

Daniel Koskelainen

Työharjoittelujärjestelmän kehitysprojektin taustatutkimus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tuotantotalouden koulutusohjelma

Opinnäytetyö

11.4.2018

Tekijä Otsikko	Daniel Koskelainen Työharjoittelujärjestelmän kehitysprojektin taustatutkimus
Sivumäärä Aika	48 sivua + 1 liite 11.4.2018
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tuotantotalouden koulutusohjelma
Suuntautumisvaihtoehto	Teollisuuden prosessit
Ohjaajat	TkT Hannu Räsänen AMK Palvelupäällikkö Kauko Ojanen
<p>Insinööriyön tavoitteena oli tutkia Metropolia Ammattikorkeakoulun työharjoittelujärjestelmää. Tavoitteina oli mm. selvittää työharjoittelujärjestelmää käyttävien harjoittelua ohjaavien opettajien käyttäjätarinat ja järjestelmää koskevat kehitystoiveet sekä tutkia työharjoittelujärjestelmän integroimisen mahdollisuutta Peppi-järjestelmään. Integrointi mahdollisuutta ajatellen oli myös selvitettävä molempien järjestelmien arkkitehtuurinen rakenne. Työn lopullisena tarkoituksena oli muodostaa tutkimuksen perusteella määritelmä, mihin suuntaan työharjoittelujärjestelmän kehitysprojektiä on mahdollista viedä.</p> <p>Tutkimuksessa kerättiin aineistoa haastattelemalla tekniikan ja liiketalouden alojen tutkinto-ohjelmien opiskelijoiden työharjoitteluprosessia ohjaavia opettajia sekä työharjoittelujärjestelmän ja Peppi-järjestelmän arkkitehtuuria tuntevia henkilöitä. Opettajien haastatteluissa selvitettiin käyttäjätarinat ja kirjattiin ylös työharjoittelujärjestelmän toiminnalliset viat ja puutteet sekä kehitystoiveet. Työharjoittelujärjestelmän ylläpidon haastattelussa selvitettiin työharjoittelujärjestelmän arkkitehtuurinen rakenne ja puitiin havaittujen järjestelmän vikojen korjausmahdollisuuksia. Peppi-asiantuntijan haastatteluissa pyrittiin selvittämään Peppi-järjestelmän arkkitehtuurinen rakenne ja kerättyjen kehitystoiveiden toteutusmahdollisuudet. Teoriaosuus pohjautui tietojärjestelmän kehittämistä, integraatiota ja arkkitehtuuria käsittelevään kirjallisuuteen.</p> <p>Työharjoittelujärjestelmän nykytila-analyysin, henkilöhaastatteluista kerätyn informaation ja Peppi-järjestelmän arkkitehtuurisen rakenteen selvityksen perusteella muodostettiin kaksi etenemisvaihtoehtoa mahdollisen kehitysprojektin suhteen, joita analysoitiin mm. SWOT-nelikenttäänalyysityökalua hyödyntämällä. Lisäksi tutkimuksen aikana muodostettiin muutamia kehitysehdotuksia työharjoittelujärjestelmän toimintojen parantamiseksi.</p> <p>Metropolia Ammattikorkeakoulun Tietohallinto voi käyttää insinööriyönä laadittua työharjoittelujärjestelmän kehitysprojektin taustatutkimusta apuna tulevan kehitysprojektin suunnittelu- ja määrittelyvaiheessa.</p>	
Avainsanat	Tarpeiden määrittäminen, taustatutkimus, arkkitehtuurikuvaus, nykytila-analyysi, tietojärjestelmän kehittäminen, integraatio

Author Title	Daniel Koskelainen Background study for the development project of the internship system
Number of Pages Date	48 pages + 1 appendix 11 April 2018
Degree	Bachelor of Engineering
Degree Programme	Industrial Engineering and Management
Professional Major	Industrial Processes
Instructors	Hannu Räsänen, D.Sc Kauko Ojanen, Service Manager
<p>The aim of this thesis was to study the internship system at the Metropolia University of Applied Sciences. The main objectives were to examine primary users' needs and development proposals, and investigate the possibility of integration of this system into the Peppi system. In order to evaluate the possibility of the system integration, it was necessary to study the architectural structure of both systems. The ultimate objective was to provide possible directions for a development project of the internship system.</p> <p>Information and data were gathered by interviewing engineering and business economics teachers and specialists who know architectural structures of the systems. In teacher interviews user stories were mapped and the detected failures in the system's functions were discussed. In addition, new desired features for the upgraded system were discussed. By interviewing the specialists of the internship system, the system's architectural structure was mapped and the possibilities of repairing detected failures in some functions were discussed. An interview with the Peppi specialist aimed at investigating the architectural structure of the Peppi system. In addition, the possibilities to implement the desired new functionalities into the upgraded internship system were discussed.</p> <p>Literature used in this thesis consisted of books and articles covering the theory of development, integration and architectural structures of information systems.</p> <p>Based on the present state analysis of the internship system, the information gathered from the interviews and architectural structure of the Peppi system, two propositions were created for a possible development project of the internship system. Propositions were also analyzed using the SWOT matrix. In addition, a number of improvement proposals were developed during this study to improve the functioning of the internship system.</p> <p>The department of Information Management at the Metropolia University of Applied Sciences can use this study to help in the definition and planning phases of the upcoming development project.</p>	
Keywords	determination of needs, research, architectural mapping, present state analysis, development of information system, integration

Sisällys

1	Johdanto	1
2	Tutkimusmenetelmät	6
3	Tietojärjestelmä	7
3.1	Tietojärjestelmä yleisesti	7
3.2	Järjestelmäarkkitehtuuri	8
3.2.1	SOA	9
3.2.2	MVC	10
3.3	Järjestelmäintegraatio	11
3.4	Tietojärjestelmän kehittäminen	12
4	Sovelletut tietojärjestelmän teknologiat	20
5	Peppi	23
5.1	Peppi-järjestelmä	23
5.2	Pepin arkkitehtuuri	23
6	Työharjoittelujärjestelmän nykytila-analyysi	27
6.1	Työharjoittelujärjestelmä	27
6.2	Työharjoittelujärjestelmän arkkitehtuuri	27
6.3	Toiminnot ja ominaisuudet	29
6.4	Työharjoittelujärjestelmän puutteet ja kehitystoiveet	32
7	Kehitysehdotukset	35
7.1	Työharjoittelujärjestelmän jatkokehittäminen Pepin ulkopuolella	35
7.2	Työharjoittelujärjestelmä osaksi Peppiä	38
7.3	SWOT-analyysi työharjoittelujärjestelmän kehitysvaihtoehdoista	40
7.3.1	Pepin ulkopuolelle rakentuva työharjoittelujärjestelmä	40
7.3.2	Pepin sisälle rakentuva työharjoittelujärjestelmä	42
8	Yhteenveto	46
	Lähteet	49

Lyhenteet ja käsitteet

CSS	Cascading Style Sheets. Tekniikka, jolla määritellään tyypillisesti sovelluksen käyttöliittymän ulkoasu.
Front-end	Front-end-kehityksellä viitataan selainpuolen ohjelmointiin. Kaikki koodi, jota ajetaan verkkoselaimessa esim. sivun rakenne, ulkoasu ja toiminnallisuudet.
Intranet	Yrityksen sisäiseen viestintään ja tietojenkäsittelyyn eristetty lähiverkko.
Rest	HTTP-protokollaan perustuva arkkitehtuurimalli rajapintojen toteuttamiseen.
Servletti	Ohjelmakomponentti, jolla laajennetaan palvelimen toiminnallisuutta.
SOAP	XML-kieleen pohjautuva tietoliikenneprotokolla.

1 Johdanto

Tietojärjestelmät ovat osa nykyaikaista jokapäiväistä elämäämme ja hyödynnämme niitä niin päivittäisissä vapaa-ajan askareissamme kuin ammatillisissa työtehtävissämme. Asioidessamme kaupoissa ja virastoissa, ostaessamme palveluja tai osallistuessamme erilaisiin tapahtumiin, olemme tekemisissä erilaisten tietojärjestelmien kanssa, jotka helpottavat ja ohjaavat toimintaamme. Monet elämämme askareista ja tarpeistamme ovat siis suoraan riippuvaisia erilaisista tietojärjestelmistä. [Paananen & Granlund, 2005.]

Tässä insinööriyössä perehdytään tietojärjestelmän käyttöön ja kehittämiseen organisaatiossa. Metropolia Ammattikorkeakoulu on organisaatio, jonka tehtävänä on kouluttaa ihmisiä ja tuottaa tutkintoja sille osoitettuja resursseja hyödyntäen. Tietojärjestelmät ovat opettajille ja opiskelijoille välttämättömiä työkaluja, joita käytetään opetuksen tukena ja opintojen suunnittelun, hallinnoinnin sekä johtamisen apuna.

Tämän insinööriyön tarkoituksena on tutkia Metropolia Ammattikorkeakoulun nykyistä työharjoittelujärjestelmää, määrittää sille asetetut käyttäjien vaatimukset ja rakentaa tämän tutkimuksen pohjalta parannus- ja kehitysehdotukset tulevaisuudessa kaavailtuun työharjoittelujärjestelmän kehitysprojektiin. Tarkemmin työn tavoitteista, rajoituksesta ja sisällöstä puhutaan tämän luvun toisessa alaotsikossa.

Metropolia Ammattikorkeakoulu

Metropolia Ammattikorkeakoulu on suomalainen kansainvälinen ja monialainen ammattikorkeakoulu, joka toimii pääkaupunkiseudulla. Metropolia Ammattikorkeakoulu sai alkunsa vuonna 2007, kun Helsingin, Espoon, Vantaan ja Kauniaisten kaupungit sekä Kirkkonummen kunta perustivat yhdessä Metropolia Ammattikorkeakoulu Oy:n. Nykyinen ammattikorkeakoulu muodostui silloisten EVTEK -ammattikorkeakoulun ja Helsingin ammattikorkeakoulu Stadian yhdistymisen tuloksena. [Metropolia Ammattikorkeakoulu.]

Metropolia Ammattikorkeakoulu kouluttaa opiskelijoita kulttuurin, liiketalouden, sosiaali- ja terveystieteiden sekä tekniikan alojen ammatteihin. Vuonna 2017 Metropolia Ammattikorkeakoulussa opiskeli 16 500 opiskelijaa ja henkilökuntaa oli noin 1000 henkeä. Yh-

teensä Metropolia Ammattikorkeakoulu tarjoaa 69 tutkinto-ohjelmaa, joista 43 on AMK-tutkinto-ohjelmaa ja 26 ylempää AMK-tutkinto-ohjelmaa. Vuonna 2017 Metropolia Ammattikorkeakoulu operoi 95 miljoonan euron kokonaisbudjetilla. [Metropolia Ammattikorkeakoulu.]

Tulevien vuosien strategiassa Metropolia Ammattikorkeakoulu tähtää voimakkaaseen uudistumiseen, yhtenäisyyden vahvistamiseen, osaamisen kehittämiseen sekä yhteiskunnallisen vaikuttavuuden lisäämiseen. Kasvua haetaan myös tutkimus-, kehittämis- ja innovaatio toiminnassa sekä liiketoiminnassa, vaikuttavuuden ja ulkoisen rahoituksen saralla. [Metropolia Ammattikorkeakoulu.]

Insinöörityön tausta

Ammattikorkeakoulujen tutkinto-ohjelmat sisältävät poikkeuksetta yhden tai useamman työharjoittelujakson, jonka opiskelija suorittaa jossain opintojensa vaiheessa. Itse työharjoittelujaksoon sisältyy paljon muutakin kuin pelkästään työssäkäynti. Opiskelijan on itse löydettävä yritys, jossa hän suorittaa työharjoittelunsa ja hänen on solmittava yrityksen kanssa harjoittelu- tai työsopimus. Ennen harjoittelun alkamista, sopimus toimitetaan työharjoittelun ohjaajan hyväksyttäväksi. Lisäksi opiskelijan on asetettava työharjoittelujaksolle oppimistavoitteita ja päämääriä sekä täytettävä erilaisia raportteja ja itsearviointeja.

Metropolia Ammattikorkeakoulun tekniikan alan tutkinto-ohjelmien opiskelijoiden käyttöön on aikanaan suunniteltu digitaalinen työharjoittelujärjestelmä, jonka avulla he voivat ilmoittaa ja hyväksyttää molemmat tutkinto-ohjelmaan sisältyvät työharjoittelunsa ja lukea muiden opiskelijoiden kirjoittamia raportteja omista työharjoitteluistaan ja yrityksistä. Työharjoittelujärjestelmä on alun perin suunniteltu palvelemaan sekä opiskelijoita että työharjoittelun ohjaajia. Opiskelijat käyttävät työharjoittelujärjestelmää lähinnä harjoittelupaikan ilmoittamis- ja hyväksyttämisportaalina. Työharjoittelun ohjaajille työharjoittelujärjestelmä toimii puolestaan työkaluna, jonka avulla he ohjaavat, seuraavat ja arvioivat opiskelijoiden työharjoittelua alusta loppuun asti.

Sittemmin työharjoittelujärjestelmä on otettu käyttöön myös liiketalouden tutkinto-ohjelmassa, ja nyt Metropolia haluaa kehittää työharjoittelujärjestelmää siten, että tekniikan ja liiketalouden opiskelijoiden lisäksi järjestelmää voisi hyödyntää kahden toisen koulutusalan: sosiaali- ja terveydenhoidon sekä kulttuurin opiskelijat. Nykyinen työhar-

joittelujärjestelmä on kuitenkin suunniteltu kattamaan ainoastaan tekniikan opiskelijoiden ja työharjoittelun ohjaajien tarpeet. Järjestelmä kaipaa siis hiomista liiketalouden työharjoitteluprosessin suhteen, puhumattakaan muiden koulutusalojen työharjoitteluprosesseista.

Insinööriyön tavoite, rajaus ja sisältö

Tavoite

Insinööriyön tavoitteena on muodostaa hyvin spesifioitu määritelmä uudelle työharjoittelujärjestelmälle. Määritelmä on muodostettava mahdollisimman tarkasti tekniikan ja liiketalouden opiskelijoiden työharjoitteluprosessien ja työharjoittelun ohjaajien tarpeiden pohjalta, jotta mahdollisesti tuleva kehitysprojekti onnistuisi kerralla oikein.

Nykyinen työharjoittelujärjestelmä on erillinen sovellus, joka ei ole suoranaisesti kytköksissä Peppi-järjestelmään. Tavoitteena onkin tutkia sekä työharjoittelujärjestelmän että Peppi-järjestelmän arkkitehtuurista rakennetta ja sitä myöten uuden työharjoittelujärjestelmän integraation mahdollisuutta Peppiin arvioiden sen tuomia hyötyjä ja parannuksia työharjoitteluprosessiin.

Tutkimuksessa kerätyn tiedon ja tietojärjestelmän teoreettisen viitekehyksen pohjalta esitetään mahdollisia vaihtoehtoja työharjoittelujärjestelmän kehitysprojektille, jonka mahdollinen toteutusajankohta on vuoden 2019 aikana. [Ojanen.]

Rajaus

Työharjoittelujärjestelmän on tarkoitus tulevaisuudessa palvella kaikkia Metropolia Ammattikorkeakoulun koulutusalojen opiskelijoita ja työharjoittelun ohjaajia. Projektin laajuuden ja suuren sisällön vuoksi sosiaali- ja terveysalan sekä kulttuurin alojen tutkinto-ohjelmien opiskelijoiden työharjoitteluprosessit jätetään tästä työstä kokonaan pois. Tässä insinööriyössä työharjoittelujärjestelmää määritellään siis vain tekniikan ja liiketalouden tutkinto-ohjelmien osalta.

Työharjoitteluprosessien tarpeiden määritysten lisäksi tässä työssä tutkitaan myös työharjoittelujärjestelmän integraation mahdollisuutta Peppi-järjestelmään. Insinööriyön

teoreettisen viitekehyksen muodostaa tietojärjestelmän teoriaan liittyvä kirjallisuus ja verkosta saatava materiaali.

Insinööri työ tehdään tulevan työharjoittelujärjestelmän kehitysprojektin pohjatyönä, eli työharjoittelujärjestelmän toteutukseen liittyvät asiat ja taloudellisen kannattavuuden arviointi jätetään tässä työssä käsittelemättä.

Sisältö

Tässä insinööri työssä tutkitaan nykyistä Metropolia Ammattikorkeakoulun työharjoittelujärjestelmää, sen rakennetta ja ominaisuuksia. Lisäksi määritellään tekniikan ja liiketalouden opiskelijoiden työharjoittelun ohjaajien käyttäjätarinat, joiden pohjalta saadaan hyvä käsitys kunkin koulutusalan työharjoitteluprosessista ja työharjoittelun ohjaajien tarpeista tietojärjestelmän suhteen.

Luvussa 2 esitellään tässä insinööri työssä sovellettuja tutkimusmenetelmiä. Luvussa 3 käydään läpi tietojärjestelmää yleisellä tasolla syventyen tarkemmin arkkitehtuurikuvaukseen, järjestelmäintegraatioon ja tietojärjestelmän kehittämiseen.

