Lappeenrannan–Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma

Kandidaatintyö

Maija Heiskanen

KOMPONENTTIKIRJASTON RAKENTAMISEN KANNATTAVUUS KOMPONENTTIPOHJAISEN KÄYTTÖLIITTYMÄKEHITYKSEN TYÖKALUKSI

Työn tarkastaja: Erno Vanhala

Työn ohjaaja: Erno Vanhala

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma
Maija Heiskanen

Komponenttikirjaston rakentamisen kannattavuus komponenttipohjaisen käyttöliittymäkehityksen työkaluksi

Kandidaatintyö 2020

45 sivua, 5 kuvaa, 4 taulukkoa, 1 liite
Työn tarkastajat: Erno Vanhala

Hakusanat: komponenttikirjasto, komponenttipohjainen ohjelmistokehitys
Keywords: component library, component-based user interface development

Tässä kandidaatintyössä tutkitaan, kannattaako CASE-yrityksen kehitystiimin rakentaa olemassa olevista komponenteistaan komponenttikirjasto komponenttipohjaisen käyttöliittymäkehityksen työkaluksi. Tutkimus toteutettiin todentamalla komponenttikirjastotyökalun yhteensopivuus kehitettävän tuotteen kanssa ja mittaamalla kyselyllä kehitystiimin kokemuksia kehityksessä esiintyvistä haasteista sekä komponenttikirjaston mahdollisuuksista auttaa niiden ratkaisemisessa. Työkalu todettiin sopivaksi tuotteen kanssa. Kyselyn perusteella kehityksen haasteina ovat dokumentaation vähyys sekä vaikeus tarkastella olemassa olevia komponentteja ja käyttää niitä oikein. Kyselyn perusteella käyttöliittymäkehittäjät eivät kokeneet heillä olevan tarpeeksi aikaa komponenttikirjaston rakentamiseksi. Yleisesti komponenttikirjasto koettiin kuitenkin tarpeelliseksi työkaluksi ja se otettaisiin mielellään käyttöön. Tulosten perusteella kehitystiimin kannattaa kokeilla komponenttikirjaston käyttämistä. Komponenttikirjaston hyödyllisyyden ehtona on, että sen ylläpitäminen saadaan luonnolliseksi osaksi kehitysprosessia.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Software Engineering
Maija Heiskanen

Profitability of building a component library as a tool in component-based user interface development

Bachelor's Thesis 2020

45 pages, 5 figures, 4 tables, 1 appendix
Examiners: Erno Vanhala

Keywords: component library, component-based user interface development

This bachelor's thesis studies whether a CASE company's development team would benefit from building a component library from their components and using it as a tool in component-based user interface development. The research was carried out by verifying the compatibility of the component library tool with the product, and by measuring the challenges faced by the development team and the component library's capability to solve them. The tool was found to be compatible with the product. According to the survey, the challenges of development are lack of documentation, difficulty of viewing existing components and using them. Based on the survey, UI developers do not feel they have enough time to build a component library. In general, however, the component library was seen as a necessary tool and the team would happily start using it. Based on the results, the development team should try out using the component library. In order to benefit from a component library, its maintenance needs to be a natural part of the development process.

ALKUSANAT

Kiitos Visma Solutions Oy ja koko Severan R&D-tiimi. Erityiskiitos Juhanalle, joka tuki suuresti teknisen osuuden toteutuksessa ja kannusti koko projektin ajan.
Kiitos ohjaajalleni Ernolle kaikesta ohjauksesta ja kannustuksesta.
Kiitos myös Nikulle, joka jaksoi kuunnella kandihuoliani ja -iloja koko syksyn.
Tätä työtä oli miellyttävää ja kiinnostavaa tehdä ja kiitos siitä kuuluu edellä mainituille tahoille.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

API	Application Programming Interface
BA	Business Analyst
CASE	Tapaus
ERP	Enterprise Resource Planning
SaaS	Software as a Service
SCRUM	Projektinhallinnan viitekehys
UI	User Interface
UX	User Experience
Y2K	Year 2 kilo, vuosi 2000

1 JOHDANTO

Kun uudelleenkäytettävien komponenttien määrät koodivarastoissa (*repository*) kasvavat, muuttuu sopivan komponentin löytäminen työläämmäksi (Casanova et al., 2003; Grundy, 2000; Han et al., 2011; Niranjan & Rao, 2010; Schmidt et al., 2010). Ketterässä kehityksessä dokumentaatiota tehdään vain harvoin (Voigt et al., 2016), vaikka se on tärkeä osa projektin onnistumista ja ylläpidettävyyttä (Prause & Durdik, 2012). Kun komponenttien dokumentaatio on vähäistä, syntyy vaikeuksia käyttää niitä oikein (Grundy, 2000).

Työ tehdään CASE-yrityksen (tapausyrityksen) yhdelle tuotekehitystiimille, joka harkitsee komponenttikirjaston rakentamista suuren SaaS-tuotteen (Software as a Service) kehityksen työkaluksi. CASE-yrityksellä on useita eri tuotteita, joiden tuotekehitystiimit ovat yksittäisiä ja itsenäisiä kehitystiimejä. Komponenttikirjastoa ei siis välttämättä pystytä laajentamaan koko yrityksen kattavaksi, johtuen eroista tuotteissa käytetyissä teknologioissa. CASE-kehitystiimi koostuu tällä hetkellä noin 30 työntekijästä, joiden työnkuvat ovat jakautuneet seuraavasti: 5 käyttöliittymäkehittäjää, 9 backend-kehittäjää, 2 mobiilikehittäjää, 2–3 UX-designeria (User Experience Designer, käyttäjäkokemussuunnittelija), 2 liiketoiminta-analyytikkoa ja noin 10 muuta työntekijää, jotka vastaavat esimerkiksi kehitysympäristöön liittyvistä toiminnoista sekä laadunvarmistuksesta.

CASE-kehitystiimissä käytetään ketterän kehityksen menetelmiä (SCRUM). Tuotetta on kehitetty jo kymmeniä vuosia ja uusinta versiota käyttöliittymästä noin kuusi vuotta. Käyttöliittymä on rakennettu käyttäen Infernoa (*InfernoJS*), Flux-arkkitehtuuria ja AdobeXD:tä. Käyttöliittymäkehitys on komponenttipohjaista ja valmiita komponentteja löytyy projektista jo satoja. Niiden dokumentaatio on kuitenkin olematonta ja löytäminen sekä käyttäminen alkaa olla haastavaa. Myös käyttöliittymän yhdenmukaisuus on alkanut kärsiä, kun samasta komponentista on tehty lukuisia eri versioita, eikä UX-suunnittelijoilla ole ollut helppoa tapaa tarkastella kaikkia olemassa olevia komponentteja systemaattisesti.

1.1 Tavoitteet ja rajaukset

Tämän kandidaatintyön tavoitteena on tutkia, ratkaiseeko käyttöliittymäkomponenteista rakennettu komponenttikirjasto käyttöliittymäkehityksessä yleisesti ilmaantuvia ongelmia, joita ovat esimerkiksi dokumentaation puute, designin yhdenmukaisuuden rikkoutuminen ja vaikeudet uudelleenkäyttää komponentteja joissakin tilanteissa. Työssä toteutetaan komponenttikirjasto käyttäen avoimenlähdekoodin työkalua ja siirtämällä 3–5 olemassa olevaa komponenttia komponenttikirjastoon. Komponentteja muokataan tarvittavilta osin ja niille luodaan tarvittava keinotekoinen data. Tässä työssä ei tutkita vaihtoehtoisia tapoja kehityksessä ilmaantuvien ongelmien ratkaisemiseksi.

Työ sisältää kaksi tutkimuskysymystä:

- **Q1:** Millaisia ongelmia käyttöliittymäkomponenttien siirtäminen komponenttikirjastoon voi ratkaista yksittäisen SaaS-tuotteen kehityksessä?
- **Q2:** Kuinka kannattavaa käyttöliittymäkirjaston luominen ja ylläpitäminen vain yksittäiselle SaaS-tuotteelle on?

1.2 Työn rakenne

Luvussa 2 kerrotaan komponenttipohjaisesta ohjelmistokehityksestä sekä komponenttikirjastoista. Luvussa 3 kuvataan käytettävä tutkimusmenetelmä. Luvussa 4 käsitellään komponenttikirjaston prototyypin implementoimista olemassa olevaan projektiin. Luvussa 5 käsitellään kyselyn tulokset. Luvussa 6 tehdään tulosten pohjalta johtopäätökset komponenttikirjaston kannattavuudesta. Luvussa 7 kootaan tämä työ ja sen tulokset tiivistelmäksi.

2 KOMPONENTTIKIRJASTOT KOMPONENTTIPOHJAISESSA OHJELMISTOKEHITYKSESSÄ

Tässä luvussa kerrotaan, mitä komponenttipohjainen ohjelmistokehitys on sekä millaisia hyötyjä ja haasteita siihen liittyy. Luvussa kerrotaan myös, mitä komponenttikirjastot ovat, millaisia hyötyjä ja haittoja niillä on, sekä raportoidaan aikaisempia havaintoja niiden käytöstä ohjelmistotuotannossa.

2.1 Komponenttipohjainen ohjelmistokehitys

Komponenttipohjainen arkkitehtuuri (*component-based architechture*) on nykyisin yksi yleisimmin käytetyistä ohjelmistoarkkitehtuureista (Dong et al., 2013, p. 1; Jha & Mishra, 2019). Komponenttipohjainen sovellus koostuu komponenteista. Komponentti on koodikokonaisuus, joka on eristetty kontekstistaan siten, että sitä voidaan uudelleenkäyttää sellaisenaan eri ohjelmistosovelluksissa (Omer et al., 2019). Komponentti on hyvin määritelty yksikkö ohjelmistoa, jolla on julkinen rajapinta ja jota voidaan yhdistellä muiden komponenttien kanssa suurempien komponenttien luomiseksi (Niranjan & Rao, 2010).

2.1.1 Komponenttipohjaisen ohjelmistokehityksen hyödyt

Komponenttipohjainen arkkitehtuuri mahdollistaa ohjelmiston uudelleenkäytön (*software reuse*) (Krueger, 1992). Ohjelmiston uudelleenkäyttämistä on olemassa olevien ohjelmakomponenttien käyttäminen uuden ohjelmiston rakentamiseksi (Frakes & Kyo Kang, 2005). Olemassa olevasta ohjelmasta uudelleenkäytetään yleisimmin lähdekoodia, mutta myös designia, vaatimusmäärittelmiä, arkkitehtuuria ja dokumentaatiota voidaan uudelleenkäyttää (Niranjan & Rao, 2010).

Uudelleenkäytöllä pyritään vähentämään ohjelmiston tuotantokustannuksia ja tuotantoon kuluvaa aikaa sekä kasvattamaan ohjelmiston tuottavuutta, ylläpidettävyyttä, siirrettävyyttä ja luotettavuutta (AL-Badareen et al., 2011). Uudelleenkäyttö parantaa myös ohjelmiston

laatua (Niranjan & Rao, 2010). Nykyään ohjelmistoja on tuotettava aiempaa nopeammin ja pienemmillä kustannuksilla (Jha & Mishra, 2019).

2.1.2 Komponenttipohjaisen ohjelmistokehityksen haasteet

Vaikka uudelleenkäyttö on laajasti tunnettu käsite ohjelmistotuotannossa, uudelleenkäyttö ei ole systemaattista ja siten niin tehokasta kuin se voisi olla. (AL-Badareen et al., 2011; Maccanti et al., 2016). Useat tutkimukset ovat kartoittaneet uudelleenkäytettäviin komponentteihin liittyviä ongelmia.

