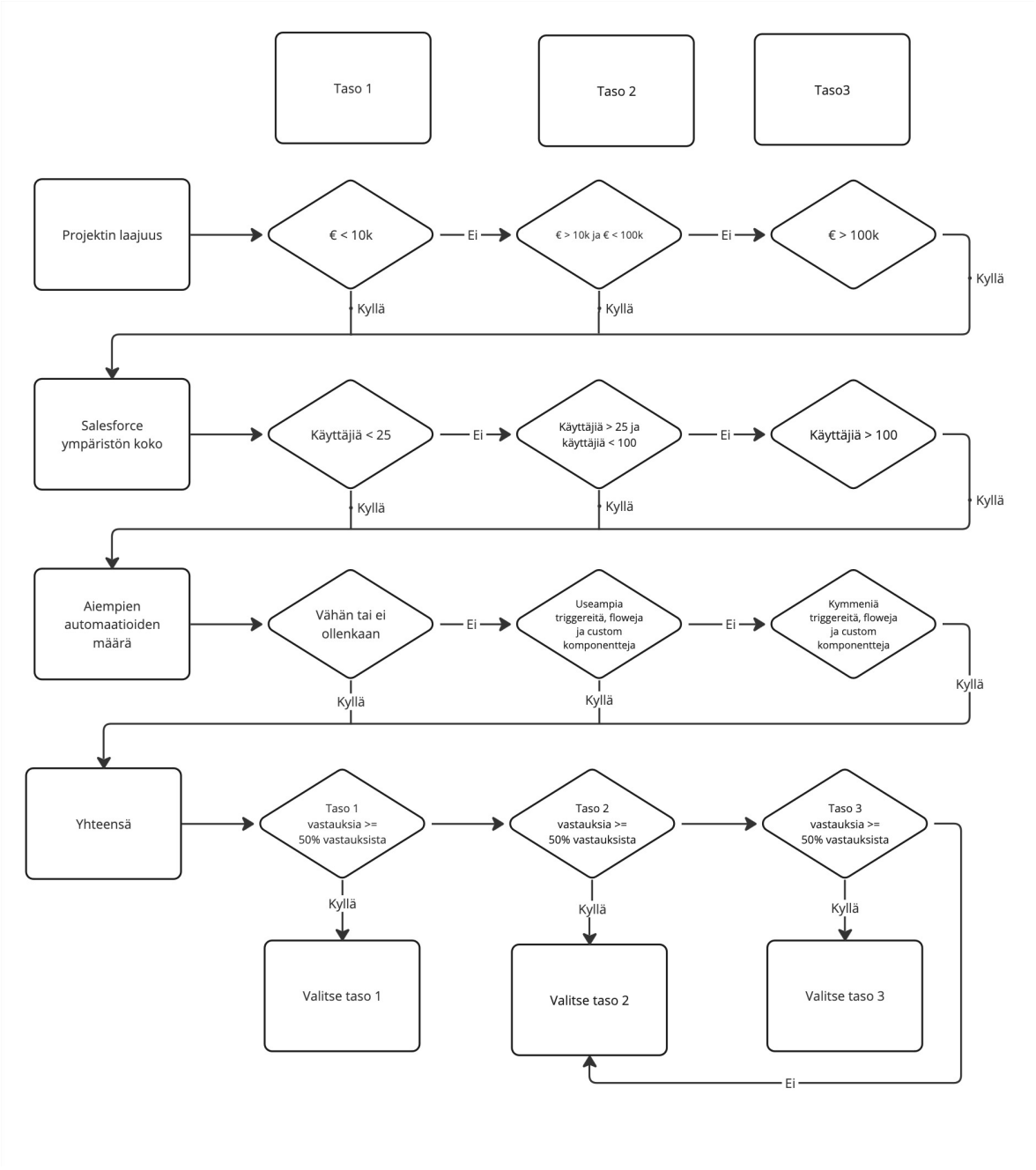
sessi saadaan pilkottua osiin eikä käyttäjän tarvitse miettiä, miten ja missä järjestyksessä prosessin tehtävät tehdään (Tidwell 2010). Seuraavaksi käydään läpi tarkemmin, miten artefakti toteutettiin.

5.3 Artefaktin toteutus

Arviointityökalu toteutettiin tutkielman kohdeorganisaation toiveiden mukaisesti Salesforce-teknologioilla. Kehitys tehtiin kohdeorganisaation määrittämässä kehitysympäristössä. Työkalun valmistuttua työkalun Apex-koodille ohjelmoitiin Apex-testit ja työkalu siirrettiin kohdeorganisaation tuotantoympäristöön sekä relevantteja käyttäjäryhmiä ohjeistettiin käyttämään työkalua.

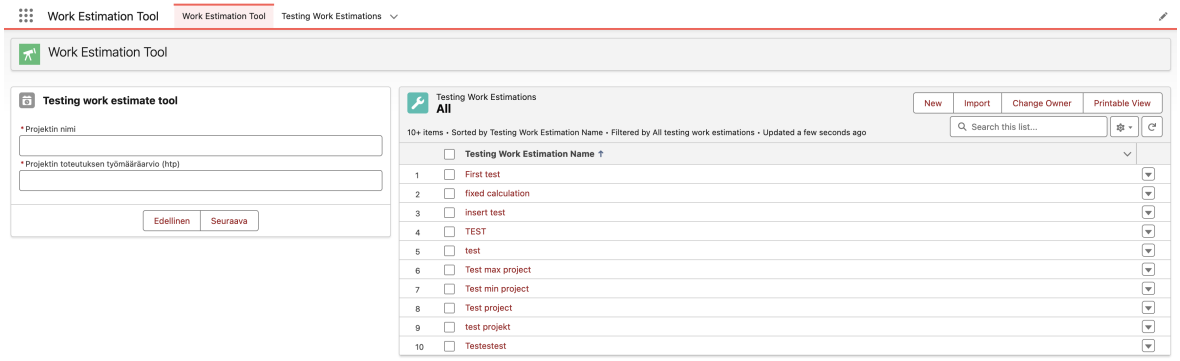
Logiikka työmäärien arviointeja varten toteutettiin Apex-koodilla. Kohdeorganisaation toiveiden mukaisesti historiatiedon tallentamista varten Salesforce-ympäristöön luotiin custom-objekti sekä objektille kentät, joihin tallennetaan käyttäjän antamat vastaukset sekä työkalun antamat tulokset. Työmäärän arviointi perustuu kohdeorganisaatiolle aikaisemmin kehitettyyn testausstrategiaan sekä testaustason valintapolkuun. Ensimmäisenä lasketaan käyttäjän antamien vastausten perusteella projektille testaustaso hyödyntäen testaustason valintapolkua. Testaustasoksi valitaan taso, johon käyttäjän vastauksia on kohdentunut yli 50 %. Jos mihinkään tasoon ei kohdennu yli 50 % vastauksista, niin valitaan taso 2 (Virtanen ja Lehikoinen 2023). Testaustason määrittämisen jälkeen lasketaan testauksen työmääräarvio määritellyn testaustason ja käyttäjän antaman alustavan projektin toteutusvaiheen työmääräarvion perusteella. Jokaiselle projektin vaiheelle annetaan testauksen tuoma lisä riippuen testaustasosta. Tämä lisä voi olla kiinteä yhden henkilötyöpäivän lisäys tai kerrannainen lisäys esimerkiksi puolitoista kertaa toteutusvaiheen työmääräarvio (Virtanen ja Lehikoinen 2023). Lopuksi käyttäjän antamat vastaukset ja niiden pohjalta laskettu työmääräarvio tallennetaan Salesforcen tietokantaan ja laskettu työmääräarvio palautetaan käyttöliittymään käyttäjää varten.