Luvussa 4 perehdytään tietojärjestelmän toteutuksessa sovellettuihin teknologioihin, joita hyödynnetään myös Pepissä ja nykyisessä työharjoittelujärjestelmässä.

Insinööri työssä perehdytään myös Peppi-järjestelmään, jota Metropolia Ammattikorkeakoulun opiskelijat ja toimihenkilöt käyttävät. Peppi-järjestelmää käydään tarkemmin läpi luvussa 5.

Tutkimuksen ja haastattelujen pohjalta muodostetaan tekniikan työharjoittelujärjestelmän nykytila-analyysi. Nykytila-analyysissä käydään perusteellisesti läpi nykyisen työharjoittelujärjestelmän rakenne ja ominaisuudet sekä käytettävyyteen liittyvät puutteet ja kehitystä kaipaavat tekijät, jotka tulivat ilmi työharjoittelun ohjaajien kanssa käytyjen haastattelujen yhteydessä. Nykytila-analyysia käsitellään luvussa 6.

Luvussa 7 esitellään nykyistä työharjoittelujärjestelmää koskevat kehitysehdotukset, joihin tämän tutkimuksen pohjalta on päästy. Kehitysehdotusten lisäksi esitellään myös kehitysprojektin mahdolliset toteutusvaihtoehdot, joita arvioidaan myös SWOT-nelikenttäanalyysia hyödyntämällä.

Insinööriyön viimeisessä luvussa 8 on esitetty tämän työn yhteenveto ja mahdolliset jatkotutkimuskysymykset.

2 Tutkimusmenetelmät

Tässä insinööriyössä on sovellettu kvalitatiivista eli laadullista tutkimusmenetelmää. Kvalitatiivinen tutkimus on luonteeltaan kokonaisvaltaista tiedonhankintaa, jossa aineisto kerätään todellisissa tilanteissa ja ihminen itse toimii tiedonkeruun työkaluna. Laadullista tutkimusmenetelmää käytetään myös usein silloin, kun tutkimuksessaan tutkija luottaa enemmän omiin havaintoihinsa ja käytyihin keskusteluihin tutkittavien henkilöiden kanssa, kuin esimerkiksi lomakkeiden tai vastaavien ”kynä-paperi”-testien avulla hankittuun aineistoon. [Hirsjärvi, Remes & Sajavaara, 2004.] Tässä insinööriyössä on oleellista kerätä työharjoittelujärjestelmän eri käyttäjien käyttäjätarinat ja tehdä niiden pohjalta tarpeiden määrittäminen.

Koska kvalitatiivisessa tutkimuksessa aineisto kootaan luonnollisissa, todellisissa tilanteissa, haastattelu on yleinen metodi kerätä aineistoa. [Hirsjärvi, Remes & Sajavaara, 2004.] Tutkimuksessa katsottiin tarpeelliseksi haastatella tekniikan ja liiketalouden opiskelijoiden työharjoittelua ohjaavia opettajia, eli työharjoittelujärjestelmää työssään käyttäviä henkilöitä, nykyisen työharjoittelujärjestelmän ylläpidosta vastaavia henkilöitä sekä Peppi-järjestelmän asiantuntijaa.

Kaikki haastattelut suoritettiin tammikuun ja helmikuun 2018 aikana. Haastattelut käytiin Metropolia Ammattikorkeakoulun Leppävaaran, Myyrmäen ja Bulevardin kampuksilla. Yksi haastattelu suoritettiin etänä, puhelinhaastatteluna. Haastattelut olivat osin strukturoituita, eli esitetyt kysymykset olivat etukäteen mietittyjä ja jäsenneltäviä, mutta haastatteluissa annettiin kuitenkin tilaa haastateltavien omille ajatuksille ja ideoille kehitysprojektin suhteen, jolloin keskustelusta saatiin paljon avoimempi ja syvällisempi.

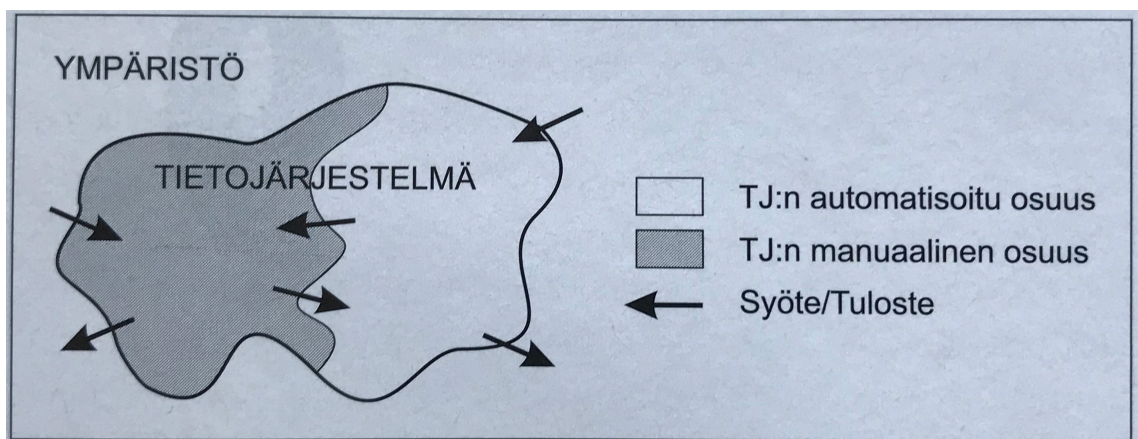
Tutkimuksen aineisto koostuu siis suurimmaksi osaksi nykyisen työharjoittelujärjestelmän analysoinnista ja Metropolia Ammattikorkeakoulun opettajien ja toimihenkilöiden haastatteluista. Tutkimuksen teoriaosuus on koottu tietotekniikkaa, tietojärjestelmää ja sen teknologioita käsittelevän kirjallisuuden ja artikkeleiden pohjalta.

3 Tietojärjestelmä

Tässä luvussa käydään läpi tietojärjestelmään liittyviä aiheita. Tämän insinööriyön kannalta on tärkeää ymmärtää tietojärjestelmän arkkitehtuurirakenne, järjestelmäintegraation mahdollisuus sekä tietojärjestelmän kehitykseen liittyvät vaiheet.

3.1 Tietojärjestelmä yleisesti

Tietojärjestelmä on kokonaisuus, joka koostuu ihmisistä, tiedosta, tietojenkäsittely- ja tiedonsiirtolaitteista sekä ohjelmistoista. Se on yksittäistä ohjelmaa tai ohjelmistoa laajempi käsite, joka palvelee jotain tiettyä toimintaa ja tehostaa tietojenkäsittelyä toiminnan toteutuksessa. [Paananen, 2005.]



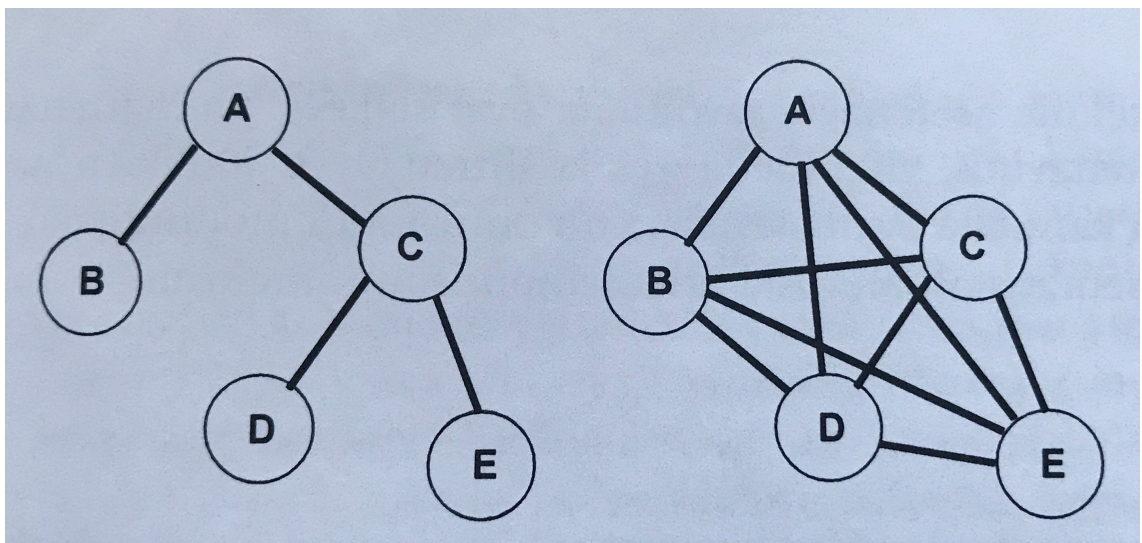
Kuva 1. Havainnollistava esitys tietojärjestelmän perusluonteesta. [Paananen, 2005]

Tietojärjestelmässä tietoa kerätään, kootaan, varastoidaan, yhdistetään, tulkitaan ja järjestellään uudelleen esimerkiksi laskutoimitusten tai muiden peräkkäisten toimintojen avulla. Tätä toimintaa kutsutaan tietojenkäsittelyksi. Tietojenkäsittely voi olla manuaalista tai automaattista. Manuaalinen tietojenkäsittely on ihmisten ohjaamaa, tiettyjen sääntöjen ja ohjeiden mukaista toimintaa. Automaattinen tietojenkäsittely puolestaan viittaa tietokoneellisesti suoritettavaan toimintaan, jolloin ihmisen osallistuminen ei ole välttämätöntä. Suuri osa nykyaikaisistakin tietojärjestelmistä koostuu kuitenkin sekä manuaalisista että automaattisista osista, jotka ovat yhteydessä toisiinsa ja omalla tavalla yhteydessä ympäristöön, jossa järjestelmä toimii. [Paananen, 2005.]

3.2 Järjestelmäarkkitehtuuri

Kuten rakennukset, myös tietojärjestelmät rakentuvat monesta eri osasesta ja komponentista. Tietojärjestelmä on monimutkainen ja laaja kokonaisuus, jonka rakennetta ja toimintaperiaatteita ei pysty selittämään yhdellä lauseella tai yksinkertaisella piirroksella. Kokonaiskuvan ymmärtämiseksi on laadittava järjestelmän arkkitehtuuri, jonka pohjalta järjestelmä rakennetaan.

Arkkitehtuurin rakenteen kannalta tärkeä käsite on *moduuli*. Moduuli on yksittäinen osa järjestelmää, jolla on oma rooli, omat toiminnalliset funktionsa ja rajapinta, jonka kautta se kommunikoi muun järjestelmän kanssa. Arkkitehtuurimallista riippuen moduuli voidaan nähdä myös palveluna (SOA), joka vastaanottaa ympäristöltään *syötteitä* ja palvelupyyntöjä. Moduuli käsittelee saamansa informaation ja lopputuloksena antaa halutun palautteen, eli *tulosteen*. Arkkitehtuurisuunnittelu pyrkii jakamaan järjestelmän mahdollisimman itsenäisiin moduuleihin, jotka olisivat minimaalisia riippuvaisia toisistaan. Mitä enemmän kytkentöjä ja riippuvuussuhteita moduulien välillä on, sitä monimutkaisemmaksi järjestelmä tulee. [Pohjonen, 2002.]



Kuva 2. Havainnollistava kuva järjestelmän monimutkaisuuden kasvusta moduulien välisten kytkentöjen lisääntymisen myötä. [Pohjonen, 2002]

Arkkitehtuurisuunnittelun rinnalla tapahtuu myös *moduulisuunnittelu*, joka kuvaa yksittäisten moduulien sisäiset rakenteet. Moduulien välisten riippuvuussuhteiden minimoinnin ohella moduulisuunnittelussa kannattaa myös tavoitella mahdollisimman pien-

tä moduulikokoa. Suuret moduulit ovat luonnostaan monimutkaisempia ja sitä myöten hankalampia ylläpitää. [Pohjonen, 2002.]

Jotta tietojärjestelmästä voitaisiin rakentaa tavoitteiden mukainen ja toimiva kokonaisuus, sen arkkitehtuurimallin on tuettava sen käyttötarkoitusta. Tietojärjestelmiä on erilaisia ja niiden rakenteelliset eroavaisuudet perustuvat sovellettujen arkkitehtuurien mukaan. Tämän insinööriyön kannalta on oleellista perehtyä palvelupohjaiseen arkkitehtuuriin ja MVC (Model View Controller)-arkkitehtuuriin.

3.2.1 SOA

SOA (Service Oriented Architecture) eli palvelupohjainen arkkitehtuuri on erityisesti liiketoimintaa tukevien tietojärjestelmien tekemiseen suunniteltu arkkitehtuurimalli, jossa toisistaan riippumattomat osaset, eli moduulit, vuorovaikuttavat keskenään palvelurajapintojen kautta. Moduulit ovat toisiinsa ”löysästi” kytkettyjä palveluita, jotka säilyttävät itsenäisen toimintansa mutta mahdollistavat tiedon liikkumisen toistensa välillä palvelurajapintojen avulla. [Hurwitz, Bloor, Kaufman & Halper, 2009.]

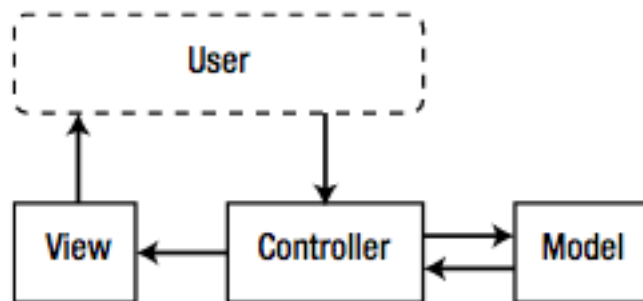
Palvelupohjainen arkkitehtuuri ei ole ainoastaan sovellettava tekninen ratkaisu tai metodologia vaan liiketoimintaa tukeva lähestymismalli. SOA pyrkii siihen, että liiketoiminnalliset ratkaisut eivät olisi enää tiettyyn teknologiaan sidonnaisia vaan eri teknologioiden tukemia. SOA poistaa näin rajoituksia ja mahdollistaa helpomman siirtymisen uusiin ratkaisuihin tai järjestelmää koskeviin muutoksiin. Näin palveluista tulee jouhevia, ja itse palvelun laatu paranee. [Hurwitz, Bloor, Kaufman & Halper, 2009.]

Yksi suurimmista SOA:n tarjoamista hyödyistä on ohjelmistoratkaisujen uudelleen käyttäminen. Pyörää ei lähdetä keksimään uudestaan, vaan hyödynnetään jo olemassa olevia ratkaisuja. Otetaan esimerkiksi iso yritys, jolla voi olla käytössään monta samankaltaista ohjelmistoa. Uudet ohjelmistot ovat syntyneet uusien tarpeiden saattamana olemassa olevia ohjelmistoja hieman mukauttaen. Lopulta tilanne voi olla se, että yrityksellä on käytössään liuta samantyyppisiä mutta varioituja ohjelmistoja samoissa prosesseissa. Tämä on tyypillinen syy siihen, että järjestelmistä kehkeytyy hyvin monimutkaisia ja kalliita ylläpitää. SOA poistaa tämän ongelman muuttamalla prosessit palveluiksi. Käytännössä tämä tarkoittaa sitä, että palvelu on kuin suljettu säiliö, jonka sisällä on tietyn funktion toteuttava ohjelmisto. Ohjelmisto on suunniteltu siten, että se on yhdistettävissä muihin prosesseihin. Näin ollen käytettävissä on tietylle funktiolle

suunniteltu palvelu, jota voidaan käyttää koko organisaation läpi. Kun jotain täytyy muuttaa koko organisaatioon vaikuttavalla mittakaavalla, riittää, että muutos tehdään yhteen paikkaan. [Hurwitz, Bloor, Kaufman & Halper, 2009.]

3.2.2 MVC

MVC (Model-View-Controller), eli suomeksi malli-näkymä-ohjain, on arkkitehtuurimalli, jota käytetään yleisesti *olio-ohjelmoinnissa*. MVC-arkkitehtuurin idea perustuu kolmen loogisen komponentin, mallin, näkymän ja ohjaimen keskinäiseen toimintaan. Alla on yksinkertainen ja havainnollistava kuvaus MVC-arkkitehtuurin rakenteesta. [Pitt, 2012.]



Kuva 3. Havainnollistava esitys malli-näkymä-ohjain-arkkitehtuurimallin elementtien välisistä kytkennöistä. [Pitt, 2012]

Malli

Ohjelmiston liiketoimintalogiikka sijaitsee mallissa. Liiketoimintalogiikka määrittelee ohjelmiston tavan hallita dataa. Jos ohjelmisto esimerkiksi haluaa hakea dataa tietokannasta, siihen soveltuva koodi sijaitsee mallissa. [Pitt, 2012.]

Näkymä

Näkymä sisältää kaikki ohjelmiston käyttöliittymään liittyvät elementit, kuten HTML-merkinnän, CSS- ja JavaScript-tiedostot. Toisin ilmaistuna näkymä määrittelee sen, mitä käyttäjä näkee. [Pitt, 2012.]