Kun uudelleenkäytettävien komponenttien määrät koodivarastoissa (*repository*) kasvavat, muuttuu sopivan komponentin löytäminen työläämmäksi (Casanova et al., 2003; Grundy, 2000; Han et al., 2011; Niranjan & Rao, 2010; Schmidt et al., 2010). Suurin ongelma komponenttien uudelleen käyttämisessä on sopivien komponenttien löytäminen, sillä tietoa ohjelmistoprojekteista ei ole yleensä organisoitu, vaan sitä on koostettu sekalaisesti tekstidokumenteista, kuvista, kaavioista, kaavoista ja lähdekoodista. (Maccanti et al., 2016). Ketterässä kehityksessä dokumentaatiota tehdään vain harvoin (Voigt et al., 2016), vaikka se on tärkeä osa projektin onnistumista ja ylläpidettävyyttä (Prause & Durdik, 2012). Kun komponenttien dokumentaatio on vähäistä, syntyy vaikeuksia käyttää niitä oikein (Casanova et al., 2003; Grundy, 2000). Dokumentaatio saattaa olla myös jälkeenjäänyttä, mikäli komponenttiin tehtyjä muutoksia ei päivitetä dokumentaatioon (Kalia & Sood, 2017).

Kun uudelleenkäytettävään komponenttiin tehdään muutoksia, muutokset vaikuttavat komponenttiin jokaisessa paikassa, jossa sitä käytetään. Näiden muutosten hallitseminen ja muutosten seurausten ennustaminen on vaikeaa, koska jokaisessa projektissa komponenttia voidaan käyttää erilaisessa kontekstissa. (Kalia & Sood, 2017)

Uudelleenkäytettävästä komponentista voi olla useita eri versioita. Jos vanhempaa versiota oleva komponentti tahdotaan päivittää uudempaan versioon, voi ilmetä ongelmia yhteensopivuuden kanssa. Komponentin käyttöpaikan konteksti voi tukea vanhempaa versiota komponentista, mutta ei uudempaa versiota. Komponentin käyttöpaikassa on myös pidettävä kirjaa siitä, mitä versiota komponentista käytetään. Eri versioiden muutosten

hallinta on vaikeaa ja aiheuttaa ongelmia (Casanova et al., 2003), jos eri versioita ja niiden eroja ei ole dokumentoitu kunnolla. (Kalia & Sood, 2017)

On harvinaista, että uudelleenkäytettävän komponentin lähdekoodi voidaan jäljittää sen vaatimusmäärittelyyn ja design-suunnitelmaan. Mahdollisten virheiden ja bugien jäljittäminen on vaikeaa, koska se vaatii lähdekoodin toiminnan ymmärtämistä ja uudelleenkäytettävän komponentin lähdekoodin on kirjoittanut joku toinen. Tilannetta pahentaa myös, jos lähdekoodissa ei ole asianmukaisia kommentteja eikä lähdekoodin alkuperäiseen tekijään ole mahdollisuutta ottaa yhteyttä. (Kalia & Sood, 2017)

Vanhentunut lähdekoodi (*legacy code*) voi aiheuttaa ongelmia tilanteissa, joissa sen ei ole suunniteltu olevan käytössä kuin tietyn aikaa. Lähdekoodin käyttämisessä suunnitellun aikajakson jälkeen voi ilmetä ongelmia, joihin ei ollut etukäteen varauduttu. Yksi esimerkki tällaisesta on Y2K-ongelma (Year 2 Kilo, vuosi 2000). Kun 1900-luvulta siirryttiin 2000-luvulle, monien vanhojen järjestelmien, joiden ei ollut tarkoitus olla käytössä enää 2000-luvulla, kanssa kohdattiin ongelmia päivämäärien siirtyessä seuraavalle vuosituhannelle. Vanhat järjestelmät on usein rakennettu teknologioilla, jotka olivat rakentamisen aikaan yleisesti käytössä ja suosittuja. Vanhoja teknologioita käyttävän järjestelmän ylläpitäminen vaikuttaa olevan vaikeaa uusilla teknologioilla. (Kalia & Sood, 2017)

2.1.3 Uudelleenkäytön haasteet käyttöliittymäkehityksessä

Käyttöliittymäkomponenttien uudelleenkäyttämiseen liittyy yleisten ongelmien lisäksi myös erityisesti niille ominaisia haasteita ylläpitoon ja uudelleenkäytettävyyteen liittyen. Suuri osa ongelmista liittyy vaikeuteen olla sisällyttämättä perustason komponentteihin bisneslogiikkaa (*business logic*) (Omer et al., 2019). Bisneslogiikalla tarkoitetaan tuotteen liiketoiminnallista kerrosta.

Käyttöliittymässä vain komponentin itsensä tulisi kyetä vaikuttamaan suoraan, ilman rajapintaa, omaan tilaansa (*state*). Bisneslogiikan ja muiden komponenttien suora vaikutus komponentin tilaan vaikeuttaa tilan muutokseen johtavan toiminnan jäljittämistä ja aiheuttaa rakenteetonta ja huonosti ylläpidettävää koodia eli niin kutsuttua ”spagettikoodia”. Myös

käyttöliittymäkomponentin uudelleenkäyttäminen vaikeutuu, jos bisneslogiikka on sisällytetty komponenttiin, sillä se johtaa lähdekoodin kahdentumiseen, koska jokaista bisneslogiikkakomponenttiparia varten on luotava uusi komponentti. Lähdekoodin määrän kasvaessa sen ylläpitäminen vaikeutuu. (Omer et al., 2019)

Dokumentaatio on erittäin tärkeä osa myös komponenttipohjaista käyttöliittymäkehitystä (Jha, 2016; Omer et al., 2019). Dokumentaatio antaa komponentin kehittäjälle mahdollisuuden kuvata, miten komponenttia ja sen rajapintoja tulisi käyttää, ilman että komponenttia käyttävä kehittäjä joutuu lukemaan lähdekoodia läpi. Ilman dokumentaatiota riski komponenttien käyttämiseen väärin kasvaa, koska komponentin käyttäjän on vaikea ymmärtää, kuinka komponentti toimii. Komponentin käyttäminen tarkoitetusta poikkeavalla tavalla johtaa erinäisten oikoteiden käyttämiseen lähdekoodissa (*hacky-code*), joita on mahdotonta ylläpitää. (Omer et al., 2019)

Keskenään samankaltaiset yleiset lähdekoodiyksiköt tulisi pitää samassa paikassa irrallaan bisneslogiikasta. Lähdekoodiyksiköiden sekoittaminen keskenään heikentää lähdekoodin luettavuutta ja ylläpidettävyyttä. Se voi myös vaikeuttaa lähdekoodin uudelleenkäyttöä, jos geneeriseen yleiskäytölliseen lähdekoodiin on yhdistetty bisneslogiikkaa. (Omer et al., 2019)

2.2 Komponenttikirjastot

Ohjelmiston uudelleenkäytön yksi isoimmista ongelmista on vaikeus löytää sopiva komponentti ja ottaa se käyttöön (Niranjan & Rao, 2010), ja siksi ohjelmistoprojektiin liittyvä tieto tulisi pystyä järjestelemään siten, että se on helposti saavutettavissa ja haettavissa (Maccanti et al., 2016). Komponentin hakua ja käyttöönottoa koskevaa ongelmaa varten on kyettävä organisoimaan ohjelmistokomponentit kokoelmiin ja hakemaan kokoelmista komponentteja nopeasti, jotta potentiaaliset vaihtoehdot saadaan huomioitua (Niranjan & Rao, 2010). Komponentteja tulisi tarvittaessa myös pystyä muokkaamaan (Niranjan & Rao, 2010). Tiedon organisoimiseksi tarvittavalla tavalla on useita eri vaihtoehtoja, joista komponenttikirjasto on yksi tapa (Frakes & Kyo Kang, 2005).

Komponenttikirjasto on komponenttitietokanta, joka sisältää komponentit sekä niiden dokumentaation eli määrittelyn käyttötarkoituksista ja ohjeet niiden käyttämiseen (Irie & Tanaka, 2002), sekä mahdollistaa käyttäjien etsiä komponentteja ja esittää ne käyttäjälle (Frakes & Kyo Kang, 2005). Komponenttikirjasto sisältää tietyn yrityksen tietyillä teknologioilla tuotetut komponentit, ja se voi olla osana suurempaa design-järjestelmää (*design system*), joka yhdistää varsinaiset toteutukset ja UX-designit (User Experience, käyttäjäkokemus) yhdeksi kokonaisuudeksi (Ruissalo, 2018). Komponenttikirjastoja on kahta tyyppiä: yritys voi joko tehdä itse oman komponenttikirjastonsa tai käyttää toisen yrityksen tarjoamaa komponenttikirjastoa (Jha & Mishra, 2019).

2.2.1 Mahdolliset hyödyt

Suunnittelutyökalut, kokoonpanonhallintatyökalut ja kuvalliset kuvaukset komponentin käyttötarkoituksesta ovat yleensä hyödyllisiä uudelleenkäytettävien komponenttien käyttämisessä ja ylläpidossa (Kalia & Sood, 2017). Komponenttikirjasto lisää komponenttien uudelleenkäyttöä ja helpottaa olemassa olevien komponenttien yhdistelemistä pienemmistä palasista isommiksi kokonaisuuksiksi (engl. *composition*) (Casanova et al., 2003). Komponenttikirjaston sisältämät kuvaukset komponenteista sekä niiden käyttötarkoituksista ja esimerkit auttavat komponenttikirjaston käyttäjää valitsemaan tarvittavat komponentit lyhyemmässä ajassa (Casanova et al., 2003; Hai-mei & Min, 2012).

Hyvin suunniteltujen komponenttien uudelleenkäyttäminen nopeuttaa ohjelmiston kehitystä (He et al., 2009). Toisaalta komponenttikirjastosta saatavaan hyötyyn vaikuttaa kehitystiimin asenne sen käyttöä kohtaan; vain jos kirjastoa käytetään aktiivisesti ja pitkäjänteisesti oikein, tuottavuus ja käyttöliittymän yhdenmukaisuus kasvavat (Derleth, 2018).

Käyttöliittymäkomponenttikirjastojen hyödyistä on vain jonkin verran tieteellistä tutkimustietoa, mutta sen lisäksi niistä on saatavilla myös epätieteellistä tietoa, erityisesti käyttöliittymäkomponenttikirjasto-ohjelmia myyvien tahojen osalta, mutta myös tekniikkaan painottuvilta blogisivustoilta. Nämä tieteelliset ja epätieteelliset lähteet esittävät keskenään osittain samoja ajatuksia komponenttikirjaston hyödyistä.

Komponenttikirjaston käyttäminen parantaa kehittäjien ja suunnittelijoiden välistä yhteistyötä (Advantages of a Component Library, 2017; Saring, 2020). Artikkelissa "Advantages of a Component Library" (2017) kerrotaan, että jos suunnittelijan tulee suunnitella uusi komponentti, hän voi aloittaa prosessin tutkimalla kirjaston avulla millaisia komponentteja on jo olemassa ja hyödyntää niitä suunnitelmissaan. Näin jo olemassa oleva design ja toteutus saadaan uudelleenkäytettyä ja suunnittelijan tarvitsee suunnitella vain uudet osat. Jos myös käyttöliittymäkehittäjät ovat ottaneet komponenttikirjaston osaksi kehitysprosessiaan, he voivat aloittaa uuden ominaisuuden implementoinnin heti, kun uudet suunnitelmat on laitettu komponenttikirjastoon. Saring (2020) puolestaan linjaa, että komponenttikirjasto mahdollistaa oikeiden koodattujen komponenttien käyttämisen suunnitelmissa, jolloin suunnittelijat sekä myös muut tuotteen parissa työskentelevät lähdekoodia ymmärtämättömät ihmiset voivat käyttää olemassa olevia koodattuja komponentteja designelementtien sijaan.

