Albert Viljanen

Vaatimustenhallinnan kehittäminen ja automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

29.5.2023

Tiivistelmä

Tekijä:	Albert Viljanen
Otsikko:	Vaatimustenhallinnan kehittäminen ja automatisointi
Sivumäärä:	30 sivua
Aika:	29.05.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Matti Välikylä, lehtori
	Mikko Pajakkala, projektihallinnan johtaja

Opinnäytetyö tehtiin kansainväliselle ohjelmistokehitysyritykselle, joka siirtyi käyttämään uutta vaatimustenhallintajärjestelmää. Yritys haluaa automatisoida vaatimustenhallintaan liittyviä toimintoja työkalun avulla ja tehostaa vaatimusten jäljitettävyyttä. Työn tarkoituksena oli luoda työkalu, jolla vaatimuksia viedään järjestelmään ja parannetaan jäljitettävyyden seurantaa.

Työ aloitettiin tutkimalla teoriaa vaatimusten ja vaatimustenhallinnan osalta. Tutkinta pohjautuu pääosin kirjallisiin lähteisiin. Teoriaa käytettiin tukena ja ohjeistuksena työkalun ja vaatimustenhallinnan toteutukseen.

Työkalu toteutettiin Python-ohjelmointikielellä, ja uutena järjestelmänä toimi Atlassian Jira. Jira oli entuudestaan yrityksellä käytössä projektinhallinnan tehtävissä. Työkalun tuli pystyä päivittämään ja lisäämään vaatimuksia järjestelmään. Vaatimukset tulevat asiakkailta Excel-tiedostoina. Kun tavoitteet ja vaatimukset olivat selvillä, oli mahdollista toteuttaa testaussuunnitelma ja käyttöönottostrategia. Työkalun etenemistä seurattiin säännöllisin väliajoin palaverien muodossa.

Testaus toteutettiin ensin erillisessä kehitysympäristössä, jonka jälkeen siirryttiin yrityksen järjestelmään. Jokainen testitapaus tuli suorittaa molemmissa ympäristöissä. Testauksessa ilmeni puutteita Excel-tiedostojen käsittelyssä. Käyttöönotto tapahtui, kun tiimi, jossa työskentelin, siirtyi uuden projektin pariin. Käyttöönottoon kuului myös koulutusta, dokumentointia ja tukea.

Työkalun ja testausautomaation avulla myös jäljitettävyyttä parannettiin. Tämä osa työkalusta irrotettiin kuitenkin erilliseksi osaksi, jotta työkalu pysyy mahdollisimman helppokäyttöisenä.

Kokonaisuudessaan työ muutti yrityksen toimintatapoja vaatimustenhallinnassa merkittävästi. Työkalulle oli selkeä tarve, ja se täytti sille asetetut tavoitteet. Vaatimustenhallinta onnistuisi tosin helpommin ja varmemmin, jos asiakkaalla ja kehittäjällä olisi yhteinen järjestelmä vaatimuksille.

Avainsanat: vaatimus, vaatimustenhallinta, automatisointi, työkalu

Abstract

Author:	Albert Viljanen
Title:	Development and Automation of Requirements Management
Number of Pages:	30 pages
Date:	29 May 2023
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisors:	Matti Välikylä, Senior Lecturer Mikko Pajakkala, Head of Project Management

---

The thesis is conducted for an international software development company that is transitioning to a new requirements management system. The company aims to automate functions related to requirements management using a tool and to enhance the traceability of requirements. The purpose of this work was to create a tool for importing requirements into the system and improving traceability tracking.

The work was initiated by studying the theory of requirements and requirements management. The investigation was mainly based on written sources. The theory was used as a support and guidance for the implementation of the tool and requirements management.

The tool was implemented using the Python programming language, and the new system used was Atlassian Jira. Jira was already in use by the company for project management tasks. The tool had to be able to update and add requirements to the system. The requirements come from customers in the form of Excel files. Once the objectives and requirements were clear, it was possible to implement a testing plan and deployment strategy. The progress of the tool was monitored at regular intervals through meetings.

Testing was first carried out in a separate development environment, after which it was moved to the company's system. Each test case had to be performed in both environments. The testing revealed deficiencies in handling Excel files. The implementation took place when the team I worked with moved on to a new project. Implementation also included training, documentation, and support.

With the help of the tool and test automation, traceability was also improved. However, this part of the tool was separated into a separate component to keep the tool as user-friendly as possible. As result, the work significantly changed the company's practices in requirements management. There was a clear need for the tool, and it met the set goals. However, requirements management could be easier and more reliable if the customer and the developer had a shared system for requirements.

Keywords: requirement, requirements management, automation, tool

Sisällys

1 Johdanto

Tässä opinnäytetyössä tutkitaan, kehitetään ja automatisoidaan ohjelmistokehitysyrityksen vaatimustenhallintaa ja vaatimusten jäljitettävyyttä. Työssä luodaan työkalu, joka automatisoi vaatimustenhallinnan osia uudessa järjestelmässä. Tämän lisäksi suunnitellaan ja luodaan ratkaisuja vaatimusten jäljitettävyyden parantamiseen. Työn tukena käytetään standardeja, kirjallisuutta, yrityksen sisäisiä määrittelyjä, ohjeistuksia ja yrityksen työntekijöiden ohjeita.

Opinnäytetyö aloitetaan kartoittamalla yrityksen tarve vaatimustenhallinnan kehittämiseen ja vaatimustenhallinnan lähtötilanne. Kartoituksessa selviää myös työn toteuttamistavat, käytettävät järjestelmät ja tarve työkalun luomiselle. Työkalu tulee toteuttamaan vaatimustenhallintaa asiakkaiden tiedostojen ja järjestelmän välillä.

Työssä tutkitaan vaatimuksiin ja vaatimustenhallintaan liittyvää teoriaa, jota käytetään tukena ja ohjeistuksena työkalun luomiseen ja vaatimustenhallinnan oikeaan toteutukseen. Teoriaosuudessa käsitellään myös yleisimpiä haittavaikutuksia heikosti toteutetun vaatimustenhallinnan seurauksena.

Tavoitteena on luoda yritykselle työkalu, joka vastaa yrityksen tarpeisiin vaatimustenhallinnan ongelmakohdissa, ja määritellä järjestelmä käytettäväksi vaatimustenhallintaa varten. Järjestelmän ollessa käytössä jäljitettävyyttä tehostetaan järjestelmän ja testausautomaatioympäristön välillä. Tavoitteena on pystyä tarkastamaan ja todentamaan vaatimukset, jotka on testattu.

Työn lopuksi pohditaan, täyttyivätkö työn tavoitteet, tehostuiko yrityksen vaatimustenhallinta ja oliko työkalun luominen sopiva ratkaisu näihin ongelmiin.

2 Vaatimustenhallinnan lähtötilanne

Tämä opinnäytetyö on tehty suomalaisen kansainvälisen ohjelmistokehitysyrityksen toimeksiannosta. Yritys on erikoistunut ohjelmistokehityksen lisäksi myös laite- ja komponenttiratkaisujen tuottamiseen, mikä mahdollistaa asiakkaiden tarpeiden entistä paremman huomioimisen ja monipuolisempien kokonaisratkaisujen tarjoamisen. Lisäksi yrityksellä on oma tuotekehitysosasto, joka vastaa uusien innovatiivisten tuotteiden kehittämisestä ja markkinoille tuomisesta. Yrityksellä on laaja asiakaskunta eri puolilta maailmaa, ja se toimii globaalilla markkinalla.

Yrityksen vaatimustenhallinta toteutetaan tällä hetkellä Visure Solutionsin vaatimustenhallintaohjelmistolla. Ohjelmistosta ollaan luopumassa, sillä ohjelmiston käyttö vaatii lisenssin, jota ei ole tarjota kaikille työntekijöille. Tästä syystä projektin vaatimustenhallinta on usein yhden ihmisen takana. Tämä rajoittaa pääsyä vaatimuksiin ja vaikeuttaa vaatimustenhallintaa. Nämä hidastavat projektin etenemistä.

Lisäksi Visuren käyttö vaatii laajempaa perehtymistä käyttäjältä. Tiedonsiirtomuodot ovat rajattuja, mikä voi aiheuttaa osassa tapauksista yhteensopivuusongelmia asiakkaiden järjestelmien kanssa. Järjestelmien välisen tiedonsiirron yhteydessä saattaa osa vaatimusten tiedoista muuttaa merkittäviltä osin muotoaan. Näitä voivat olla esimerkiksi kuvat, taulukot ja vaatimushierarkiat. Tämä johtaa lisätyöhön vaatimustenhallinnassa.

Vaatimustenhallinta vaatimustiedostojen käsittelyn osalta tapahtuu suurilta osin manuaalisesti, mikä vie aikaa ja resursseja. Manuaalinen tietojenkäsittely on myös alttiimpi virheille. Asiakkaan toiveet tai toteutuksen suunnitelma voivat muuttua useasti, mikä on normaalia. Tämän vuoksi on tärkeää, että vaatimuksien päivittyessä ne on helppo päivittää luotettavasti ja nopeasti järjestelmään. Vaatimukset tulevat asiakkailta pääsääntöisesti Excel-tiedostoina.