Ohjain

Ohjain on puolestaan komponentti, joka toimii mallin ja näkymän välissä. Ohjelmiston käyttäjän pyynnöt tulevat ensin ohjaimeen, joka sitten välittää pyynnön malliin ja mallista takaisin näkymän kautta käyttäjälle. Lyhyesti sanottuna ohjaimen rooli kiteytyy ohjelmiston toiminnan ohjaamiseen. [Pitt, 2012.]

3.3 Järjestelmäintegraatio

Järjestelmäintegraatio mahdollistaa keskenään yhteensopimattomien tietoteknisten sovellusten automatisoidun kommunikaation. Sami Tähtinen määrittelee järjestelmäintegraation kirjassaan ”Järjestelmäintegraatio” [Tähtinen, 2005] *”toimintamalleiksi ja tekniikoiksi, joiden avulla voidaan saattaa vähintään kaksi eri toiminnallisuutta tarjoavaa tietojärjestelmää jakamaan informaatiota siten, että informaation siirto ja muunnokset ovat kontrolloitavissa ja monitoroitavissa yhdestä tai useammasta keskitetystä pisteestä”*.

Kun pyritään integroimaan kaksi keskenään yhteen sopimatonta ja keskenään kommunikointia järjestelmää, on järjestelmäintegraatiossa oltava yksinkertaisimmillaan kyse

1. informaation siirtämisestä integroitavien järjestelmien välillä
2. tietomuunnoksista näiden järjestelmien sisäisten esitysmuotojen välillä
3. kokonaisprosessin (tiedonsiirto sekä tietomuunnokset) kontrolloinnista sekä tähän liittyvästä valvonnasta ja raportoinnista. [Tähtinen, 2005.]

Lähes kaikki tietojärjestelmät on suunniteltu siten, että niiden sisäisiin tietovarastoihin on mahdollista päästä käsiksi. Erityisesti yrityksen toiminnan kannalta tärkeät sovellukset on tyypillisesti suunniteltu avoimien tietovarastojen päälle. Poikkeuksena ovat kuitenkin tiettyihin erikoistarkoituksiin tarkoitettut ”murtovarmat” sovellukset kuten sirukortit. [Tähtinen, 2005.]

Informaation siirtyminen integroitavien järjestelmien välillä on mahdollista vain, jos näiden välillä on *rajapinnat*, joiden välityksellä järjestelmät voivat hakea ja syöttää informaatiota. Vastaanotettavaa tietoa kutsutaan syötteeksi ja lähetettävää tietoa tulosteeksi.

si. [Pohjonen, 2002.] Toimivat rajapinnat ovat tärkeä tekninen edellytys järjestelmäintegraation kannalta [Tähtinen, 2005].

Informaation liikkuminen palvelujen, sovellusten tai järjestelmien välillä vaatii jonkin fyysisen siirtotien, kuten tietoverkon tai tietoverkkojen päällä toimivan sanomajärjestelmän. Yksinkertaisimmillaan informaatiota siirtävänä mediana voivat toimia magneettiset tai optiset tallennusvälineet kuten levykkeet, magneettinauhat tai CD-levyt. Edellä mainitut mediat eivät kuitenkaan kykene nopeatempoiseen automatisoituun tiedonsiirtoon, vaan sitä varten on turvauduttava tietoliikenneverkkojen puoleen. Sisäinen järjestelmien kommunikointi luonnistuu tavallisimmin LAN (local area network)-paikallisverkon avulla, mutta ulkoinen kommunikointi vaatii tyypillisesti Internetin välityksellä tapahtuvaa liikennöintiä. [Tähtinen, 2005.]

Järjestelmäintegraatiossa tärkeäksi kysymykseksi nousee tietoturva. Kun järjestelmien välinen tiedonsiirto tapahtuu julkisen tietoverkon kuten Internetin kautta, tiedonsiirto on tietovuodon uhan alaisena. Tietoturvaa varten on olemassa tietoturvyökaluja, joiden avulla on mahdollista suojautua järjestelmäintegraation tietoturvariskeiltä. Tietoturvaratkaisuja voivat olla erilaiset VPN (Virtual Private Network)-ratkaisut tai siirrettävän informaation salaaminen ja allekirjoittaminen. [Tähtinen, 2005.]

3.4 Tietojärjestelmän kehittäminen

Tietojärjestelmän kehittämisen idea syntyy tarpeesta kehittää uutta tai ylläpitää vanhaa. Usein tietojärjestelmän kehitystyön taustalla voi olla asiakkaan tarpeet, uuden teknologian tarjoama hyöty ja uudet mahdollisuudet, tai jokin laajempi kartoitustyö, kuten *kokonaistutkimus*, jonka sivutuotteena myös tietojärjestelmä kokee kasvojen kohotuksen. [Pohjonen, 2002.]

Kun organisaatio kartoittaa tietojärjestelmiensä tilat ja selvittää kehitystarpeet, puhutaan kokonaistutkimuksesta. Kokonaistutkimus on pitkän tähtäimen selvitys organisaation tietotekniikan ja tietojenkäsittelyn tarpeista, jossa luodaan priorisoitu suunnitelma tietojärjestelmien kehityshankkeiden suhteen. [Pohjonen, 2002.]

Tietojärjestelmien kehittäminen on systemaattista työtä. Kehitystyön läpi vienti koostuu toinen toistaan seuraavista vaiheista, jotka sisältävät tiettyjä tehtäväkokonaisuuksia. Ohjelmistokehitysprojekteissa sovelletaankin usein ns. *Vesiputousmallia*, jossa tuotan-

toprosessi etenee suunnittelusta toteutukseen vaihe vaiheelta alaspäin kuin vesiputouksessa. Yleensä tietojärjestelmän kehittäminen alkaa selvityksestä, onko kehitysprojektin toteutus tarpeellista tai edes mahdollista. Tätä vaihetta kutsutaan *esitutkimukseksi*. Tämän jälkeen siirrytään *vaatimusmäärittely-* ja *analyysivaiheeseen*, jossa tehdään mahdollisimman tarkka määritelmä järjestelmän käyttötarkoituksesta ja käyttäjien tarpeista. Seuraavaksi seuraa järjestelmän *suunnittelu-*, *toteutus-* ja *testausvaiheet*, ja viimeiseksi valmiin tuotteen *ylläpitovaihe*. [Pohjonen, 2002.]

Esitutkimus

Kehitysprosessin ensimmäinen vaihe on esitutkimus, jonka tarkoitus on selvittää hankkeen toteuttamisen mahdollisuus ja organisaation edellytykset siihen. Esitutkimuksessa selvitetään, miksi uusi järjestelmä halutaan rakentaa, mitkä ovat kehityshankkeen tavoitteet ja mitä vaihtoehtoja sen toteuttamiselle on olemassa. Tässä vaiheessa ei siis rakenneta, ohjelmoida tai päätetä sovellettavien teknologioiden suhteen mitään, vaan määritellään mahdollisimman tarkat lähtökohdat mahdolliselle tietojärjestelmän kehityshankkeelle. Esitutkimuksen perusteella tehdään päätös kannattaako kehitysprojekti vai ei. [Pohjonen, 2002.]

Vaatimusmäärittely

Jos esitutkimuksen jälkeen kehityshankkeelle näytetään vihreää valoa, voidaan siirtyä vaatimusmäärittelyyn. Vaatimusmäärittely on dokumentti, joka sisältää asiakkaiden/sidosryhmien/käyttäjien tarpeet ja tietojärjestelmälle asetetut vaatimukset. Vaatimukset voidaan jakaa *toiminnallisiin* ja *ei-toiminnallisiin* vaatimuksiin. Dokumentti ei ota kantaa tekniseen toteutukseen eikä teknologioihin, joiden avulla vaatimukset käytännössä täytetään. [Pohjonen, 2002.]

Toiminnalliset vaatimukset kuvaavat sen, mitä tietojärjestelmän odotetaan tekevän. ”Tietojärjestelmän on tuettava kertakirjautumista” on yksi esimerkki toiminnallisesta vaatimuksesta. Ei-toiminnalliset vaatimukset taas määrittävät toimintakehykset toiminnallisille vaatimuksille. Esimerkiksi ”tietojärjestelmän on toimittava sekä Windows- että Mac-käyttöliittymällä” on ei-toiminnallinen vaatimus. On myös olemassa erikoistapauksia, *Rajoitteita*, jotka asettavat tietyt rajoitteet toiminnallisille vaatimuksille. ”Kertakirjautumisen tulee olla sallittua ainoastaan tietojärjestelmän omissa liitännäisissä” on esimerkki tällaisesta rajoitteesta. [Pohjonen, 2002.]

Vaatimusten keräämiseen ei ole yhtä yksittäistä keräysmenetelmää, vaan tarpeeksi kattavan lopputuloksen saavuttamiseksi on sovellettava erilaisia strategioita samanaikaisesti. Hyväksi todettu keräysmenetelmä on sidosryhmien edustajien haastattelu, jossa vaatimusten ohessa pyritään myös kerätä taustatietoa tietojärjestelmän tulevista käyttäjistä ja tietojärjestelmän ympäristöstä. Tärkeäksi tekijäksi nouseekin oikean haastateltavan löytäminen kustakin sidosryhmästä. [Pohjonen, 2002.]

Muita metodeja vaatimusten keräämiseen ovat esimerkiksi ryhmä ideointipalaverit ja aivoriihet. Myös markkinatutkimukset ja tietyille käyttäjäryhmille kohdenetut kyselyt voivat olla hyödyksi vaatimusten keräämisessä. Lisäksi olemassa olevia tietojärjestelmiä ja toimintatapoja voidaan hyödyntää, näitä tarkastelemalla ja analysoimalla. [Pohjonen, 2002.]

Vaatimukset tulisi dokumentoida mahdollisimman helposti luettaviksi, tulkittaviksi ja ymmärrettäviksi. Tulkinnallisesti epäselvät vaatimukset voivat aiheuttaa vaikeuksia kehityshankkeen myöhemmissä vaiheissa. Jos vaatimus on esitetty muodossa: ”järjestelmän tulee nopeuttaa tietokantakyselyä”, sen toteutumisen arviointi voi olla haasteellista. Edellä ilmaistun vaatimuksen tehokkuuden kasvua ei ole tarkemmin määritelty eikä sitä voida verrata mihinkään. Siksi sama vaatimus olisi järkevämpää kirjata ylös seuraavasti: ”järjestelmän tulee nopeuttaa automaattista tietokantakyselyä 35 % manuaaliseen verrattuna”. [Pohjonen, 2002.]

Asiakkaan vaatimusten kerääminen ja tarpeiden määrittäminen on tärkeä ja vaativa työvaihe. Se muodostaa keskeisen perustan tietojärjestelmän kehityshankkeelle ja dokumentointiin tulisi kiinnittää erityistä huomiota. Risto Pohjosen kirjan ”Tietojärjestelmien kehittäminen” [Pohjonen, 2002] mukaan vaatimusmäärittelydokumentista tulisi löytyä seuraavat asiat:

- kuvaus kehittämishankkeen toimeksiannosta
- yleiskuvaus organisaation nykytilanteesta
- kuvaus tietojärjestelmästä ja sille asetetuista tavoitteista
- jokaisen toiminnallisen vaatimuksen kuvaus
- jokaisen ei-toiminnallisen vaatimuksen kuvaus
- jokaisen rajoitteen kuvaus
- vaatimukset ja rajoitteet numeroituina ja priorisoituina

- mahdolliset lisäselvitykset.

Usein tietojärjestelmää rakennetaan versio kerrallaan. Tällöin myös vaatimukset olisi syytä priorisoida, sillä toiminnallisuuksia ei luonnollisestikaan saada mahtumaan yhteen versioon. Myös aikataulu- tai kustannusongelmat saattavat vaikuttaa kehityshankkeen etenemiseen, jolloin prioriteetiltään vähiten merkitsevistä toiminnallisuuksista voidaan jopa kokonaan luopua. [Pohjonen, 2002.]

Järjestelmäanalyysi

Vaatimusmäärittelyn jälkeen suoritetaan kehitettävän järjestelmän määrittely, jota kutsutaan järjestelmäanalyysiksi. Analyysin tarkoitus on selvittää kehitettävän järjestelmän käyttötarkoitus ja lopuksi rakentaa toiminnallinen määrittely analysoimalla edeltävässä vaiheessa tunnistetut vaatimukset. Toiminnallisen määrittelyn tulisi sisältää kuvaukset järjestelmän käyttötarkoituksesta, toiminnasta, käyttäjistä, rajoitteista, jokaisesta toiminnosta, tietokannoista ja rajapinnoista. Lisäksi järjestelmän suorituskyky, käytettävyys, tietoturvalliset kysymykset ja toteutusta rajoittavat tekijät, kuten standardit ja teknologiarajoitteet tulisi olla määritelty. [Pohjonen, 2002.]

Järjestelmäanalyysin selvitykset olisi syytä dokumentoida huolellisesti, sillä kehityshankkeen seuraavat vaiheet ovat hyvin riippuvaisia järjestelmän toiminnallisesta määrittelystä. Lisäksi järjestelmän kehittäminen voidaan aikataulullisista syistä tai resurssisyistä jakaa useampaan samanaikaisesti suoritettavaan työhön, jolloin jako suoritetaan toiminnallisen määrittelyn pohjalta. [Pohjonen, 2002.]

Suunnittelu

Järjestelmäanalyysia seuraa suunnitteluvaihe. Tämän vaiheen tärkeys heijastuu sille asetetuista tavoitteista, jotka ovat selkeys, ymmärrettävyys, tehokkuus, luotettavuus, ylläpidettävyys ja siirrettävyys. Suunnittelussa luodaan määrittely, joka kuvaa järjestelmän toteutuksen. Edellisessä vaiheessa asiakkaan vaatimusten pohjalta tehty toiminnallinen määrittely on nyt muutettava *tekniseksi määrittelyksi*. [Pohjonen, 2002.] Suunnittelu voidaan jakaa kahteen osaan: arkkitehtuuri- ja moduulisuunnitteluun. Arkkitehtuuri- ja moduulisuunnittelua on jo käsitelty tämän luvun toisessa kappaleessa.

Teknisen määrittelyn dokumentaation tulisi käydä ilmi ainakin seuraavat asiat:

- tiivistelmä käyttötarkoituksesta
- sovellusalue ja järjestelmän rooli
- sovelletut teknologiat
- arkkitehtuurisuunnittelu
- moduulisuunnittelu
- järjestelmän toteutuksen reunaehdot
- toteutusrajoitteiden määrittelyt
- teknisten ratkaisujen kuvaukset
- vaihtoehtoisten tai hylättyjen ratkaisujen kuvaukset
- muut toteutukseen vaikuttavat asiat.

Kaikki edellä mainitut seikat ovat tärkeitä osasia teknistä määrittelyä, joiden pohjalta järjestelmä toteutetaan. Pätevästä arkkitehtuurisuunnittelusta seuraa onnistunut moduulijako, joka on järjestelmäanalyysin kannalta keskeinen tekijä ja edellytys onnistuneelle toteutuksen suunnittelulle. [Pohjonen, 2002.]

Toteutus

Suunnittelua seuraa järjestelmän toteutus. Tietojärjestelmä toteutetaan valituilla taustakehyksillä ja ohjelmointikielellä. Usein tietojärjestelmä kehitetään osissa, moduuli kerrallaan, jolloin lopuksi valmiit ohjelmamoduulit integroidaan yhdeksi toimivaksi kokonaisuudeksi. Itse toteutusvaihe on suoraviivaista tekemistä, sillä kaikki tärkeimmät järjestelmän rakennetta ja toiminnallisuutta koskevat seikat on määritelty jo aikaisemmissa vaiheissa. [Pohjonen, 2002.]

Nykyään markkinoilta on saatavilla lukuisia teknologioita ja tietojärjestelmän toteutusta tukevia työkaluja, jotka antavat toteutukselle tietyt kehykset. Sopivan taustakehyksen, tyylikehyksen ja ohjelmointikielen valinta määräytyy tietojärjestelmän käyttötarkoituksen ja sovelletun järjestelmäarkkitehtuurin mukaan. Tietyt käyttötarkoitukset edellyttävät tietyn toteutusvälineen käyttöä. Lisäksi tietyt teknologiat toimivat paremmin tietyn arkkitehtuurimallin kanssa ja toiset teknologiat toimivat paremmin toisen arkkitehtuurimallin kanssa. [Pohjonen, 2002.]

Onnistunut toteutusteknologian ja ohjelmointikielen valinta on kuitenkin vasta yksi kolikon puolista. Tärkeimmäksi tekijäksi nousee se, kuinka hyvin järjestelmä vastaa sille

asetettuja vaatimuksia ja tavoitteita sekä toiminnallisia ja teknisiä määrittelyjä. Avainasemassa onkin edellisten vaiheiden huolellinen läpivienti ja onnistunut suunnittelu. [Pohjonen, 2002.]

Toteutuksessa kannattaa myös kiinnittää huomiota ohjelmointityyliin. Tietojärjestelmän elinkaaren aikana sen ylläpidosta vastaavat henkilöt voivat vaihtua useaan otteeseen. Tämän takia on tärkeää ohjelmoida sellaisella tyylillä, että koodista saa selvää ja sitä pystyy seuraamaan joku muukin kuin sen alkuperäinen laatija. Hyvän dokumentoinnin lisäksi hyviksi todettuja keinoja selkeän koodin tuottamiseen on tuotetun koodin kommentoiminen ja järjestelmällisten sekä kuvaavien nimeämiskäytäntöjen noudattaminen. [Pohjonen, 2002.]