Käyttöliittymää varten työkalulle luotiin kohdeorganisaation Salesforce-ympäristöön oma Lightning-sovellus. Työkalun käyttöliittymä on kuvion 4 mukainen. Käyttöliittymän vasemmalla puolella olevassa listauksessa näytetään käyttäjien tekemiä työmääräarviointeja. Listan



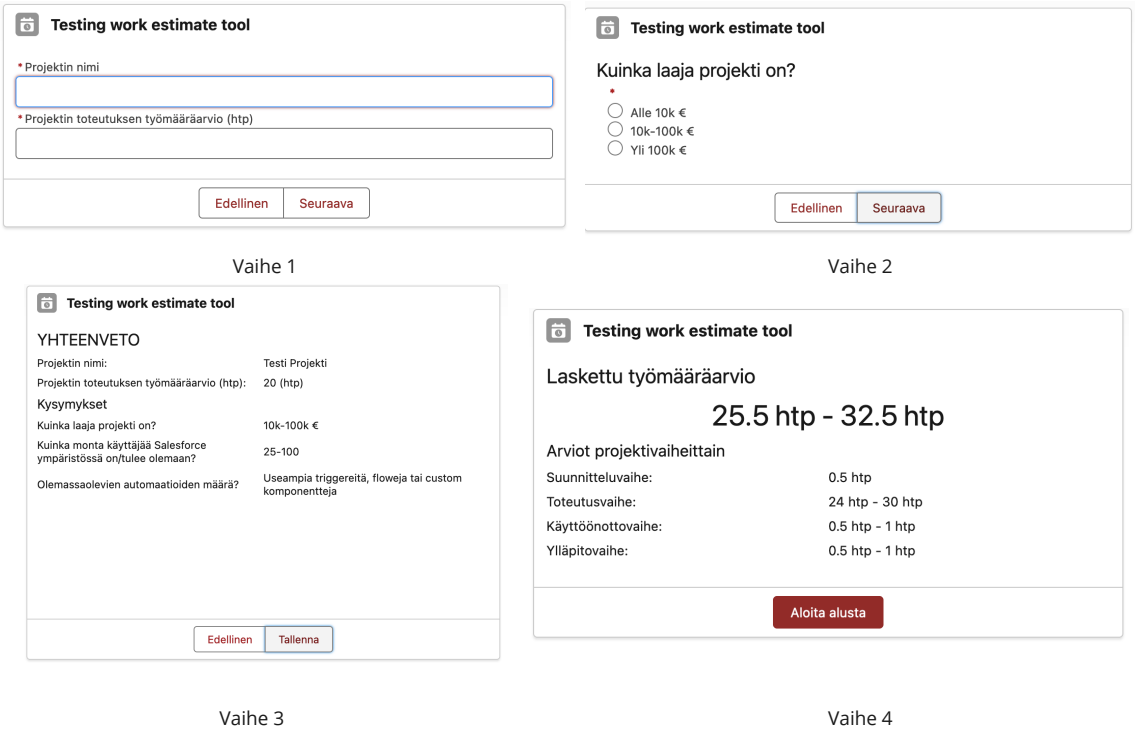
Kuvio 3. Kohdeorganisaatiolle kehitetty testaustason valintapolku (Virtanen ja Lehikoinen 2023)

avulla käyttäjä voi tarkastella muiden tekemiä työmääräarviointeja sekä palata tarvittaessa käyttäjän itsensä tekemiin työmääräarviointeihin. Käyttöliittymän vasemmalla puolella on testauksen työmäärän arviointityökalu, jolla käyttäjä voi laskea projektille testauksen työmääräarvion. Työkalu sisältää neljä vaihetta. Eri vaiheissa näytettävät ruudut ovat nähtävissä kuviossa 5. Työkalu kehitettiin LWC-komponenttina ja se pyrittiin kehittämään siten, että toimiakseen sillä ei ole riippuvuuksia muihin kohdeorganisaation Salesforce-sovelluksiin tai kustomointeihin. Kohdeorganisaatio voi tarvittaessa sijoittaa työkalun haluamalleen sivulle.



Kuvio 4. Työmäärän arviointityökalun käyttöliittymä

Ensimmäisessä vaiheessa käyttäjältä kysytään projektin perustiedot. Näitä ovat projektin nimi sekä projektin toteutusvaiheen alustava työmääräarvio. Työmääräarvio tulee antaa henkilötyöpäivinä. Toisessa vaiheessa käyttäjältä kysytään yksi kerrallaan kohdeorganisaatiolle kehitetyn testaustason valintapolussa määriteltyjä kysymyksiä. Seuraavaksi käyttäjälle näytetään esikatseluvaihe, jossa käyttäjä voi tarkastella antamiaan vastauksia ennen testauksen työmääräarvion laskentaa. Tarvittaessa käyttäjä voi palata takaisin aikaisempiin vaiheisiin muuttamaan vastauksia. Jos käyttäjä on tyytyväinen antamiinsa vastauksiin, hän voi painaa ”Tallenna”-painiketta, jonka jälkeen työkalu laskee annettujen vastausten perusteella testauksen työmääräarvion ja näyttää eritellyn tuloksen käyttäjälle. Halutessaan käyttäjä voi tehdä uuden työmääräarvioinnin painamalla ”Aloita alusta”-painiketta.



Kuvio 5. Työmäärän arviointityökalun eri vaiheissa näytettävät ruudut

6 Artefaktin arviointi

Tässä luvussa käydään läpi artefaktiin liittyvät asiat: artefaktin toimivuus, käyttäjäkokemukset sekä arviointi artefaktin hyödyllisyydestä kohdeorganisaatiolle. Alaluvussa 6.1 käydään läpi datankeruuprosessi ja alaluvussa 6.2 esitellään haastattelun avulla käyttäjiltä saadut tulokset.

6.1 Datan keruu

Haastattelu suoritettiin puolistrukturoituna ryhmähaastatteluna, kuten alkuhaastattelukin. Haastattelussa mukana olleet osallistujat tutustuivat ja käyttivät työkalua ennen haastattelua. Haastateltavat henkilöt olivat eri henkilöitä, kuin alkuhaastattelussa olleet. Työkalu julkaistiin kohdeorganisaation Salesforce-ympäristössä noin kaksi kuukautta etukäteen, joten koko henkilökunnalla oli hyvin aikaa perehtyä siihen. Haastattelun kysymykset liittyvät siihen, millaisia kokemuksia käytöstä on saatu ja ovatko työkalun kysymykset selkeitä. Myös työkalun hyödyllisyydestä on tärkeää saada tietoa.

Kohdeorganisaatio on melko pieni kasvuyritys, minkä vuoksi myös haastateltavien ryhmä jäi pieneksi. Ryhmässä oli neljä haastateltavaa, jotka olivat kuitenkin kohderyhmänä sellaisia henkilöitä, jotka tulevat jatkossa hyödyntämään työkalua. Venable, Pries-Heje ja Baskerville (2016) mukaan ryhmän kooksi suositellaan 6–8 henkilöä, joten tähän suositeltuun ryhmän kokoon nähden otantamme on pieni. Kontekstiin nähden oleellinen informaatio saatiin kuitenkin haastattelussa irti pienemmältäkin ryhmältä.

6.2 Tulokset

Artefaktina toteutetun työkalun toimivuutta arvioidaan haastattelemalla käyttäjiä heidän käyttökokemuksistaan. Tässä alaluvussa paneudutaan haastattelun tuloksien läpikäyntiin sekä arvioidaan yleisesti artefaktin toimivuutta ja onnistuneisuutta sille osoitettuun käyttötarkoitukseen.

Haastattelun ensimmäinen kysymys liittyi projektin nykyiseen tilanteeseen. Miten testauk-

seen annettu työmääräarvio näkyy myytyjen projektien työmääräarvioissa ja toteutuksissa. Alkuhaastattelun perusteella olimme saaneet käsityksen siitä, miten työmääräarvioita toteutetaan tällä hetkellä kohdeorganisaatiossa, mutta halusimme näkemyksen myös jälkimmäiseltä ryhmältä.

Henkilöiden vastauksissa huomaa pieniä eroja. Tämä voi johtua siitä, että kohdeorganisaatiossa työntekijöille on epäselvää tai heillä ei ole riittävästi tietoa siitä, miten työmääräarvioita tehdään tai miten ne näkyvät projektien arvioissa. Alkuhaastattelusta saadun tiedon perusteella testauksen työmääräarviot ovat usein tietyn suuruinen osuus kokonaisprojektin työmääräarviosta, mutta loppuhaastattelussa mukana olleet eivät havaitse tätä osuutta helposti. Henkilöstä riippuen osuus voi näkyä vain pienenä työmääräarviona, pakollisena lisänä projektiin tai sitä ei hahmota ollenkaan työmääräarviossa. Yleisesti vastaukset olivat kuitenkin samankaltaisia kuin alkuhaastattelussa ja työmäärän arvioinnin haastavuus tuli ilmi myös tässä keskustelussa työmääräarvioiden antamiseen.

