Opinnäytetyö (AMK)

Tietotekniikan ko.

Hyvinvointiteknologia

2012

Aki Tahvanainen

JÄRJESTELMÄINTEGRAATIO TERVEYDENHUOLTOALALLA

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

Turun ammattikorkeakoulu

Tietotekniikan ko. | Hyvinvointiteknologia

Talvi 2012 | 56

Ohjaajat: yliop. Mika Luimula

Aki Tahvanainen

JÄRJESTELMÄINTEGRAATIO TERVEYDENHUOLTOALALLA

Opinnäytetyössä suunniteltiin ja toteutettiin järjestelmäintegraatio Mirth Connect -nimisellä integraatioalustalla. Mirth Connect oli toimeksiantajan löytämä vaihtoehtoinen ratkaisu heidän nykyisen kankean integraationalustan tilalle. Opinnäytetyöprojektissa haluttiin selvittää mihin Mirth Connect pystyy, sekä saada tarkempaa tietoa sen ominaisuuksista ja käytettävyydestä.

Järjestelmäintegraatio on yksinkertaisesti kuvattuna tietojärjestelmien yhdistämistä toisiinsa. Sen avulla eri tietojärjestelmät pystyvät keskustelemaan toistensa kanssa sovelluskomponenttien taholla. Järjestelmäintegraatioita suunniteltaessa on hyvä ottaa huomioon sen tuleva arkkitehtuuri. Akkitehtuuriratkaisulla voidaan vaikuttaa esimerkiksi tulevan integraation vasteaikoihin ja hallittavuuteen.

Terveydenhuoltoalalla järjestelmäintegraatioiden yleisin standardi on HL7. HL7 on lyhennys sanoista Health Level 7, jolla viitataan OSI-mallin seitsemänteen kerrokseen eli sovelluskerrokseen. HL7 yleisin käytetty sanomastandardi on HL7 v2.x. HL7 v2.x on ASCII-muotoista tekstiä, joka perustuu segmentteihin, joihin tieto säilötään. Tietosegmentit erotellaan toisistaan putkimerkeillä ennaltamäärätyssä järjestyksessä.

Mirth Connect on integraatioalustamoottori, joka on suunnitteltu yksinomaan HL7-sanomastandardille. Opinnäytetyössä oli tarkoitus käyttää Mirth Connect -integraatioalustaa ja tehdä sanomakäsittelijät nykyisen integraatioalustan tilalle, jotta pystyttäisiin arvioimaan Mirth Connectin suorituskykyä ja sopivuutta yrityksen käyttöön. Mirth Connectia alettiin miettiä vakavasti vaihtoehdoksi, sillä nykyiset sanomien käsittelijät on toteutettu itsetehdyillä Perl-moduuleilla. Perl-moduulien vaihtaminen tuli ajankohtaiseksi, koska niiden hallittavuus on melko heikko. Mirth Connectin hallintaliittymän uskottiin olevan ratkaisu hallittavuusongelmiin.

Mirth Connect oli yllättävän helppo ottaa käyttöön, sen asennus ja esikonfiguraatio oli ohitse muutamassa minuutissa. Käyttöliittymään valikot näyttivät samanlaisilta, eikä käyttöliittymä ohjannut käyttäjää tarpeeksi selvästi. Valikoiden loogisuus alkoi kuitenkin paljastua käytön edetessä, vaikka tämä ei muuta sitä, että käytettävyys voisi olla innovatiivisempi.

Mirth Connect todettiin erittäin tehokkaaksi työkaluksi hallita eri integraatiokanavia. Kanavien hallinta monipuolistuu huomattavasti, kun niiden hallinta siirretään visuaaliseen käyttöliittymään. Käyttöliittymän kautta tehtävät muutoksen onnistuivat helposti transformereiden kautta. Transformerien hyödyllisyys huomattiin, kun piti tehdä nopea ja yksinkertainen muokkaus sanomankäsittelyyn. Muutoksen tekeminen sanomankäsittelyyn tapahtui transformereiden avulla nopeasti ja niiden käyttöönotto oli myös todella nopeaa.

ASIASANAT:

järjestelmäintegraatio, integraatio, HL7, Health Level 7, Mirth, Mirth Connect

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information technology | Health informatics

Winter 2012 | 56

Instructor: Mika Luimula

Aki Tahvanainen

ENTERPRISE APPLICATION INTEGRATION IN THE HEALTH CARE SECTOR

One of the main issues of this thesis was to design and develop system integration with a program called Mirth Connect. Mirth Connect was the program that the client selected to replace their current integration solution which was slow to maintain. The client wanted to find out about Mirth Connect and see it in action, in a real testing environment.