Testaus

Ennen käyttöönottoa toteutettu tietojärjestelmä on vielä testattava virheiden varalta. Testausta varten on kehitetty ns. V-malli, jolloin testaaminen jaetaan kolmeen eri tasoon: moduulitestaukseen, integrointitestaukseen ja järjestelmätestaukseen.

Moduulitestaus suunnitellaan moduulisuunnittelun yhteydessä ja siinä etsitään nimensä mukaan virheitä yksittäisistä moduuleista. Vastaavasti integrointitestaus suunnitellaan jo arkkitehtuurisuunnittelun yhteydessä ja siinä etsitään virheitä koko järjestelmän moduulien välisestä kommunikoinnista. Järjestelmätestauksessa puolestaan etsitään vikoja ja koko järjestelmän toiminnoista ja suorituskyvystä vertaamalla valmista järjestelmää sen toiminnalliseen määrittelyyn. Testaus ja testiaineisto ei siis perustu ohjelmakoodiin vaan määrittelyn ja suunnittelun tuloksiin. [Pohjonen, 2002.]

Käyttöönotto

Järjestelmä voidaan ottaa käyttöön, kun testaus on suoritettu kokonaisuudessaan. Järjestelmän käyttöönotto ei suinkaan ole selvä yhdellä napin painalluksella. Uuden järjestelmän käyttöönottoon liittyviä tekijöitä on useita, ja ne tulee huomioida ja valmistella huolellisesti ja ajoissa. [Pohjonen, 2002.]

Usein tietojärjestelmän kehittäminen on nimenomaan kehitysprojekti, jonka tarkoitus on kehittää uudempi/parempi/päivitetty versio vanhasta järjestelmästä. Tällöin on otettava huomioon myös vanhan järjestelmän olemassaolo ja mahdollisten tiedostojen ja tieto-

kantojen siirtäminen uuteen järjestelmään. Myös järjestelmän käyttäjät ja ylläpitohenkilöstä vaativat perehdytyksen. Joissakin tapauksissa koulutusten järjestäminen on ehdotonta, jos uusi järjestelmä poikkeaa radikaalisti vanhasta ja siihen on ladattu liuta uusia työkaluja ja toiminnallisuuksia. Usein taas pelkkä asianmukainen käyttöohjeistus riittää, etenkin kokeneiden järjestelmän käyttäjien keskuudessa. [Pohjonen, 2002.]

Ylläpito

Käyttöönoton jälkeen seuraa tietojärjestelmän elinkaaren pisin yksittäinen vaihe: ylläpito. Ylläpito huolehtii järjestelmän mutkattomasta toiminnasta korjaamalla ilmeneviä virheitä ja parantamalla järjestelmän ominaisuuksia jatkokehityksellä päivitysten muodossa. Ylläpito vaihe kestää aina järjestelmän käyttöönotosta sen käytön loppuun saakka. [Pohjonen, 2002.]

Ylläpito voidaan lokeroittaa neljään eri tapaukseen: *korjaavaan*, *sopeuttavaan*, *täydentävään ja ennakoivaan* ylläpitoon. Korjaava ylläpito vastaa tuotantokäytössä olevan järjestelmän virheiden korjaamiseen. Sopeuttavasta ylläpidosta puhutaan, kun järjestelmä halutaan siirtää uusiin ympäristöihin. Täydentävä ylläpito puolestaan vastaa järjestelmän jatkokehityksestä, ja ennakoiva ylläpito huolehtii järjestelmän tarkasta dokumentaatiosta helpottaakseen järjestelmän tulevia ylläpitotilanteita. [Pohjonen, 2002.]

Ylläpidon kannalta on tärkeää, että järjestelmän dokumentaatio on jatkuvaa ja kattavaa aina ensimmäisestä kehitysvaiheesta lähtien. Kun täydentävä ylläpito suunnittelee järjestelmään päivityksiä tai uusia ominaisuuksia, on ehdottoman tärkeää ymmärtää järjestelmän taustalla vaikuttavia suunnittelu- ja toteutusratkaisuja. Nämä taustatekijät olisi syytä olla dokumentoitu kattavasti, selkeästi ja ymmärrettävästi, jouhevamman jatkokehittämisen kannalta. On myös tärkeää, että jatkokehityksen aikana toteutetut lisäykset ja järjestelmän muutokset dokumentoidaan selkeästi ja kattavasti. [Pohjonen, 2002.]

Usein tietojärjestelmä voi koostua monesta eri versiosta, jotka on käyttöönotettu eri aikoina. Samasta järjestelmästä on nimittäin voitu tehdä asiakaskohtaisia räätälöityjä versioita. Tällaisen järjestelmän ylläpito ei ole enää niin yksinkertaista. Ongelmaksi voi muodostua tiettyä versiota varten laadittu päivitys, joka aiheuttaa ei-toivottuja sivuvaikutuksia toisissa järjestelmän versioissa, mikä johtuu siitä, että järjestelmän moduulit ovat voimakkaasti riippuvaisia toisistaan. Ongelma voidaan kuitenkin osittain taklata jo

arkkitehtuurisuunnittelussa kapseloimalla toiminnot erillisiin, mahdollisimman itsenäisiin ”säiliöihin”, jolloin järjestelmän virheiden paikallistaminen on nopeaa ja niiden korjaaminen on helppoa aiheuttamatta sivuvaikutuksia tai häiriöitä järjestelmän muihin moduuleihin. [Pohjonen, 2002.] Juuri tähän palvelupohjainen arkkitehtuuri pyrkii.

Ylläpidon ollessa tietojärjestelmän elinkaaren pisin yksittäinen vaihe on hyvä muistaa, että se syö valtavan osan kehitysprojektille osoitetuista resursseista. Tämä on otettava huomioon jo määrittely- ja suunnitteluvaiheessa tarjoamalla hyvällä dokumentaatiolla ja suunnittelulla mahdollisimman helpot asetelmat ylläpitovaiheen toimenpiteille. Tuolloin ylläpitovaiheen helpottamiseen sijoitetut resurssit voivat vaikuttaa hukkaan heitetyiltä, mutta todellisuudessa tästä saatu hyöty voi säästää huomattavia summia kokonaisbudjetista. Ylläpitovaiheen syödessä jopa 70 % järjestelmään suunnatuista resursseista pitkän tähtäimen suunnittelu on ensiarvoisen tärkeää. [Pohjonen, 2002.]

4 Sovelletut tietojärjestelmän teknologiat

Suunniteltaessa tietojärjestelmää on valittava kehityksessä sovellettavat ohjelmointikieliset, ohjelmistokehykset ja teknologiat. Tässä luvussa esitellään lyhyesti tämän insinöörityön kannalta oleelliset ohjelmointikieliset, sovelluskehykset ja teknologiat, joista Peppi-järjestelmä ja nykyinen työharjoittelujärjestelmä on rakennettu.

Ohjelmointikieli on väline, jolla tietokonetta ohjelmoidaan ja ohjataan. Ohjelmointikieliä on erilaisia, ja ne voidaan jaotella eri ryhmiin esimerkiksi ohjelmointimallien mukaan. Internetsivujen ja -sovellusten ohjelmoinnissa käytetään usein olio-ohjelmointia tukevia kieliä, kuten Javaa ja C++ -kieltä. [Paananen, 2005.]

Sovelluskehys on teknologia, joka helpottaa ja nopeuttaa tietokoneohjelmiston kehittämistä. Sovelluskehysten käyttö perustuu uudelleen käytettäviin suunnittelumalleihin ja komponentteihin, joista ohjelmistot pääosin koostuvat. Kun jokaista koodin pätkää ei tarvitse enää kirjoittaa uudelleen, säästyy aikaa ja resursseja, mikä puolestaan näkyy tuottavuuden kasvussa. Markkinoilla on kymmeniä sovelluskehysteknologioita, jotka on suunniteltu kattamaan erilaisten ohjelmistojen ja käyttöjärjestelmien tarpeet. Sovelluskehysten toteutusteknologiat, toiminnallisuudet ja ominaisuudet sekä yhteen pelaaminen muiden sovelluskehityksessä käytettävien teknologioiden kanssa vaihtelee näiden käyttötarpeiden ja sovellusratkaisujen mukaan. [Riehle, 2000.]

Java

Java on suuren suosion saavuttanut järjestelmäriippumaton oliopohjainen ohjelmointikieli. Se muistuttaa syntaksiltaan suosittua C++ -kieltä, mutta myös eroaa siitä monin tavoin. Järjestelmäriippumattomuus mahdollistaa saman koodin toimimisen eri käyttöjärjestelmissä ja -ympäristöissä, joiden määrä kasvaa jatkuvasti. Java-ohjelmia voidaan siis ajaa Linuxissa, Windowsissa ja Macissa sekä esimerkiksi matkapuhelimissa ja digitelevisioissa. [Java perusteita.]

PHP

PHP on dynaamisten web-sivujen toteutukseen suunniteltu ohjelmointikieli. PHP-kieli lukeutuu tulkittaviin ohjelmointikieliin, joita kutsutaan yleisesti komentosarjakieliksi eli skriptikieliksi. PHP-koodi upotetaan HTML-kielen sekaan ja suoritetaan palvelimella

ennen kuin se lähetetään selaimelle. Näin ollen PHP-kieli ei vaadi tukea selaimelta, ja sitä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä. Syntaksiltaan PHP-koodi muistuttaa mm. C-kieltä ja Javaa. [PHP-ohjelmoinnin perusteet.]

Apache Struts

Apache Struts on yksi maailman suosituimmista sovelluskehysistä. Se on avoimen lähdekoodin servlet-teknologiaan perustuva teknologia, jota hyödynnetään Java pohjaisten web-sovellusten kehittämisessä. [Apache Struts]

Liferay Portal

Liferay Portal on Javalla toteutettu avoimen lähdekoodin työkalu, jota käytetään erityisesti monista palveluista koostuvien verkkosivujen eli portaalien toteutuksessa. Liferay Portal sisältää yli 60 erilaista portlettia, eli pienempää verkkosovellusta, liitännäistä tai teemaa, joista voi helposti rakentaa monista palveluista koostuvan verkkosivun ilman, että tarvitsisi itse kirjoittaa ohjelmointikoodia. Liferay Portal tukee palvelupohjaisen arkkitehtuurimallin periaatteita ja hyödyntää mm. Rest- ja SOAP-tyyppisiä rajapintoja. [Liferay Portal]

Apache ServiceMix

Apache ServiceMix on Javalla toteutettu avoimen lähdekoodin integraatioalusta, joka on rakennettu Apachen omien teknologioiden pohjalta. ServiceMixin avulla voidaan hyödyntää samassa paketissa mm. Apache ActiveMQ:n, Camelin ja CXF:n toimintaa. ServiceMixi on ESB-alusta (Enterprise Service Bus), jonka avulla on mahdollista liittää toisiinsa erilaisilla arkkitehtuureilla toteutettuja järjestelmiä. ServiceMix on yleisesti käytetty teknologia palvelupohjaisen arkkitehtuurimallin sovelluksissa. [Apache ServiceMix.]

Apache ActiveMQ on teknologia, jolla toteutetaan järjestelmän komponenttien välinen viestittely, ja Apache Camel on monipuolinen ja selkeä sovelluskehys, joka toimii reitintopalveluna helpottaen järjestelmäintegraatioiden tekemistä. Apache CXF on puolestaan *frontend*-kehityksessä käytetty sovelluskehys, joka hyödyntää kaikkia yleisimpiä rajapintateknologioita. [ActiveMQ, Camel, CXF.]

MariaDB

MariaDB on yksi maailman suosituimmista avoimen lähdekoodin tietokannoista, jota käyttävät muun muassa Wikipedia ja Google. MariaDB on nopea, skaalautuva ja vahva SQL-tyyppinen tietokanta, joka soveltuu hyvin monenlaisten sovellusratkaisujen tietokannaksi hyvien ominaisuuksiensa tähden. [MariaDB.]

Elasticsearch

Elasticsearch on suosittu Javalla toteutettu avoimen lähdekoodin hakukoneteknologia, joka kykenee hakemaan suuria määriä dataa hyvin nopeassa ajassa. Elasticsearch hyödyntää HTTP protokollaa ja JSON-tiedostoformaattia REST-tyyppisten rajapintojen avulla, mikä tekee datasta helposti haettavaa ja luetteloitavaa. [ElasticSearch.]

Symfony

Symfony on PHP:lla toteutettu avoimen lähdekoodin sovelluskehys, jota käytetään Internetsivujen ja -sovellusten kehittämisessä. Symfonyn päällimmäisenä ominaisuutena on uudelleen käytettävät PHP-komponentit ja kirjastot. [Symfony.]

Foundation

Foundation on avoimen lähdekoodin frontend-sovelluskehys. Se sisältää uudelleen käytettäviä HTML-, CSS- ja JavaScript-komponentteja ja suunnittelumalleja, joita voi hyödyntää Internet-sivujen kehittämisessä. [Foundation.]

Shibboleth SP

Shibboleth Service Provider on standardoitu avoimen lähdekoodin ohjelmisto, jonka avulla voidaan toteuttaa web-sovellusten kertakirjautumisominaisuus. [Shibboleth.]

5 Peppi

Metropolia Ammattikorkeakoulu käyttää opetuksen ja koulutuksen suunnittelun sekä toteutuksen tukena Peppi-järjestelmää. Tässä luvussa käydään läpi Peppi-järjestelmää yleisellä tasolla perehtyen tarkemmin sen arkkitehtuuriin.

5.1 Peppi-järjestelmä

Peppi on Peppi-konsortion kehittämä järjestelmäkokonaisuus, joka on suunniteltu tukemaan eri koulutusorganisaatioiden opetukseen ja koulutukseen liittyviä prosesseja. Peppi-konsortio on eri koulutusorganisaatioiden yhtymä, joka hallinnoi ja kehittää Peppi-järjestelmäkokonaisuutta. Konsortio on alun perin perustettu kahden koulutusorganisaation, Metropolia Ammattikorkeakoulun sekä Tampereen ammattikorkeakoulun toimesta. Konsortio on avoin kaikille suomalaisille koulutusorganisaatiolle, ja nykyään siihen kuuluu yli 80 % Suomen ammattikorkeakouluista ja yliopistoista sekä kaksi toisen asteen ammatillista oppilaitosta (helmikuu 2018). [Peppi-konsortio.]

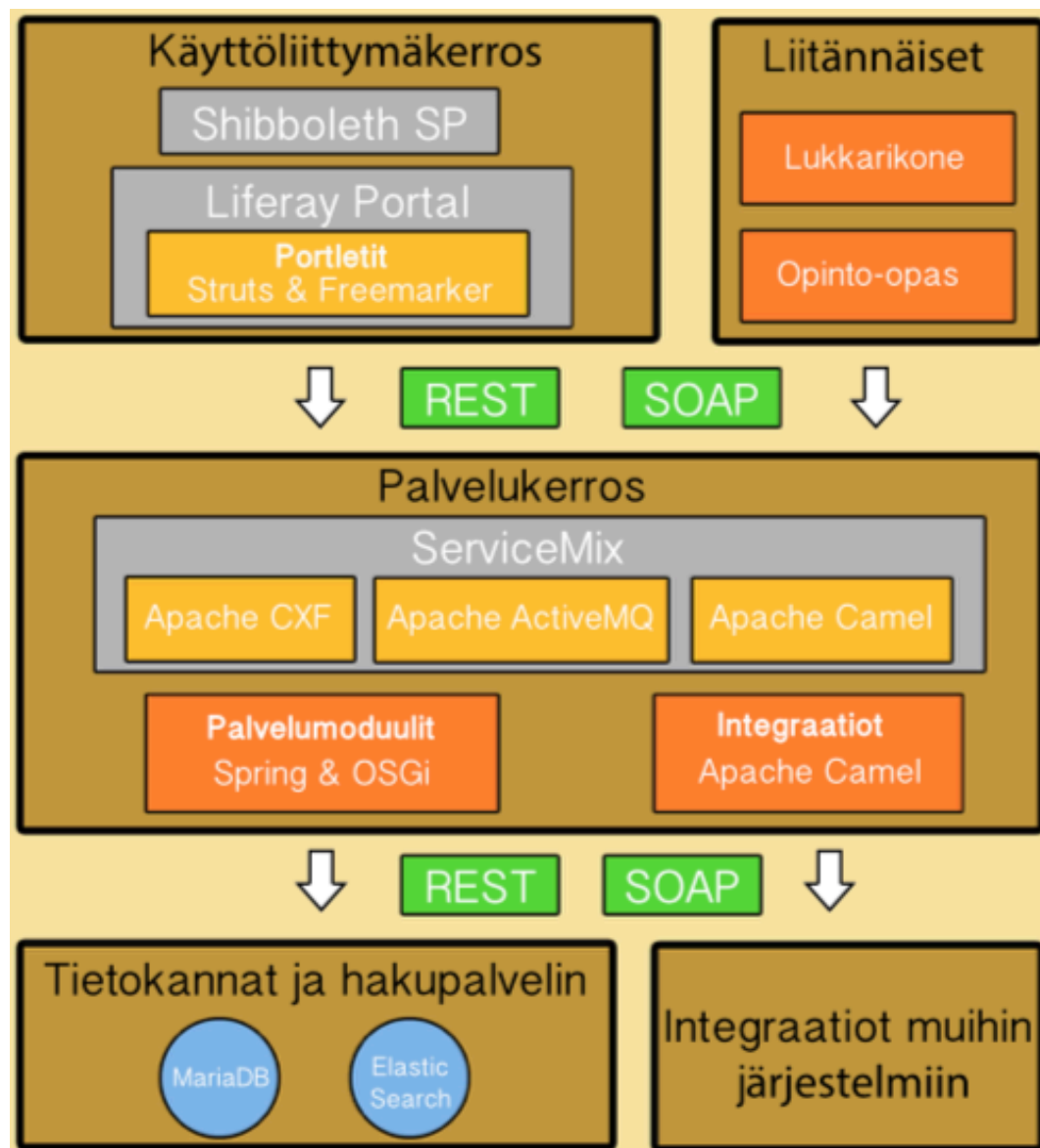
Pepin käyttäjiä ovat opiskelijat, opettajat, opintohallinto sekä järjestelmän ylläpito. Jokaisella käyttäjäryhmällä on käytössään oma räätälöity käyttöliittymä, jonka avulla järjestelmää käytetään omissa prosesseissa ja työssä. Tarkemmin kuvailtuna nämä räätälöidyt käyttöliittymät ovat roolikohtaisia sähköisiä työpöytiä, joihin on määritelty tietyt ominaisuudet ja toiminnallisuudet. Esimerkiksi opiskelija käyttää Peppiä sekä yleisen työpöydän että opiskelijan oman työpöydän kautta. Yleinen työpöytä on käytännössä Metropolian Intranet, jonka kautta opiskelija pystyy seuraamaan koulun tapahtumia ja opintoja koskevia tiedotteita, työpaikkailmoituksia ja uutisia. Opiskelijan työpöytä taas toimii enemmän työkaluna, johon kuuluvat tietyt toiminnallisuudet ja ominaisuudet, joita opiskelija hyödyntää opintojensa suunnittelussa ja hallinnassa. [Peppi-konsortio.]