Tämän lisäksi vaatimusten jäljitettävyyttä halutaan parantaa. Nykyisellään myös jäljitettävyys on toteutettu manuaalisesti, ja sitä halutaan automatisoida. Yrityksellä ei ole käytössä ratkaisua, mikä helpottaisi kyseisiä ongelmakohtia.

Yritys haluaa tehostaa ja automatisoida vaatimustenhallintaa sekä siirtää vaatimustenhallinnan jo käytössä olevaan Jira-projektinhallintajärjestelmään, jolloin projektin eri osa-alueiden seuranta tapahtuu keskitetysti ja jäljitettävyys paranee. Laajasti käytössä oleva Jira mahdollistaa huomattavasti helpomman integroinnin muihin järjestelmiin, jolloin esimerkiksi testausautomaatio ja jäljitettävyys onnistuvat nykyistä järjestelmää paremmin.

3 Vaatimustenhallinnan kehityskohdat

Työn tavoitteena on kehittää työkalu, jonka avulla vaatimukset voidaan tuoda uuteen järjestelmään, sekä määritellä uusi järjestelmä vaatimuksien osalta yrityksen toiveiden mukaisesti.

Opinnäytetyössä keskitytään vaatimustenhallinnan automatisointiin ja tehokkuuden parantamiseen, jotta yritys voi käyttää resurssejaan tehokkaammin ja saavuttaa parempia tuloksia. Vaikka vaatimusten vaikutus osana ohjelmistokehitystä on tärkeä, tämä opinnäytetyö keskittyy erityisesti vaatimustenhallintaan ja sen kehittämiseen. Automatisoinnin avulla voidaan vähentää manuaalista työtä, mikä vähentää inhimillisiä virheitä ja nopeuttaa työprosessia. Tehostettu vaatimustenhallinta puolestaan parantaa projektinhallinnan laatua ja nopeuttaa tuotteen kehittämistä.

Työssä luodaan Python-ohjelmointikielellä työkalu, joka käsittelee vaatimuksia ja tuo ne Atlassian Jira-projektinhallintajärjestelmään. Työkalun tulee pystyä tuomaan vaatimuksia järjestelmään Excel-tiedostoista. Tämän vuoksi työkalun tulee pystyä käsittelemään Excel-tiedostoja mahdollisimman yleisellä tasolla, jotta erilaiset tiedostomuodot saadaan tuotua järjestelmään mahdollisimman sujuvasti. Tämä tekee vaatimustenhallinnasta joustavampaa ja vähentää manuaalista työtä tiedostojen käsittelyssä. Samalla se parantaa vaatimustenhallinnan tehokkuutta ja tarkkuutta, mikä auttaa saavuttamaan projektin tavoitteet aikataulussa.

3.1 Käytettävät työkalut ja järjestelmät

Opinnäytetyössä käytettävä järjestelmä vaatimustenhallinnassa on Atlassian Jira -projektinhallintajärjestelmä. Jira on entuudestaan käytössä yrityksen kaikilla työntekijöillä. Jira on monipuolinen järjestelmä, joka mahdollistaa projektinhallinnan lisäksi esimerkiksi vaatimusten hallinnan, testauksen, dokumentaation ja paljon muuta. Jira käsittelee yksittäisiä ongelmia tai tapauksia järjestelmässä nimellä *issue* (ongelma). Jokaisella Jira-issuella on yleensä oma uniikki tunnisteensa, joka helpottaa sen tunnistamista järjestelmässä.

Opinnäytetyössä käytetään työkalun luomiseen Python-ohjelmointikieltä, sillä se oli yksi työkalun vaatimuksista. Yrityksessä käytetään laajasti Pythonia, jolloin tarpeen vaatiessa myös muut yrityksen työntekijät voivat jatkokehittää työkaluja tai muita Pythonilla luotuja toteutuksia. Tällä ja hyvällä dokumentoinnilla voidaan varmistaa, että työkaluja voidaan käyttää jatkossakin, vaikka alkuperäinen kehittäjä lähtisi yrityksestä.

Jäljitettävyyden parantaminen tapahtuu Robot Framework -ympäristössä. Robot Framework on yrityksellä käytössä testausautomaatiossa. Testeihin integroidaan ohjelma, joka vertailee vaatimuksia useasta eri lähteestä ja luo helposti luettavaa informaatiota tuloksista.

3.2 Vaatimustenhallinnan ongelmakohdat

Yrityksen suurin ongelmakohta vaatimustenhallinnassa on rajattu pääsy nykyiseen järjestelmään. Yrityksessä projektitiimi koostuu yleensä noin 5–10 työntekijästä, ja pahimmassa tapauksessa vain yhdellä tiimin jäsenellä on pääsy vaatimustenhallintajärjestelmään. Tämä voi aiheuttaa viivästyksiä, jos henkilö ei ole saatavilla tai hänellä on liian paljon muuta työtä, eikä hän pysty vastaamaan vaatimuksiin liittyviin kysymyksiin. Atlassian Jira on yrityksellä valmiiksi käytössä projektinhallintajärjestelmänä, ja kaikilla työntekijöillä on pääsy järjestelmään. Tästä syystä vaatimustenhallinnan siirtäminen Jiraan mahdollistaa kaikille yrityksen työntekijöille pääsyn tarkastelemaan vaatimuksia.

Vaatimustiedostojen manuaalinen käsittely on myös yksi ongelmakohdista. Manuaalinen käsittely vie aikaa ja altistaa inhimillisille virheille. Tämä voi johtaa vaatimusten puutteellisuuteen tai virheisiin, jotka voivat vaikuttaa projektin onnistumiseen. Työkalu, joka automatisoi vaatimusten käsittelyn, voi auttaa vähentämään tätä riskiä ja tehostaa projektin etenemistä.

Myös jäljitettävyyden parantaminen on tärkeä osa vaatimustenhallinnan tehostamista. Tämä voidaan saavuttaa automatisoimalla vaatimusten käsittelyprosessi ja tekemällä vaatimukset kaikkien saataville. Tällä tavoin kaikki tiimin jäsenet voivat seurata projektin etenemistä ja todentaa, että vaatimuksia seurataan.

Työkalun avulla myös testausautomaation kattavuuden tarkistus voidaan automatisoida. Turvallisuuskriittisissä projekteissa toteutusprosessi edellyttää jäljitettävyyttä, jolloin vaatimus sekä sen jäljitettävyys on pystyttävä todentamaan toteutuksessa ja testauksessa. Tehostettu jäljitettävyys myös auttaa varmistamaan projektin aikana, että vaatimukset täyttyvät ja että projektin tavoitteet ja eteneminen saavutetaan ajallaan.

4 Vaatimus

Vaatimus on ohjelmistolle asetettu vaatimus tai vaatimuskuvaus, joka kuvaa järjestelmän toiminnallisuutta, ominaisuuksia tai vaadittavaa suorituskykyä. Vaatimuksia käytetään ohjelmistokehityksen aikana ohjelmiston toiminnallisuuden ja vaatimusten määrittelyn hallintaan. Vaatimukset auttavat varmistamaan, että ohjelmisto toteuttaa asiakkaan tarpeet ja vaatimukset. (Wiegers & Beatty 2013: 5–7.)

Vaatimukset voidaan jakaa kahteen tyyppiin: toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnalliset vaatimukset kuvaavat, mitä ohjelmiston tulee tehdä ja mitä ominaisuuksia sen tulee tarjota käyttäjille. Ei-toiminnalliset vaatimukset keskittyvät puolestaan ohjelmiston suorituskykyyn, turvallisuuteen, käytettävyyteen ja muihin ei-toiminnallisiin piirteisiin. (Wiegers & Beatty 2013: 7.)

Vaatimusten käyttö on keskeistä ohjelmistokehitysprosessin kaikissa vaiheissa alkaen suunnittelusta ja jatkuen toteutukseen, testaukseen ja ylläpitoon asti. Vaatimukset toimivat ohjelmiston kehitystiimille lähtökohtana ohjelmiston suunnittelulle ja toteutukselle. (Wiegers & Beatty 2013: 52–53.)

Vaatimuksia käytetään myös tärkeänä viestintävälineenä asiakkaan ja kehitystiimin välillä. Vaatimukset auttavat asiakasta ymmärtämään, mitä hän voi odottaa ohjelmistolta ja millaisia toiminnallisuuksia tai ominaisuuksia hän voi käyttää. Kehitystiimi puolestaan käyttää vaatimuksia ohjelman määrittelyyn ja suunnitteluun, jotta lopputulos vastaa asiakkaan odotuksia. Lisäksi vaatimuksia voidaan käyttää laadunvarmistuksessa. Kun vaatimukset on määritelty ja dokumentoitu selkeästi, voidaan käyttää niitä mittarina, jonka avulla voidaan arvioida ohjelmiston laadun tasoa. (Withall 2007: 34–38.)