Käyttöliittymät, jotka ovat yhtenäisiä, mahdollistavat käyttäjän hyödyntää aikaisemmin oppimaansa ja ovat siten käytettävämpiä ja opittavampia kuin käyttöliittymät, jotka eivät ole yhtenäisiä (Derleth, 2018). Ilman kehittäjien ja suunnittelijoiden yhteistä alustaa, jossa molemmat voivat etsiä funktionaalisia sekä visuaalisia komponentteja, syntyy monia versioita samoista käyttöliittymäkomponenteista, mikä puolestaan heikentää käyttöliittymän yhtenäisyyttä (Derleth, 2018). Komponenttikirjasto auttaa pitämään käyttöliittymän yhtenäisenä (Advantages of a Component Library, 2017; Derleth, 2018; Saring, 2020; Weber, 2019, 2019).

Uudelleenkäyttö on tärkein tekijä ohjelmistokehityksen tuottavuudessa (Banker & Kauffman, 1991). Komponenttikirjastoa käytettäessä ohjelmiston kehittämisen nopeus kasvaa (Derleth, 2018; Saring, 2020; Weber, 2019). Koska komponenttikirjasto mahdollistaa komponenttien tehokkaan etsimisen ja käyttöönoton, lähdekoodin kahdentuminen vähenee (Derleth, 2018; Weber, 2019) ja sen ylläpitäminen helpottuu, koska samoja muutoksia ei tarvitse tehdä useaan paikkaan (Derleth, 2018; Saring, 2020).

Saring (2020) kertoo, että komponenttikirjaston avulla uusien työntekijöiden perehdyttäminen helpottuu. Komponenttikirjasto kokoaa käytetyt komponentit yhteen ja uusi työntekijä näkee nopeasti, millaisia komponentteja ohjelmistossa on käytössä.

2.2.2 Mahdolliset haitat

Ohjelmistoprojektiin liittyvän tiedon jäsentäminen ja asettaminen helposti saavutettavaksi vie yrityksen resursseja. Pelkkä tiedon uudelleenjärjestäminen ei riitä, vaan tietokokoelman esillepanoa varten on rakennettava oma infrastruktuurinsa, jossa tietoon pääsee käsiksi ja jossa tietokokoelmasta voi etsiä tarvittavia tietoja. (Maccanti et al., 2016)

Komponenttikirjastojen tarpeellisuutta on kyseenalaistettu uudelleenkäytön tutkimuksessa. Uudelleenkäytön epäonnistumista tutkittaessa on kuitenkin havaittu, että komponentin on oltava saatavilla, löydettävissä ja ymmärrettävä, mikä puoltaa komponenttikirjastojen tarpeellisuutta. On myös pohdittu, että suurin osa komponenttikokoelmista koostuu niin pienestä määrästä komponentteja, että niiden käyttö ei tarvitse edistynyttä kirjastoa tuekseen. (Frakes & Kyo Kang, 2005)

Pelkkä komponenttikirjasto käsittää vain kehitetyt komponentit, mutta ei ole suoraan yhdistettynä UX-suunnitteluun ja designiin. Tämä voi johtaa tyylioppaiden vanhentumiseen, koska ne eivät ole suoraan linkitettyinä kehitykseen, mikä puolestaan johtaa tuottavuuden laskuun. Koska ohjelmistokehittäjien tulkinnat suuripiirteisimmistä suunnitelmista voivat vaihdella, UX-suunnittelijoiden on aina suunniteltava ja mallinnettava kaikki yksityiskohdat, mikä johtaa suureen työmäärään verrattuna komponenttien käyttämiseen suunnitelmissa. (Ruissalo, 2018)

Beranek and Kovar (2020) esittävät, että komponenttikirjasto ja sen rajattu valikoima erilaisia komponentteja voivat rajoittaa UX-suunnittelijan luovuutta käyttöliittymää suunniteltaessa. Tämän takia ominaisuuksien käytettävyys ja käyttöliittymän yleisilme voivat kärsiä. Myös Gordon (1992) mainitsee, että komponenttien uudelleenkäytöstä seuraa se, että tuotettavien ohjelmistojen ominaisuudet ovat rajoitettuja. Gordon kertoo myös, että komponenttien tekijöiden ja komponenttien käyttäjien välillä on oltava paljon

kommunikaatiota, jotta komponentteja saadaan tarvittaessa muokattua. Kyky muokata komponentteja helposti on erittäin tärkeää.

2.3 Komponenttikirjastojen aikaisempi käyttö ja tutkimus

Monet ohjelmistoyritykset, jotka käyttävät uudelleenkäytön periaatteita, ovat alkaneet kehittää omia komponenttikirjastojaan ja saaneet huomattavaa hyötyä siitä. Yritysten tavoitteena on kehittää komponentteja, joita voidaan uudelleenkäyttää mahdollisimman monessa paikassa ilman muutoksia tai vain hyvin pienillä muutoksilla. Komponentit dokumentoidaan ja sertifioidaan kattavasti, jotta niiden käyttäminen eri laitteilla ja käyttöjärjestelmillä olisi mahdollista. (Jha & Mishra, 2019)

Komponenttikirjaston käyttöä ja komponenttipohjaista ohjelmistotuotantoa on tutkittu esimerkiksi ERP-järjestelmän (Enterprise Resource Planning, toiminnanohjausjärjestelmä) rakentamisen kautta. Prosessissa määriteltiin ohjelmistolle vaatimukset, jotka analysoitiin ja jaettiin pienempiin osiin, komponentteihin. Komponenttien luomisen sijasta komponentit etsittiin valmiista komponenttikirjastosta niiden vaatimusmäärittelyiden perusteella. Valmiita atomisia komponentteja yhdistelemällä luotiin ohjelmistossa tarvittavat komponentit. Kun lopulta yhä isompia ja isompia komponentteja yhdistettiin keskenään, ERP-järjestelmä oli valmis ja itsessään yksi iso komponentti. Menetelmän todettiin parantavan kehitystyön tehokkuutta ja ohjelmiston mukautuvuutta. (He et al., 2009)

(Ruissalo, 2018) diplomityössä käytetyssä CASE-yrityksessä oli käytössä tyyliopas (*style guide*), kuvakirjasto (*pattern library*), komponenttikirjasto sekä käyttöliittymäpakki suunnittelutyökalulle. Kyseisessä CASE-yrityksessä komponenttikirjastoa käytettiin osana laajempaa tyylijärjestelmää (*design system*). Designsuunnitelmien ja komponenttikirjaston yhdistämisen design-järjestelmäksi todettiin säästävän aikaa, kun pelkkien abstraktien design-suunnitelmien sijasta käytettävissä oli toteutettuja käyttöliittymäkomponentteja. (Ruissalo, 2018)

Méndez et al., (2007) tutkivat, kuinka komponenttikirjastoa voidaan hyödyntää uuden komponentin luomisessa. He käyttivät IMO.Net -komponenttikirjastoa, joka on suunniteltu kiteyttämään Matlab-ohjelmiston toiminnallisuudet. Uusi toiminnallinen komponentti luotiin komponenttikirjastoa käyttäen. Kokeilussa koettiin komponenttikirjasto hyödylliseksi, sillä kirjaston komponentit oli helppo integroida osaksi luotavaa komponenttia.

Tapionlinna (2010) toteutti komponenttikirjaston verkkosivujen suunnitteluun yritykselle, joka ei aikaisemmin käyttänyt komponenttipohjaista kehitystä. Komponenttikirjaston luomiseen käytettiin 75 henkilötyöpäivää. Työssä analysoitiin tarvittavat komponentit ja toteutettiin ne. Tämän jälkeen mitattiin, kuinka komponenttikirjaston käyttäminen vaikutti testiprojektin kestoon ja laatuun. Komponenttikirjasto vähensi projektin suunnitteluun kuluvaa aikaa 32 % ja projektin toteuttamiseen kuluvaa aikaa 16 %. Koko projektin osalta aikaa säästyi 23 %. Tulosten odotettiin vielä parantuvan, kun kirjasto tulee käyttäjilleen tutuksi.

3 TUTKIMUSMENETELMÄ

Tässä luvussa esitellään käytetty tutkimusmenetelmä. Tutkimusmenetelmänä toimii pääasiassa kysely, mutta sen suorittamiseksi on ensin todennettava, että komponenttikirjaston lisääminen kehitettävään tuotteeseen on mahdollista.

3.1 Komponenttikirjaston käyttöönoton tekninen toteutus

Tutkitaan tarjolla olevat työkalut komponenttikirjaston luomiseksi ja valitaan niistä sopivin. Luodaan komponenttikirjasto valitulla työkalulla ja siirretään 3–5 keskenään erityyppistä komponenttia komponenttikirjastoon. Tutkitaan, soveltuvatko tuotteessa käytetyt teknologiat komponenttikirjaston luomiseen. Luodaan tarvittaessa tuki tuotteessa käytetyille teknologioille, jos mahdollista. Havainnoidaan komponenttikirjaston luomisen ja ylläpitämisen työläyttä. Tehdään luodusta komponenttikirjaston prototyypistä demovideo, joka voidaan liittää kyselyyn.

3.2 Kysely komponenttikirjaston käyttöönoton kannattavuudesta

Kyselyssä kerätään dataa käyttöliittymäkehityksessä ilmaantuvista haasteista sekä komponenttikirjaston mahdollisuuksista ratkaista niitä kyselyllä. Kysely valittiin tutkimusmenetelmäksi, koska se mahdollistaa vastaajien vastata heille sopivana ajankohtana ja anonyymisti. Kyselyn kohderyhmä työskentelee tuotteen kehityksen parissa ja osaa siten parhaiten arvioida komponenttikirjaston sopivuutta kyseisen tuotteen kehityksessä. Kyselyssä kysytään ensin vastaajien taustatiedot, esimerkiksi työtehtävä, työkokemus ja aikaisempi kokemus komponenttikirjastoista. Taustatietojen jälkeen vastaajalle esitetään kategorisoituja (Kitchenham & Pfleeger, 2002, p. 3) väittämiä käyttöliittymäkehityksen toimivuudesta. Haasteiden kartoittamisen jälkeen vastaajalle esitellään komponenttikirjasto lyhyen demovideon avulla, jossa käydään läpi komponenttikirjaston päätoiminnallisuudet sekä myös sen ylläpitämiseen kuuluvat välivaiheet. Komponenttikirjaston esittelyn jälkeen vastaajalle esitetään kategorisoituja väittämiä komponenttikirjaston vaikutuksista

käyttöliittymäkehitykseen sekä sen hyödyllisyydestä yleisellä tasolla. Kysymyksissä kysytään vastaajan omia kokemuksia aiheisiin liittyen (Kitchenham & Pfleeger, 2002, p. 3). Kyselyn aihe ja sen tulokset todennäköisesti kiinnostavat vastaajia, joten voidaan olettaa vastaajien jaksavan käyttää aikaa vastaamiseen ainakin 20-30 minuuttia (Kitchenham & Pfleeger, 2002, p. 3).

Vastaajat vastaavat väittämiin Likert-asteikon (Likert, 1932) avulla. Valitussa asteikossa on yhteensä kuusi vastausvaihtoehtoa, jotka on kuvattu taulukossa Taulukko 1. Vastausvaihtoehtoihin jätettiin neutraali vaihtoehto sekä mahdollisuus olla ilmaisematta kantaansa, mikäli vastaaja kokee, ettei tiedä aiheesta tarpeeksi. Tämä mahdollistaa useamman kysymyksen kysymisen kaikilta vastaajaryhmiltä. Jokaisen väittämäkategorian jälkeen vastaajalla on myös mahdollisuus vastata kategorian väittämiin sanallisesti niin halutessaan (Kitchenham & Pfleeger, 2002, p. 3).