5.2 Pepin arkkitehtuuri

Pepin arkkitehtuurin perusteknologiana toimii Java ja erilaiset avoimen lähdekoodin palvelinohjelmat ja sovelluskehikset. Peppi-järjestelmä on toteutettu palvelupohjaisen arkkitehtuurin mukaisesti erottamalla palvelut käyttöliittymistä erillisiksi moduuleiksi. Käyttöliittymät ja palvelut kommunikoivat keskenään Rest- ja SOAP-standardien mukaisen rajapintojen avulla. Palvelumoduuleista erotettuja käyttöliittymiä voidaan hel-

posti uudistaa aiheuttamatta sivuvaikutuksia palvelukerroksen toimintaa. Uusien käyttöliittymien lisääminen järjestelmään on myös helppoa olemassa olevien palvelurajapintojen vuoksi. [Lavikainen, Arkkitehtuurin kuvaus.]

Peppi-järjestelmän rakennetta on helpompi tutkia jakamalla se osiin. Järjestelmän oma rakenne voidaan jakaa kolmeen kerrokseen: *käyttöliittymä-, palvelin- ja tietokantakerrokseen*. Lisäksi järjestelmään kuuluvat ulkoiset, Pepin ulkopuolelle toteutetut *liitännäiset ja integraatiot muihin järjestelmiin*.



Kuva 4. Havainnollistava esitys Peppi-järjestelmän arkkitehtuurista.

Käyttöliittymäkerros

Pepin käyttöliittymien sovelluskehysinä on käytetty Apache Strutsia ja FreeMarkeria. Käyttöliittymäkerroksessa on hyödynnetty Liferay portal-teknologiaa, jonka avulla käyttöliittymämoduulit, eli portletit, koostetaan roolipohjaisiksi sähköisiksi työpöydiksi.

Peppi-järjestelmä tukee kertakirjautumista, ja se toteutetaan oletuksellisesti Shibbolethin avulla. Kertakirjautuminen on menetelmä, jonka avulla käyttäjä voi yhdellä kirjautumisella käyttää koko järjestelmäkokonaisuuden eri sovelluksia. Tällä menetelmällä vähennetään ylimääräisiä käyttäjän tunnistamisia ja kirjautumisia organisaation sisäisten sovellusten välillä. [Lavikainen, Arkkitehtuurin kuvaus.]

Palvelukerros

Pepin palvelukerros muodostuu erillisistä palvelumoduuleista, jotka on toteutettu hyödyntämällä Apache ServiceMix in tuotteita sekä Spring-sovelluskehysten avulla tehtyjä OSGi-bundleja. Tämän ratkaisun idea pohjautuu siihen, että palvelut saadaan kytkettyä toisiinsa OSGi:n tarjoaman palvelurekisterin avulla. OSGi:n avulla Pepin palveluista voidaan rakentaa palvelukokonaisuuksia, jotka toimivat tehokkaasti ServiceMix in ympäristössä. [Lavikainen, Arkkitehtuurin kuvaus.]

Palvelukerroksen toteutuksessa on siis käytetty useita eri teknologioita. Ratkaisu auttaa palvelujen resurssitarpeiden tarkemmassa hallinnassa, ja palvelut voidaan myös jakaa erilleen niiden tietosuojatasojen mukaan. Palvelumoduulit ovat rakenteeltaan samanlaisia ja kommunikoivat keskenään Rest- ja SOAP-tekniikalla toteutettujen rajapintojen kautta. [Lavikainen, Arkkitehtuurin kuvaus.]

Tietokantakerros

Pepin tietokantatyypinä on SQL, ja järjestelmä käyttää MariaDB-tietokantaa. Alunperin järjestelmä käytti myös MySQL-tietokantaa, josta on sittemmin luovuttu kokonaan. [Lavikainen.]

Tietokannan lisäksi palvelut käyttävät Elasticsearch-hakupalvelinta, jonka avulla järjestelmään on toteutettu monipuoliset hakutoiminnot ja nopeat lukurajapinnat. [Lavikainen, Arkkitehtuurin kuvaus.]

Liitännäiset

Pepi-järjestelmään kuuluu myös integroituja sovelluksia, eli liitännäisiä, kuten esimerkiksi lukkarikone ja opinto-opas. Järjestelmän ulkopuoliset sovellukset hyödyntävät samoja Rest- ja SOAP-rajapintoja kuin käyttöliittymät. Tietoturvalisistä syistä Pepin ulkopuolisilta järjestelmiltä on evätty kirjoitusoikeudet, eli sovellukset voivat hakea esimerkiksi käyttäjätietoja tietokannoista, mutta tiedon siirtäminen järjestelmän ulkopuolisesta lähteestä itse järjestelmään ei ole mahdollista. [Lavikainen.]

Palvelupohjainen arkkitehtuuri mahdollistaa järjestelmän ulkopuolisten sovellusten integroimisen jatkossakin, eikä konsortio aseta liitännäisten toteutukselle mitään rajoituksia. [Lavikainen.]

Integraatiot muihin järjestelmiin

Pepin integraatiot muihin järjestelmiin on toteutettu Apache Camel -teknologialla. Camelin avulla voidaan toteuttaa integraatiot vanhemmalla teknologialla toteutettuihin järjestelmiin, joihin ei ole käytössä moderneja etärajapintoja. [Lavikainen, Arkkitehtuurin kuvaus.]

6 Työharjoittelujärjestelmän nykytila-analyysi

Tässä luvussa käydään läpi tekniikan ja liiketalouden nykyistä työharjoittelujärjestelmää ja sen rakennetta, ominaisuuksia ja eroavaisuuksia. Lopuksi tarkastellaan myös nykyisen järjestelmän puutteita ja heikkouksia, jotka ovat tulleet esille harjoitteluinsinöörien sekä harjoittelusta vastaavien opettajien haastatteluissa.

6.1 Työharjoittelujärjestelmä

Metropolia Ammattikorkeakoulun nykyinen työharjoittelujärjestelmä on selainpohjainen ja löytyy osoitteesta harjoittelu.edu.metropolia.fi. Työharjoittelujärjestelmän keskeisin tehtävä on toimia opiskelijan työharjoitteluprosessia tukevana työkaluna, joka helpottaa harjoitteluun liittyvien prosessien hoitamista. Järjestelmällä on siten kaksi pääkäyttäjärhmää: harjoittelusta vastaavat opettajat ja opiskelijat. Opettajat käyttävät työharjoittelujärjestelmää harjoittelupaikan vahvistamiseen, opiskelijan harjoittelun aikaiseen ohjaamiseen ja lopuksi työharjoittelun arviointiin. Opiskelijoille työharjoittelujärjestelmä on työkalu, johon he keskittävät koko työharjoitteluprosessinsa aikaiset toimenpiteet. [Aalto, Rupponen.]

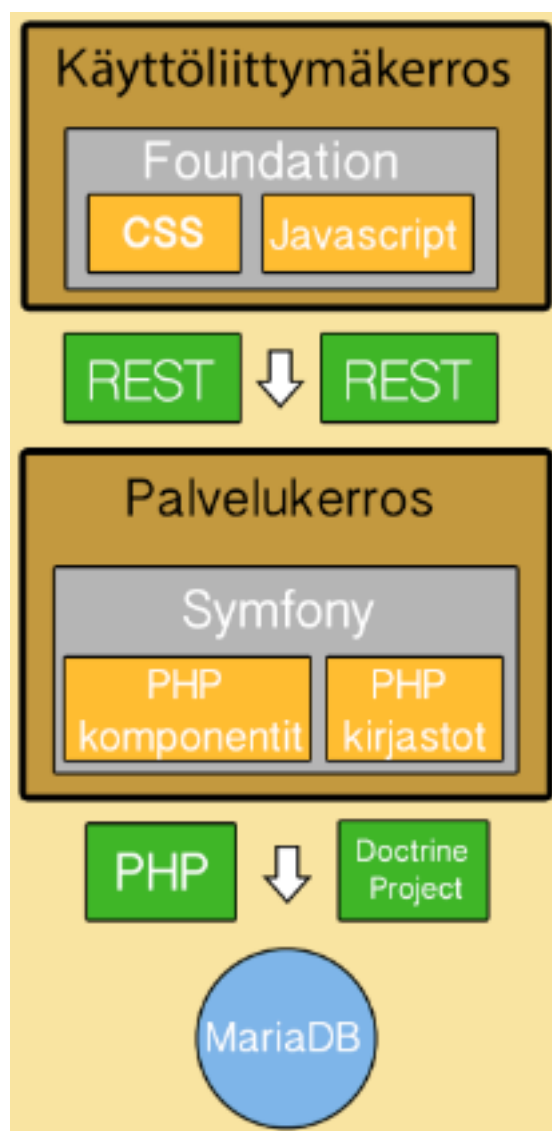
Työharjoittelujärjestelmän ovat aikoinaan kehittäneet Metropolia Ammattikorkeakoulun tieto- ja viestintätekniikan opiskelijat. Tieto- ja viestintätekniikan opiskelijoista koostuva ryhmä tunnetaan tutummin Karhukoplana, joka on Metropolia Ammattikorkeakoulun entisen opettajan perustama tieto- ja viestintätekniikan opiskelijoista koostuva ryhmä. Karhukoplan kokoonpano on aina vaihdellut uusien ja valmistuvien opiskelijoiden mukaan. Nykyinen Karhukopla koostuu kahdesta opiskelijasta, jotka ovat aloittaneet opin- tonsa syksyllä 2014. Itse Karhukoplassa he ovat toimineet toisen opiskeluvuoden kevästä lähtien. [Karhukopla.]

6.2 Työharjoittelujärjestelmän arkkitehtuuri

Työharjoittelujärjestelmä on toteutettu MVC-arkkitehtuurilla, joka on yleinen arkkitehtuurimalli graafisten käyttöliittymien suunnittelussa ja ohjelmoinnissa. Järjestelmän ohjelmointikielenä on käytetty PHP:tä, jonka lähdekoodi on päivitetty kesän 2017 aikana viimeisimpään PHP 7-versioon. [Karhukopla.]

Työharjoittelujärjestelmän toteutuksessa on hyödynnetty kahta erilaista sovelluskehystä. Palvelinkerros on toteutettu Symfony-tekniologialla ja käyttöliittymässä on puolestaan hyödynnetty Foundation-tekniologiaa. Tämän lisäksi käyttöliittymässä on käytetty myös JavaScript-komentosarjakieltä. Pepin tavoin myös työharjoittelujärjestelmä hyödyntää Rest-tyyppistä rajapintaa. [Karhukopla.]

Työharjoittelujärjestelmän tietokantatyypinä on SQL, ja järjestelmän käyttämä tietokanta on Pepin tavoin MariaDB. Vaikka PHP itsessään tukee suoria tietokantakyselyjä, kyselyjen helpottamiseksi järjestelmässä on myös hyödynnetty Doctrine Project -tekniologiaa, joka tarjoaa valmiita PHP-kirjastoja. [Karhukopla.]

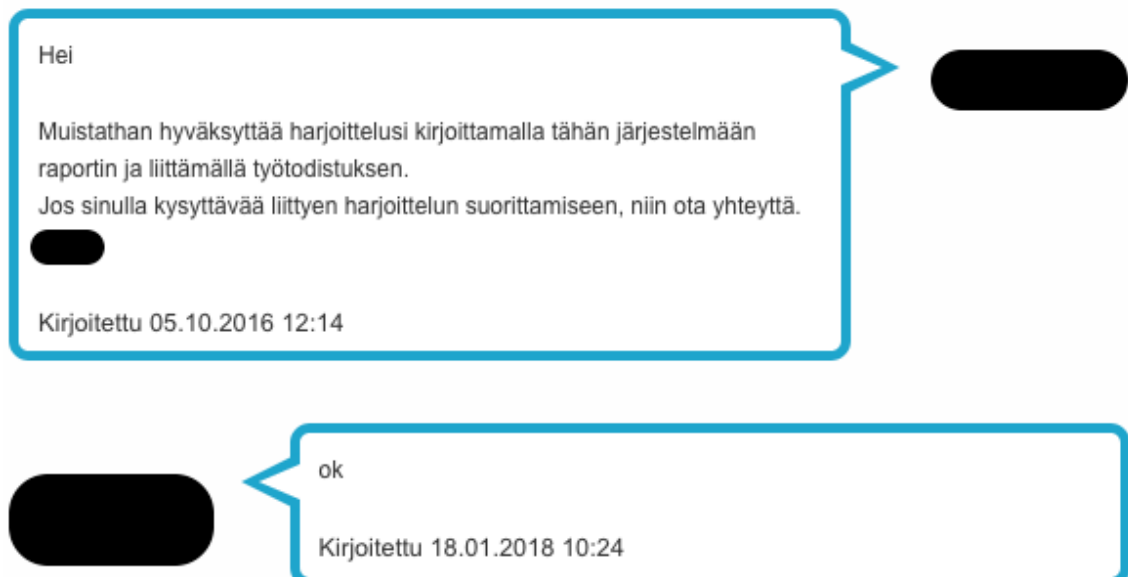


Kuva 5. Kuvaus työharjoittelujärjestelmän arkkitehtuurista.

6.3 Toiminnot ja ominaisuudet

Työharjoittelujärjestelmän keskeisimmät toiminnot rakentuvat työharjoitteluprosessin ympärille. Opettajien kannalta tärkeimmät toiminnot ovat opiskelijan ilmoittaman työharjoittelupaikan hyväksyminen, ohjaaminen ja lopulta itse harjoittelun hyväksyminen. Opiskelijan tehdessä työharjoittelupaikkailmoituksen järjestelmä osaa itse laskea opiskelijan ilmoittaman ajankohdan mukaisen laajuuden, mikä auttaa opettajaa arvioimaan onko harjoittelu aika tarpeeksi pitkä vaadittujen opintopisteiden ansaitsemiseksi. Opiskelijan harjoittelupaikkailmoituksesta selviää luonnollisesti myös yritys, jonne työharjoittelu suoritetaan ja opiskelijan päätehtävät kyseisessä positiossa. Näin ollen opettajan on myös helppo arvioida, onko harjoittelu vaativuudeltaan sopiva ja täyttääkö tämä esimerkiksi ammattiharjoittelujakson vaatimat kriteerit. [Aalto, Rupponen.]


Tarvittaessa harjoittelusta vastaava opettaja voi ohjata opiskelijaa työharjoittelujärjestelmän työpöydän chat-toiminnon avulla. Chat-toiminnolla opettaja ja opiskelija voivat vaihtaa reaaliajassa viestejä. Vastaanotetusta viestistä tulee myös aina sähköpostivahvistus, joka lähetetään automaattisesti opiskelijan tiedoista löytyvään sähköpostiosoitteeseen. [Aalto.]



Kuva 6. Esimerkki opiskelijan ja opettajan välisestä chat-keskustelusta työharjoittelujärjestelmän avulla.