4.1 Vaatimusmäärittely

Vaatimusmäärittely on osa ohjelmistokehityksen prosessia, jossa määritellään ohjelmistolle asetetut vaatimukset ja tarpeet. Vaatimukset voivat olla asiakkaan toiveita, lakisääteisiä vaatimuksia tai muita rajoitteita, jotka on otettava

huomioon ohjelmiston suunnittelussa ja toteutuksessa. (Sommerville 2011: 22.) Vaatimusmäärittelyssä pyritään ymmärtämään asiakkaan tarpeita ja toiveita ja muuttamaan ne selkeiksi, mitattaviksi ja toteutuskelpoisiksi vaatimuksiksi. (Sommerville 2011: 111–114.)

Vaatimusmäärittely on tärkeä osa ohjelmistokehitystä, sillä se auttaa varmistamaan, että ohjelmisto vastaa asiakkaan tarpeita ja että sen kehitys on suunniteltu ja toteutettu mahdollisimman tehokkaasti. Hyvin laadittu vaatimusmäärittely auttaa myös välttämään virheitä ja puutteita ohjelmiston suunnittelussa ja toteutuksessa, mikä voi säästää aikaa ja kustannuksia pitkällä aikavälillä. (Sommerville 2011: 58–62.)

Vaatimusmäärittelyprosessi alkaa yleensä vaatimusten keräämisellä ja tunnistamisella, joka voi tapahtua erilaisin menetelmin, kuten haastatteluilla, kyselyillä ja havainnoinnilla. Vaatimukset tulee määritellä selkeästi ja yksiselitteisesti, jotta ne voidaan ymmärtää ja tulkita oikein. Vaatimukset tulee myös priorisoida ja valita niistä tärkeimmät toteutettavaksi. (Paakki 2011.)

Vaatimusten määrittelyssä tulee ottaa huomioon myös erilaiset käyttäjäryhmät ja heidän tarpeensa, jotta ohjelmisto vastaa käyttäjien odotuksia ja tarpeita. Vaatimusten määrittelyprosessi on iteratiivinen, ja vaatimuksia voidaan muokata ja täydentää kehitysprosessin edetessä. (Paakki 2011.)

Huono vaatimustenmäärittely voi johtaa projektin keston pidentymiseen ja budjetin kasvamiseen jopa kolminkertaiseksi. Tämä johtuu siitä, että huonot vaatimukset voivat aiheuttaa muutoksia ja viivästyksiä projektin eri vaiheissa. Heikosti toteutettu vaatimustenmäärittely voi myös aiheuttaa laatuongelmia, kun kehittäjät eivät saa tarpeeksi tarkkaa kuvaa kehitettävästä ohjelmistosta, mikä johtaa muutoksiin ja korjauksiin projektin edetessä. (Egeland 2019.)

4.2 Vaatimustenhallinta

Vaatimustenhallinnalla tarkoitetaan prosessia, joka kattaa vaatimusten keräämisen, dokumentoinnin, analysoinnin, validoinnin, varmentamisen ja hallinnan

(Wundenberg 2015: 32). Vaatimukset ovat asiakkaan ja sidosryhmien tarpeiden ilmaisemista toiminnallisesti, suorituskyvyllisesti, laadullisesti ja turvallisuuteen liittyen (Wundenberg 2015: 7–8). Vaatimustenhallinta on erityisen tärkeää suurissa ja monimutkaisissa tietointensiivisissä projekteissa, joissa vaatimukset voivat olla ristiriitaisia ja epäselviä. Vaatimustenhallinta auttaa varmistamaan, että lopputuote vastaa asiakkaiden ja sidosryhmien tarpeita, että se täyttää laatuvaatimukset, että se on turvallinen ja että se on toteutettu tehokkaasti ja taloudellisesti. (Wundenberg 2015: 39–41.)

Vaatimustenhallinnan prosessi koostuu useista vaiheista, joista ensimmäinen on vaatimusten kerääminen. Tämä sisältää vaatimusten tunnistamisen ja niiden dokumentoinnin. Seuraava vaihe on vaatimusten analysointi, joka sisältää vaatimusten järjestämisen tärkeysjärjestykseen, ristiriitojen tunnistamisen ja selvittämisen sekä vaikutusten arvioinnin. Kun vaatimukset on analysoitu, ne tulee validoida, varmentaa ja hyväksyä. Tämä varmistaa, että vaatimukset on oikein kirjattu ja täydellisesti toteutettavissa ja että ne vastaavat asiakkaiden ja sidosryhmien tarpeita. Lopuksi vaatimukset tulee dokumentoida ja hallita vaatimusten mahdollisia muutoksia, jotta varmistetaan, että ne pysyvät ajan tasalla ja ovat käytettävissä koko projektin ajan. (Wundenberg 2015: 5–8.)

Vaatimustenhallintaan liittyy myös useita menetelmiä ja työkaluja, joita voidaan käyttää prosessin eri vaiheissa. Näitä ovat esimerkiksi haastattelut, kyselyt, ryhmätyöskentely, prototyyppien kehittäminen, simulointi ja mallinnus. Lisäksi on olemassa erilaisia ohjelmistoja ja järjestelmiä, jotka voivat auttaa vaatimustenhallinnan prosessissa, kuten vaatimustenhallintajärjestelmät ja jäljitettävyystyökalut. (Wundenberg 2015: 5–7.)

Hyvä vaatimustenhallinta on tärkeää projektin onnistumiselle ja sen avulla voidaan varmistaa, että lopputuote vastaa asiakkaan tarpeita ja odotuksia (Young 2004: 2–4). Puutteellinen vaatimustenhallinta voi aiheuttaa useita ongelmia, jotka vaikuttavat projektin lopputulokseen ja laatuun. Näitä ovat esimerkiksi kustannusten kasvu, aikataulujen ylitykset, laadun heikkeneminen, kommunikaatioongelmat, asiakastyytymättömyys sekä riskien kasvu. (Young 2004: 206.)

Kustannusten kasvu: puutteellisesti toteutettu vaatimustenhallinta voi johtaa projektin lisätyöaikoihin, tarpeettomiin muutoksiin ja uudelleen suunnitteluun, mikä kasvattaa projektin kustannuksia (Young 2004: 61–62).

Aikataulun ylitykset: puutteellisesti toteutettu vaatimustenhallinta voi aiheuttaa aikataulun ylityksiä, koska vaatimukset eivät ole selvät tai ne muuttuvat kesken projektin (Young 2004: 18).

Laadun puute: puutteellisesti toteutettu vaatimustenhallinta voi johtaa lopputuotteen laadun puutteisiin, koska vaatimukset eivät vastaa asiakkaan tarpeita tai niitä ei ole määritelty selkeästi (Young 2004: 18).

Kommunikaatio-ongelmat: puutteellisesti toteutettu vaatimustenhallinta voi johtaa kommunikaatio-ongelmiin eri projektiryhmien välillä, mikä voi hidastaa projektin etenemistä (Young 2004: 2).

Asiakastyytymättömyys: puutteellisesti toteutettu vaatimustenhallinta voi johtaa asiakastyytymättömyyteen, koska lopputuote ei vastaa asiakkaan odotuksia (Young 2004: 175–176).

Riskien kasvu: puutteellisesti toteutettu vaatimustenhallinta voi johtaa riskien kasvuun, koska vaatimusten epäselvyys tai puutteellisuus voi johtaa virheisiin tai puutteisiin lopputuotteessa (Young 2004: 38).

Vaatimustenhallinnan automatisointi tarkoittaa vaatimustenmäärittelyn prosessin osien tai kokonaisuuden automatisointia tietokoneohjelmien tai erilaisten työkalujen avulla. Tavoitteena on yksinkertaistaa, nopeuttaa ja tehostaa vaatimustenhallintaprosessia sekä parantaa sen laatua. Automaation avulla voidaan esimerkiksi helpottaa vaatimusten kokoamista, järjestämistä ja analysointia. Automatisointi voi myös vähentää inhimillisten virheiden riskiä. (Pohl & Rupp 2015: 141–144.)

Vaatimustenhallinnan automatisointi vaatii huolellista suunnittelua ja toteutusta, jotta se toimii tehokkaasti ja luotettavasti. Työkalun tulee tukea organisaation käyttämiä vaatimustenhallinnan prosesseja ja standardeja. On tärkeää

ymmärtää ja saavuttaa juuri ne hyödyt, joita organisaatio tarvitsee. Työkalun tulee olla skaalautuva, jotta se pystyy käsittelemään suuria määriä vaatimuksia, sekä tarjota raportointimahdollisuuksia, jotta prosessin etenemistä voi seurata. Työkalun pitää myös olla turvallinen ja tietoturvallinen. (Lauesen 2002: 273–275.)