1. Miten testaus näkyy tällä hetkellä projekteissa työmääräarvioissa ja toteutuksessa?

> *"Yleensä testaukselle annettua työmääräarviota ei ole eritelty suuremmin omaksi kokonaisuudekseen, vaan se on osana projektin koko työmääräarviota." H1*
>
> *"Jos testaukseen on erillinen työmääräarvio, on se usein pieni ns. pakollinen lisä, esimerkiksi 1 henkilötyöpäivä." H2*
>
> *"Testauksen työmääräarviointi saattaa olla hankalaa. Ei tiedetä, mikä on riittävästi." H4*

Haastatteluissa keskeistä oli selvittää työkalun käytettävyyttä ja ymmärrettävyyttä heti ensimmäisten kokeilukertojen jälkeen. Muutama kysymys liittyi yleisesti työkalun käyttöön ja sen ymmärrettävyyteen. Lisäksi kysyimme, minkälaiseksi haastateltavat kokivat prosessin läpikäynnin.

Yleisellä tasolla työkalun käyttö miellettiin yksinkertaiseksi, helppokäyttöiseksi ja nopeaksi. Testauksen työmääräarvion tekeminen ei vaatinut kovin paljon aikaa arvion saamiseksi, mikä on hyvä asia käyttöönoton kannalta. Haastateltavat olivat asiasta samaa mieltä, joten voimme tämän perusteella todeta työkalun olevan käyttöliittymältään selkeä. Ilman haasteita

ei kuitenkaan selvitty, sillä projektin alkuvaiheessa ei ole aivan selvää, kuinka paljon projektin aikana tullaan tekemään esimerkiksi erilaisia automaatioita järjestelmään. Täten tässä vaiheessa kaikkiin kysymyksiin ei ole vielä selvää vastausta tiedossa. Joihinkin kysymyksiin on annettava siten parhaaksi katsottu arvio, mikä voi vaikuttaa arvioiden oikeellisuuteen.

Haastateltavat ehdottivat, että työkalun antamat tulokset esitetään sellaisessa muodossa, että työkalun antamia arvioita olisi mahdollisimman helppo tulkita. Toteutuksemme perustuu täysin aikaisemmin toteutettuun testausstrategiapohjaan, johon perustuen kaikki arviot on toteutettu. Haastattelun aikana ilmeni, etteivät haastateltavat olleet tietoisia testaustrategiasta, mikä voi osaltaan vaikuttaa siihen, että haastateltavien mielestä arviot eivät vaikuttaneet selkeiltä. Testausstrategia on kuitenkin julkaistu kohdeorganisaatiossa ja on kaikkien työntekijöiden saatavilla, joten tiedottaminen siitä ei ole täysin onnistunut. Epäselvyyttä voi aiheuttaa myös muotoilu, jolla arviot ilmaistaan työkalussa. Työkalun ensimmäisenä kysymyksenä kysytään koko projektin toteutusvaiheen arvioitua työmäärää, johon testausta ei ole laskettu mukaan. Lopuksi työkalu antaa vastauksena työmääräarvion, johon on sisällytetty projektin työmääräarvio sekä lisätty työkalun laskema testauksen työmääräarviointi. Tästä voi päätellä harhaanjohtavasti, että kyseessä on testauksen työmääräarvio. Tämä konteksti täytyy nostaa selkeästi esille. Myös testauksen työmääräarvio on hyvä nostaa erilliseksi kohdaksi, jotta ainakin tämän ongelman aiheuttamat epäselvyydet saadaan korjattua.

2. Minkälaiseksi koit prosessin läpikäynnin?

> *"Prosessi itsessään oli yksinkertainen, helppo ja nopea." H1, H2, H3, H4*
>
> *"Jos arvioinnin teki projektin alkupäässä, ei välttämättä tiennyt vastausta kaikkiin kysymyksiin." H1*
>
> *"Työkalun antamia vastauksia voisi selkeyttää, jotta ne olisivat helpommin tulkittavissa; tarkemmat erittelyt tuloksille sekä selitykset mitä mikäkin arvo tarkoittaa." H4*

Kuten kysymyksestä 2 käy ilmi, työkalun käyttäminen koettiin yksinkertaiseksi ja helpoksi. Saimme myös kehitysideoita jo tämän kysymyksen kohdalla. Halusimme vielä saada enemmän ehdotuksia työkalun parantamiseksi, joten kysyimme haastateltavilta kehitysideoita. Prosessin alkuun ehdotettiin listausta tulevista kysymyksistä, jotta kaikki tarvittava tieto

olisi esillä heti alussa. Kun kaikki tieto on valmiiksi esillä tai muistissa, niin prosessin suoraviivainen läpikäynti on helpompaa, eikä siihen tule keskeytyksiä. Tämä on hyvä kehitysehdotus, mitä ei kehittämisvaiheessa osattu ottaa huomioon. Muutos on nopea ja yksinkertainen toteuttaa, joten tämä saadaan lisättyä helposti käyttäjien tueksi.

Kysymyksen 2 vastauksista ilmeni, että projektin alussa on vähän tietoa projektin etenemistavasta. Haastateltavat toivoivat mahdollisuutta luoda useampia työmääräarvioita erilaisilla skenaarioilla. Tällöin olisi mahdollista tarkastella erisuuruisia arvioita ja muodostaa niiden pohjalta jonkinlaista keskiarvoa. Tällä hetkellä usean arvion tekeminen onnistuu, mutta uudet arviot eivät ole linkittyneitä keskenään. Oletettavasti tällainen linkitys olisi mahdollista toteuttaa jollain tavalla. Näin erillisiä arvioita ei tarvitse etsiä kaikkien arvioiden joukosta, vaan ne olisivat helposti löydettävissä yhdestä paikasta. Yleisesti ajatus on hieman ristiriidassa toteutetun testausstrategian kanssa, jos prosessia muokataan käyttämään monen arvion keskiarvoa tai jos monesta arviosta pitäisi valita yksi, joka oikeasti otetaan käyttöön projektissa. Tällä tavalla työkalun arvioiden oikeellisuutta on vaikea analysoida ja poikkeamia voi syntyä molempiin suuntiin. Toki on ymmärrettävää, että arvioiden toteuttaminen on haasteellista projektin alkuvaiheessa.

3. Tuliko prosessin aikana kehitysideoita työkalun käytettävyyden parantamiseksi?

> *"Työkalussa olisi hyvä eritellä tarkemmin työkalun antamat tulokset ja avata mitä ne tarkoittavat." H4*
>
> *"Ennen prosessin aloittamista olisi hyvä olla listaus kysymyksistä, jotta tiedetään mitä tietoa tarvitsee arvioinnin suorittamiseksi." H1*
>
> *"Työkalu voisi tukea eri skenaarioiden tekemisistä samalle projektille." H1*

Ensimmäisessä kysymyksessä viitattiin tämänhetkiseen testauksen tilanteeseen ja näkyvyyteen projekteissa. Artefaktin hyödyllisyys ja tärkeys ovat tässä tilanteessa oleellista tietoa, joten halusimme alustavia mielipiteitä työkalun mahdollistamasta muutoksesta testauksen toteutumiseen projekteissa. Haastateltavien keskuudessa työkalu koettiin hyödylliseksi. Työkalun antamat arviot koettiin tekijänä, joka voi parantaa projektien konkretiaa, kun testaukseen on selkeästi allokoitu tietty aika. Myös projektien testauksen laatua voidaan parantaa

tämän myötä.

Koska haastattelu oli puolistrukturoitu, haastateltavilta kysyttiin myös motivaatiota suorittaa testausta. Haastateltavat totesivat, että työmäärän laajuus ei saisi vaikuttaa motivaatioon, vaan motivaatio tulisi löytyä muualta. Hyvä motivaatio on lähtökohta, jotta asiakkaille voidaan tarjota parhaita mahdollisia ratkaisuja. Kuten kysymyksen 4 vastauksesta käy ilmi, työmäärä ilmaisee, miten paljon projektiryhmälle on varattu aikaa testaukseen, jotta testaus toteutetaan tarpeeksi kattavasti ja suunnitellusti. Tästä näkökulmasta työkalusta on selkeää hyötyä projektien testauksen parantamisessa. Usein projektiaikataulut ovat syynä testauksen puuttumiselle tai vähäisyydelle, joten selkeä testaamiseen allokoitu aika parantaisi asiaa huomattavasti.