Opettaja hyväksyy tai hylkää opiskelijan työharjoittelun tämän laajuuden ja opiskelijan kirjoittaman harjoitteluraportin sekä hänen toimittamansa työtodistuksen perusteella.

Itse arviointia opettaja ei kuitenkaan pysty työharjoittelujärjestelmän kautta tekemään, vaan se on tehtävä Pepin perusrekisterin kautta. Kun opiskelija on hyväksyttävästi suorittanut molemmat työharjoittelunsa, hänen työharjoittelujärjestelmän työpöydälle ilmestyy ”kultainen pokaali”, joka viittaa koko opetussuunnitelman mukaisen hyväksytysti suoritettuun työharjoittelujaksoon. [Aalto, Rupponen.]



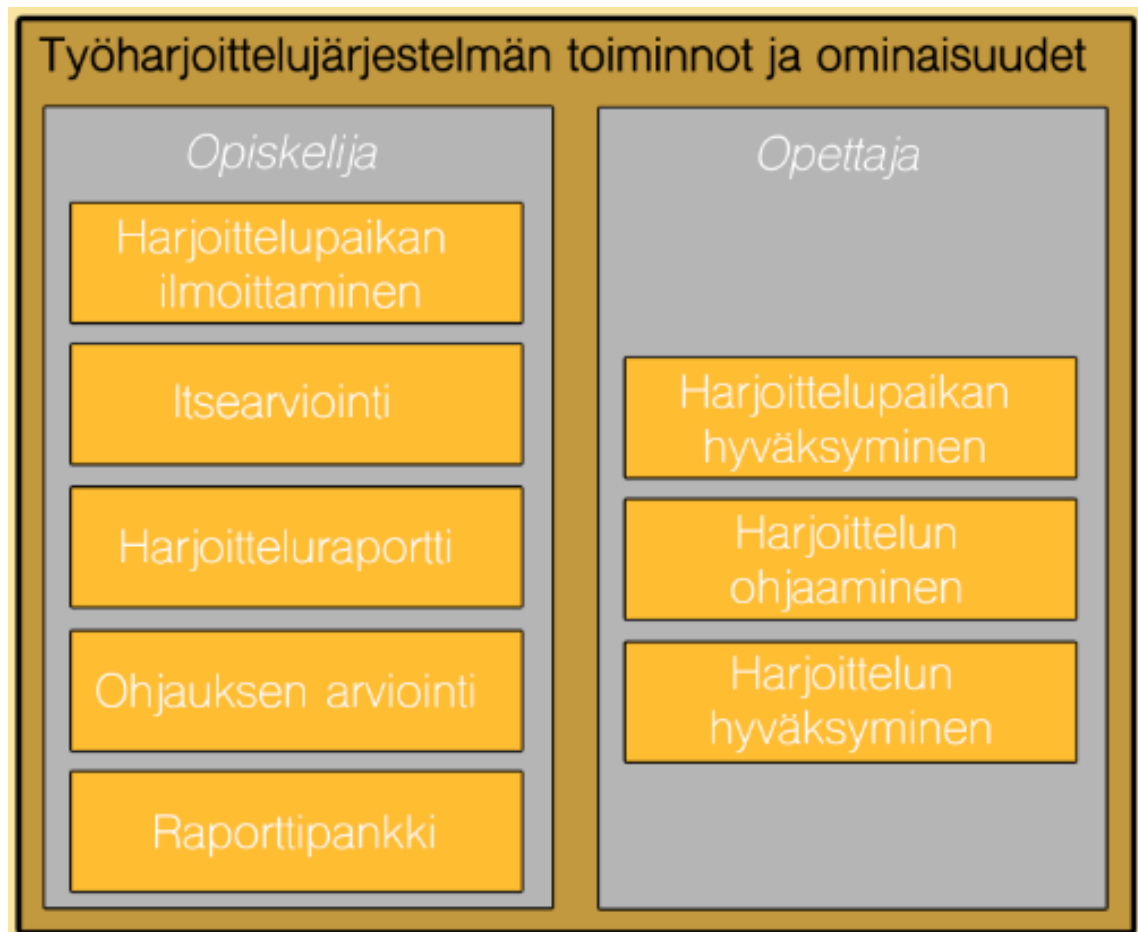
Tapahtuma	Aika	Näytä kaikki
★ [redacted] muokkasi itsearviointia	13.02.2018 10:27	
💬 [redacted] lähetti viestin	18.01.2018 10:24	
👤 [redacted] täytti harjoittelun ohjauksen arvioinnin	18.01.2018 10:23	
🏆 [redacted] hyväksyi koko harjoittelun	10.10.2016 12:29	
✅ [redacted] hyväksyi yhden harjoittelusuorituksen	10.10.2016 12:29	

Kuva 7. Esimerkki järjestelmän tuottamasta ilmoituksesta hyväksytysti suoritetusta työharjoittelusta.

Opiskelijan kannalta työharjoittelujärjestelmän tärkeimmät toiminnallisuudet ovat työharjoittelupaikan ilmoittaminen, itsearvioinnin kirjoittaminen ja lopuksi työharjoittelun päätteeksi harjoitteluraportin kirjoittaminen. Raporttiin liitetään myös harjoittelusta saatu työtodistus. Aivan lopuksi opiskelijan on myös tehtävä työharjoittelun ohjauksen arviointi, joka on käytännössä samantapainen raportti, mitä opiskelija on jo täyttänyt harjoittelupaikan ilmoittamisen, itsearvioinnin ja harjoitteluraportin yhteydessä. Harjoittelun ohjauksen arvioinnin tarkoitus on selvittää opiskelijan kokema ohjauksen laatu koko

työharjoitteluprosessin aikana, aina harjoitteluinfosta ja yleisestä työharjoitteluun liittyvästä tiedottamisesta hänen henkilökohtaiseen harjoittelun ohjaamiseen. Tässä arvioinnissa opiskelija voi myös arvioida kokemansa perehdytyksen työhön työpaikan ohjaajan puolesta. [Aalto.]

Työharjoittelujärjestelmästä löytyy lisäksi osio nimeltä ”Selaat yrityksiä/Raportteja”. Tämän järjestelmän ominaisuuden voisi luokitella opiskelijaa aidosti auttavaksi ominaisuudeksi. Selaamalla raporttipankkia opiskelija voi tutkia muiden opiskelijoiden kirjoittamia raportteja jo suoritetuista työharjoittelujaksoista. Muiden opiskelijoiden raporttien lukeminen voi auttaa opiskelijaa harjoittelupaikan etsimisessä ja antaa hyvät referenssit harjoittelijoita ottavista yrityksistä, joista opiskelijan voisi olla mielenkiintoista kysyä harjoittelupaikan mahdollisuutta. [Aalto, Rupponen.]



Kuva 8. Havainnollistava esitys työharjoittelujärjestelmän toiminnoista ja ominaisuuksista.

6.4 Työharjoittelujärjestelmän puutteet ja kehitystoiveet

Tekniikka

Nykyinen työharjoittelujärjestelmä on alun perin kehitetty palvelemaan tekniikan alan tutkinto-ohjelmien opiskelijoita ja harjoitteluinsinöörejä. Järjestelmän toiminnallisuuksia suunniteltaessa on siten otettu huomioon ainoastaan tekniikan opiskelijoiden työharjoitteluprosessin tarpeet. Tekniikan harjoitteluinsinöörin haastattelun perusteella voidaan todeta, että nykyinen työharjoittelujärjestelmä on toiminnallisuuksiltaan sellaisenaan riittävä kattamaan tekniikan opiskelijoiden työharjoitteluprosessin tarpeet jatkossakin. Tästä huolimatta osa toiminnallisuuksista kaipaa pientä jatkokehittämistä ja hienosäätöä.

Raporttien hakeminen

Tällä hetkellä raporttipankin raportteja ei ole erikseen nimetty millään tavalla. Samassa yrityksessä suoritettujen työharjoittelujen raportit löytyvät samasta listauksesta, josta selviävät ainoastaan harjoittelun suoritettu ajankohta, tutkinto-ohjelma ja koulutusala. Nyt raportteja voi suodattaa ainoastaan koulutusohjelman mukaan, eli esim. kaikki tuotantotalouden opiskelijoiden työharjoitteluraportit löytyvät samasta listauksesta riippumatta opiskelijoiden pääaineesta. Työharjoittelun työtehtävät ja työpaikkakuvaus eivät siis selviä muuten kuin avaamalla ja tarkastelemalla raporttia, mikä ei ole kovin käytännöllistä. [Aalto.]

Liiketalous

Vaikka tekniikan ja liiketalouden opiskelijoiden työharjoitteluprosessi on monilta osin samantyyppinen, haastattelujen perusteella voidaan todeta, ettei liiketalouden työharjoitteluprosessista vastaavat opettajat koe nykyisen järjestelmän täyttävän liiketalouden työharjoitteluprosessin tarpeita tarpeeksi hyvin. Itse toiminnallisuuksiltaan työharjoittelujärjestelmä koetaan riittäväksi, mutta niitä tulisi hienosäätää, jotta järjestelmä palvelisi paremmin juuri liiketalouden työharjoitteluprosessia.

Ulkomailla suoritettut työharjoittelut

Liiketaloudessa on monta eri pääainetta ja osa työharjoitteluista suoritetaan myös ulkomailla. Tällä hetkellä liiketalouden raportteja voi hakea kyllä pääainekohtaisesti, mutta se ei auta löytämään ulkomailla suoritettujen työharjoittelujen raportteja. Kuten ylempänä on jo mainittu, raportteja ei ole nimetty millään tavalla, vaan ne löytyvät yhtenä listauksena yrityksen nimen alta. Raporteista ei siis päällepäin näe, onko kyseessä ulkomailla vai Suomessa suoritettu työharjoittelu, varsinkin kun nykyään monella firmalla on hyvin kansainvälinen nimi. [Rupponen, Väkiparta-Lehtonen.]

Englanninkielinen käyttöliittymä

Suurimpana yksittäisenä puutteena on nostettu esiin järjestelmän huono käännöstyö. Järjestelmän käyttöliittymä on alun perin tehty suomenkieliseksi, mutta myöhemmin käännetty englanniksi. Vaikka käännöstyötä on tarkistettu useampaan otteeseen, sen englanninkielinen käyttöliittymä koetaan edelleen liian kehnoksi, jotta sitä voisi käyttää sujuvasti. Tästä johtuen esimerkiksi kansainväliset opiskelijat eivät käytä työharjoittelujärjestelmää lainkaan, vaan heillä on omat toimintatavat työharjoitteluun liittyvien prosessien suhteen. [Rupponen, Väkiparta-Lehtonen.]

Harjoittelun ajanjakso

Työharjoittelujärjestelmä laskee automaattisesti harjoittelun laajuuden opiskelijan ilmoittaman ajanjakson perusteella. Kun harjoittelusta vastaava opettaja hyväksyy työharjoittelupaikkailmoituksen, harjoittelun ilmoitettua ajankohtaa ei pysty enää jälkeenpäin muokata. Joskus kuitenkin syystä tai toisesta toteutunut työharjoittelu-aika ei vastaa ilmoitettua ajanjaksoa, eikä järjestelmän ilmoittama harjoittelun laajuus siten pidä paikkaansa. [Rupponen, Väkiparta-Lehtonen.]

Harjoittelusta vastaavan opettajan valinta

Haastattelussa liiketalouden opettajat nostivat esille nykyisessä järjestelmässä esiintyvän ongelman, joka hankaloittaa työharjoittelun ohjausta merkittävästi. Liiketaloudessa on useita harjoittelua ohjaavia opettajia, ja tällä hetkellä järjestelmä valitsee ohjaavat opettajat automaattisesti opiskelijan suorittamien esivalintojen pohjalta. Liiketaloudessa on kaksi työharjoittelujaksoa, joista ensimmäinen on nimikkeeltään perusharjoittelu ja

toinen ammattiharjoittelu. Perus- ja ammattiharjoittelussa on usein eri harjoittelua ohjaavat opettajat, ja järjestelmä valitsee ammattiharjoittelun ohjaajaksi automaattisesti sen opettajan, joka on ollut opiskelijan ohjaajana perusharjoittelussa. Harjoittelusta vastaavaa opettajaa ei pysty järjestelmässä vaihtamaan manuaalisesti, ja tästä johtuen kaikki opiskelijan ja opettajan välinen viestittely järjestelmän sisällä ei toimi oikein. Opiskelijan lähettämä viesti ohjautuu automaattisesti järjestelmässä merkitylle ohjaavalle opettajalle, mikä ei välttämättä pidä paikkaansa. Tässä tilanteessa ”väärä” opettaja joutuu välittämään opiskelijan viestin manuaalisesti ”oikealle” opettajalle, joka sitten vastaa opiskelijalle sähköpostin välityksellä, eikä itse järjestelmän kautta. [Rupponen, Väkiparta-Lehtonen.]

Käyttöliittymä

Edellä mainittujen puutteiden ja vikojen lisäksi myös järjestelmän käyttöliittymän ulkoasu sai kritiikkiä osakseen. Liiketalouden opettajien mukaan työharjoittelujärjestelmän nykyinen käyttöliittymä ei ole kovin käyttäjäystävällinen ja ulkoasultaan liian vanha. Käyttöliittymän ulkoasuun liittyvää palautetta tuli myös tietohallinnon puolelta, jossa uuteen järjestelmään toivottiin modernimpaa ja enemmän Peppi-tyylistä ulkoasua. [Rupponen, Väkiparta-Lehtonen, Lavikainen.]

Arvosanan vienti Peppiin

Sekä tekniikan että liiketalouden harjoittelua ohjaavat opettajat toivoivat työharjoittelujärjestelmään ominaisuutta, jonka avulla opiskelijan arvosanan voisi viedä harjoitteluraportin hyväksymisen jälkeen automaattisesti Peppiin. Tällä hetkellä prosessi toimii siten, että harjoittelun hyväksymisen jälkeen opettajan on siirryttävä työharjoittelujärjestelmästä Peppiin ja tehtävä harjoittelun arviointi Pepin perusrekisterissä. [Aalto, Rupponen, Väkiparta-Lehtonen.]

7 Kehitysehdotukset

Tämän insinööriyön selvityksen ja tutkinnan perusteella voidaan esitellä kaksi mahdollista tapaa edetä tietojärjestelmän kehitysprojektin suhteen. Järjestelmää voidaan jatkossakin kehittää Pepin ulkopuolella tai siitä voidaan rakentaa Pepin sisäinen palvelu. Tässä luvussa esitellään kaksi mahdollista vaihtoehtoa, joiden mukaan työharjoittelujärjestelmän kehitystä voidaan viedä eteenpäin.

7.1 Työharjoittelujärjestelmän jatkokehittäminen Pepin ulkopuolella


Nykyisen työharjoittelujärjestelmän jatkokehittäminen Pepin ulkopuolella on nopeasti katsottuna helpoin ja paras ratkaisu. Peppi nimittäin koostuu tälläkin hetkellä sekä omista palveluista että järjestelmän ulkopuolisista integraatioista. Pepin palvelupohjaisen arkkitehtuurimallin johdosta järjestelmään on helppo kytkeä liitännäispalveluja hyödyntämällä SOAP- ja REST-teknologioita, joita myös nykyinen työharjoittelujärjestelmä hyödyntää.

Rakentamalla järjestelmä kokonaan Pepin ulkopuolelle voidaan hyödyntää maksimaalisesti järjestelmän nykyistä ohjelmakoodia ja jo olemassa olevia toimintoja. Järjestelmän ohjelmakoodikin on päivitetty jo valmiiksi viimeisimpään PHP 7-versioon, mikä antaa hyvät lähtökohdat jatkokehittämiseen ja toimintojen paranteluun. [Karhukopla] Lisäksi järjestelmän jatkokehittämiselle annetaan hyvin vapaat kädet, sillä Peppikonsortio ei aseta rajoitteita liitännäisten kehittämiseksi. Nykyinen ohjelmakoodi on kuitenkin käytävä tarkasti läpi tietoturvallisuuden takia, eli se on validoitava huolellisesti. Tämä on hyvin tärkeää erityisesti silloin, jos työharjoittelujärjestelmä otetaan Metropolia Ammattikorkeakoulun viralliseen ja kaupalliseen käyttöön. [Lavikainen.]

Jopa tällaisenaan nykyinen työharjoittelujärjestelmä on periaatteessa riittävä kattamaan tekniikan ja liiketalouden opiskelijoiden työharjoitteluprosessien tarpeet. Mutta kuten edellisessä luvussa käy ilmi, nykyinen työharjoittelujärjestelmä kaipaa kuitenkin joidenkin toimintojen osalta parantelua ja jatkokehittämistä. Pitämällä työharjoittelujärjestelmä Pepistä irrallisena liitännäisenä tulisi ensisijaisesti kiinnittää huomiota havaittujen puutteiden ja vikojen korjaamiseen sekä tiettyjen toimintojen parantamiseen.