Taulukko 1 Kyselyssä käytettävä Likert-asteikko

Numeroarvo	**Sanallinen kuvaus**
1	Täysin eri mieltä
2	Jokseenkin eri mieltä
3	Ei samaa eikä eri mieltä
4	Jokseenkin samaa mieltä
5	Täysin samaa mieltä
6	En osaa sanoa

Saatujen vastausten perusteella voidaan selvittää, missä kehityksen osa-alueissa koetaan olevan haasteita ja missä kehityksen osa-alueissa komponenttikirjaston käytöstä olisi apua. Kyselyllä kartoitetaan myös yleistä asennetta komponenttikirjaston käyttöönottoa kohtaan, sillä komponenttikirjaston hyödyllisyys riippuu myös paljon tiimin jäsenten asenteista sitä kohtaan. Kysely kohdistetaan sitä todennäköisimmin käyttäville eli käyttöliittymäkehittäjille, UX-designereille sekä liiketoiminta-analyytikoille. Kysely jaetaan myös muille kehitystiimin jäsenille. Kyselyä ei jaeta kehitystiimin ulkopuolelle, esimerkiksi asiakaspalvelijoille tai myyjille, koska komponenttikirjasto tulisi vahvasti kehityksen työkaluksi, eikä siitä olisi luultavasti juurikaan hyötyä muille.

4 KOMPONENTTIKIRJASTON KÄYTTÖÖNOTTO

Tässä luvussa kerrotaan, miten komponenttikirjaston käyttöönotto toteutettiin ja millaisia työvaiheita sen käyttämiseen sisältyy. Luvussa arvioidaan myös työvaiheisiin kulunutta aikaa.

4.1 Valittu työkalu

Valitun työkalun tulisi olla käytettävissä ilman ohjelmointiosaamista ja soveltua myös Infernolle. Työkalun tulisi näyttää komponentit visuaalisesti ja mahdollistaa niiden kehittämisen erotettuna toimivasta applikaatiosta. Komponenttikirjasto täytyy olla mahdollista rakentaa kokonaan omista komponenteista ja lisätä se jo olemassa olevaan projektiin.

Suurimmaksi rajoittavaksi tekijäksi työkalun valinnassa muodostuivat työkalujen tukemat kehykset (*framework*). Mikään tutkituista komponenttikirjastotyökaluista ei tarjonnut valmiiksi tukea vähemmän käytetylle Infernolle. Tutkitut vaihtoehdot ja niiden soveltuvuustekijät on kuvattu taulukossa Taulukko 2.

Komponenttikirjasto päädyttiin toteuttamaan Storybook-työkalulla (“Storybook,” n.d.), sillä Storybook oli ainoa työkalu, joka tarjosi ohjeet uuden kehyksen lisäämiselle ja mainosti sitä ominaisuutenaan. Storybook mahdollistaa myös useiden erilaisten lisäosien (*addon*) lisäämisen, esimerkiksi Adobe XD -integraatiolla voidaan Adobe XD -suunnitelmat lisätä suoraan komponenttien yhteyteen. Storybook on avoimen lähdekoodin komponenttikirjastonhallintatyökalu, joka tarjoaa mahdollisuuden kehittää komponentteja eristyksissä kehitettävästä ohjelmasta sekä tarkastella komponentteja ja niiden eri tiloja tarinoiden (*story*) avulla. Tarina on komponentin tallennettu tila, jossa sille on annettu tietyt ominaisuudet (*prop, properties*).

Taulukko 2. Työkaluvaihtoehdot

Työkalun nimi	**Hintatyyppi**	**Tuki Infernolle**	**Tuki lisättävissä**	**Muuta**
Bit.Dev	Kuukausimaksu	Ei	Ehkä	Selvityksen ainoa kaupallinen vaihtoehto, asiakastuki saatavilla
Bluekit	Ilmainen	Ei	Ei	Komponentin tilojen välillä ei voi liikkua, tehty Reactille
Chromatic	-	Ei	-	Ylläpito lopetettu, ohjattu käyttämään Storybookia
Electrode Explorer	Ilmainen	Ei	Ei	Komponentin tilojen välillä ei voi liikkua, tehty Reactille
React Cosmos	Ilmainen	Ei	Ehkä	Tehty Reactille
React Styleguidist	Ilmainen	Ei	Ei	Tehty Reactille
Storybook	Ilmainen	Ei	Kyllä	Laajasti käytössä, paljon lisäosia

4.2 Tuen lisääminen Infernolle

Infernon API (Application Programming Interface, ohjelmointirajapinta) on hyvin samankaltainen kuin React-kehyksen API (“Inferno,” n.d.; “React Top-Level API,” n.d.), mikä mahdollisti valmiin lähdekoodin (“Storybook repository,” n.d.) osittaisen uudelleenkäyttämisen. Storybook tarjosi ohjeistusta tuen lisäämiseksi uudelle kehykselle (“Scaffolding a new framework,” n.d.), mutta koska Storybook on avoimen lähdekoodin ohjelmisto, ohjeet ja dokumentaatio olivat hajallaan ja osittain vanhentuneita sekä puutteellisia, mikä aiheutti haasteita ja projektin viivästymistä.

Jotta tuki Infernolle voitiin lisätä Stoorybookiin, oli luotava uusi paketti (*package*) @storybook/inferno. Paketissa oli määriteltävä Inferno-komponenttien renderöiminen sekä webpack-konfiguraatio. Koska tässä kandidaatintyössä toteutetaan vain prototyyppi, ei luotua pakettia julkaistu missään, vaan se linkitettiin projektiin paikallisesti ”yarn link” -komennolla. Komento luo linkin paketin ja projektin välille siten, että sen sijaan, että paketinhallintatyökalu lataisi paketin normaalisti, se käyttääkin paketin lokaalia versiota.

Aikaa työvaiheeseen kului arviolta 30 työtuntia. Tuen lisääminen tarvitsee kuitenkin tehdä vain kerran, joten työkalun varsinaiseen käyttöön ja ylläpitoon kuluvaan aikaan se ei vaikuta.

4.3 Komponenttikirjaston lisääminen projektiin

Storybookin lisäämiseksi olemassa olevaan projektiin, tuli projektiin luoda ”.storybook” -kansio, joka sisälsi työkalun konfiguraatiotiedostot. Luotu @storybook/inferno -paketti linkitettiin projektiin. Projektiin luotiin myös uusi kansio tarinoita varten. Tarinakansioon luotiin kansiorakenne, joka auttaa hallitsemaan tarinoita ja niihin liittyviä tiedostoja. Koska projektissa komponentteja on toteutettu käyttämällä periytymistä, on komponenttien vastaanottamien ominaisuuksien dokumentaatio luotu siten, että ylemmän luokan ominaisuuksia voidaan käyttää suoraan alemman tason komponentin dokumentaatiossa.

Komponenttikirjastoon otettiin käyttöön kaksi lisäosaa, jotka ovat kontrollit (*controls-addon*) sekä Adobe XD -integraatio (*xd-designs-addon*). Kontrollit mahdollistavat komponentin saamien ominaisuuksien dynaamisen muokkaamisen. Adobe XD -integraatio mahdollistaa Adobe XD -suunnitelmien suoran linkittämisen olemassa oleviin komponentteihin.

Komponenttikirjaston käyttöönottoon ilman komponenttien siirtämistä kului kaikkineen noin 10 työtuntia. Käyttöönotto tarvitsee kuitenkin tehdä vain kerran, joten työkalun varsinaiseen käyttöön ja ylläpitoon kuluvaan aikaan se ei vaikuta.

4.4 Komponenttien lisääminen komponenttikirjastoon

Komponentin lisääminen komponenttikirjastoon sisältää taulukossa Taulukko 3 kuvatut työvaiheet. Yksittäisen komponentin komponenttikirjastoon siirtämiseen aikaan vaikuttivat komponentit kompleksisuus, sen vastaanottamien ominaisuuksien määrä ja laatu, sekä

komponentin aikaisemman dokumentaation määrä. Aikaa yksittäisen komponentin siirtämiseen kului noin 1–6 tuntia. Ajan voi odottaa lyhentyvän käytön rutinoituessa.

Taulukko 3 Komponentin lisääminen kirjastoon.

Työvaihe	**Pakollinen**
Komponentin lähdekoodin tuominen *(import)* komponenttikirjastoon.	Kyllä
Komponentin perustilan luominen tarinaksi ja muiden tilavariaatioiden luominen tarinoiksi perustilan pohjalta.	Kyllä
Ominaisuuksien dokumentoiminen ja kontrollien luominen.	Suositeltava
Designin linkittäminen.	Suositeltava
Somisteiden (*decorator*) lisääminen.	Ei

5 KYSELYN TULOKSET

Tässä luvussa esitellään kyselyn tulokset olennaisilta osin. Kyselyn tulokset löytyvät tiivistämättöminä liitteestä 1.

5.1 Tiivistetty koonti olennaisimmista tuloksista

Tiivistelmä olennaisimmista tuloksista löytyy taulukosta Taulukko 4. Taulukossa on listattu väittämiä käyttöliittymäkehityksessä esiintyvistä haasteista, komponenttikirjaston mahdollisuuksista ratkaista ne sekä komponenttikirjaston käyttöönoton kannattavuudesta. Jokaisen väittämän yhteydestä löytyy vastausten keskiarvo sekä mediaani. ”En osaa sanoa” -vastaukset on jätetty huomiotta keskiarvoa ja mediaania laskettaessa.

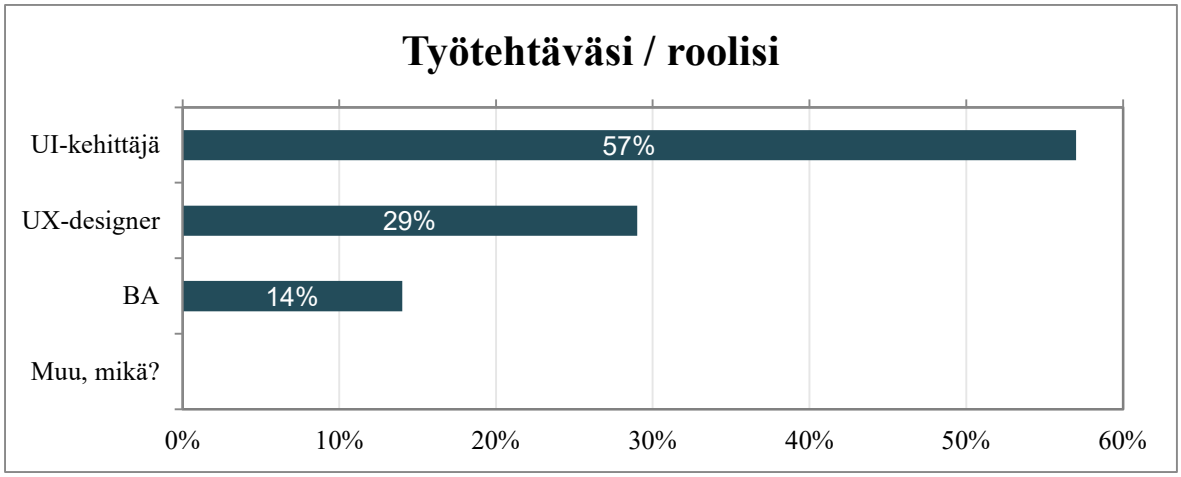
Taulukko 4 Tiivistelmä kyselyn tuloksista, missä 1 = Täysin eri mieltä, 2 = Jokseenkin eri mieltä, 3 = Ei samaa eikä eri mieltä, 4 = Jokseenkin samaa mieltä, 5 = Täysin samaa mieltä, 6 = En osaa sanoa (jätetty huomiotta). Punaisella on korostettu väitteet, joista vastaajat olivat keskimäärin eri mieltä ja vihreällä väitteet, joista vastaajat olivat keskimäärin samaa mieltä.