4. Miten työkalu voisi helpottaa testauksen toteutumisen kanssa?

> *"Työkalun voisi lisätä projektiryhmän tietoisuuttaa testauksesta ja sen tärkeydestä. Jos projektille on arvioitu tietty aika testaamiselle, niin se voisi motivoida projektiryhmää allokoimaan enemmän aikaa testaamiseen. Nyt testaus saatetaan tehdä nopeasti kehityksen ohella, mutta ei välttämättä tarpeeksi kattavasti." H2*

Salesforce-ympäristössä on paljon erilaisia tuotteita, joista kaikkia ei ole päästy käyttämään kohdeorganisaation projekteissa. Välillä tulee tilanteita, jolloin projektiin täytyy sisällyttää uusi tuote, joka vaatii huomattavasti enemmän aikaa perehtymiseen ja testaamiseen. Tämän vuoksi tiedustelimme näkemystä työkalun käytöstä osana projektien työmääräarviointia, joissa on mukana tuntematon tuote. Työkalun käyttö koettiin tässä kohtaa jonkin verran hyödylliseksi, sillä se voisi antaa suuntaa antavan arvion tai vähintään minimiarvion projektin vaatimalle testaukselle. Näiden perusteella tiedetään varata testaamiselle riittävästi aikaa. Uusien tuotteiden haasteet ilmenevät usein vasta projektin toteutusvaiheessa tuotteiden ominaisuuksien ja ongelmien konkretisoituessa käytössä. Ilmenneitä haasteita täytyy selvitellä projektin aikana. Näiden haasteiden takia työkaluun ehdotettiin valintaruutua uusia tuotteita varten. Tavoitteena on huomioida uuden tuotteen vaatima lisäaika testausarviossa.

5. Voisiko työkalun antamat arviot auttaa tuntemattomien tuotteiden projektien kanssa?

> *"Työkalu voi antaa pientä apua työmääräarvioiden tekemistä tuntemattomien tuottei-*

den kanssa projektia tehdessä. Haasteet toteutuksessa ilmenevät vasta kun projektia on alettu toteuttamaan ja erot arvioidun ja toteutuneen työmääräarvion välillä ovat yleensä toteutusvaiheessa." H1, H2

"Työkalu voisi antaa minimi tason testauksen työmääräarvioille." H2

Viimeisessä kysymyksessä suuntasimme katsetta jo tulevaisuuteen ja mahdollisiin työkalun lisäversioihin. Toteuttamamme työkalu keskittyy toistaiseksi vain testauksen työmääräarvioihin. Halusimme tietää, olisiko yleinen työkalu projektien kokonaistyömäärän arvioinnissa hyödyllinen. Saimme vastaukseksi hyvän huomion siitä, että työkalua voisi laajentaa projektien eri osa-alueisiin. Koko projektia arvioivaa työkalua pidettiin myös hyvänä ja hyödyllisenä ideana. Yksi haastateltava toivoi työkalua, joka hyödyntäisi tehtyjen projektien arvioiden ja toteutumien tietoja. Työkalu oppisi arvioimaan vastaavia projekteja näiden mukaisesti, jolloin olisi mahdollisuus tehdä parhaita arvioita.

6. Jos työkalusta tulisi lisäversio yleisesti projektien työmääräarvioiden tekemiseksi, näkisitkö tällaiselle käyttöä tai toisiko se lisäarvoa työskentelyyn?

"Työkalu voisi laajentua kattamaan muita projektin osa-alueita tai koko projektin arviointia." H1, H2, H3, H4

"Tulevaisuudessa työkalu voisi olla “oppiva” ja se osaisi ottaa huomioon edellisten projektien toteuman ja tehdyt arviot... Työkalu näyttäisi esimerkiksi kuvaajassa edellisten samankaltaisten projektin toteumat ja arviot, joiden pohjalta voi määritellä tulevan projektin arviota." H2

Projekteissa esiintyy välillä tilanteita, joissa joudutaan miettimään projektin sisältöön alun perin kuulumattomille asioille uusia työmääräarvioita. Yleisenä kommenttina haastattelussa esitettiin, että työkalu voisi helpottaa uusia työntekijöitä tilanteissa, joissa heillä ei ole vielä laajempaa kokemusta Salesforce-projekteista. Tällöin työmääräarvioiden antaminen on haasteellista. Tämä kommentti liittyy sekä jo toteutettuun työkaluun että mahdolliseen laajennettuun kokonaistyömääräarviota tuottavaan versioon.

Haastattelun pohjalta voidaan todeta, että työmäärän arviointi nähdään monimutkaisena prosessina, jonka yksinkertaistamisesta ja helpottamisesta työkalulla voi olla organisaatiolle

paljon hyötyä. Historiatiedon hyödyntämistä työmäärän arvioinnissa pidetään hyödyllisenä. Historiatiedon liiallinen painotus voi olla ongelmallista. Organisaation kasvaessa ja kehittyessä aikaisemmat arviot eivät välttämättä pidä paikkaansa, koska organisaatiolla saattaa olla esimerkiksi paremmin resursseja käytettävissä. Myös analogioiden löytäminen arvioitavalle projektille voi olla haastavaa (Keung 2009). Työkalun tulisi ottaa huomioon organisaation erityispiirteitä, pystyä mukautumaan organisaation kasvaessa sekä kehityskäytäntöjen muuttuessa ja kehittyessä.

7 Pohdinta

Tutkimusprosessi on kulkenut aaltoilevasti, kuten varmasti monet tämänkaltaiset prosessit usein menevät. Alaluvussa 7.1 pohdimme artefaktin hyötyjä kohdeorganisaatiolle ja alaluvussa 7.2 pohdimme, miten hyvin vastasimme tutkimuskysymyksiin. Alaluvussa 7.3 arvioimme tutkielman onnistumisia ja puutteita sekä vertaamme toteutettua artefaktia kirjallisuudessa esitettyihin työmäärän arviointimenetelmiin. Viimeisessä alaluvussa 7.4 pohdimme tutkimuksen luotettavuutta ja ehdotamme jatkotutkimusaiheita.

7.1 Artefaktin hyödyt kohdeorganisaatiolle

Kuten aiemmin tutkimuksessa on jo mainittu, toteutettua testausstrategiaa ei ole laajemmin jalkautettu käyttöön kohdeorganisaatiossa. Tekemämme artefaktin avulla pyrimme helpottamaan ja nopeuttamaan projektien testaukseen vaadittavien työmääräarvioiden määrittämistä. Tämä vaikuttaa projekteissa käytettävään testausaikaan, jolla varmistetaan tuotettujen ominaisuuksien toimivuus projekteissa ennen varsinaista tuotantokäyttöä. Testausstrategian varsinainen käyttöönotto tapahtuu myös helpommin, kun projekteissa on selkeä aika varattuna testaamiseen. Tarkoituksena on saada testausstrategiasta automaattisesti huomioitava asia projekteissa, jotta täysi hyöty saadaan irti toteutetusta työstä.

Alunperin tutkimuksessa oli tarkoituksena selvittää toteutetun artefaktin hyödyllisyyttä sekä myös artefaktin antamien arvojen oikeellisuutta uusien projektien parissa. Aikataulullisista syistä jouduimme kuitenkin jättämään arvojen tarkastelun pois tästä tutkimuksesta, koska projekteista ei ehditty saada ajoissa riittävästi dataa. Myös vanhojen projektien vertaaminen on haastavaa, koska vanhoissa projekteissa tuntimerkintöjen kohdentumisesta testaukseen ei ole varmuutta. Emme halua vaarantaa artefaktin oikeellisuuden arviointia virheellisen datan johdosta. Tätä on hyvä selvittää tulevaisuudessa. Tutkimuksessamme painotimme työkalun selkeyttä ja sen hyödyllisyyttä tulevaisuuden projekteissa.

Lukuihin 5 ja 6 viitaten kohdeorganisaation nykyistä työmäärän arvioinnin toteutustapaa voisi yhtenäistää. Erityisesti testauksen työmääräarvio vaatii vielä lähempää tarkastelua, sillä testausta ei ole välttämättä huomioitu omana kohtanaan vaan se on osa kokonaisarviointia.

Tällöin työmäärä voi kohdentua suurelta osin projektin toteutusvaiheeseen ja järjestelmän testaaminen jäädä vajavaiseksi. Toteuttamamme artefakti yhtenäistää testauksen työmääräarvioiden tekemisen ja antaa helpon käyttöliittymän niiden toteuttamiseen jo ensimmäisillä käyttökerroilla. Tämä vahvistaa kohdeorganisaation kykyä tehdä työmääräarvioita sekä yhtenäistää arvioita riippumatta henkilöstä, joka työmääräarvion toteuttaa.