Raporttien hakutoiminto

Ensimmäisenä parannusta kaipaavana asiana nousikin esille raporttipankin hakutoiminto. Nykyisen järjestelmän tekniikan raporttien hakutoiminto on hyvin yksinkertainen, sillä raportteja pystyy suodattamaan ainoastaan koulutusohjelman ja vapaan sanahaun mukaan. Koska esimerkiksi tuotantotalouden opetussuunnitelmat vaihtuvat vuosittain, mutta pääaineita on aina vähintään kaksi, opiskelijoiden raportit olisi järkevää pystyä suodattamaan pääaineen mukaan. Tällöin esimerkiksi ICT-opiskelijoiden on helpompi löytää yrityksiä, joihin muut pääaineenaan ICT:tä lukevat tuotantotalouden opiskelijat ovat suorittaneet työharjoittelujaan ja toimitusketjun hallintaa lukevat löytäisivät helpommin omaan työkuvaan sopivat raportit.



Pikahaku

Koulutusohjelma

Tuotantotalous

Pääaine

ICT

Hae

Kuva 9. Hahmotelmaehdotus tekniikan raporttien hakutoimintojen kehittämiseksi.

Liiketalouden raporttihaku on hieman tekniikan raporttihakua monipuolisempi. Vapaan sanahaun lisäksi raportteja on mahdollista suodattaa toimialan ja koulutusohjelman mukaan. Liiketalouden puolella toivottiin helpotusta ulkomailla suoritettujen työharjoitteluraporttien hakemiseen, jolloin hakutoimintoon olisi järkevä lisätä yhdeksi hakukriteeriksi ulkomailla suoritettut työharjoittelut. Tällä tavalla opiskelijoiden olisi helpompi löytää yritykset, jotka tarjoavat työharjoittelumahdollisuutta Suomen ulkopuolella.

Kuva 10. Hahmotelmaehdotus liiketalouden raporttien hakutoiminnon kehittämiseksi.

Puutteet ja kehitystoiveet

Työharjoittelujärjestelmässä on muutamia huomion arvoisia vikoja ja puutteita, jotka täytyy saada kuntoon. Ensimmäisenä on nostettava esiin molempien työharjoittelujärjestelmien huono englanninkielinen käyttöliittymä, jonka seurauksena kansainväliset opiskelijat eivät käytä työharjoittelujärjestelmää lainkaan. Tähän asiaan on saatava muutos, sillä saman organisaation sisällä olevat prosessit olisi järkevää yhdenmukaistaa mahdollisuuksien mukaan. Työharjoittelujärjestelmä tarjoaa mahdollisuuden koko työharjoitteluprosessin keskittämiseen yhteen keskukseseen, mutta sitä hyödyntävät tällä hetkellä ainoastaan opiskelijat, jotka suorittavat opintojaan suomenkielellä. Englanninkielisen käännöstyön tarkistus ja korjaaminen on siis tulevan kehitysprojektin kannalta tärkeä toimenpide.

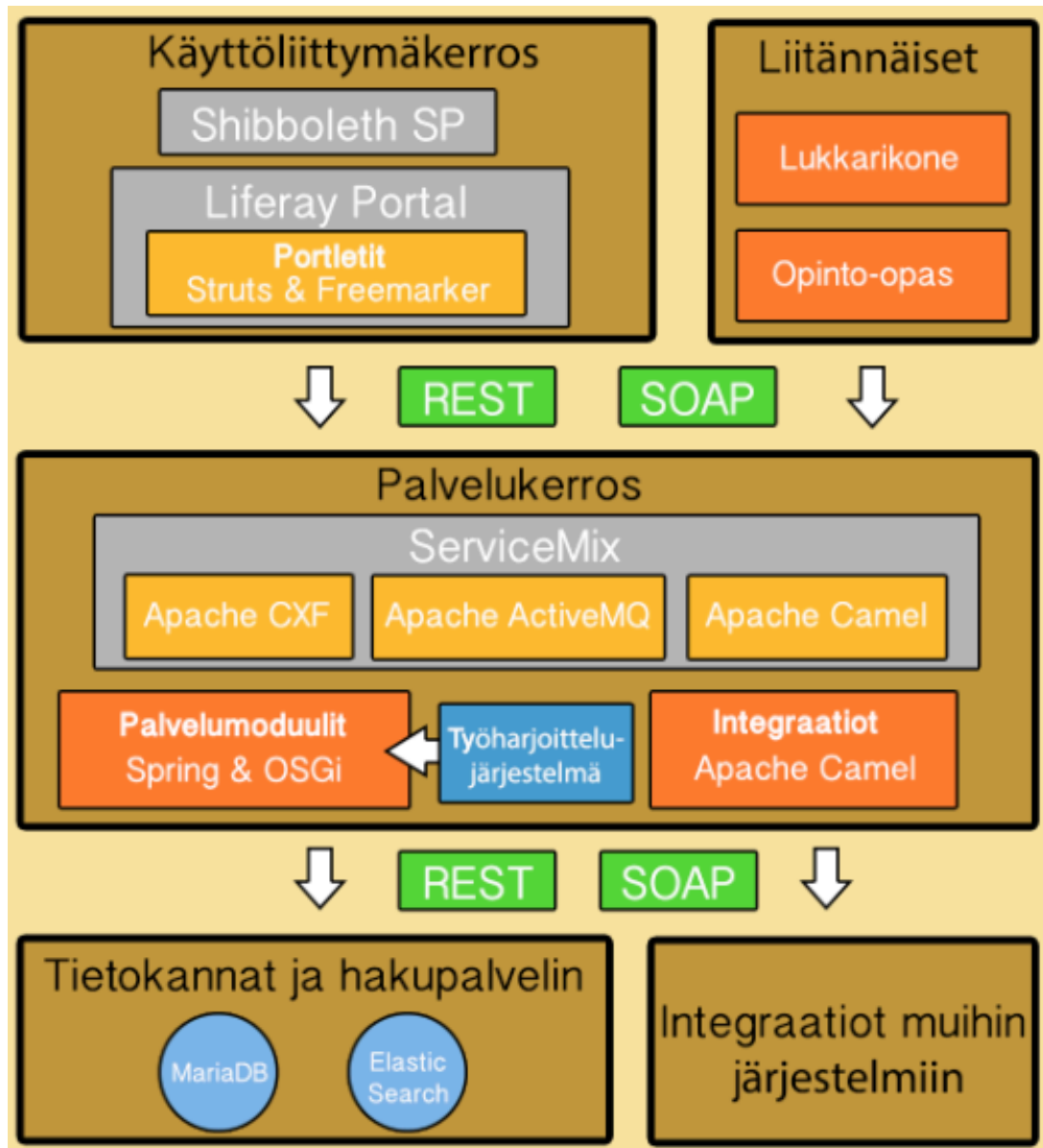
Huonon englanninkielisen käännöstyön lisäksi järjestelmässä on toiminnallisuuksia, jotka toimivat virheellisesti. Ensimmäinen ongelma koskee järjestelmän määrittelemää harjoittelun ajanjakson pituutta. Joskus työharjoittelun aika voi poiketa alun perin suunnitellusta esimerkiksi ennenaikaisesti loppuneen tai pidempään jatkuneen harjoittelun takia. Tällöin järjestelmän ilmoittama harjoittelun laajuus on virheellinen, sillä järjestelmä laskee harjoittelun ajanjakson automaattisesti opiskelijan tekemän harjoittelupaikan ilmoituksen tietojen perusteella eikä oikeasti toteutuneen harjoitteluajan perusteella. Tämä on helppo korjata muuttamalla kyseinen toiminto laskemaan harjoittelun laajuuden opiskelijan harjoitteluraportin perusteella, joka täytetään vasta työharjoittelun jälkeen.

Toinen ongelma koskee harjoittelujärjestelmän liiketalouden versiota, jossa järjestelmä automaattisesti kiinnittää opiskelijalle väärän harjoittelua ohjaavan opettajan. Virhe johtuu siitä, että tieto perusharjoittelun yhteydessä opiskelijalle kiinnitetystä opettajasta jää järjestelmän dataan, jota järjestelmä hyödyntää uudelleen saman opiskelijan tehdessä ammattiharjoittelun harjoittelupaikkailmoitusta. Karhukoplan henkilöitä haastateltaessa selvisikin, että kyseisessä toiminnossa on ”bugi”, joka on helppo korjata ohjelmoimalla toiminto priorisoimaan opiskelijan ammattiharjoittelun harjoittelupaikkailmoituksessa täyttämät tiedot perusharjoittelun harjoittelupaikkailmoituksen sijaan.

Uutena ominaisuutena työharjoittelujärjestelmään toivottiin toimintoa, jonka avulla opiskelijan harjoittelun arviointi voidaan viedä suoraan järjestelmästä Pepin perusrekisteriin. Perusrekisterissä opettaja tekee lopulliset kurssit ja työharjoittelut koskevat arvioinnit. Tietoturvalisistä syistä johtuen, työharjoittelujärjestelmän ollessa Pepin ulkopuolinen liitännäinen, sille ei voida myöntää kirjoitusoikeuksia Peppiin. Suoranaisesti tätä kehitystoivetta ei siis voida toteuttaa, mutta mahdollinen kompromissi voi olla suora linkitys työharjoittelujärjestelmän ja Pepin perusrekisterin välillä. Liitännäisellä ei siis voi olla kirjoitusoikeuksia Peppiin, mutta tietoa voi edelleen hakea Pepin tietokannoista rajapintojen kautta. [Lavikainen.]

7.2 Työharjoittelujärjestelmä osaksi Peppiä

Toisena vaihtoehtona on rakentaa työharjoittelujärjestelmä Pepin sisälle. Peppi on kokonaisuus, joka koostuu omista palveluista ja integroiduista palveluista – liitännäisistä. Tässä tapauksessa työharjoittelujärjestelmästä tulisi Pepin oma palvelu, joka on rakennettu lähelle Pepin ydintä.



Kuva 11. Havainnollistava esitys Pepin sisälle rakennetun työharjoittelujärjestelmän sijoittumisesta Pepin arkkitehtuurikuvauksessa.

Toisin kuin liitännäisten suhteen Peppi-konsortio asettaa tiettyjä rajoitteita palvelujen kehittämiseksi. Rajoitteet koskevat lähinnä teknologioita ja komponentteja, joilla palvelu toteutetaan. Palvelu on toteutettava konsortion määrittelemiä teknologioita hyödyntäen, ja kuten kaikkien Pepin palvelujen sen on tuettava kertakirjautumista. Lisäksi käyttöliittymän ulkoasun pitää olla yhdenmukainen muun Pepin kanssa, eli sen täytyy käyttää Pepin käyttöliittymän komponentteja. [Lavikainen, Arkkitehtuurin kuvaus.]

Peppi on palvelupohjaiseen arkkitehtuurimalliin perustuva Java-pohjainen sovellus, kun taas työharjoittelujärjestelmä on MVC-arkkitehtuurimalliin perustuva PHP:lla toteutettu

sovellus. Sovelluksissa on siis kriittisiä rakenteellisia eroja, jotka estävät työharjoittelujärjestelmän ohjelmakoodin suoranaisen hyödyntämisen Peppi-pohjaisen järjestelmän kehittämisessä. Lisäksi PHP ei kuulu Peppi-konsortion määrittelemien teknologioiden joukkoon. [Lavikainen.] Nykyisen työharjoittelujärjestelmän olemassa olevia ominaisuuksia ja toiminnallisuuksia, sekä tämän insinööriyön tutkimusta voidaan hyödyntää uuden järjestelmän esitutkimus- ja vaatimustenmäärittelyvaiheessa, mutta kokonaisuudessa voidaan puhua projektista, johon kuuluvat kaikki kehitysprojektin vaiheet määrittelystä ylläpitoon.

Koska Peppi on toteutettu avoimen lähdekoodin teknologioilla, se ei luo riippuvuutta tiettyyn toimittajaan. [Arkkitehtuurin kuvaus.] Näin ollen palvelujen laajentaminen järjestelmään on helppoa. Jos uusi järjestelmä rakennetaan puhtaalta pöydältä osaksi Peppi-järjestelmää, on tärkeää ottaa huomioon edellisessä luvussa mainitut toiminnallisuuksien puutteet ja kehitystoiveet, jotka tulevat suoraan järjestelmän varsinaisilta käyttäjiltä. Järjestelmän ollessa Pepin oma palvelu sille voidaan myös sallia kirjoitusoikeudet Peppiin, eli harjoittelun arviointiprosessista tulee huomattavasti suoraviivaisempi ja nopeampi, kuin se tällä hetkellä on.

7.3 SWOT-analyysi työharjoittelujärjestelmän kehitysvaihtoehdoista

Tässä luvussa esitettyjen SWOT-analyysien avulla voidaan analysoida Pepin sisälle ja ulkopuolelle rakentuvan työharjoittelujärjestelmän vahvuuksia, heikkouksia, mahdollisuuksia ja uhkia. SWOT on yksinkertainen nelikenttäanalyysi ja yleisesti käytetty työkalu, jota organisaatiot käyttävät oman toiminnan arvioimiseen. Vahvuudet ja heikkoudet -kentät muodostuvat organisaation sisäisistä tekijöistä, kun taas mahdollisuudet ja uhat -kentät muodostuvat organisaation ulkopuolisista tekijöistä. [SWOT.]

7.3.1 Pepin ulkopuolelle rakentuva työharjoittelujärjestelmä

Vahvuudet

Olemassa olevan työharjoittelujärjestelmän jatkokehittämisessä Pepin ulkopuolella on omat hyötynsä. Suurin etu on tietysti se, ettei ”pyörää tarvitse keksiä uudestaan”. Järjestelmän perusrunko on valmis, kun ominaisuudet ja toiminnot on todettu toimiviksi. Järjestelmä kaipaa siis vain hienosäätöä ja joidenkin vikojen ja puutteiden korjaamista.

Kun ohjelmistokoodikin on päivitetty PHP:n viimeisimpään versioon, voidaan olemassa olevaa koodia hyödyntää maksimaalisesti. Lisäksi järjestelmän käyttäjät, oppilaat ja opettajat, ovat jo tottuneet nykyisen järjestelmän toimintaan, eikä ”kasvojen kohotuksen” kokeva järjestelmä tule vaatimaan mitään erityistä käyttöönotto-perehdytystä. [Pohjonen, 2002.]

Heikkoudet

Työharjoittelujärjestelmän jatkokehittäminen Peppi-järjestelmän ulkopuolella vaatii ensinnäkin havaittujen toiminnallisuuksia koskevien virheiden ja puutteiden korjaamisen. Lisäksi kritiikkiä saanut vanhahko käyttöliittymän ulkoasu huutaa modernisointia. Kaikki viat ja puutteet ovat toki helposti korjattavissa, mutta vaativat osaltaan huomiota kehitysprojektissa. Suurin miinus työharjoittelujärjestelmän jatkokehittämisessä Pepin ulkopuolelle lienee kuitenkin se, että järjestelmälle ei voida sallia toivottuja kirjoitusoikeuksia Peppiin. [Lavikainen.]

Mahdollisuudet

Peppi-konsortio ei aseta rajoitteita liitännäisten kehittämiseksi. Nykyinen työharjoittelujärjestelmän käyttöliittymä on saanut osakseen paljon kritiikkiä etenkin ulkoasun suhteen. Rakentamalla työharjoittelujärjestelmä Pepin ulkopuolelle kehittäjillä on hyvin vapaat kädet toteuttaa järjestelmän käyttöliittymä haluamallaan teknologioilla ja muovata ulkoasusta modernimman, kuin se tällä hetkellä on. Peppi-järjestelmän palvelupohjaisen arkkitehtuurin ja samojen rajapinta-teknologioiden vuoksi siihen on helppo kytkeä järjestelmän ulkopuolelle rakentuva sovellus.

Uhat

Yksi kehitystoiveista oli lisätä työharjoittelujärjestelmään ominaisuus, jonka avulla opettajat pystyvät viemään opiskelijan arvosanan työharjoittelujärjestelmästä suoraan Pepin perusrekisteriin. Tietoturvasyistä johtuen tätä ominaisuutta ei suoranaisesti pysty toteuttamaan. Sallimalla kirjoitusoikeudet Pepin perusrekisteriin Peppi-järjestelmän ulkopuolisesta lähteestä, nostaa riskiä ko. ominaisuuden väärinkäytön mahdollisuudelle, sillä liitännäisten tietoturvaa ei pysty valvomaan yhtä tarkasti kuin Peppi-järjestelmän omaa tietoturvaa. Lavikaisen mukaan järjestelmien välille on mahdollista rakentaa suora linkitys, mikä nopeuttaa opettajan työharjoittelun arviointi-prosessia,

mutta kaikkiin ylimääräisiin ”virityksiin” liittyy aina korkeampi tietoturvariski. Lavikainen myös korosti, että nykyinen ohjelmistokoodi on validoitava tarkasti mahdollisten tietoturvallisten puutteiden vuoksi ennen kuin sovellukselle avataan yhteydet Peppiin rajapintojen välityksellä. [Lavikainen.]



Kuva 12. Esitys SWOT-nelikenttäanalyysin avulla Pepin ulkopuolelle rakentuvan työharjoittelujärjestelmän vahvuuksista, heikkouksista, mahdollisuuksista ja uhista.