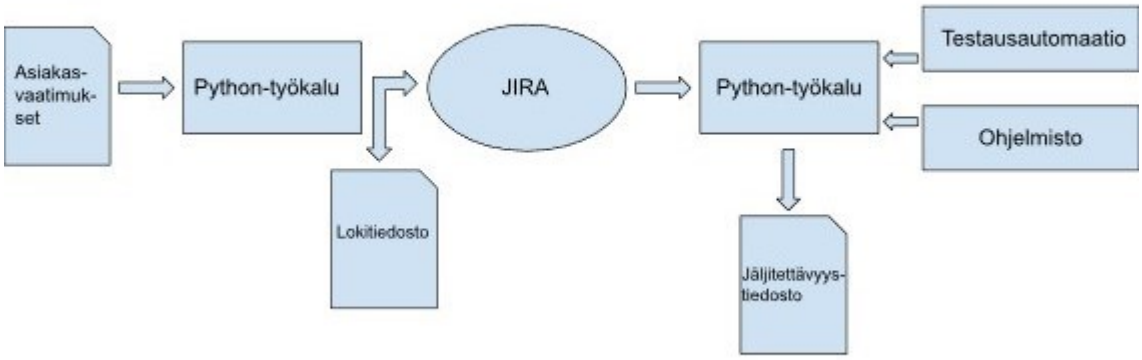
5 Työkalun toteutus

Työkalu toteutetaan projektitiimille, joka on viimeistelemässä nykyistä projektiaan ja siirtyy puolen vuoden päästä toisen projektin pariin. Tavoite on, että uudessa projektissa vaatimustenhallinta toteutetaan uudessa järjestelmässä työkalun avulla.

Ennen varsinaista vaatimustenhallinnan työkalun kehittämistä on tärkeää suorittaa perusteellinen valmisteluprosessi, joka auttaa varmistamaan, että lopputulos vastaa yrityksen tarpeita ja odotuksia.

5.1 Valmisteluprosessi

Työkalun suunnitteleminen aloitettiin kartoittamalla sekä projektitiimin että yrityksen tarpeita ja odotuksia. Kartoitus aloitettiin palavereilla yrityksen työntekijöiden kanssa, jotka olivat avainasemassa vaatimustenhallinnan kehittämisessä. Tapaamisissa selvisi, millaisia ominaisuuksia ja toimintoja työkalulta odotetaan ja millaisia vaatimuksia sen tulee täyttää. Lisäksi teknisinä yksityiskohtina selvisi, että työkalu tulisi toteuttaa Python-ohjelmointikielellä ja järjestelmänä toimisi Atlassian Jira. Työkalu siirtää vaatimukset järjestelmään Excel-tiedostosta. Alla on havainnointikuva 1, mitä työkalun tulisi tehdä ja millaisiin järjestelmiin se on pystyttävä integroimaan.



Kuva 1. Havainnointikuva työkalun toiminnasta.

Kun toteutustapa oli selvillä, tiedusteltiin, minkä verran asiakkaiden lähettämiä tiedostoja tulisi manuaalisesti käsitellä, jotta nämä saataisiin järjestelmään. Tällä tavoin luotiin ensimmäinen suunnitelma yleisestä Excel-mallista, joka toimisi kaikkien asiakkaiden kanssa. Työkalu haluttiin toteuttaa myös niin, ettei työkalun käyttö itsessään vaadi osaamista.

Teknisten ratkaisujen kartoittaminen toteutuksen tasolla jatkui yhteistyöllä muiden sidosryhmien kanssa. Projektitiimi, jonka projekti tulee toimimaan työkalun pilottina, oli näistä merkittävin. Tämä yhteistyö jatkui aktiivisesti aina vaatimustenhallinnan työkalun kehitysprosessista jatkokehitykseen saakka.

Tässä vaiheessa työkalulle oli määritelty riittävästi vaatimuksia (taulukko 1), joiden pohjalta työkalua oli mahdollista ryhtyä toteuttamaan. Sekä yrityksen että projektitiimien vaatimukset koottiin yhteen ja dokumentointiin. Näiden pohjalta luotiin toteutussuunnitelma ja kehitettiin testaus- ja käyttöönottostrategia.

Taulukko 1. Työkalun vaatimukset.

Työkalun vaatimukset
Työkalu toteutetaan Python-ohjelmointikielellä
Työkalu toimii Jira-järjestelmässä
Työkalu noutaa, päivittää ja lisää vaatimuksia järjestelmässä
Käyttäjä voi kirjautua ja määrittää projektin työkalulla
Työkalu lukee ja käsittelee Excel-tiedostoja
Työkalun toiminnan vaiheet dokumentoidaan lokitiedostoon
Työkalu luo raportin vaatimuksista, jotka ovat uusia, päivitettiin tai epäonnistui

Tämän jälkeen aloitettiin tiedonhaku tehokkaimmista mahdollisista toteutustavoista, kun halutaan käsitellä dataa Jira-järjestelmässä käyttäen Python-ohjelmaa. Tiedonhakuun kuului samankaltaisen toteutuksien opiskelu internetissä, Jira-projektinhallintajärjestelmän käyttöoppaan lukeminen, erilaisten Python-kirjastojen soveltuvuuden tutkiminen sekä yrityksen työntekijöiden kanssa keskustelu.

5.2 Määrittelyt

Työn ensimmäisiin vaiheisiin kuului sekä vaatimusten että vaatimustenhallintajärjestelmän määrittely olemassa olevan tiedon ja työkalun vaatimusten pohjalta. Tämä tuli suunnitella ja toteuttaa huolella, sillä määrittely ohjaisi koko työkalun luontia jatkossa. Työkalun tulisi lukea ja käsitellä vaatimuksia Excel-tiedostosta mahdollisimman yleisellä tasolla ja lisätä vaatimukset vaatimustenhallintajärjestelmään.

Vaatimusten määrittely pohjautui alalla yleisesti käytettyihin standardeihin sekä yrityksen sisäisiin dokumentteihin. Näiden avulla saatiin näkemys, mitä kaikkea vaatimuksesta tulisi dokumentoida. Lisäksi yhteistyössä sidosryhmien kanssa sovittiin, mitä eri asiakkaille ominaisia tietoja ei ole syytä tuoda yrityksen järjestelmään. Kun vaatimustenhallinnan prosessi on yhtenäinen, se mahdollistaa

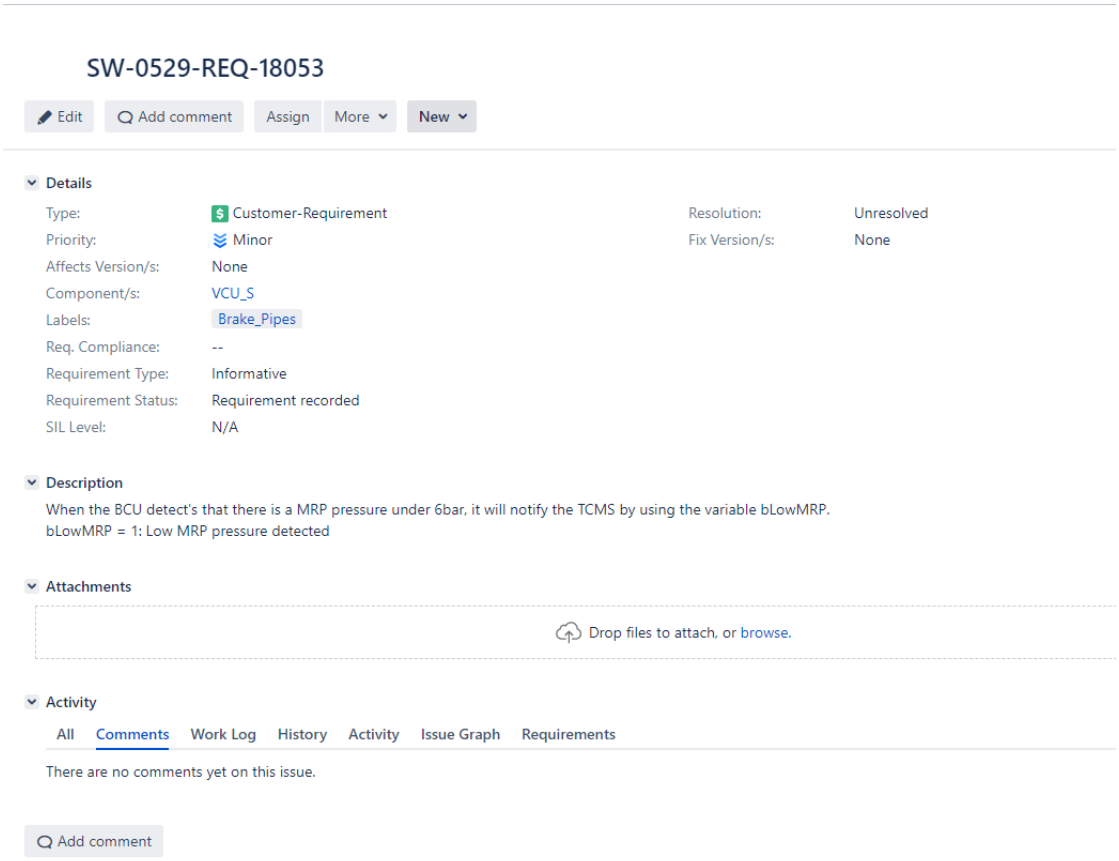
tehokkaan ja läpinäkyvän vaatimustenhallinnan koko yrityksessä. Työkalulta ei myöskään vaadita monimutkaisia ominaisuuksia, jolloin sen toiminta on varmempaa eikä vaadi ylimääräistä osaamista käyttäjältä. Kuvassa 2 on asiakkaan lähettämä Excel-tiedosto.