Väite	**Keskiarvo**	**Mediaani**
Dokumentaatiota on riittävä määrä	2,43	2
Dokumentaatiota on liikaa	2,00	2
Dokumentaatio on ajan tasalla	3,00	3
Dokumentaatio on helposti löydettävissä	2,86	3
Minun on helppo jäljittää komponentti sen vaatimusmäärittelyyn	2,29	2
Minun on helppo jäljittää käytössä oleva komponentti sen designiin	3,14	4
Uusien työntekijöiden perehdyttäminen työtehtäväsi on helppoa	3,86	4
Työssäni on tärkeää pystyä tarkastelemaan applikaatiossa käytössä olevia komponentteja	4,43	5
Käytössä olevien komponenttien tarkasteleminen on helppoa	2,86	2
Minun on helppo tarkistaa komponenttiin tehtyjen muutosten toimivuus	3,50	3,5
Komponenttien uudelleenkäyttö on tehokasta	4,14	4
Komponenttien uudelleenkäyttö on systemaattista	4,00	4
Näen komponentteja käytettävän tarkoitetusta poikkeavalla tavalla	3,86	4
Komponentista on olemassa vain yksi versio	2,83	2,5

Komponenttikirjasto auttaisi…		
...dokumentoimaan komponentit paremmin	4,71	5
...pitämään dokumentaation ajan tasalla	4,00	4
..löytämään dokumentaation helpommin	4,71	5
...jäljittämään komponentin sen vaatimusmäärittelyyn	4,50	4,5
...jäljittämään käytössä oleva komponentin sen designiin	4,83	5
...uusien työntekijöiden perehdyttämisessä	4,71	5
...tarkastelemaan käytössä olevia komponentteja	4,83	5
...tarkistamaan komponenttiin tehtyjen muutosten toimivuuden	4,50	4,5
...uudelleenkäyttämään komponentteja tehokkaammin	4,43	4
...uudelleenkäyttämään komponentteja tehokkaammin systemaattisemmin	4,29	4
...vähentämään komponenttien käyttöä tarkoitetusta poikkeavalla tavalla	4,29	4
...ehkäisemään komponenttien useiden eri versioiden syntymistä	4,43	5
Komponenttikirjaston luomiseen on aikaa yleisellä tasolla	3,00	3
Minulla on aikaa, jota voin käyttää komponenttikirjaston luomiseen	3,00	3
Komponenttikirjaston ylläpitämiseen on aikaa yleisellä tasolla	3,67	4
Minulla on aikaa, jota voin käyttää komponenttikirjaston ylläpitämiseen	3,50	3,5
Komponenttikirjasto olisi mielestäni tarpeellinen työkalu	4,57	5
Ottaisin komponenttikirjaston mielelläni käyttöön	4,43	5
Komponentteja on niin vähän, että komponenttikirjastoa ei tarvita	1,50	1

5.2 Vastaajien taustatiedot

Kyselyyn vastasi neljä käyttöliittymäkehittäjää, kaksi UX-designeria ja yksi BA (Business Analyst, liiketoiminta-analyytikko). Vastaajien suhteelliset osuudet työtehtävän mukaan näkyvät kuvassa Kuva 1. Kysely tavoitti lähes koko kohderyhmän. Tuloksissa on huomioitu vain ne kysymykset, joihin kaikki vastaajat vastasivat.



Kuva 1 Vastaajien työtehtävät, n = 7.

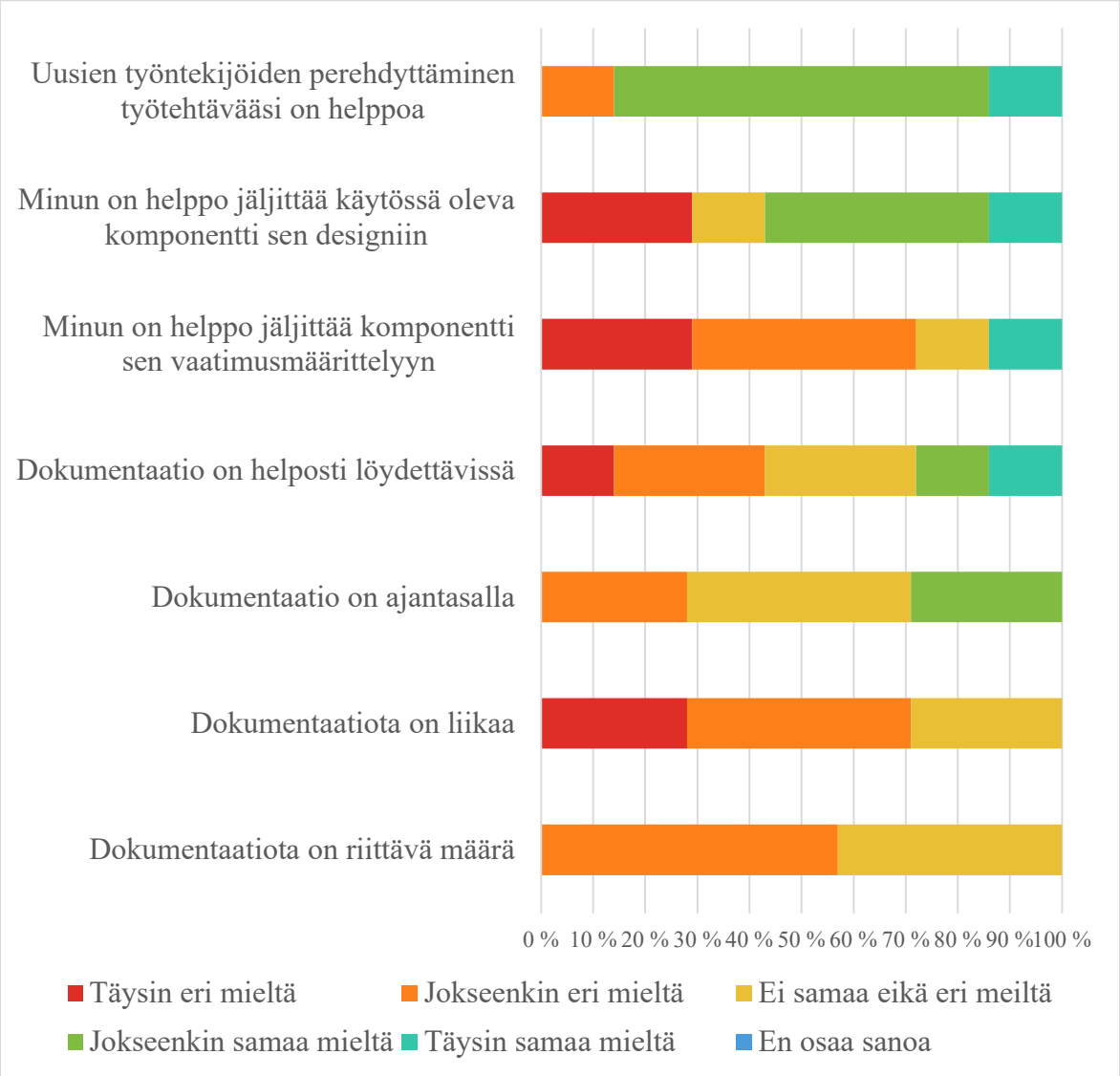
UI-kehittäjillä (käyttöliittymäkehittäjillä) on kokemusta tuotteen parissa työskentelemisestä 5–10 vuotta, mutta UX-designereilla ja liiketoiminta-analyytikoilla kokemusta on alle vuosi. UI-kehittäjät ovat keränneet kokemuksensa UI-kehityksestä suurimmaksi osaksi kyseessä olevan tuotteen parissa, mutta UX-designerit ovat hankkineet kokemusta muualtakin. Siinä missä UI-kehittäjille kehitettävä tuote on tuttu useiden vuosien ajalta, UX-designereille ja liiketoiminta-analyytikoille tuote on paljon uudempi. Osalla tiimistä on aikaisempaa kokemusta komponenttikirjaston käyttämisestä ja osalla ei.

5.3 Käyttöliittymäkehityksen haasteet

Kyselyn vastausten sekä kuvan Kuva 2 perusteella käyttöliittymäkehityksen haasteiksi koettiin erityisesti dokumentaation vähyys ja vastauksissa nousikin esille tarve selkeämmälle dokumentaatiolle:

”-- komponenteilla ei ole selkeää dokumentaatiota, miten niiden pitäisi toimia. Yksi komponentti saattaa olla rakennettu turhan moneen käyttötarkoitukseen ja esimerkiksi komponenteille annettujen propertyjen [eli ominaisuuksien] määrä on järkyttävän iso.”, UI-kehittäjä.

”Tällä hetkellä komponenteilla ei ole virallista dokumentaatiota”, UI-kehittäjä.



Kuva 2 Käyttöliittymäkehityksen haasteet: dokumentaatio

Toisaalta vastauksissa nousi esille myös, että vuosien kokemus tuotteen parissa työskentelystä korvaa ainakin osittain dokumentaation:

” Olen itse ollut alusta asti rakentamassa nykyisiä komponentteja, joten pystyn sen takia melko hyvin hahmottamaan mitä komponentteja on ja miten ne toimivat. Toki komponentteja alkaa olla aika paljon, niin kaikkien komponenttien kaikkia ominaisuuksia en tunne.”, UI-kehittäjä.

” Toisaalta dokumentaation puute ei ole ainakaan minulle ollut suuri ongelma, koska tunnen koodin aika hyvin ja tiedä mitä missäkin tilanteessa voi yleensä hyödyntää. Mutta varmaan uusille työtekijöille voi kuitenkin aiheuttaa haasteita.”, UI-kehittäjä.

Osalla vastaajista on myös ajoittain vaikeuksia tarkastella olemassa olevia komponentteja. Komponenttien uudelleenkäyttö sen sijaan koettiin suurimmaksi osin tehokkaaksi ja systemaattiseksi, vaikkakin komponentteja käytetään myös tarkoituksestaan poikkeavalla tavalla. Haasteiksi koettiin myös käyttöliittymän yhtenäisyyden puute:

”Tällä hetkellä käyttöliittymästä löytyy aika paljon sekalaisia ratkaisuja niin värien kuin esim. ikonien suhteen.”, UX-designer.