7.3.2 Pepin sisälle rakentuva työharjoittelujärjestelmä

Vahvuudet

Työharjoittelujärjestelmän toteuttaminen Pepin sisälle tuo mukanaan lisääntyviä toiminnallisia mahdollisuuksia. Suurimpana uutena ominaisuutena voidaan toteuttaa järjestelmän käyttäjien toivoma mahdollisuus arvosanojen suorasta viemisestä Pepin perusrekisteriin. Toisin kuin järjestelmän ulkopuolisesta lähteestä Peppi-järjestelmän pal-

velintason sisälle rakennetusta moduulista tulevat syötteet ovat paremmin valvottavissa mahdollisten tietovuotojen varalta. [Lavikainen.]

Toinen huomattava Peppi-järjestelmän sisälle rakentuvan työharjoittelujärjestelmän vahvuus on sen yhdenmukaisuus muun Peppi-järjestelmän kanssa. Peppi-konsortion asettamien rajoitteiden vuoksi järjestelmän moduulit on toteutettava samoilla teknologioilla ja niiden käyttöliittymät samoilla käyttöliittymän komponenteilla kuin muut Pepin moduulit ja niiden käyttöliittymät. Työharjoittelujärjestelmän ollessa yksi Pepin moduuleista sen ylläpitäminen helpottuu huomattavasti, sillä sen arkkitehtuurinen rakenne perustuu Peppi-järjestelmän palvelupohjaiseen arkkitehtuuriin. Näin ollen työharjoittelujärjestelmän ylläpito saadaan ”saman katon alle” muun Peppi-järjestelmän ylläpidon kanssa.

Heikkoudet

Tämän insinööriyön tutkimuksen perusteella voidaan todeta, että työharjoittelujärjestelmän toteuttaminen Pepin sisälle ei sisällä konkreettisia huonoja puolia. Ainoana heikkoutena voidaankin puhua siitä, että uuden järjestelmän suunnittelu ja kehitystyö on aloitettava alusta eikä olemassa olevaa ohjelmistokoodia pystytä hyödyntämään lainkaan, sillä PHP ei kuulu Peppi-konsortion määrittelemien teknologioiden joukkoon. Tästä syystä uuden järjestelmän kehittämisessä voidaan puhua enemmän uhkista ja mahdollisuuksista kuin heikkouksista.

Mahdollisuudet

Rakentamalla uusi työharjoittelujärjestelmä Pepin sisälle tarjoutuu uusia mahdollisuuksia viemään järjestelmää kohti koko Metropolia Ammattikorkeakoulun käyttöön soveltuvaa kokonaisuutta. Tietyssä mielessä voidaan puhua aloittamisesta ”puhtaalta pöydältä”. Nimittäin tulevan järjestelmän ominaisuuksiin ja toimintoihin voidaan vaikuttaa paljon enemmän kuin jatkokehittämällä jo olemassa olevaa työharjoittelujärjestelmää. Toisaalta nykyinen työharjoittelujärjestelmä tarjoaa jo valmiiksi hyväksi todetut ominaisuudet. On myös hyvä muistaa, että toimiviksi todettujen ominaisuuksien suora implementointi uuteen järjestelmään säästää paljon aikaa ja resursseja kehitysprojektin määrittely- ja suunnitteluvaiheessa. [Pohjonen, 2002.]

Uhat

Uuden järjestelmän kehittämiseen liittyy aina enemmän uhkia kuin vanhan järjestelmän jatkokehittämiseen. Tässä tapauksessa uhkiksi voidaan lukea Peppi-konsortion asettamat rajoitteet, isosta kehitysprojektista aiheutuvat resurssikustannukset sekä uuden järjestelmän käyttöönottoon ja ylläpitoon liittyvät haasteet.

Konsortion asettamat määritykset toteutusteknologioiden ja -komponenttien suhteen rajoittavat työharjoittelujärjestelmän kehittämistä. Itsessään rajoitteet eivät ole huono asia, sillä samasta materiaalista rakennettu kokonaisuus toimii varmasti paremmin kuin vastaava eri materiaaleista koostuva rakennelma. Mutta toisaalta rajoitteet velvoittavat kehittäjiä noudattamaan tiettyjä sääntöjä ja sitouttavat käyttämään ennalta määriteltyjä teknologioita.

Toinen uhka on uuden järjestelmän kehittämisestä aiheutuvat kustannukset. Tietojärjestelmän kehitysprojekti on iso projekti, johon kuluu paljon organisaation resursseja. Vaikka projektille on myönnetty määräraha vuodeksi 2019 [Ojanen], on mahdollista, että suunniteltu budjetti ei veny kattamaan koko kehitysprojektista aiheutuvia kustannuksia, etenkin mahdollisten vastoinkäymisten sattuessa.

Kolmas uhka on uuden työharjoittelujärjestelmän käyttöönotosta ja ylläpidosta aiheutuvat haasteet ja kustannukset. Uuden järjestelmän ylläpito syö tyypillisesti n. 70 % järjestelmään suunnatuista resursseista. [Pohjonen, 2002.] Uuden asian omaksuminen voi joskus viedä aikaa ja käyttöönotto voi kohdata jopa vastustusta tietyiltä käyttäjiltä, mikäli he ovat hyvin tottuneet ja tyytyväisiä nykyisen työharjoittelujärjestelmän käyttöön ja toimintaan. Toki tässä tapauksessa Peppi-järjestelmän sisälle rakennettu työharjoittelujärjestelmä tuo mukanaan parannuksia ja käyttäjien toivomia uusia toimintoja, jolloin uuden järjestelmän käyttöönotto ei todennäköisesti tule kohtaamaan mitään suurempaa vastustelua.



Kuva 13. Esitys SWOT-nelikenttäanalyysin avulla Pepin sisälle rakentuvan työharjoittelujärjestelmän vahvuuksista, heikkouksista, mahdollisuuksista ja uhista.

8 Yhteenveto

Tämän insinööriyön tavoitteena oli tutkia Metropolia Ammattikorkeakoulun tekniikan ja liiketalouden alojen tutkinto-ohjelmien opiskelijoiden ja opettajien käytössä olevaa työharjoittelujärjestelmää, ja muodostaa tutkimuksen pohjalta mahdollisimman tarkka järjestelmän ominaisuuksia, toimintoja ja käyttötarkoituksia koskeva määritelmä mahdollista kehitysprojektia ajatellen. Tavoitteena oli myös selvittää sekä työharjoittelujärjestelmän että Peppi-järjestelmän tietotekninen rakenne ja muodostaa selvityksen perusteella molempien järjestelmien arkkitehtuurikuvaus, jonka pohjalta voidaan arvioida työharjoittelujärjestelmän integrointimahdollisuutta tai sisällytystä osaksi Peppi-järjestelmää.

Insinööriyön tekijän koulutustaustasta johtuen tietojärjestelmät, sovellusteknologiat ja –arkkitehtuurimallit eivät olleet entuudestaan hänen vahvinta tuntemusalaan, minkä takia tutkimuksen teoriaosuus ja järjestelmien nykytilan analysointi vei paljon aikaa. Työ oli aikataulutettu siten, että aluksi suoritettiin kohdehenkilöiden haastattelut ja koottiin teoreettista viitekehystä käsittelevä materiaali, joiden pohjalta tehtiin järjestelmän käyttäjien tarpeiden määrittäminen ja työharjoittelujärjestelmän sekä Peppi-järjestelmän nykytila-analyysit sekä arkkitehtuurikuvaukset. Käyttäjätarinoiden, kehitystoiveiden ja nykytila-analyysin pohjalta esiteltiin kaksi mahdollisuutta viedä kehitysprojektia eteenpäin. Samalla arvioitiin näiden vaihtoehtojen tuomia hyötyjä ja mahdollisuuksia parantamaan tekniikan ja liiketalouden alojen tutkinto-ohjelmien työharjoitteluprosessia entisestään.

Johtopäätökset

Insinööriyön selvityksen perusteella on selvää, että nykyistä työharjoittelujärjestelmää on jatkokehitettävä pikimmiten. Nykytila-analyysin ja järjestelmäkäyttäjien haastattelujen perusteella kävi ilmi, että nykyisessä järjestelmässä on puutteita, jotka hankaloittavat työharjoittelun ohjaamista. Suurin nykyisen työharjoittelujärjestelmän puute on huono englanninkielinen käyttöliittymä, minkä seurauksena englanninkielisten koulutusohjelmien kansainväliset opiskelijat eivät käytä työharjoittelujärjestelmää lainkaan. Lisäksi osa työharjoittelujärjestelmän toiminnoista kaipaa uudelleenohjelmoimista ja parantamista.

Työharjoittelujärjestelmän rakentaminen osaksi Peppiä on ehdottomasti oikea suunta viedä tätä kehitysprojektia eteenpäin. Kun työharjoittelujärjestelmästä tulee Pepin oma

palvelu, siihen on mahdollista toteuttaa kaivattu kirjoitusoikeus, ja itse palvelun käyttämisestä tulee helpompaa ja suoraviivaisempaa. Lisäksi palvelu olisi jatkossa helposti löydettävissä opiskelijan omalta Peppi-työpöydältä Pepin muiden palvelujen joukosta, kun taas nykyiseen työharjoittelujärjestelmään siirtyminen tapahtuu työharjoittelun työtilan kautta tai suoraan työharjoittelujärjestelmä verkkotunnuksen, eli domainin kautta.

Tietojärjestelmän kehitysprojekti on kuitenkin pitkä ja kivinen tie, jota ei kuljeta hetkessä. Jotta työharjoittelun ohjaamista voidaan jatkossakin jatkaa tuttuun totuttuun tapaan, ainoa oikea tapa toimia on aluksi korjata nykyisen työharjoittelujärjestelmän tunnistetut viat ja puutteet. Siksi työharjoittelujärjestelmä on pidettävä toistaiseksi Pepistä irrallisena liitännäis-palveluna, kuten tähänkin asti. Jotta työharjoittelujärjestelmä olisi helpommin löydettävissä, voidaan sille rakentaa linkitys suoraan opiskelijan työpöydältä samalla tavalla kuin esim. Outlook-sähköposti-palvelun linkitys on toteutettu opiskelijan työpöydän työkalupalkkiin.



Kuva 14. Konkreettinen ehdotus työharjoittelujärjestelmän linkittämiseksi opiskelijan työpöydän työkalupalkkiin.

Kun työharjoittelujärjestelmä pystyy sujuvasti kattamaan molempien koulutusohjelmien työharjoitteluprosessien tarpeet, voidaan toimivan järjestelmän rinnalla aloittaa uuden, Pepin sisälle rakennettavan, työharjoittelujärjestelmän kehittäminen. Peppi-järjestelmä on toteutettu avoimenlähdekoodin teknologioilla ja rakennettu palvelupohjaisen arkkitehtuurimallin mukaan, mikä takaa hyvät lähtökohdat uuden palvelun kehittämiseksi. Hiomalla ensin olemassa olevasta työharjoittelujärjestelmästä mahdollisimman täydellinen työkalu ja lopulta konvertoimalla siitä Pepin oma palvelu, Metropolia Ammattikorkeakoululla olisi käytössään varmasti hyvin spesifioitu ja kattava digitaalinen keskus tekniikan ja liiketalouden opiskelijoiden työharjoittelujen ohjaamiseen. Hyvin toteutettu palvelu tarjoaisi myös hyvät lähtökohdat kahden muun koulutusalan työharjoitteluprosessien ohjausten keskittämiseksi yhteen, koko organisaation tarpeet kattavaan järjestelmään.

Jatkotutkimushankkeita

Tämä insinööriyö rajattiin tutkimaan ainoastaan tekniikan ja liiketalouden alojen tutkinto-ohjelmien työharjoitteluprosessia. Jos kehitysprojektia päätetään viedä jatkossa eteenpäin Peppi-orientoituneesti, on järkevää tutkia myös Metropolia Ammattikorkeakoulun kahden muun koulutusalan työharjoitteluprosesseja. Etenkin sosiaali- ja terveysalan tutkinto-ohjelmien työharjoitteluprosessi on niin paljon laajempi ja monimutkaisempi kuin tekniikan ja liiketalouden alojen tutkinto-ohjelmien työharjoitteluprosessi, että tarkan ja hyvin spesifioidun tarvemääritelmän tekeminen vaatii hyvin laajan tutkimisen ja nykytila-analyysin suorittamisen.

Tätä insinööriyötä tehdessä syntyi myös ajatus siitä, että tulevaisuudessa työharjoittelujärjestelmän kautta voisi ohjata opinnäytetöitä. Opiskelijan opinnäytetyö vaatii usein paljon intiimimpää ja aktiivisempaa ohjausta kuin opiskelijan työharjoittelu, joten digitaalisen järjestelmän hyödyntäminen tässä asiassa kuulostaa varsin järkevältä. Nykyinen opinnäytetöiden ohjaaminen ei ole mihinkään ennalta määriteltyyn toteutustapaan sidonnainen, vaan ohjaus tapahtuu puhtaasi opiskelijan ja ohjaavan opettajan kesken sovitulla tavalla, esim. sähköpostin tai puhelimen välityksellä.

Lähteet

- Paananen, J., Granlund, K. 2005. Tietotekniikan peruskirja. 1. painos. Jyväskylä: Docendo.
- Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. 2. painos. Jyväskylä: Docendo.
- Tähtinen, S. 2005. Järjestelmäintegraatio. 1. painos. Helsinki: Talentum.
- Hurwitz, J., Bloor, R., Kaufman, M., Halper, F. 2009. Service Oriented Architecture For Dummies. 2. painos. Hoboken: Wiley Publishing.
- Hirsjärvi, S., Remes, P., Sajavaara, P. 2004. Tutki ja kirjoita. Helsinki: Tammi.
- Pitt, C. 2012. Pro PHP MVC. New York: Springer.
- Riehle, D. 2000. Framework Design: A Role Modeling Approach. Zürich: ETH Zürich.
- Ojanen, Kauko. AMK palvelupäällikkö. Opinnäytetyö-palaveri. 9.1.2018
- Aalto, Tiina. Harjoitteluinsinööri. Haastattelu. 18.1.2018
- Rupponen, Mari. Opintoneuvoja. Haastattelu. 6.2.2018
- Väkiparta-Lehtonen, Pia. Lehtori. Puhelinkeskustelu. 7.2.2018
- Lavikainen, Mika. Projektipäällikkö. Haastattelu. 20.2.2018
- Karhukopla. Haastattelu. 13.2.2018 ja 22.2.2018
- Arkkitehtuurin kuvaus. Peppi-järjestelmä. Tietohallinto. PDF-dokumentti.
- Javan perusteita. Verkkodokumentti. <<http://ohjelmointi.medianurkka.com/wp-content/uploads/2010/04/java.pdf>> Luettu 2.3.2018
- Tarmia, M., Sarja, J. 2015. PHP-ohjelmoinnin perusteet. Otava Opisto.
- Peppi-konsortio. Verkkodokumentti. <<http://www.peppi-konsortio.fi/>> Luettu 25.2.2018
- Metropolia Ammattikorkeakoulu. Verkkodokumentti. <<http://www.metropolia.fi/tietoa-metropoliasta/>> Luettu 10.2.2018
- Apache Struts. Verkkodokumentti. <<https://struts.apache.org/>> Luettu 2.3.2018
- Liferay Portal. Verkkodokumentti. <<https://web.liferay.com/community/wiki/-/wiki/Main/Liferay+Portlets>> Luettu 2.3.2018
- Apache ServiceMix. Verkkodokumentti. <<http://servicemix.apache.org/docs/5.x/user/what-is-smx4.html>> Luettu 2.3.2018
- Apache ActiveMQ. Verkkodokumentti. <<http://activemq.apache.org/>> Luettu 2.3.2018
- Apache Camel. Verkkodokumentti. <<http://camel.apache.org/>> Luettu 2.3.2018

Apache CXF. Verkkodokumentti. <<http://cxf.apache.org/>> Luettu 2.3.2018

MariaDB. Verkkodokumentti. <<https://mariadb.org/about/>> Luettu 2.3.2018

ElasticSearch. Verkkodokumentti. <<https://aws.amazon.com/elasticsearch-service/what-is-elasticsearch/>> Luettu 2.3.2018

Symfony. Verkkodokumentti. <<https://symfony.com/what-is-symfony>> Luettu 2.3.2018

Foundation. Verkkodokumentti. <<https://foundation.zurb.com/showcase/about>> Luettu 2.3.2018

SWOT. Verkkodokumentti. <<https://www.pk-rh.fi/tools/swot.html>> Luettu 5.4.2018

Shibboleth. Verkkodokumentti. <<https://www.shibboleth.net/index/>> Luettu 2.3.2018

Haastattelukysymykset

Opettajien haastattelut

Mitkä ovat nykyisen työharjoittelujärjestelmän tärkeimmät ominaisuudet ja toiminnot?

Mitkä ovat nykyisen työharjoittelujärjestelmän viat ja puutteet?

Mitä kehitettävää nykyisessä työharjoittelujärjestelmässä on? (kehitystoiveet, uudet ominaisuudet)

Karhukoplan haastattelu

Mikä on karhukopla?

Mikä on nykyisen työharjoittelujärjestelmän arkkitehtuuri?

Mitä teknologioita työharjoittelujärjestelmän toteuttamisessa on käytetty?

Mika Lavikaisen haastattelu

Minkälainen on Peppi-järjestelmän rakenne?

Mitä teknologioita Peppi-järjestelmän toteuttamisessa on käytetty?