ID	Type	Description	SIL	Vhl Type	TCMS Subsystem	Status	Comments
SW-0529-REQ-131		**1 GENERAL INFORMATION**	N/A			WF - Work Finished	
SW-0529-REQ-132		**1.1 Purpose**	N/A			WF - Work Finished	
SW-0529-REQ-133	I	This document describes the safety functions requirements for the STCMS system software.	N/A			WF - Work Finished	
SW-0529-REQ-134		**1.2 Scope**	N/A			WF - Work Finished	
SW-0529-REQ-135	I	The scope of this document is the project F073.	N/A			WF - Work Finished	
SW-0529-REQ-142		**1.3.2 Referenced Standards**	N/A			WF - Work Finished	
SW-0529-REQ-144	I	**IEC 61375-2-3:2015+A11:2017** - Electronic railway equipment - Train communication network (TCN) Part 2-3: TCN communication profile. Safety protocol (SDTv2)	N/A			WF - Work Finished	
SW-0529-REQ-145	I	**EN 50159:2010** Railway applications - Communication, signalling and processing systems – Safety related communication in transmission systems.	N/A			WF - Work Finished	
SW-0529-REQ-223		**3.1.2.1.1 Common**	N/A			WF - Work Finished	
SW-0529-REQ-466	I	**Signals read by the MIO_S in the powerhead/locomotive**:	N/A			WF - Work Finished	
SW-0529-REQ-224	I	($DIp_OccupiedCab1$): Cabin ocupy relay (K1) energized in loco and cab-car in cabin A, primary [110 Vdc]	SIL-2			WF - Work Finished	
SW-0529-REQ-225	I	($DIs_OccupiedCab1$): Cabin ocupy relay (K1) energized in loco and cab-car in cabin A, secondary [110 Vdc]	SIL-2			WF - Work Finished	
SW-0529-REQ-218		**3.1.4 Communication Interfaces**	N/A			WF - Work Finished	
SW-0529-REQ-1598	P	The communication through MVB in the powerheads have to be according to the interface control document REQ-139.	SIL-2			WF - Work Finished	
SW-0529-REQ-1744	P	The communication through MVB in the coches have to be according to the interface control document REQ-140.	SIL-2			WF - Work Finished	
SW-0529-REQ-481		**3.2.2.3 General requirements for the VCU_S**	N/A			WF - Work Finished	
SW-0529-REQ-1621		**3.2.2.3.1 Redundancy management**	N/A			WF - Work Finished	13/12/20: Reviewed OK, without changes
SW-0529-REQ-1625	P	The follower device supervises the leader device through life signs sent through ETH point-to-point connection between the VCU_S halves.	SIL-2	Vhl_T01 Vhl_T02	VCU_S	WF - Work Finished	
SW-0529-REQ-1627	P	When the leader life sign is lost on both channels the follower automatically assumes the role as leader and begins to send data for redundant safety applications.	SIL-2			WF - Work Finished	

Kuva 2. Esimerkki asiakkaan lähettämästä vaatimustiedostosta.

Järjestelmän määrittelyn osalta tärkeintä oli luoda ja muokata järjestelmä soveltuvaksi yrityksen vaatimustenhallintaa varten. Tämä tarkoitti, että järjestelmällä olisi valmius dokumentoida ja käsitellä kaikki tiedot, jotka haluttiin järjestelmään tuoda. Vaatimustenhallintajärjestelmään luotiin vaatimusta varten uusi ongelmatyyppi, asiakasvaatimus. Vaatimukset on näin ollen helppo erotella muista projektin ongelmista, ja sekä informaationhaku että dokumentointi onnistuvat helpommin. Lisäksi asiakasvaatimus-ongelmatyypille luotiin useita tietokenttiä,

joihin vaaditut tiedot on mahdollista ohjata. Kuvassa 3 on kuvattuna vaatimus Jira-järjestelmässä.



Kuva 3. Vaatimusnäkymä uudessa järjestelmässä.

5.3 Testaus- ja käyttöönottostrategia

Testausta ja käyttöönottoa varten luotiin oma irrallinen kehitysympäristö, sillä yrityksen Jira-järjestelmä oli aktiivisesti käytössä useissa projekteissa. Tällä tavoin vältyttiin mahdollisilta ongelmilta, kun työkalun testaus tapahtui muualla. Testausta varten luotiin erillinen henkilökohtainen Jira-tunnus ja määriteltiin kehitysympäristö vastaamaan kriittisiltä osin yrityksen ympäristöä.

Työkalun testaaminen suunniteltiin testattavaksi toiminto kerrallaan. Työkalun kehitysvaiheen ajan testaus tullaan suorittamaan erillisessä kehitysympäristössä. Yksittäisen toiminnon toteutus ja testaus on Python-ohjelmointikielen funktioiden myötä tehokas tapa edetä. Tällä tavoin kehitys pysyy seurattavana

ja muutoksia voi tehdä nopeasti. Taulukossa 2 on esitetty irrallisessa kehitysympäristössä toteutettava testaussuunnitelma sisältäen testin järjestysnumeron, toiminnallisuuden ja yhteisesti sovitun valmiin tai hyväksyttävän testin määritelmän (DoD, Definition of Done).

Taulukko 2. Työkalun testaussuunnitelma.

Testitapaus	**Toiminto**	**DoD**
1	Kirjautuminen/yhdistäminen	Käyttöliittymä. Kirjautuminen voidaan toteuttaa useammalla käyttäjällä. Yhdistäminen Jiraan onnistuu.
2	Projektin haku	Projektin lyhenne ja issue count tulostetaan.
3	Ongelman haku	Issuen avain ja kaikki tiedot tulostetaan.
4	Issuen luonti, tiedot ohjelmassa	Issue on luettavissa järjestelmässä.
5	Issuen päivittäminen, tiedot ohjelmassa	Issuen tiedot päivittyvät ja on luettavissa järjestelmässä. Todennetaan päivittyminen tarkistamalla uniikki avain.
6	Excel-tiedoston luku	Tulostaa viiden eri Excel-tiedoston tiedot. Tarvittaessa kirjataan havainnot.
7	Excel-tiedoston käsittely	Ohjaa otsikoiden alla olevat tiedot vastaavan Jira-kentän alle. Muuttaa tyhjät kentät ja lyhenteet Jiraan sopiviksi. Osaa yhdistää olemassa olevan issuen Jira-avaimen Excelin vaatimukseen päivittämistä varten. Tietorakenteet tallentavat vaatimuksen tiedot oikein.
8	Issuen luonti, Excelistä	Issue on luettavissa järjestelmässä.
9	Issuen päivittäminen, Excelistä	Issuen tiedot päivittyvät ja ovat luettavissa järjestelmässä. Todennetaan päivittyminen tarkistamalla uniikki avain.
10	Dokumentointi	Ohjelma luo luettavan tiedoston, josta selviää eri vaiheet, onnistuneiden luontien/päivitysten määrät, sekä listaa epäonnistuneet määrällisesti ja nimellisesti.

Testitapaus	Toiminto	DoD
11	Kuormitustesti	Määrät menevät oikein. Dokumentointi toimii. Ohjelma ei kaadu tai häiritse muuta työskentelyä.
12	Prosessin tehostaminen	Testissä #11 mahdollisesti esiin tulleiden tapausten tehostaminen.
13	Simulaatiotesti	Vaatimukset järjestelmässä määritetyllä tavalla. Täyttää työkalun vaatimukset ja toimii odotetulla tavalla.

Käyttöönottostrategia pohjautui yrityksen toimintatapojen mukaan työkalun toiminnallisuudelle soveltuvaksi. Taulukossa 3 on esitetty työkalun käyttöönottostrategian eri vaiheiden ajankohta ja toteutus.

Taulukko 3. Työkalun käyttöönottostrategia.

Tehtävä	Ajankohta ja toteutus
Ympäristön määrittäminen	Ennen työkalun toteutusta.
Käyttöliittymän suunnittelu	Työkalun ensimmäinen toiminto.
Työkalun kehitysvaiheen dokumentointi	Jatkuva, työkalun kehitysvaiheen yli.
Testaus	Jatkuva, toiminto kerrallaan testaussuunnitelman mukaisesti.
Pilottikäyttöönotto	Kehitysvaiheen jälkeen.
Käyttöopas ja dokumentointi	Ennen työkalun jakamista yrityksen käyttöön.
Koulutus	Työkalun siirtyessä yrityksen käyttöön. Käyttöönotto palaveri.
Tuki ja ylläpito	Jatkuva.