Luvun 6 loppuhaastattelusta saatujen kommenttien perusteella voidaan todeta, että työkalulle on käyttöä tulevaisuudessa ja se koetaan hyödyllisenä. Myös käyttökokemukset olivat kaikin puolin myönteisiä, pieniä korjauksia lukuun ottamatta, jotka on helppo toteuttaa.

7.2 Tutkimuskysymyksiin vastaaminen

Tutkimuksemme lähtökohtana ja tarkoituksena oli toteuttaa työkalu, joka helpotta projektien testauksen työmääräarvioiden tekemistä sekä samalla jalkauttaa aiemmin toteutetun testausstrategian käyttöönottoa osaksi projekteja. Näiden lähtökohtien perusteella mietimme tutkimuskysymyksiä, joiden avulla saisimme mahdollisimman paljon ideoita ja ehdotuksia toimivan työkalun kehittämiseksi.

Ensimmäinen tutkimuskysymyksemme oli:
Miten Salesforce-projektin testauksen työmäärän arviointia voi tukea työkalulla?

Tutkimuksen lähtökohtana oli luoda artefakti, eli tässä kontekstissa työkalu, joka laskee Salesforce-projektille testauksen työmääräarvion koko projektin toteutusvaiheen työmääräarvion sekä neljän lisäkysymyksen perusteella. Tavoite oli helpottaa kohdeorganisaatiossa työmääräarvioita tekevien henkilöiden työtä verrattuna aikaisemmin käytössä olleeseen testausstrategian arviointimenetelmään.

Työkalulle kehitettiin monia ominaisuuksia, jotka tukevat testauksen työmäärän arviointia verrattuna siihen, että arvioija käy lukemassa ja tarkastelemassa testausstrategiadokumenttia sekä pohtii asioita mielessään. Vielä haastavampaa olisi tehdä arvioita omien näkemysten ja kokemuksen pohjalta. Tämä oli kohdeorganisaation tilanne ennen testausstrategiadokumentin luomista. Verrattuna dokumenttiin työkalusta on paljon hyötyä arvioijalle. Sen käyttö on yksinkertaista ja se helpottaa arvioiden yleistä toteutusta.

Kuten loppuhaastattelusta ilmeni, työkalun kysymyksiin vastaaminen ja vastauksien saaminen oli nopeaa ja suoraviivaista. Alkuperäistä toteutustapaa eli dokumenttia käyttämällä työaikaa olisi kulunut huomattavasti enemmän. Dokumenttia käytettäessä arvioijan pitäisi pohtia vastauksista ensimmäisenä tasoa ja tämän jälkeen selata dokumentin osioita läpi, jotta hän saisi kaikki kertoimet ja työmäärien lisäykset omaan arvioonsa. Loppuhaastattelun vastauksista selviää, että työkalun antamia vastauksia on selkeytettävä siten, että kaikille käyttäjille on heti selvää, mitä arviot tarkoittavat. Vastaukset on kategorisoitu ryhmittäin ja kokonaisarvio tulee erikseen yhtenä kenttänä. Kenttien nimet ja niiden ohjetekstit pitää vielä laatia selkeiksi. Työkalun käytön pitää olla mahdollisimman sujuvaa ja suoraviivaista, jotta arvioijat ovat motivoituneita käyttämään sitä myös jatkossa omien arvioiden tekemisessä. Käyttäjän vaatimukset työkalun käytettävyydelle riippuu siitä, kuinka usein hän käyttää työkalua. Yksinkertaisella käyttöliittymällä saadaan motivoitua käyttäjiä käyttämään työkalua myös tulevaisuudessa, ensimmäisien kokeilukertojen jälkeen. (Tidwell 2010)

Molemmista haastatteluista kävi ilmi, että työmääräarvioiden tekeminen koetaan yleensä haasteelliseksi, oli sitten kyseessä koko projektin toteutus tai vain yksi osa-alue, kuten testaus. Työkalun tuoma toimintalogiikka ohjaa arvioijia työmääräarvioiden määrittämisessä. Tutkimuksen aikana ei saatu kerättyä tarpeeksi dataa projektien toteutuneista työmääristä. Tämän takia ei ollut mahdollista toteuttaa työkalun arvioiden paikkaansa pitävyyttä tutkimuksessamme. Kun projekteista saatua tietoa toteutuneista työmääräarvioineista kertyy, selviää työkalun mahdollinen potentiaali paremmin.

Lyhyenä yhteenvetona kysymykseen. Työkalu tuo helppokäyttöisen ja nopean tavan toteuttaa testauksen työmääräarvioita projekteille. Työkaluun kaivataan joitakin muutoksia selkeyttämään annettuja vastauksia sekä ilmoittamaan alkuvaiheessa kaikki esitettävät kysymykset, jotta vastaaja osaa ennakoida kaiken tarvittavan materiaalin saataville ennen vastaamista. Tämän perusteella voimme todeta, että työkalusta on hyötyä ja se helpottaa käyttäjien toimintaa testauksen työmääräarvioiden tekemisessä.

Toinen tutkimuskysymyksemme oli:
Millä tavoin työkalu yksinkertaistaa työmäärän arviointia?

Tähän kysymykseen löytyi tutkimuksen edetessä monia vastauksia. Osittain vastaukset ovat

testausstrategian ansiota. Vastaukset liittyvät myös niin paljon työkaluun, että voidaan olettaa niiden vastaavan kysymykseen tässä tutkimuksessa. Yksi työmäärän arviointia yksinkertaistava ominaisuus on, että työkalu antaa valmiit kysymykset työmääräarviota tekevälle henkilölle. Tämä helpottaa arvioijan työtä, kun hän tietää valmiiksi, minkälaisiin kysymyksiin hänen täytyy miettiä vastauksia. Kun prosessista poistetaan jokaisen arvioijan omat näkemykset ja ajatukset, saadaan yhtenäistettyä arvioiden toteutusprosessia sekä arvioiden suuruutta. Yhtenäiset toimintatavat organisaation sisällä helpottavat asiakkaiden tasapuolista kohtelua, kun arviointia tehdään yhtenäisen prosessin mukaisesti.

Testausstrategian dokumentti on ollut Confluence-dokumenttina ja se on ollut kaikkien kohdeorganisaation työntekijöiden saatavilla. Työkalu on toteutettu kohdeorganisaation Salesforce-ympäristöön, jossa se on myös kaikkien organisaation työntekijöiden saatavilla. Saavutettavuus edistää työkalun käyttöönottoa, kun on tiedossa mistä se on saatavissa.

Loppuhaastattelussa nousi esiin miten uudet työntekijät tekevät työmäärän arviointia. Projektien aikana on yleistä, että uusia toteutuksia halutaan aiemman projektitoteutuksen lisäksi. Tällöin uusille työntekijöille työmääräarvioiden tekeminen on todella haastavaa. Työkalu yksinkertaistaa prosessia testauksen työmäärän arviosta ja tämä helpottaa arvioiden tekemistä.

Työkalun yksi hyödyllisin ominaisuus verrattuna testausstrategian mallipohjaan on valmis laskentalogiikka työkalun taustalla. Näin työntekijöiden ei tarvitse itse laskea ja tehdä laskutoimituksia arvojen ja kertoimien kanssa. Työkalu hoitaa nämä kysymyksien ja vastauksien pohjalta. Tällä saadaan helpotettua työmäärän arviointia tekevien työtä sekä nopeutettua prosessin läpikäyntiä huomattavasti.

Loppuhaastattelussa tuli vielä esiin yksi tärkeä asia. Miten arvioida testauksen vaatima työmäärä, kun kyseessä on tuote, josta kohdeorganisaatiolla ei ole aikaisempaa kokemusta. Aiemmin tällaisten projektien kohdalla on tehty työmääräarvio kehitystyölle ja testaukselle työmääräarvioinnista saadun yleisen kokemuksen pohjalta. Jatkossa kehitetyn työkalun avulla voidaan arvioida suuntaa antava minimiarvo testauksen työmäärälle arvioidun toteutuksen työmäärän avulla.