5.4 Komponenttikirjaston mahdolliset hyödyt

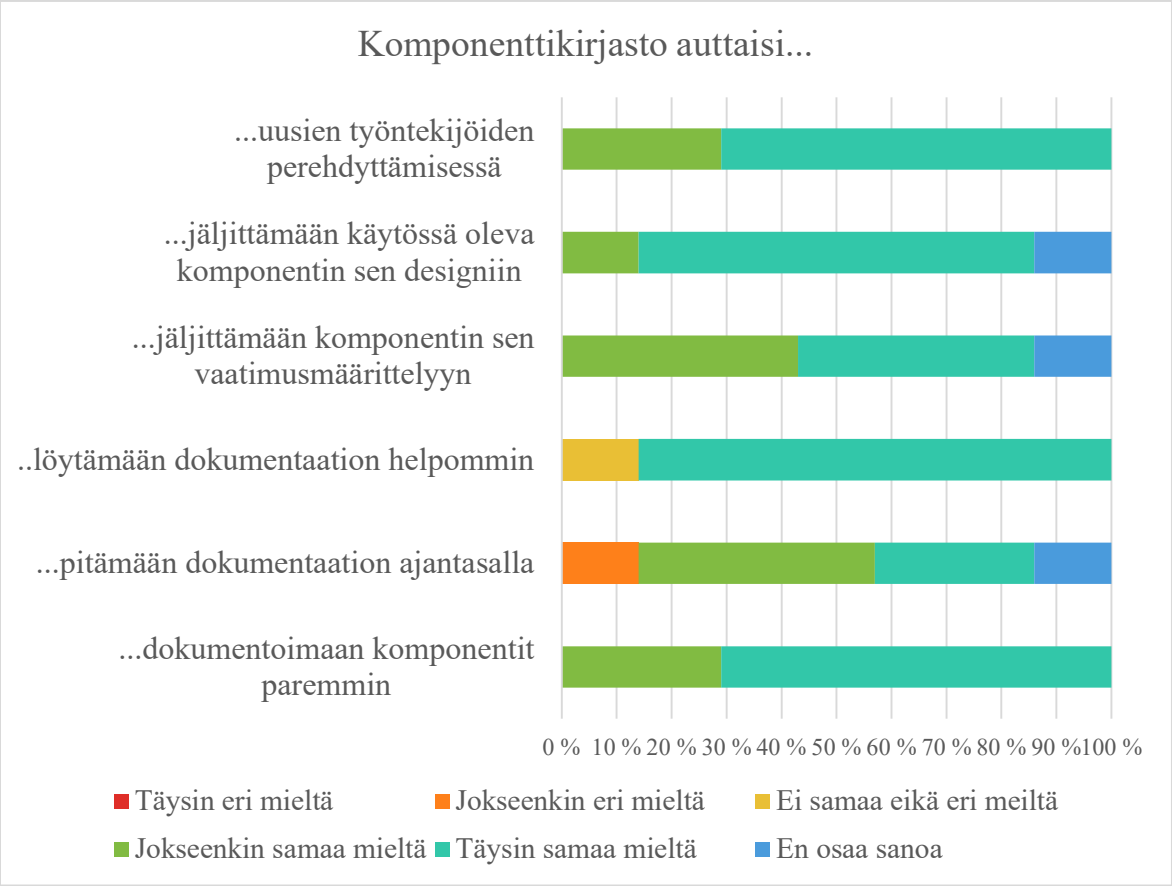
Kuvan Kuva 3 mukaan komponenttikirjaston koettiin voivan auttaa etenkin komponenttien dokumentoinnissa ja dokumentaation löytämisessä sekä komponentin yhdistämisessä sen designiin. Myös komponenttien tarkastelemisen ja uudelleenkäytön koettiin voivan helpottua komponenttikirjaston avulla. Dokumentaation suhteen huolta herätti kuitenkin se, pysyykö dokumentaatio ajan tasalla sekä kuinka sitä päivitetään:

”Vaatii sitoutumista kaikilta, että kirjastoa ylläpidetään –”, BA.

”Kuka pitää dokumentaation ajan tasalla? -- joutuisiko dokumentaatiota ylläpitämään kahteen paikkaan (designerit)? Mistä tietää mistä silloin löytyy uusin tieto?”, UX-designer.

Komponenttikirjaston todettiin helpottavan erityisesti muita kuin kehittäjä tarkastelemaan olemassa olevia komponentteja:

”[Komponenttikirjasto] Helpottaa myös ei-koodareita tutkimaan tiettyjä komponentteja ja bongaamaan "väärinkäytöksiä".”, BA.



Kuva 3 Komponenttikirjaston mahdolliset hyödyt: dokumentaatio

5.5 Käyttöönoton kannattavuus ja asenteet

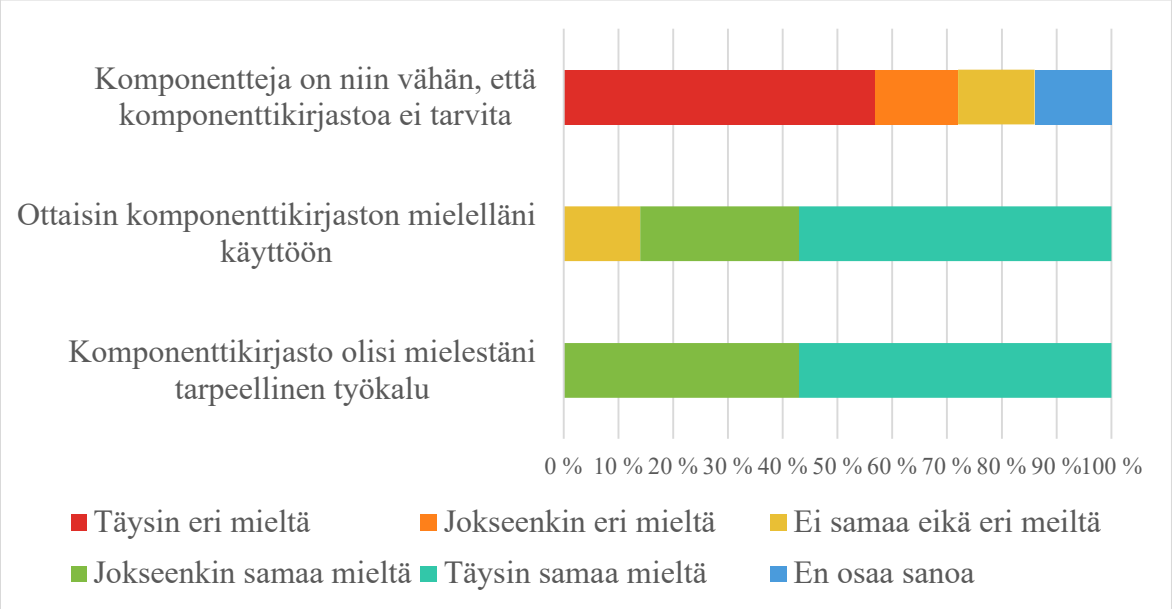
Vastausten perusteella osalla vastaajista on jonkin verran aikaa komponenttikirjaston luomiseen sekä ylläpitämiseen ja osalla vastaajista ei ole aikaa. Toisaalta komponenttikirjaston todettiin säästävän aikaa tulevaisuudessa:

”Yksi niistä monista asioista, joihin lähtökohtaisesti kaikki sanovat, ettei ole aikaa, mutta joka helpottaa tulevaisuutta.”, BA.

”Komponenttikirjaston luomiseen joutuisi käyttämään aikaa, mutta se aika on säästöä jatkossa, koska ei tarvitse kaikkea piirtää alusta asti joka kerta.”, UX-designer.

Komponenttikirjaston ylläpitoa varten vastauksissa ehdotettiin generaattorin käyttämistä sekä komponenttikirjaston päivittämisen ottamista osaksi normaalia kehitystyötä. Kuvasta Kuva 4 nähdään, että kaikki vastaajat kokivat komponenttikirjaston jokseenkin tai täysin tarpeelliseksi työkaluksi ja he ottaisivat sen mielellään käyttöön. Komponenttikirjasto koettiin kaiken kaikkiaan työkaluksi, josta olisi apua, mutta jonka ylläpitäminen voi olla ongelmallista:

”Komponenttikirjastossa on paljon hyvää mutta suurin ongelma on sen luominen ja ylläpitäminen. Jos nämä saisi jotenkin automatisoitua, niin se tekisi siitä oivan apuvälineen UI-kehityksessä.”, UI-kehittäjä.



Kuva 4 Komponenttikirjaston hyödyllisyys

6 KESKUSTELU

Tässä luvussa pohditaan komponenttikirjaston hyödyllisyyttä ja sen rakentamisen kannattavuutta kyselyn tulosten sekä teknisen toteutuksen kokemusten pohjalta.

6.1 Tulosten tulkinta

Tulosten pohjalta voidaan todeta, että komponenttikirjaston rakentamisesta käyttöliittymäkehityksen työkaluksi olisi todennäköisesti hyötyä. Kyselyssä ei noussut esiin muita haittoja, kuin komponenttikirjaston rakentamiseen ja ylläpitämiseen kuluvat resurssit. Saatavien hyötyjen ja kuluvien resurssien suhteesta ei kuitenkaan tulosten pohjalta voida tehdä johtopäätöksiä. Komponenttikirjasto kannattaa siis sopivassa vaiheessa ottaa käyttöön ja kokeilla sen käyttöä suuremmassa mittakaavassa. Jos myöhemmin osoittautuu, että se ei olekaan toimiva työkalu, voidaan sen käytöstä luopua. Tulokset kuitenkin osoittavat, että komponenttikirjaston käyttämistä kannattaa ainakin kokeilla, sillä kehitystiimi koki sen tarpeelliseksi työkaluksi.

Tulosten perusteella haasteita näyttäisi esiintyvät dokumentaatiossa, komponenttien tarkastelemisessa ja komponenttien käyttämisessä oikeissa käyttötarkoituksissa. Komponenttien tarkasteleminen koetaan vaikeaksi ehkä siksi, että kaikilla ryhmillä on omat keinonsa tarkastella omia komponenttejaan. UX-designereilla on designit komponenteista ja niiden käyttötilanteista, mutta näihin designeihin eivät muut pääse yhtä helposti käsiksi. Käyttöliittymäkehittäjillä puolestaan on käytettävissään kaikki lähdekoodi, jota UX-designerit eivät pääse tarkastelemaan. Lähdekoodista on myös vaikeaa löytää tarvittavaa komponenttia, jos ei jo valmiiksi tiedä sen nimeä.

Tulosten perusteella komponenttikirjasto auttaisi eniten dokumentaatiossa, komponenttien tarkastelemisessa sekä niiden jäljittämisessä niiden designiin. Kaikki kyselyssä esitetyt hyödyt koettiin mahdollisiksi. Vastaajat kokivat, että komponenttikirjastosta saisi vähiten hyötyä dokumentaation ylläpitämiseen, mikä selittyy avoimilla vastauksilla, joissa ilmaistiin

huoli siitä, että komponenttikirjasto ei pysy ajan tasalla ja että lopulta uusinta tietoa joutuu etsimään monesta eri paikasta.

Tuloksista (Taulukko 4) voidaan huomata, että vastaajat kokivat mahdollisuuden tarkastella olemassa olevia komponentteja osana työtään tärkeäksi (4,43), mutta eivät kokeneet sen olevan erityisen helppoa (2,84). Toinen tuloksista esiin noussut huomio on, että monessa avoimessa vastauksessa kerrottiin, että dokumentaation vähyys ei ole ollut ongelma, koska vastaaja on ollut mukana projektissa alussa asti ja siten tuntee koodikannan hyvin. Yhdessä vastauksessa tosin myös todettiin, että tämä ei päde kaikkeen, sillä koodia ja erilaisia komponentteja alkaa olla niin paljon, että kaikkien toimintaa ei voi tuntea. Onkin hyvä pohtia, kuinka pahasti tuotteen kehitys kärsii, jos osa tai kaikki kehitystiimin jäsenet vaihtuvat.

Yleinen asenne komponenttikirjaston käyttöönottoa kohtaan oli positiivinen (4,43) ja se koettiin tarpeelliseksi työkaluksi (4,57). UX-designerit olivat jokseenkin sitä mieltä, että komponenttikirjaston luomiseen on aikaa yleisellä tasolla, mutta käyttöliittymäkehittäjät ja liiketoiminta-analyytikot sen sijaan kokivat suurimmaksi osin, että komponenttikirjaston luomiseen ei olisi aikaa yleisellä tasolla. Komponenttikirjaston ylläpidon suhteen aikaa koettiin kaikissa ryhmissä löytyvän hieman enemmän, joskin edelleen vain joidenkin vastaajien toimesta. Avoimista vastauksista käy kuitenkin ilmi, että vaikka komponenttikirjaston luominen ja ylläpitäminen koetaan aikaa vievinä asioina, komponenttikirjaston ennustetaan säästävän aikaa tulevaisuudessa. Ongelmana lienee siis se, että tällä hetkellä komponenttikirjaston rakentamiseen ei ole varattu aikaa sprinteille.