5.4 Työkalun kehittäminen

Työkalun kehittäminen oli mahdollista aloittaa, kun tarvittavat valmistelut oli tehty. Työkalulle osoitetut vaatimukset oli dokumentoitu, tiedonhaku toteutettu, määrittelyt tehty ja sekä toteutussuunnitelma että testaus- ja käyttöönottostrategia laadittu.

Kehitys tapahtui jatkuvan tiedonhaun ohella, sillä yrityksellä ei ollut Jira-asiantuntijaa, joka olisi voinut nopeuttaa prosessia. Tästä huolimatta kehitysvaihe eteni sujuvasti ja aikataulun mukaisesti, sillä projektitiimi, joka toimisi pilottitiiminä työkalulle, tuki ja opasti järjestelmän käytön kanssa.

Työkalun kehittämisessä jouduttiin palaamaan tiettyjen aikaisempien toimintojen kehitykseen useita kertoja. Tähän oli myös varauduttu. Huolimatta siitä, että kehitys ja testaus tapahtui toimintojen osalta samanaikaisesti, simulaatiotesteissä paljastui ongelmia aikaisempien toimintojen osalta. Ongelmia käydään läpi tarkemmin seuraavassa luvussa.

5.5 Työkalun testaaminen

Työkalun testaaminen aloitettiin testaussuunnitelman mukaisesti kehitysympäristössä. Testitapaukset suoritettiin suunnitellussa järjestyksessä. Testauksen aikana jouduttiin kuitenkin poikkeamaan suunnitelman järjestyksestä paremman toiminnallisuuden vuoksi. Luvun 5.3 taulukon 2 kohtaan 7, ”Excel-tiedoston käsittely”, jouduttiin palaamaan useita kertoja.

Ongelmia ilmeni, kun työkalulle suoritettiin simulaatiotestejä. Simulaatiotesteihin haluttiin sisällyttää useita eri Excel-tiedostomuotoja, jotka olivat rakenteellisesti poikkeavia toisistaan. Työkalu ei pystynyt käsittelemään näitä vaaditulla tavalla, joten toiminnallisuutta piti muuttaa. Tämä tarkoitti myös testaussuunnitelman osalta palaamista luvun 5.3 taulukon 2 kohtaan 7, Excel-tiedoston käsittelyn testaamiseen.

Tiedoston käsittelyn ollessa puutteellista suurin ongelma oli päivitystoiminto. Ongelmaa ratkoessa tuli esille, että kehitys- ja testaussuunnitelmasta puuttui merkittävä toiminnallisuus, järjestelmästä tulevan tiedon käsittely. Tämä lisättiin suunnitelmiin jälkikäteen.

Kehitysympäristössä suoritetun testauksen jälkeen oli aika siirtyä testaamaan työkalua yrityksen järjestelmään. Testaaminen aloitettiin projektissa, joka oli tarkoitettu testiympäristöksi. Testaaminen suoritettiin myös yrityksen järjestelmässä testaussuunnitelman mukaan.

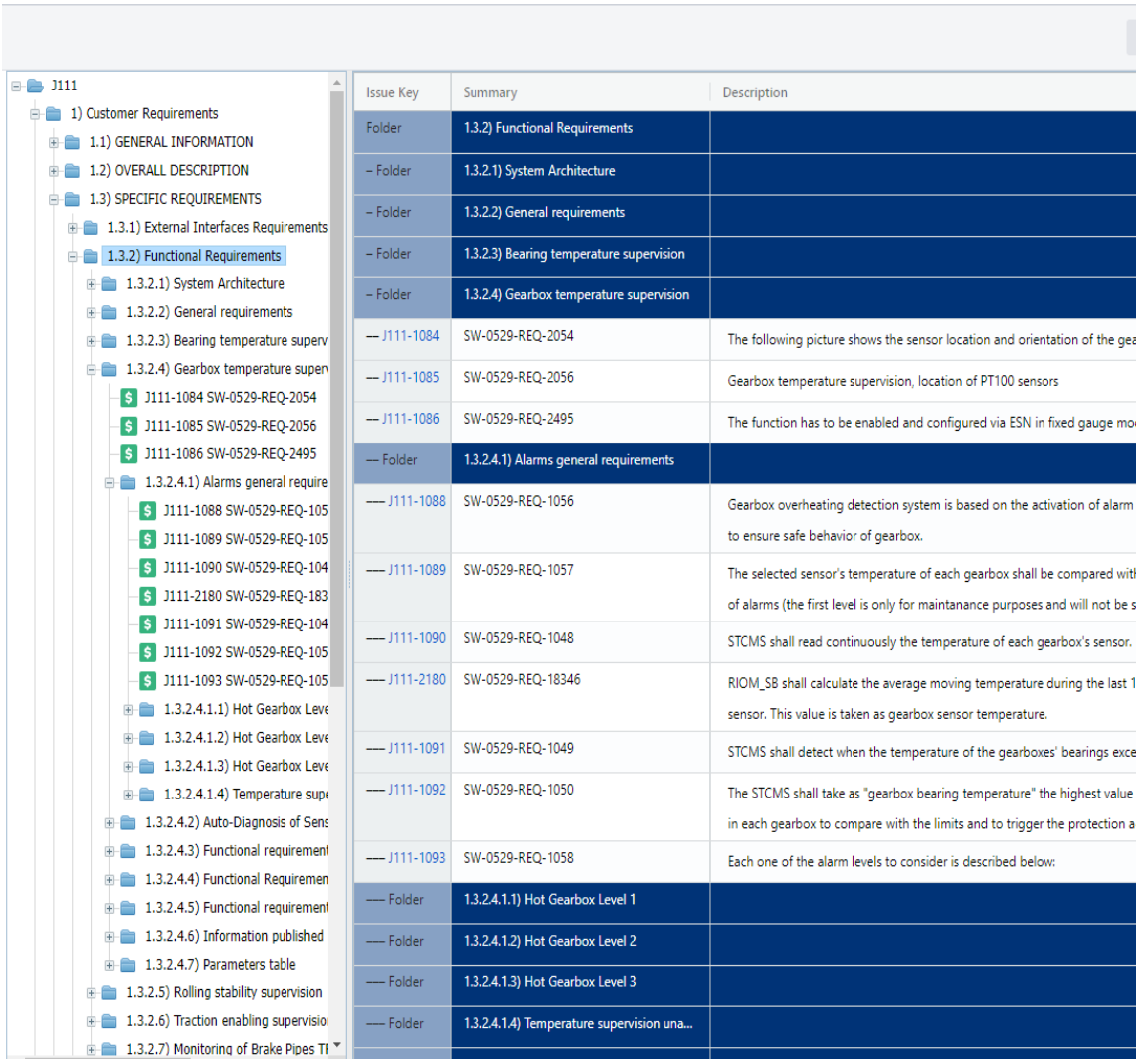
Tämä tarkoitti myös muutoksia työkaluun, sillä Jirassa ei ole mahdollista määritellä itse eri kenttien tunnisteita. Työkalun osalta tämä tarkoittaa tietyn tiedon ohjaamista oikean kentän alle ja ennalta määriteltyjen arvojen tunnistenumeroiden muokkaamista.

Jira on yrityksellä entuudestaan käytössä projektinhallintajärjestelmänä. Tästä syystä järjestelmässä on todella paljon dataa. Iso osa tästä datasta ei koske vaatimuksia tai vaatimustenhallintaa, joten työkalun toimintaa täytyi tehostaa. Tehostaminen toteutettiin muuttamalla järjestelmästä tulevan tiedon hallintaa ja suodatusta työkalussa.

6 Työkalun käyttöönotto ja kehitysmahdollisuudet

Testauksen jälkeen siirryttiin työkalun käyttöönottoon. Käyttöönotossa seurattaisiin aikaisemmin luvun 5.3 kuvatun taulukon 2 mukaista strategiaa.

Käyttöönotto aloitettiin projektitiimin sisällä. Projektitiimillä oli tarve siirtää yli 2000 uuden projektin vaatimusta järjestelmään. Käyttöönottotilanne toimi samalla myös alustavana koulutuksena tiimin työntekijöille. Vaatimukset saatiin siirrettyä onnistuneesti järjestelmään. Projekti käyttää uutta järjestelmää ainoana tapana vaatimustenhallinnassa. Kuvassa 4 on näkymä vaatimustenhallinnasta Jirassa.



Kuva 4. Vaatimushierarkia järjestelmässä.

Tämän jälkeen alkoi prosessin analysointi ja raportointi, jotta työkalua voi jatkokehittää tai sen tuomia etuja hyödyntää muissa käyttökohteissa. Esiin tulevat huomiot saattaisivat myös täydentää työkalun dokumentointia tai käyttöopasta.

Dokumentointi ja käyttöoppaan luominen on tärkeä prosessi. Selkeä, hyvin kirjoitettu käyttöopas on edellytys jokaiselle kehitetylle työkalulle yrityksessä. Työkalun ollessa helposti käytettävä käyttöopas keskittyy enimmäkseen järjestelmän määrittelemiin vaatimuksiin tiedon hallinnasta. Yleisen pohjan malli ja eri kenttien mahdolliset arvot täytyi esittää mahdollisimman selkeästi.