Näiden perusteluiden pohjalta voimme todeta vastanneemme myös tähän tutkimuskysymyk-

seen. Testauksen työmäärän arviointi on yksinkertaisempaa ja suoraviivaisempaa työkalun avulla perustelluista syistä ja samalla se yhtenäistää kohdeorganisaation henkilöiden toimintaprosessia.

7.3 Tutkimuksen arviointi

Kuten tutkielman aiemmissa luvuissa on jo tullut esiin, tarkoituksena on ollut luoda yhtenäinen ja yleisesti testauksen työmäärien arviointia helpottava työkalu testausstrategian pohjaa mukaillen. Mielestämme työkalu hoitaa tämän asian hyvin. Työkalun avulla työntekijät tekevät työmäärien arviointia työkalun ohjaaman prosessin mukaisesti, jolloin arvioista tulee yhtenäisempiä. Myönteistä oli myös kuulla loppuhaastattelussa, että työkalua pidettiin potentiaalisena ja helpottavana lisänä prosessissa. Arvioiden yhtenäistäminen ei kuitenkaan ollut ainoa päämäärä työkalua kehitettäessä vaan myös työntekijöiden työn helpottaminen. Tuntui, että tämä toteutui.

Kun pohditaan tarkemmin tutkielman onnistumisia, on nostettava esiin artefaktin onnistuminen yleisellä tasolla. Suunnitteluvaiheessa saimme paljon hyvää pohjatietoa testausstrategian tekijöiltä. Tämä helpotti omaa suunnitteluamme. Heti alusta alkaen kehittämistyö onnistui hyvin ja valmis työkalu täytti kaikki sille etukäteen asetetut vaatimukset. Vaatimuksia olivat yksinkertainen ja selkeä käyttöliittymä, testausstrategian mukainen toimintalogiikka, eritellyt työmääräarviot osa-alueittain ja selkeä kokonaisarvio sekä kaikkien saatavilla oleva työkalu. Nämä kriteerit pääosin toteutuivat tavoitteiden mukaisesti. Ainoastaan vastauksien selkeyteen annettiin loppuhaastattelussa kehitysehdotuksia.

Yllä mainittujen onnistumisten taustalla vaikuttaa vahvasti toteutetun testausstrategian selkeä toteutus, minkä pohjalta työkalu on ollut helppo toteuttaa. Yleisesti artefaktin toteutusvaihe oli suoraviivaista, kun testausstrategiasta sai valmiit arvot ja suuntaviivat, joiden pohjalta työkalu oli mahdollista kehittää.

Tutkimukselle ja artefaktille jää tämän tutkielman jälkeen vielä useita kulmia ja aukkoja, joiden avulla tutkimukseen olisi saanut lisää syvyyttä. Suurimpana asiana työkalun antamien työmääräarvioiden paikkansapitävyys projekteissa. Nyt työkalun antamille arvioille ei ole mitään tietoa, kuinka oikeellista tai edes lähelle meneviä arvioita se tuottaa. Tämä on yksi

merkittävimmistä asioista, joka vaikuttaa työkalun hyödyllisyyteen ja käytettävyyteen jatkossa. Tämä täytyy selvittää pikimmiten. Kun saadaan tarpeeksi monen projetin arvioista tuloksia, niin on helpompaa miettiä, mihin suuntaan arvioita tulisi muuttaa, jotta niistä tulisi parempia.

Työkalumme pohjautuu täysin testausstrategiaan ja siinä määriteltyyn taustalogiikkaan, joten työkalu ei varsinaisesti käytä luvussa 2 esiteltyjä malleja. Kehitetylle työkaluratkaisulle voidaan osittain soveltaa asiantuntija-arviointia ja testipisteanalyysia, mutta varsinaista tieteellistä pohjaa arvioiden antamiseen ei saada. Asiantuntija-arvion näkökulmasta työkalun tuottamien arvioiden voidaan olettaa pohjautuvan tuetusti asiantuntija-arviointiin, sillä kohdeorganisaation henkilöt ovat toteuttaneet kertoimia ja määriä testausstrategiaan, joiden avulla arviot lasketaan. Toisaalta työkalu myös laajentaa perinteistä asiantuntija-arviointia. Kuten Molokken ja Jorgensen (2003) tutkimuksessaan toteavat, hyödynnetään asiantuntija-arvioinnissa yhden tai useamman henkilön kokemusta projekteista sekä tietämystä toimialasta ja organisaatiosta. Työkalun tapauksessa nämä on sisällytetty osaksi työkalun toimintalogiikkaa. Testipisteanalyysin osalta työkalun toiminnallisuus toimii osittain sovelletusti ja hieman karsitulla mallilla. Kuten Veenendaal ja Dekkers (1999) tutkimuksessa ja tutkielmamme luvussa 2 kerrotaan, on kolme asiaa, joista tarvitaan tietoa ennen testipisteanalyysin käyttämistä. Ne ovat testattavan ohjelmiston koko, testausstrategia sekä tuottavuuden taso. Työkalun tapauksessa testattavan ohjelmiston koko on korvattu koko projektin toteutuksen työmääräarviolla. Tämä on ensimmäinen työkalussa kysytty tieto. Testausstrategiaa hyödynnetään arvioiden teossa, jonka pohjalta työkalun antama taso ja arvion suuruus määräytyvät. Tuottavuuden taso on jo haasteellisempi määritettävä tekijä, koska Salesforce-kehitys eroaa normaalista sovelluskehityksestä. Tuottavuuden tason toisena alakategoriana ovat ympäristötekijät. Näihin tekijöihin lukeutuvat muun muassa työkalut ja testausympäristöt. Salesforcen kohdalla näitä ei tarvitse ottaa mukaan huomioihin, koska alusta itsessään tarjoaa valmiit testausalustat ja tarvittavat työkalut.

Toteuttamamme työkalu eroaa testipisteanalyysistä muun muassa tarvittavien esitietojen osalta, jotta menetelmää voi käyttää. Testipisteanalyysiä käytettäessä tarvitaan hyvät pohjatiedot menetelmän käytöstä. On osattava määrittää testipisteet oikein ja lopuksi käyttää monimutkaisia matemaattisia kaavoja, joiden avulla työmääräarvio saadaan laskettua (Vee-

nendaal ja Dekkers 1999). Toteuttamamme työkalun avulla käyttäjän ei tarvitse hakea taustalogiikasta mitään esitietoja. Riittää, kun tiedossa on tarvittavat projektiin liittyvät tiedot. Testipisteanalyysissä myös oletetaan, että testauksessa on osana erillinen testausprojekti, jota kohdeorganisaation kontekstissa ei ole käytössä. Testaus kohdeorganisaatiossa tapahtuu enemmän osana kokonaisprojektia kuin erillisenä kokonaisuutena.

7.4 Tutkimuksen luotettavuuden pohdinta ja jatkotutkimusaiheita

Tutkimuksen luotettavuuteen ja paikkansapitävyyteen liittyen on hyvä huomioida muutamia asioita, jotka voivat vaikuttaa tuloksiin. Sekä alkuhaastattelu- että loppuhaastatteluryhmämme olivat pieniä verrattuna suositeltuihin haastatteluryhmien kokoihin. Tämä voi vaikuttaa haastattelutuloksiin. Kohdeorganisaatiokin on pieni, joten suurempia työmääräarviointeihin perehtyneitä haastatteluryhmiä oli mahdoton koota. Näin ollen koemme saavamme luotettavimmat tulokset valitsemiltamme pieniltä haastatteluryhmiltä. Kaikilla haastatteluryhmän jäsenillä oli kokemusta työmäärien arvioinnista.

Tuloksia on vaikea yleistää kaikille organisaatioille sopiviksi, koska alusta asti lähtökohtana on ollut toteuttaa kohdeorganisaatiolle soveltuva työkalu. Taustalla on myös kohdeorganisaatiolle toteutettu testausstrategia, joka haastaa entisestään tuloksien yleistettävyyttä. Muut organisaatiot voivat kuitenkin saada tutkimuksesta suuntaviivoja ja ideoita. Vaikka tutkimus pohjautuu vahvasti kohdeorganisaatioon ja Salesforce-kontekstiin, voi tutkimuksen tuloksia soveltaa myös muihin ohjelmistokehitysalustoihin. On myös hyvä ottaa huomioon, että tutkimuksen toteuttajat ovat kohdeorganisaation työntekijöitä. Tästä huolimatta olemme pyrkineet pitämään kaikki tulokset ja pohdinnat läpinäkyvinä ja puolueettomina, eikä tuloksia tai haastateltavien sanomisia ole muovattu todellisuudesta poikkeaviksi.