Komponenttikirjaston ylläpitämisestä pitää saada yhtä luonnollinen osa kehitysprosessia kuin esimerkiksi yksikkötestien kirjoittamisesta. Jokainen komponentti lisätään komponenttikirjastoon ja komponenttiin tehdyt muutokset on päivitettävä myös komponenttikirjastoon. Avoimien vastausten perusteella komponenttikirjaston luominen ja ylläpitäminen näyttäisi olevan raskainta käyttöliittymäkehittäjille. Tätä voi ainakin osittain selittää se, että UX-designereilla on jo designelementeistä tehty komponenttikirjasto tehtynä. Komponenttikirjaston luomiseen tulee siis varata riittävästi aikaa ja ottaa sen luominen hallitusti osaksi sprinttejä.

6.2 Havainnot toteutusprosessista

Suurin työ lienee komponenttikirjaston luomisessa, eikä niinkään sen ylläpitämisessä ja komponenttien päivittämisessä. Komponentin päivittämiseen liittyvä työmäärä muodostuu mahdollisesta tekstin päivittämisestä, eli miten komponenttia tulee käyttää, missä paikoissa ja mihin tarkoitukseen, sekä komponentin tarinoiden päivittämisestä, eli vanhojen tarinoiden päivittämisestä ja uusien tarinoiden luomisesta. Komponenttikirjaston luominen sen sijaan vaatii kaikkien tarinoiden luomisen kaikille komponenteille sekä myös päätökset komponentin mahdollisista käyttöpaikoista ja tarkoituksista, joita aletaan noudattamaan.

6.3 Jatkokehityksen tarpeet

Komponenttikirjaston ylläpitämisen ja rakentamisen helpottamiseksi voitaisiin esimerkiksi tutkia mahdollisuuksia generoida komponentin ominaisuuksien kontrollit automaattisesti komponentin lähdekoodin perusteella. Mahdollisuutta eri käännösten tarkastelemiseen komponenttikirjaston avulla on tutkittava. Myös joidenkin komponenttien lähdekoodia täytyy luultavasti järjestää uudelleen ja joitakin komponentteja hajauttaa useammaksi eri komponentiksi. Komponenttikirjaston teknisen puolen ylläpitämisen helpottamiseksi Inferno-tuki pitäisi saada osaksi virallista avoimen lähdekoodin projektia, jolloin esimerkiksi pakettienhallinta hoidettaisiin sitä kautta.

Komponenttikirjasto tulisi lähtökohtaisesti yhden laajan SaaS-tuotteen kehityksen työkaluksi. CASE-yrityksellä on kuitenkin useampi samaa visuaalista tyyliä noudattava tuote, joiden kehityksessä voitaisiin mahdollisesti hyödyntää CASE-tuotteen komponenttikirjastoa jakamalla joitakin komponentteja.

7 JOHTOPÄÄTÖKSET

Tässä työssä tutkittiin, kannattaako CASE-yrityksen kehitystiimin rakentaa komponenttikirjasto yksittäisen SaaS-tuotteen komponenttipohjaisen käyttöliittymäkehityksen työkaluksi. Tutkimus toteutettiin todentamalla komponenttikirjastotyökalun yhteensopivuus kehitettävän tuotteen kanssa ja mittaamalla kyselyllä kehitystiimin kokemuksia kehityksessä esiintyvistä haasteista sekä komponenttikirjaston mahdollisuuksia auttaa niiden ratkaisemisessa.

Työssä tarkasteltiin aikaisempia tutkimuksia komponenttipohjaisen ohjelmistokehityksen hyödyistä ja haitoista, komponenttipohjaisen käyttöliittymäkehityksen hyödyistä ja haitoista, sekä komponenttikirjastojen käyttämisestä. Aikaisemmissa tutkimuksissa komponenttikirjastoista todettiin olevan hyötyä. Kirjallisuudesta löydettyjen hyötyjen ja haittojen perusteella luotiin kysely, jolla selvitettiin, millaisia haasteita tuotekehityksessä esiintyy ja millaisissa asioissa komponenttikirjaston koettiin voivan auttaa. Kyselyyn vastasivat kehitystiimin käyttöliittymäkehittäjät, UX-designerit ja liiketoiminta-analyytikot. Kyselyn lisäksi todennettiin, että valittu komponenttikirjastotyökalu on yhteensopiva tuotteen kanssa. Testaaminen tapahtui luomalla komponenttikirjasto tuotteen yhteyteen ja lisäämällä viisi keskenään erilaista komponenttia komponenttikirjastoon. Komponenteille luotiin tarinat ja ominaisuuksien dokumentaatio. Luotu komponenttikirjasto esiteltiin vastaajille kyselyn aikana.

Komponenttikirjastotyökalu todettiin yhteensopivaksi tuotteen kanssa. Kyselyn tulosten perusteella kehityksen haasteita ovat dokumentaation vähyys sekä vaikeus tarkastella olemassa olevia komponentteja ja käyttää niitä oikein. Tulosten perusteella komponenttikirjaston koettiin voivan auttaa kaikilla esitetyillä tavoilla, mutta erityisesti dokumentaatiossa, komponenttien tarkastelemisessa sekä niiden designin jäljittämisessä. Komponenttikirjasto koettiin tarpeelliseksi työkaluksi ja se otettaisiin vastaajien keskuudessa mielellään käyttöön. Ainoa huolenaihe vastaajien keskuudessa oli ajan riittämättömyys komponenttikirjaston rakentamiseen ja ylläpitämiseen.

Tulosten perusteella suositellaan komponenttikirjaston kokeilemista käyttöliittymäkehityksen työkaluna sopivana ajankohtana. Komponenttikirjaston käyttäminen pitää silloin ottaa osaksi kehitysprosessia, jotta se pysyy ajan tasalla. Jotta komponenttikirjaston käytön kannattavuudesta yksittäisen SaaS-tuotteen kehityksessä saataisiin varmempaa tietoa, komponenttikirjastoa on tutkittava täysikokoisena ja osana oikeaa tuotekehitystä.

LÄHTEET

Advantages of a Component Library [WWW Document], 2017. Open Social. URL https://www.getopensocial.com/blog/open-source/advantages-component-library (Viitattu: 21.9.2020).

AL-Badareen, A.B., Selamat, M.H., Jabar, M.A., Din, J., Turaev, S., 2011. Reusable Software Components Framework. Advances in Communications 6.

Banker, R.D., Kauffman, R.J., 1991. Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study. MIS Quarterly 15, 375. https://doi.org/10.2307/249649

Beranek, M., Kovar, V., 2020. A component-based method for developing cross-platform user interfaces for mobile applications. Advances in Intelligent Systems and Computing 1041, 201–217. https://doi.org/10.1007/978-981-15-0637-6_17

Casanova, M., Van Der Straeten, R., Jonckers, V., 2003. Supporting evolution in component-based development using component libraries, in: Seventh European Conference OnSoftware Maintenance and Reengineering, 2003. Proceedings. Presented at the Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings., pp. 123–132. https://doi.org/10.1109/CSMR.2003.1192419

Derleth, T., 2018. Corporate Component and Service Libraries — A concept for creating, maintaining and managing a company-specific user interface component library in the field of frontend web development 12.

Dong, R., Faber, J., Ke, W., Liu, Z., 2013. rCOS: Defining Meanings of Component-Based Software Architectures, in: Liu, Z., Woodcock, J., Zhu, H. (Eds.), Unifying Theories of Programming and Formal Engineering Methods: International Training School on Software Engineering, Held at ICTAC 2013, Shanghai, China, August 26-30, 2013, Advanced Lectures, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 1–66. https://doi.org/10.1007/978-3-642-39721-9_1

Frakes, W.B., Kyo Kang, 2005. Software reuse research: status and future. IEEE Transactions on Software Engineering 31, 529–536. https://doi.org/10.1109/TSE.2005.85

Gordon, D., 1992. A graphical user interface in Ada for domain-specific reuse libraries, in: Proceedings of the Conference on TRI-Ada '92, TRI-Ada '92. Association for Computing Machinery, New York, NY, USA, pp. 309–320. https://doi.org/10.1145/143557.143750

Grundy, J., 2000. Storage and retrieval of software components using aspects. Presented at the Proceedings - 23rd Australasian Computer Science Conference, ACSC 2000, pp. 95–103. https://doi.org/10.1109/ACSC.2000.824386

Hai-mei, Z., Min, G., 2012. A component library information model supporting component composition, in: 2012 IEEE International Conference on Mechatronics and Automation. Presented at the 2012 IEEE International Conference on Mechatronics and Automation, pp. 475–479. https://doi.org/10.1109/ICMA.2012.6282956

Han, H., Gao, P., Oyama, K., 2011. Retrieval, description and security: Towards the large-scale UI component-based reuse and integration, in: 2011 IEEE International Conference on Information Reuse Integration. Presented at the 2011 IEEE International Conference on Information Reuse Integration, pp. 193–199. https://doi.org/10.1109/IRI.2011.6009545

He, S., Chang, H., Wang, Q., 2009. Component Library-Based ERP Software Development Methodology, in: 2009 International Conference on Interoperability for Enterprise Software and Applications China. Presented at the 2009 International Conference on Interoperability for Enterprise Software and Applications China, pp. 34–38. https://doi.org/10.1109/I-ESA.2009.47

Inferno [WWW Document]. Inferno.js. URL https://www.infernojs.org/ (Viitattu: 25.10.2020).

Irie, Y., Tanaka, S., 2002. Software component library management system. US20020194578A1.

Jha, S.K., 2016. Multi criteria based retrieval techniques for reusable software components from component repository. IJEAST.

Jha, S.K., Mishra, D.R.K., 2019. A Review on Reusability of Component Based Software Development. Reliability: Theory & Applications 14.

Kalia, A., Sood, S., 2017. Concerns in Maintaining Reusable Software Components and the Possible Solutions [WWW Document]. Indian Journal of Science and Technology. URL https://indjst.org/ (Viitattu: 21.9.2020).

Kitchenham, B.A., Pfleeger, S.L., 2002. Principles of survey research: part 3: constructing a survey instrument. SIGSOFT Softw. Eng. Notes 27, 20–24. https://doi.org/10.1145/511152.511155

Krueger, C.W., 1992. Software Reuse. ACM Computing Surveys (CSUR) 24, 131–183. https://doi.org/10.1145/130844.130856

Likert, R., 1932. A technique for the measurement of attitudes. Archives of Psychology 22 140, 55–55.

Maccanti, S., Al-Jaroodi, J., Sirinterlikci, A., 2016. Knowledge Management Framework for Software Reuse, in: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC). Presented at the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), pp. 155–160. https://doi.org/10.1109/COMPSAC.2016.147

Méndez, A.J., Roselló, E.G., Lado, M.J., Dacosta, J.G., Cota, M.P., 2007. Integrating Matlab Neural Networks Toolbox functionality in a fully reusable software component library. Neural Computing and Applications 16, 471–479. https://doi.org/10.1007/s00521-006-0075-5

Niranjan, P., Rao, C.V.G., 2010. A Mock-Up Tool for Software Component Reuse Repository. IJSEA 1, 1–12. https://doi.org/10.5121/ijsea.2010.1201

Omer, N., Jha, S.K., Kumar Khatri, S., 2019. Maintaining Reusable Software Components, in: 2019 International Conference on Intelligent Computing and Control Systems (ICCS). Presented at the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), pp. 1350–1352. https://doi.org/10.1109/ICCS45141.2019.9065845

Prause, C.R., Durdik, Z., 2012. Architectural design and documentation: Waste in agile development?, in: 2012 International Conference on Software and System Process (ICSSP). Presented at the 2012 International Conference on Software and System Process (ICSSP), IEEE, Zurich, Switzerland, pp. 130–134. https://doi.org/10.1109/ICSSP.2012.6225956

React Top-Level API [WWW Document], n.d. URL https://reactjs.org/docs/react-api.html (Viitattu: 25.10.2020).

Ruissalo, M., 2018. Operating a design system in a large software company (Diplomityö).