Käyttöoppaan valmistuttua oli aika esitellä työkalu yleisesti yrityksen työntekijöille. Tämä tapahtui yrityksen palaverissa, johon kaikki työntekijät oli kutsuttu. Työkalun toiminnallisuuksista kerrottiin ja käyttötarkoitus avattiin. Lisäksi esittelyn aikana työkalua käytettiin testiprojektissa. Tämä palaveri toimi osittain myös alustavana koulutuksena työkalun käyttöönottoa ja käyttöä varten.

Lisäksi esiteltiin työkalun vaikutusta, kun se on otettu projektissa käyttöön. Pilottitiimin vaatimustenhallinta oli kokonaisuudessaan uudessa järjestelmässä ja päivittäisessä käytössä. Tiimin muut työntekijät kertoivat myös omia mielipiteitään, miten uudistunut vaatimustenhallinta on helpottanut työntekoa. Kommentit olivat positiivisia.

Työkalun käyttöönoton jälkeen sitä muokattiin hieman vastaamaan paremmin tarpeita. Työkaluun lisättiin uusi yrityksen määrittelemä kenttä, jonka avulla vaatimuksen tilaa prosessissa voisi seurata. Tämän avulla voidaan selvittää, mitkä vaatimukset ovat aktiivisia tai toteutettavissa ja mitkä vaatimukset eivät täytä vaatimusmäärittelyn kaikki kohtia.

Tilan seurannan avulla voidaan tehostaa projektin kulkua ja työn parissa työskentelevät ovat ajan tasalla toteuttamiskelpoisista toiminnoista. Tämä vähentää väärinkäsityksiä ja nopeuttaa myös vaatimustenmäärittelyä asiakkaiden kanssa.

Mahdollisiin kehityskohtiin tulevaisuudessa kuuluu työkalun käytön aikaisten vaiheiden parempi dokumentointi, laajemmat mahdollisuudet tila-kentän hallinnassa, jäljitettävyyden automatisointi ja muutoslokin luominen ja tarkistaminen.

Dokumentoinnin parantaminen auttaisi käyttäjää seuraamaan työkalun käyttämisen vaiheita paremmin. Jos työkalun käyttämisessä tai vaatimusten tuonnissa järjestelmään ilmenee ongelmia, on syyn selvittäminen helpompaa hyvän lokin myötä.

Tila-kentän hallinnan laajennus mahdollistaisi toiminto- tai vaatimuskohtaisesti nykytilan räätälöinnin. Tämä olisi merkittävä uudistus, sillä työkaluun lisättiin kentän huomiointi vain uusien vaatimusten kohdalla, jolloin tilana toimii ”Vaatimus dokumentoitu”. Uusi kenttä ja tästä syystä toiminnallisuuden huomioiminen myös työkalussa tuli kehitystyön myöhäisessä vaiheessa, joten toiminnallisuus jäi tästä syystä yksinkertaiseksi.

Muutoslokin luominen ja tarkistaminen on myös yksi kehityskohteista. Työkalu pystyy jo tunnistamaan, kuinka moni vaatimuksista on päivitettyjä. Lisäämällä toiminto, jonka avulla luodaan päivitetyistä vaatimuksista lista tai tiedosto, voitaisiin verrata tätä asiakkaan lähettämään muutoslokiin. Tämän avulla voidaan jälleen kerran varmistua, että kaikki tiedot pitävät paikkansa.

Muutoslokin ohella haluttiin myös parantaa jäljitettävyyttä. Työkalun kehittämiseen käytyjen keskusteluiden edetessä kävi ilmi, että näitä kahta toimintoa ei kannata yhdistää työkaluun. Työkalu toimi tehtävässään hyvin, ja se haluttiin pitää irrallisena muista vaatimuksiin liittyvistä toimenpiteistä. Toimintojen lisääminen voi johtaa liian monimutkaiseen työkaluun ja pahimmassa tapauksessa virheelliseen toimintaan.

7 Jäljitettävyyden automatisointi

Jäljitettävyyden automatisoinnin kehitystarpeisiin kuului muutoslokin tarkistaminen ja vaatimusten seuraaminen ja jäljittäminen sekä testitapauksissa että ohjelmistojen koodissa. Keskusteluiden edetessä muutoslokista kuitenkin luovuttiin, sillä vastaavan tapainen toteutus oli jo olemassa. Tämän toteutuksen muokkaaminen käyttötarkoitukseen sopivaksi oli helpompi ja nopeampi tapa kuin kokonaan uuden toteutuksen luonti.

Asiakasvaatimukset löytyvät entuudestaan merkittynä koodi- ja testitiedostoissa. Nykyisellään jäljitettävyyttä seurataan ja merkitään manuaalisesti. Vaatimukset kirjataan tiedostoon ja lisätään testitapaukset ja kooditiedostot, joissa vaatimus on katettu.

Tämä prosessi halutaan automatisoida. Jäljitettävyys-toiminnon tulisi olla erillinen työkalusta, ja se tulisi suorittaa aina kun testiajo suoritetaan. Testiajo suoritetaan Robot Framework -ympäristössä, joten jäljitettävyyden automatisointi toteutetaan kyseisellä alustalla. Lopputuloksena tulisi olla helposti luettava tiedosto, josta ilmenee vaatimus, testaukseen liittyvät vaatimuksen ominaisuudet, testitapaukset joissa vaatimus on katettu sekä tiedostot, joissa vaatimus on katettu kooditasolla.

7.1 Toteutus

Jäljitettävyyden automatisointia varten luotiin testitapaus Robot Framework -ympäristöön, mikä ajetaan aina ensimmäisenä testinä testiajossa. Testissä käytetään osittain samoja lähestymistapoja kuin vaatimustenhallinnan työkalussa, kun haetaan vaatimukset Jira-järjestelmästä. Testiajojen ollessa automaattisesti käynnistyviä käyttäjän asettamia asetuksia eikä täten myöskään käyttöliittymää tarvita.

Kun vaatimukset on haettu Jirasta, ne suodatetaan ja listataan halutulla tavalla. Tämän jälkeen käydään läpi kaikki testitapaukset, jotka on tarkoitus ajaa

testiajossa. Testitiedoston nimi sekä testin kattamat vaatimukset otetaan talteen. Sama prosessi pienin muokkauksin toteutetaan kooditiedostoille.

Tämän jälkeen voidaan aloittaa tietojen yhdistely. Jirasta haettuun vaatimukseen yhdistetään kaikki testitapaukset ja kooditiedostot, joista vaatimus löytyy. Kun tiedot on yhdistetty, on nämä helppo tuoda luettavaan Excel-tiedostoon. Jos vaatimus löytyy vaadituista testeistä ja kooditiedostoista, lasketaan se katetuksi. Toiminto ilmoittaa prosenttimääräisen kattavuuden testin päätteeksi, mikä helpottaa projektin etenemisen seuraamista.

7.2 Kehityskohdat

Toiminto otettiin projektissa käyttöön jo kehitysvaiheessa. Kun toiminto oli valmis, sen toimivuutta ja tuotoksia seurattiin aluksi satunnaisesti, projektin tilan niin salliessa. Testitapauksien lukumäärien kasvaessa myös katettujen vaatimusten määrä lisääntyi. Tällöin huomattiin, että toiminto ei toimi vaaditulla tavalla.

Ongelmana olivat vaatimukset, joissa oli useampi komponentti. Alun perin vaatimus, jolla on useampi komponentti, listataan useampaan kertaan komponenttien lukumäärän mukaan. Tätä ei ollut otettu huomioon, kun vaatimukseen yhdistettiin testejä ja tiedostoja, joissa vaatimus oli katettu. Lopputuloksena esiintyi kuvan 5 mukaisia virheitä.

	A	B	C	D
1	Requirement	Component	Test coverage	Code coverage
2	REQ-001	X	Y/test1	Z/code.c
3	REQ-001	Y	Y/test1	Z/code.c
4	REQ-001	Z	Y/test1	Z/code.c
5				

Kuva 5. Jäljitettävyystiedoston virheellinen raportointi.

Kuvassa 5 on vaatimus, jolla on komponentit X, Y ja Z. Ohjelma laskee vaatimuksen virheellisesti katetuksi, sillä vaatimus löytyy sekä testeistä että koodista

ottamatta kantaa komponenttiin. Rivillä kaksi esiintyvä vaatimus koskee komponenttia X, on testattu komponentin Y osalta ja on koodissa Z komponentin osalta.

Tämä korjattiin uudistamalla tiedonkäsittelyä. Ohjelma suodattaa testitapaukset ja kooditiedostot nimen, tunnisteen tai kansion nimen perusteella ja yhdistää ne oikeaan vaatimukseen oikean komponentin mukaan. Kuvassa 6 on korjattu lopputulos.

	A	B	C	D
1	**Requirement**	**Component**	**Test coverage**	**Code coverage**
2	REQ-001	X	X/test1 X/test2	
3	**REQ-001**	**Y**	**Y/test1**	**Y/code.c**
4	REQ-001	Z		Z/code.c

Kuva 6. Korjattu näkymä jäljitettävyystiedostossa.