Tämän tutkimuksen aikana työkalusta on toteutettu ensimmäinen versio, jota on tutkittu käytännöllisyyden ja käyttäjäystävällisyyden näkökulmasta. Työkalua voidaan vielä monin tavoin parantaa. Kehittämistarpeita on tullut ilmi tutkimuksen aikana esimerkiksi luvussa 6 ja aiemmin tässä luvussa. Tärkeimpänä kehittämistoimenpiteenä nostetaan esiin arvioiden paikkansapitävyys. Tätä ei voi liikaa korostaa. Toiseksi kehitettäväksi ominaisuudeksi nostetaan työkalun laajentaminen koskemaan yleisesti projektin työmääräarvioiden sekä eri osa-

alueiden työmääräarvioiden laskentaa. Tällaiselle ominaisuudelle olisi suuri tarve kohdeorganisaation sisällä. Näiden pohjalta ehdotamme jatkotutkimusaiheita, jotka ovat seuraavanlaiset:

1. Työkalun antamien työmääräarvioiden paikkaansapitävyyden tutkiminen
2. Työkalun laajentaminen muihin projektien osa-alueisiin sekä kokonaisuutena koko projektiin

Viimeiseksi nostamme esiin parannusehdotukset, joita saimme loppuhaastattelujen aikana. Parannusehdotukset kohdistuivat työkalun selkeyttämiseen ja prosessin helpottamiseen. Näitä olivat alkuun lisättävä listaus kaikista kysymyksistä ennen arviointiprosessin aloittamista sekä annettujen arvioiden selkeyttäminen ymmärrettävämpään muotoon. Nämä muutokset on tarkoitus tehdä kohdeorganisaatiossa. Tämän tutkimuksen kannalta muutoksien tarkastelu ei ole oleellista, koska emme toteuta uutta iteratiivista prosessia.

8 Yhteenveto

Tutkielman johdannossa todettiin, että ohjelmistokehitysprosessin kustannuksia on vaikea arvioida. Prosessi sisältää useita tehtäväalueita, joten eri osa-alueisiin kuluvaa aikaa on vaikea ennustaa. Tutkielmassa keskitymme testauksen osa-alueeseen. Onnistuneella testauksella on suora yhteys laatuun ja myönteiseen asiakaskokemukseen. Testauksen tavoite on huomata virheet ajoissa, jo ennen tuotantokäyttöä, mikä on kustannustehokasta. Tutkielmassa liikutaan testauksen työmääräarvioinnin osalta aihealueella, josta on saatavilla vähän tutkimustietoa. Kirjallisuuden perusteella yleinen tapa on arvioida testaukseen käytettävää aikaa prosenttiosuutena projektin kokonaiskustannuksista. Asiantuntijat luottavat kokemukseensa aikaisemmista projekteista, erityisesti pienissä organisaatioissa.

Tämän tutkimuksen pääpainona oli toteuttaa kohdeorganisaatiolle testauksen työmääräarvioita laskeva työkalu sekä samalla edistää toteutetun testausstrategian käyttöönottoa. Työkalu toteutettiin kohdeorganisaation Salesforce-ympäristöön, jossa toteutetut arviot tallennetaan järjestelmään tietueina. Tutkimuksen aihe tuli jatkotutkimusaiheena Virtanen ja Lehikoinen (2023) toteuttamasta tutkimuksesta, jossa kohdeorganisaatiolle toteutettiin testausstrategia.

Tutkimus toteutettiin suunnittelutieteellä, jonka ideana on toteuttaa artefakti, jota arvioidaan DSRM-prosessin mukaisesti iteratiivisessa syklissä (Peffers ym. 2007). Tämän tutkimuksen artefaktina toimi testauksen työmääräarvioita laskeva työkalu. Työkalun toimivuutta tarkasteltiin itse testaamalla toiminnallisesti sekä haastattelemalla kohdeorganisaation henkilöitä, jotka tulevat hyödyntämään työkalua työssään tulevaisuudessa.

Tutkimuksen osalta vastasimme määriteltyihin tutkimuskysymyksiin ja pääsimme haluttuun lopputulokseen. Tutkimuksesta jäi pois työkalun tuottamien aika-arvioiden kohdentumisen tarkkuus, sillä projektit, joissa työkalua kohdeorganisaatiossa käytetään ovat vielä kesken. Muutoin tutkimus onnistui mielestämme suunnitellun mukaisesti. Toivomme, että tutkimus antaa hieman uutta tietoa liittyen ohjelmistotestauksen työmäärien arviointiin. Lisäksi toivomme, että jatkossa työkalusta on kohdeorganisaatiossa hyötyä tuottamalla kohdilleen osuvia testauksen työmäärien arviointeja.

Lähteet

Ali, Asad ja Carmine Gravino. 2019. “A systematic literature review of software effort prediction using machine learning methods” [kielellä en]. *Journal of Software: Evolution and Process* 31, numero 10 (lokakuu): e2211. ISSN: 2047-7473, 2047-7481, viitattu 2. marraskuuta 2023. https://doi.org/10.1002/smr.2211. https://onlinelibrary.wiley.com/doi/10.1002/smr.2211.

Ammann, Paul ja Jeff Offutt. 2016. *Introduction to software testing*. Cambridge University Press.

Arora, Ankit ja Abhinav Gupta. 2013. *Force. com tips and tricks*. Packt Publishing.

Bluemke, Ilona ja Agnieszka Malanowska. 2022. “Software Testing Effort Estimation and Related Problems: A Systematic Literature Review” [kielellä en]. *ACM Computing Surveys* 54, numero 3 (huhtikuu): 1–38. ISSN: 0360-0300, 1557-7341, viitattu 2. marraskuuta 2023. https://doi.org/10.1145/3442694. https://dl.acm.org/doi/10.1145/3442694.

Boehm, Barry, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy ja Richard Selby. 1995. “Cost models for future software life cycle processes: COCOMO 2.0”. Publisher: Springer, *Annals of software engineering* 1:57–94.

Coxon, Hayley. 2022. *Salesforce Environments: Which Do I Use and When?* [Kielellä en-GB], heinäkuu. Viitattu 17. marraskuuta 2023. https://www.salesforceben.com/salesforce-environments-which-do-i-use-and-when/.

Grimstad, Stein ja Magne Jørgensen. 2007. “Inconsistency of expert judgment-based estimates of software development effort”. *Journal of Systems and Software* 80 (11): 1770–1777. ISSN: 0164-1212. https://doi.org/https://doi.org/10.1016/j.jss.2007.03.001. https://www.sciencedirect.com/science/article/pii/S0164121207000714.

Grimstad, Stein, Magne Jørgensen ja Kjetil Moløkken-Østvold. 2006. “Software effort estimation terminology: The tower of Babel” [kielellä en]. *Information and Software Technology* 48, numero 4 (huhtikuu): 302–310. ISSN: 09505849, viitattu 2. marraskuuta 2023. https://doi.org/10.1016/j.infsof.2005.04.004. https://linkinghub.elsevier.com/retrieve/pii/S0950584905000674.

Gupta, Radhika, Sahil Verma ja Kavita Janjua. 2018. “Custom Application Development in Cloud Environment: Using Salesforce”. Teoksessa *2018 4th International Conference on Computing Sciences (ICCS),* 23–27. Elokuu. Viitattu 2. marraskuuta 2023. https://doi.org/10.1109/ICCS.2018.00010. https://ieeexplore.ieee.org/document/8611028/authors#authors.

Harris, Ivan. 2022. “Package Development” [kielellä en]. Teoksessa *Beginning Salesforce DX: Versatile and Resilient Salesforce Application Development,* toimittanut Ivan Harris, 457–529. Berkeley, CA: Apress. ISBN: 978-1-4842-8114-7, viitattu 11. marraskuuta 2023. https://doi.org/10.1007/978-1-4842-8114-7_13. https://doi.org/10.1007/978-1-4842-8114-7_13.

Huang, Jianglin, Yan-Fu Li ja Min Xie. 2015. “An empirical analysis of data preprocessing for machine learning-based software cost estimation” [kielellä en]. *Information and Software Technology* 67 (marraskuu): 108–127. ISSN: 09505849, viitattu 27. marraskuuta 2023. https://doi.org/10.1016/j.infsof.2015.07.004. https://linkinghub.elsevier.com/retrieve/pii/S0950584915001275.