Saring, J., 2020. 15 Reasons to Build Your Component Library in Bit.dev [WWW Document]. Medium. URL https://blog.bitsrc.io/15-reasons-to-build-your-component-library-in-bit-dev-93a514878863 (Viitattu: 21.9.2020).

Scaffolding a new framework [WWW Document], n.d. URL https://storybook.js.org/docs/react/api/new-frameworks (Viitattu: 25.10.2020).

Schmidt, M., Polowinski, J., Johannes, J., Fernández, M.A., 2010. An integrated facet-based library for arbitrary software components. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6138 LNCS, 261–276. https://doi.org/10.1007/978-3-642-13595-8_21

Storybook repository [WWW Document]. GitHub. URL https://github.com/storybookjs/storybook (Viitattu: 25.10.2020).

Storybook: UI component explorer for frontend developers [WWW Document], n.d. URL https://storybook.js.org (Viitattu: 25.10.2020).

Tapionlinna, M., 2010. Komponenttikirjasto verkkosivujen suunnitteluun. Component library for designing Web pages.

Voigt, S., von Garrel, J., Müller, J., Wirth, D., 2016. A Study of Documentation in Agile Software Projects, in: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16. Association for Computing Machinery, New York, NY, USA, pp. 1–6. https://doi.org/10.1145/2961111.2962616

Weber, D., 2019. Benefits of a Component Library [WWW Document]. DEV Community. URL https://dev.to/domysee/benefits-of-a-component-library-2baa (Viitattu: 21.9.2020).

LIITE 1. Kyselyn tulokset

Työtehtäväsi / roolisi

	n	Prosentti
UI-kehittäjä	4	57,14 %
UX-designer	2	28,57 %
BA	1	14,29 %
Muu, mikä?	0	0 %

Työkokemus nykyisiä työtehtäviäsi vastaavista töistä (vuosina)

	n	Prosentti
0-0,5 vuotta	0	0 %
0,5-1 vuotta	1	14,29 %
1-3 vuotta	0	0 %
3-5 vuotta	4	57,14 %
5-10 vuotta	0	0 %
yli 10 vuotta	2	28,57 %
En osaa sanoa	0	0 %

Kauanko olet työskennellyt tuotteen parissa nykyisissä tehtävissäsi? (vuosina)

	n	Prosentti
0-0,5 vuotta	0	0 %
0,5-1 vuotta	3	42,86 %
1-3 vuotta	0	0 %
3-5 vuotta	2	28,57 %
5-10 vuotta	2	28,57 %
yli 10 vuotta	0	0 %
En osaa sanoa	0	0 %

Tiedän, mikä on komponenttikirjasto

	n	Prosentti
Kyllä	7	100 %
En	0	0 %
En osaa sanoa	0	0 %

Minulla on aikaisempaa kokemusta komponenttikirjastoista ja niiden käyttämisestä

	n	Prosentti
Kyllä	4	57,14 %
Ei	3	42,86 %
En osaa sanoa	0	0 %

(jatkuu)

LIITE 1. (jatkuu)

Kehityksen haasteet, 1 = Täysin eri mieltä, 2 = Jokseenkin eri mieltä, 3 = Ei samaa eikä eri mieltä, 4 = Jokseenkin samaa mieltä, 5 = Täysin samaa mieltä, 6 = En osaa sanoa

	1	**2**	**3**	**4**	**5**	**6**
Dokumentaatiota on riittävä määrä	0	4	3	0	0	0
Dokumentaatiota on liikaa	2	3	2	0	0	0
Dokumentaatio on ajan tasalla	0	2	3	2	0	0
Dokumentaatio on helposti löydettävissä	1	2	2	1	1	0
Minun on helppo jäljittää komponentti sen vaatimusmäärittelyyn	2	3	1	0	1	0
Minun on helppo jäljittää käytössä oleva komponentti sen designiin	2	0	1	3	1	0
Uusien työntekijöiden perehdyttäminen työtehtävääsi on helppoa	0	1	0	5	1	0
Työssäni on tärkeää pystyä tarkastelemaan applikaatiossa käytössä olevia komponentteja	0	0	1	2	4	0
Käytössä olevien komponenttien tarkasteleminen on helppoa	0	4	1	1	1	0
Minun on helppo tarkistaa komponenttiin tehtyjen muutosten toimivuus	0	1	2	2	1	1
Komponenttien uudelleenkäyttö on tehokasta	0	1	0	3	3	0
Komponenttien uudelleenkäyttö on systemaattista	0	1	1	2	3	0
Näen komponentteja käytettävän tarkoitetusta poikkeavalla tavalla	0	1	1	3	2	0
Komponentista on olemassa vain yksi versio	0	3	2	0	1	1

(jatkuu)

LIITE 1. (jatkuu)

Kehityksen haasteet, vapaat kommentit

Olen itse ollut alusta asti rakentamassa nykyisiä komponentteja, joten pystyn sen takia melko hyvin hahamottamaan mitä komponentteja on ja miten ne toimivat. Toki komponentteja alkaa olla aika paljon, niin kaikkien komponenttien kaikkia ominaisuuksia en tunne. Koodia lukemalla kyllä löytää kun tietää mistä etsii. Uusien työntekijöiden tuntuu olevan melko haastavaa löytää tarvittavat komponentit. Komponenttien määrä on iso ja niitä on rakennettu aina käyttöpaikkakohtaisesti, joten kenelläkään ei ole varmaan selkeää kuvaa miten niiden pitäisi tarkalleen toimia. Tiettyihin komponentteihin on vaikeaa tehdä muutoksia ja tyylejä joutuu tekemään aina jokaiseen käyttöpaikkaan erikseen.
Tällä hetkellä komponenteilla ei ole virallista dokumentaatiota, ellei sellaiseksi haluta sitten ymmärtää jotain yksittäistä design leiskaa, jonka pohjalta komponentti on aikanaan rakennettu. Lisäksi alkuperäiset komponentit ovat saaneet monia ajan myötä uusia ominaisuuksia ja laajennuksia, joita ei ole dokumentoitu mihinkään. Toisaalta dokumentaation puute ei ole ainakaan minulle ollut suuri ongelma, koska tunnen koodin aika hyvin ja tiedä mitä missäkin tilanteessa voi yleensä hyödyntää. Mutta varmaan uusille työtekijöille voi kuitenkin aiheuttaa haasteita.
On tärkeää että se on yhdessä paikassa ja että se paikka tosi helposti löydettävissä.
Tällä hetkellä käyttöliittymästä löytyy aika paljon sekalaisia ratkaisuja niin värien kuin esim. ikonien suhteen. Komponenttikirjaston olemassaolo tukisi varmasti kokonaisuuden yhtenäistämistä.
Iso haaste minusta on, että komponeteilla ei ole selkeää dokumentaatiota miten niiden pitäisi toimia. Yksi komponentti saattaa olla rakennettu turhan moneen käyttätarkoitukseen ja esimerkiksi komponenteille annettujen propertyjen määrä on järkyttävän iso.
Mikään ohjeistus tai komponenttikirjasto ei poista design reviewin tarvetta. Korostaisin myös laadukkaan kommunikoinnin tärkeyttä.

(jatkuu)

LIITE 1. (jatkuu)

Komponenttikirjaston hyödyt ja ajankäyttö. 1 = Täysin eri mieltä, 2 = Jokseenkin eri mieltä, 3 = Ei samaa eikä eri mieltä, 4 = Jokseenkin samaa mieltä, 5 = Täysin samaa mieltä, 6 = En osaa sanoa. Komponenttikirjasto auttaisi minua…

	1	**2**	**3**	**4**	**5**	**6**
...dokumentoimaan komponentit paremmin	0	0	0	2	5	0
...pitämään dokumentaation ajantasalla	0	1	0	3	2	1
..löytämään dokumentaation helpommin	0	0	1	0	6	0
...jäljittämään komponentin sen vaatimusmäärittelyyn	0	0	0	3	3	1
...jäljittämään käytössä oleva komponentin sen designiin	0	0	0	1	5	1
...uusien työntekijöiden perehdyttämisessä	0	0	0	2	5	0
...tarkastelemaan käytössä olevia komponentteja	0	0	0	1	5	0
...tarkistamaan komponenttiin tehtyjen muutosten toimivuuden	0	0	0	3	3	0
...uudelleenkäyttämään komponentteja tehokkaammin	0	0	0	4	3	0
...uudelleenkäyttämään komponentteja tehokkaammin systemaattisemmin	0	0	0	5	2	0
...vähentämään komponenttien käyttöä tarkoitetusta poikkeavalla tavalla	0	0	1	3	3	0
...ehkäisemään komponenttien useiden eri versioiden syntymistä	0	0	1	2	4	0
Komponenttikirjaston luomiseen on aikaa yleisellä tasolla	0	3	1	3	0	0
Minulla on aikaa, jota voin käyttää komponenttikirjaston luomiseen	1	1	2	3	0	0
Komponenttikirjaston ylläpitämiseen on aikaa yleisellä tasolla	0	1	1	3	1	1
Minulla on aikaa, jota voin käyttää komponenttikirjaston ylläpitämiseen	0	1	2	2	1	1
Komponenttikirjasto olisi mielestäni tarpeellinen työkalu	0	0	0	3	4	0
Ottaisin komponenttikirjaston mielelläni käyttöön	0	0	1	2	4	0
Komponentteja on niin vähän, että komponenttikirjastoa ei tarvita	4	1	1	0	0	1

(jatkuu)

LIITE 1. (jatkuu)

Komponenttikirjaston hyödyt ja ajankäyttö, vapaat kommentit

Vaatii sitoutumista kaikilta, että kirjastoa ylläpidetään, mutta varmasti toimiva tapa. Helpottaa myös ei-koodareita tutkimaan tiettyjä komponentteja ja bongaamaan "väärinkäytöksiä".
Kuka pitää dokumentaation ajantasalla? Komponenttikirjastoon ei varmaankaan tultaisi tekemään kaikkea mahdollista, joten joutuisiko dokumentaatiota ylläpitämään kahteen paikkaan (designerit)? Mistä tietää mistä silloin löytyy uusin tieto?
Muutosten toimivuus riippuu niin paljon asiayhteydestä, että ehkä käyttöliittymästä käsin se lopullinen tarkastaminen. Mutta antaa ainakin hyvän suunnan.
Auttaa varmasti nykytilanteessa, jossa tietystä komponentista on useita variaatioita ja saman tyyliseen käyttöön tehdystä variaatiosta vielä lisävariaatioita.
pitää käyttä generaatori
Yksi niistä monista asioista, joihin lähtökohtaisesti kaikki sanovat ettei ole aikaa, mutta joka helpottaa tulevaisuutta. Mielestäni pitäisi priorisoida tavalla tai toisella, koska hyödyt ovat selkeät.
Komponenttikirjaston luomiseen joutuisi käyttämään aikaa, mutta se aika on säästöä jatkossa, koska ei tarvitse kaikkea piirtää alusta asti joka kerta. Oletan myös, että virheiden mahdollisuus vähenee ja päivitettävyys paranee?
Vaatii resurssointia, hallitusti sprinteille
pitää käyttä generaatori
Hankala sanoa. Jos ylläpito on helppoa kuten sanot, niin tuskin ajankäyttö silloin on ongelma.
Onko ajankäyttöä suurempi ongelma kuitenkin se, että pysyykö kirjasto ajantasalla jatkossa?
Jos kirjasto otetaan käyttöön niin ylläpidon pitää kuulua yhtenä osana taskeihin joissa komponentteja päivitetään
Komponenttikirjastossa on paljon hyvää mutta suurin ongelma on sen luominen ja ylläpitäminen. Jos nämä saisi jotenkin automatisoitua, niin se tekisi siitä oivan apuvälineen UI-kehityksessä.
Joka osiossa toistuvat elementit, paljon käytetyt elementit. Myös tuotteen omat värit ja fonttityylit?