Ohjelma ei myöskään laske vaatimusta katetuksi, sillä ohjelma katsoo kaikki komponentit läpi. Kuvan 6 vaatimus REQ-001 on katettu vain yhden komponentin osalta.

8 Toteutuksen seuranta

Työn edistymistä seurattiin palaverien avulla. Palavereja pidettiin eri sidosryhmien ja henkilöiden kanssa.

Opinnäytetyön edistymistä seurattiin noin kahden viikon välein palaverilla, johon osallistui oma esimieheni. Näissä palavereissa ei keskitytty itse työn toteutukseen, vaan seurattiin ja tuettiin kirjoitusprosessia.

Työkalun toteutusta ja toimintoja seurattiin noin kuukauden välein pidettävien palaverien avulla. Palavereihin osallistui esimiehiä ja yrityksen toimihenkilöitä. Näin varmistettiin, että työkalun suunta on oikea ja tälle asetetut vaatimukset täyttyvät.

Viikoittain pidettiin myös pienempiä tapaamisia oman projektitiimin kanssa. Näissä keskeisin asia oli työn teknisten ratkaisujen pohdinta ja työn vaiheen raportointi eteenpäin.

Vaatimustenhallinnan automatisointi työkalun avulla oli yritykselle uusi tapa toteuttaa vaatimustenhallintaa. Tämän vuoksi oli usein uusia tilanteita, joihin ratkaisuja piti etsiä pohtimalla ja kokeilemalla.

9 Johtopäätökset ja pohdinta

Työn tavoitteena oli vaatimustenhallinnan automatisointi yrityksen siirtyessä käyttämään uutta järjestelmää vaatimustenhallinnassa. Lisäksi vaatimusten jäljitettävyyttä tuli tehostaa ja automatisoida raportointi. Toteutustapa oli yritykselle uusi, minkä vuoksi työn kestoa oli vaikea arvioida. Yritys toivoi, että työkalu olisi osittain valmis kuuden kuukauden sisällä, jolloin arvioitiin suurin tarve uuden projektin alkaessa.

Työtä varten kerättiin tietoa, miten vaatimuksia ja vaatimustenhallintaa tulee käsitellä, mitkä ovat yrityksen tarpeet ja mikä on täsmällisin ja tehokkain tapa toteuttaa työ. Tietoa kerättiin standardeista, kirjallisuudesta, internetistä ja yrityksen työntekijöiltä. Uutena toteutustapana työhön kuului paljon ongelmatilanteita, joiden ratkaisua täytyi etsiä pohtimalla ja kokeilemalla.

Työn alkaessa suurimmaksi ongelmaksi nousi se, ettei yrityksellä ollut tarjota koulutusta uuden järjestelmän käyttöön ja kehitykseen. Tämän lisäksi ei ollut erillistä kehitysympäristöä, jossa työkalua olisi voinut testata. Nämä aiheuttivat viivästyksiä itseopiskelun muodossa. Kyseisiin ongelmakohtiin on suunnitteilla parannuksia tulevaisuudessa.

Erityisesti esille nousi nykyisen vaatimustenhallinnan vaatima työmäärä ja hankaluus. Kehittäjät ja testaajat eivät päässeet suoraan käsiksi vaatimuksiin, mikä vaikeutti ja hidasti toteutusta. Lisäksi kun vaatimukset muuttuivat, mikä on kohtalaisen yleistä, niiden tarkistaminen ja päivittäminen oli aikaa vievää ja vaati tekijältä paljon manuaalista työtä. Haluan myös nostaa esiin, että kansainvälisesti toimivan yrityksen tulee huomioida mahdolliset kulttuuriset erot. Vaatimus saattaa olla ajateltu eri tavoin osapuolten välillä. Näin ollen myös vaatimusten selvittämiseen tulee olla varattuna aikaa ja mahdollisesti tarjota yhteneväinen alusta, jossa vaatimukset on dokumentoitu ja niihin voi kommentoida huomioita. Tämä tosin pätee kaikkiin projekteihin asiakkaasta riippumatta.

Uusi järjestelmä ja työkalu ovat olleet merkittävässä roolissa näiden ongelmien poistamisessa. Jokaisella työntekijällä on nyt mahdollisuus lisätä ja päivittää

vaatimuksia. Tämän lisäksi prosessi on nopeampi ja helpommin seurattavissa. Projektin jäsenillä on jatkuva ajantasainen pääsy vaatimuksiin, mikä edesauttaa kehitystyötä ja testaamista. Tulevaisuudessa tarkoitus on, että myös asiakkaan on mahdollista päästä yrityksen järjestelmään tarkastelemaan ja kommentoimaan vaatimuksia, mikä auttaa väärinymmärrysten ehkäisemistä. Vaatimusten tuominen järjestelmään työkalulla on mielestäni tehokkaampi tapa kuin työntekijä, joka toimisi vain tässä tehtävässä, vaikkakin useammassa projektissa. Tosin ongelmatilanteissa tai muissa suurissa muutoksissa työkalu ei pysty reagoimaan ja vaatisi suuremman määrän selvittelyä.

Koska työkalua on mahdollista käyttää jokainen, eikä sen käyttöä järjestelmässä ole rajoitettu, voi sillä myös saada aikaan ikäviä muutoksia. Työkalu on tosin suunniteltu niin, että se toimii vain annetussa projektissa eikä sillä voi poistaa vaatimuksia. Lisäksi päivittäminen tapahtuu aina järjestelmän tunnisteen pohjalta. Kuitenkin voidaan esimerkiksi lisätä toista tuhatta vaatimusta väärään projektiin tai päivittää vaatimukset vanhempaan versioon. Kaikkia ongelmatilanteita ei ole kuitenkaan tiedostettu ja testattu, joten täyttä varmuutta ei ole.

Työkalu otettiin ensimmäiseksi käyttöön projektissa, jonka jäsenenä itsekin nykyään työskentelen. Se on kerännyt kiitosta ja mielenkiintoa toimihenkilöiltä ja muista projektitiimeistä. Projektit kestävät kuitenkin yleensä useamman vuoden, joten työkalu ei ole vielä laajasti käytössä. Vaatimusten päivittäminen onnistuu muutamassa minuutissa, jolloin niiden arviointiprosessin voi aloittaa heti eikä arvioinnin tarvitse tapahtua sille erikseen varattuun aikaan. Tämä on merkittävä ajallinen etu vanhaan verrattuna, jolloin tiimi kerääntyi yhteen ilman selvää mahdollisuutta tutkia vaatimuksia etukäteen.

Työ täytti kaikki sille asetetut vaatimukset. Vaatimukset tosin olivat todella yleisluontoisia eivätkä kovin spesifejä. Monen asian kohdalla toteutustapa oli määrittelemätön, jolloin yhdessä pohdittiin sille toimivin ratkaisu. Myös aikataulullisesti työ valmistui määräajassa, juuri ennen uuden projektin alkamista.

Yrityksen tuotteen elinkaari voi kuitenkin olla esimerkiksi 30 vuotta, joten on epätodennäköistä, että järjestelmä ja työkalut nykyisellään seuraisivat tuotetta

sen elinkaaren ajan. Vaatimustenhallintaa voisi kehittää niin, että luodaan ratkaisu ja järjestelmä, jota käyttävät sekä asiakas- että kehittäjäyritys yhdessä. Tällöin vaatimukset olisivat reaaliajassa dokumentoituina, niiden määrittely ja arviointi tapahtuisivat yhdessä asiakkaan kanssa ja vaatimustenhallinnan ympärille ei tarvitsisi rakentaa työkaluja tai määritellä tiedostoformaatteja. Tällaisen toteutus tulisi tosin olla integroituna järjestelmään, joka olisi yleisesti ja laajasti käytössä maailmalla, eikä se välttämättä siltikään olisi ratkaisu jokaisen asiakkaan kanssa.

Lähteet

Egeland, Brad. 2019. Poor Requirements Can Triple The Length Of The Project. Verkkoaineisto. BA Times. <https://www.batimes.com/articles/poor-requirements-can-triple-the-length-of-the-project/>. 25.2.2019. Luettu 28.2.2023.

Lauesen, Soren. 2002. Software Requirements: Styles & Techniques. Glasgow: Addison-Wesley Professional.

Paakki, Jukka. 2011. Ohjelmistojen vaatimusmäärittely. Luentomoniste. Helsingin yliopisto.

Pohl, Klaus & Rupp, Chris. 2015. Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level. Santa Barbara: Rocky Nook.

Sommerville, Ian. 2011. Software Engineering. Ann Arbor: Pearson.

Wiegers, Karl & Beatty, Joy. 2013. Software Requirements. Sebastopol: Microsoft Press.

Wundenberg, Sven-Michael. 2015. Requirement Engineering for Knowledge-Intensive Processes. E-kirja. SpringerGabler.

Withall, Stephen. 2007. Software Requirement Patterns. E-kirja. Microsoft Press.

Young, Ralph Rowland. 2004. The Requirements Engineering Handbook. E-kirja. Artech House.