Jayakumar, Kamala Ramasubramani ja Alain Abran. 2013. “A Survey of Software Test Estimation Techniques”. *Journal of Software Engineering and Applications* 06 (10): 47–52. ISSN: 1945-3116, 1945-3124, viitattu 18. marraskuuta 2023. https://doi.org/10.4236/jsea.2013.610A006. http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jsea.2013.610A006.

Kemerer, Chris F. 1987. “An empirical validation of software cost estimation models” [kielellä en]. *Communications of the ACM* 30, numero 5 (toukokuu): 416–429. ISSN: 0001-0782, 1557-7317, viitattu 27. marraskuuta 2023. https://doi.org/10.1145/22899.22906. https://dl.acm.org/doi/10.1145/22899.22906.

Keung, Jacky. 2009. “Software Development Cost Estimation Using Analogy: A Review”. Teoksessa *2009 Australian Software Engineering Conference,* 327–336. Gold Coast, Australia: IEEE. ISBN: 978-0-7695-3599-9, viitattu 27. marraskuuta 2023. https://doi.org/10.1109/ASWEC.2009.32. http://ieeexplore.ieee.org/document/5076655/.

Lin, Geng, David Fu, Jinzy Zhu ja Glenn Dasmalchi. 2009. “Cloud Computing: IT as a Service”. Conference Name: IT Professional, *IT Professional* 11, numero 2 (maaliskuu): 10–13. ISSN: 1941-045X, viitattu 11. marraskuuta 2023. https://doi.org/10.1109/MITP.2009.22.

López-Martín, Cuauhtémoc. 2022. “Machine learning techniques for software testing effort prediction” [kielellä en]. *Software Quality Journal* 30, numero 1 (maaliskuu): 65–100. ISSN: 0963-9314, 1573-1367, viitattu 2. marraskuuta 2023. https://doi.org/10.1007/s11219-020-09545-8. https://link.springer.com/10.1007/s11219-020-09545-8.

Mahnič, Viljan ja Tomaž Hovelja. 2012. “On using planning poker for estimating user stories” [kielellä en]. *Journal of Systems and Software* 85, numero 9 (syyskuu): 2086–2095. ISSN: 01641212, viitattu 27. marraskuuta 2023. https://doi.org/10.1016/j.jss.2012.04.005. https://linkinghub.elsevier.com/retrieve/pii/S0164121212001021.

Manchar, Anuradha ja Ankit Chouhan. 2017. “Salesforce CRM: A new way of managing customer relationship in cloud environment”. Teoksessa *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT),* 1–4. Helmikuu. Viitattu 10. marraskuuta 2023. https://doi.org/10.1109/ICECCT.2017.8117887. https://ieeexplore.ieee.org/abstract/document/8117887.

Mathew, Reena ja Ryan Spraetz. 2009. “Test Automation on a SaaS Platform”. Teoksessa *2009 International Conference on Software Testing Verification and Validation,* 317–325. ISSN: 2159-4848. Huhtikuu. Viitattu 2. marraskuuta 2023. https://doi.org/10.1109/ICST.2009.46. https://ieeexplore.ieee.org/document/4815365.

Molokken, K. ja M. Jorgensen. 2003. “A review of software surveys on software effort estimation”. Teoksessa *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* 223–230. Rome, Italy: IEEE Comput. Soc. ISBN: 978-0-7695-2002-5, viitattu 3. marraskuuta 2023. https://doi.org/10.1109/ISESE.2003.1237981. http://ieeexplore.ieee.org/document/1237981/.

Patel, Jigar ja Ankit Chouhan. 2016. “An approach to introduce basics of Salesforce.com: A cloud service provider”. Teoksessa *2016 International Conference on Communication and Electronics Systems (ICCES),* 1–8. Lokakuu. Viitattu 26. lokakuuta 2023. https://doi.org/10.1109/CESYS.2016.7889991. https://ieeexplore.ieee.org/abstract/document/7889991.

Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger ja Samir Chatterjee. 2007. “A Design Science Research Methodology for Information Systems Research”. Publisher: Routledge _eprint: https://doi.org/10.2753/MIS0742-1222240302, *Journal of Management Information Systems* 24, numero 3 (joulukuu): 45–77. ISSN: 0742-1222, viitattu 23. marraskuuta 2023. https://doi.org/10.2753/MIS0742-1222240302. https://doi.org/10.2753/MIS0742-1222240302.

Quadri, SMK ja Sheikh Umar Farooq. 2010. “Software testing–goals, principles, and limitations”. *International Journal of Computer Applications* 6 (9): 1.

Radliński, Łukasz. 2023. “The Impact of Data Quality on Software Testing Effort Prediction” [kielellä en]. *Electronics* 12, numero 7 (maaliskuu): 1656. ISSN: 2079-9292, viitattu 2. marraskuuta 2023. https://doi.org/10.3390/electronics12071656. https://www.mdpi.com/2079-9292/12/7/1656.

Salesforce. 2023a. “Get Started with Lightning Web Components”. Viitattu 16. marraskuuta 2023. https://developer.salesforce.com/docs/platform/lwc/guide.

———. *Sandbox Types and Templates* [kielellä en-US]. Viitattu 17. marraskuuta 2023. https://help.salesforce.com/s/articleView?id=sf.create_test_instance.htm&language=en_US&type=5.

Shekhar, Shivangi ja Umesh Kumar. 2016. “Review of various software cost estimation techniques”. Publisher: Foundation of Computer Science, *International Journal of Computer Applications* 141 (11): 31–34.

Singh, Brajesh Kumar ja AK Misra. 2012. “Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for nasa software projects”. Publisher: Citeseer, *International Journal of Computer Applications* 59 (9).

Singh, Sanjay Kumar ja Amarjeet Singh. 2012. *Software testing.* Vandana Publications.

Suri, PK ja Pallavi Ranjan. 2012. “Comparative analysis of software effort estimation techniques”. Publisher: Citeseer, *International Journal of Computer Applications* 48 (21): 12–19.

Tidwell, Jenifer. 2010. *Designing interfaces: Patterns for effective interaction design.* "O’Reilly Media, Inc."

Trendowicz, Adam. 2013. *Software cost estimation, benchmarking, and risk assessment: the software decision-makers’ guide to predictable software development.* The Fraunhofer IESE series on software and systems engineering. OCLC: ocn794710374. Heidelberg ; New York: Springer. ISBN: 978-3-642-30763-8.

Trendowicz, Adam ja Ross Jeffery. 2014. *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success* [kielellä en]. Cham: Springer International Publishing. ISBN: 978-3-319-03629-8, viitattu 26. marraskuuta 2023. https://doi.org/10.1007/978-3-319-03629-8. https://link.springer.com/10.1007/978-3-319-03629-8.

Walkerden, Fiona ja Ross Jeffery. 1999. “An Empirical Study of Analogy-based Software Effort Estimation”. *Empirical Software Engineering* 4 (2): 135–158. ISSN: 13823256, viitattu 27. marraskuuta 2023. https://doi.org/10.1023/A:1009872202035. http://link.springer.com/10.1023/A:1009872202035.

Veenendaal, E.P.W.M., van ja T. Dekkers. 1999. “Test point analysis : a method for test estimation” [kielellä English]. Teoksessa *Project control for software quality : proceedings of the combined 10th European Software Control and Metrics conference and the 2nd SCOPE conference on software product evaluation, April 27-29, 1999, Herstmonceux, England,* toimittanut R. J. Kusters, A. Cowderoy ja F. J. Heemstra, 47–59. Shaker-Verlag. ISBN: 90-423-0075-2.

Venable, John, Jan Pries-Heje ja Richard Baskerville. 2016. “FEDS: a Framework for Evaluation in Design Science Research” [kielellä en]. *European Journal of Information Systems* 25, numero 1 (tammikuu): 77–89. ISSN: 0960-085X, 1476-9344, viitattu 7. joulukuuta 2023. https://doi.org/10.1057/ejis.2014.36. https://www.tandfonline.com/doi/full/10.1057/ejis.2014.36.

Virtanen, Otto ja Samuel Lehikoinen. 2023. “Testausstrategia Salesforce-kehitykseen” [kielellä fi]. Accepted: 2023-05-10T05:53:12Z, viitattu 12. tammikuuta 2024. https://jyx.jyu.fi/handle/123456789/86842.