System integration, or otherwise called enterprise application integration, is simply defined as connecting enterprise applications. With the help of enterprise application integration, systems are able to communicate with each other at the level of applications. When designing a brand new enterprise application integration, it is a worth remembering that the architecture of the enterprise application integraion will play a major role in message distributing. The correct choice of enterprise application integration architecture can affect response times and even manageability.

The most common messaging standard in health care industry is HL7. HL7 stands for Health Level 7 which refers to the seventh layer of the OSI-model. The most common message format in the HL7 standard is the HL7 v2.x. HL7 v2.x is ASCII formatted text which is divided into segments with pipe characters. There is always a certain order for the pipe characters and always the same information will be stored into same segment in that certain type of message.

Mirth Connect is an integration engine which was designed especially for the health care industry. That is why in this thesis it was necessary to use an integration engine such as Mirth Connect. With Mirth Connect we were able to test out an alternate solution for the enterprise application integration. The old and stiff Perl integration modules were just too hard to maintain so Mirth Connect offered a fresh new approach to that challenge. With the help of the Mirth Connect user interface, it was easy to perform the same functions as with old Perl modules.

Mirth Connect was surprisingly easy to set up, configure and operate. The user interface menus seemed slightly confusing, because they all had the same appearance. The user interface should have been more innovative so that it would be easier to use.

Mirth Connect was found out to be a quite effective tool for controlling enterprise application integrations. Channel control was more efficient with the visual user interface. Changes were really actually made with transformers, which were the most efficient tools in Mirth Connect. Transformers enabled fast changes and really fast deployment of changes.

KEYWORDS:

system integration, enterprise application integration, HL7, Health Level 7, Mirth, Mirth Connect

SISÄLTÖ

LIITTEET

KÄYTETYT LYHENTEET, SANASTO

ASCII	lyhenne sanoista American Standard Code for Information Interchange, 7-bittinen tietokoneiden merkistö. (Wikipedia Foundation)
HTTP	lyhenne sanoista Hypertext Transfer Protocol, protokolla jonka avulla selain ja www-palvelin keskustelevat.
HTML	lyhenne sanoista Hypertext Markup Language, standardi jolla merkitään www-sivujen rakenne.
Java	avoimen lähdekoodin ohjelmointikieli, joka vaaditaan useissa www-sovelluksissa asennettavaksi.
Linux	Unix-pohjainen käyttöjärjestelmä.
Ubuntu 8.04.4 LTS	Linux-jakelupaketti, avoimen lähdekoodin ohjelmisto, käyttöjärjestelmä. (Wikipedia Foundation)
FTP	lyhenne sanoista File Transfer Protocol, tiedostojensiirtoprotokolla.
XML	lyhenne sanoisat eXtensible Markup Language, tarkoitettu tiedonsiirtoon ja tiedon tallentamiseen tietyssä muodossa.
SOAP	lyhenne sanoista Simple Object Access Protocol, XML-pohjainen protokolla, joka antaa mahdollisuuden siirtää tietoja HTTP:n avulla. (W3Schools)
WSDL	lyhenne sanoista Web Service Description Language, XML-perustainen kieli, jolla kuvataan verkkopalvelun ominaisuudet (Web Service). (Wikipedia Foundation)
Javascript	skriptikieli, jolla voidaan lisätä toiminallisuuksia pääasiassa www-sivuille. (W3Schools)

1 JOHDANTO

Järjestelmäintegraation päätarkoituksena on yhdistää tietojärjestelmiä toisiinsa. Tietojärjestelmien yhdistäminen toisiinsa ei ole aina helppoa, koska tietojärjestelmiä on erilaisia ja niitä harvoin suunnitellaan ajatellen, että joku toinen tietojärjestelmä kytkeytyy myöhemmin kiinni suunniteltavaan tietojärjestelmään. Terveydenhuoltoalalla on jouduttu siihen pisteeseen, että terveyskeskuksilla ja sairaaloilla on lukuisia erilaisia tietojärjestelmiä. Lukuisiin tietojärjestelmiin syötetätään samoja tietoja tai joudutaan tekemään asioita useita kertoja, koska sama tieto ei siirry järjestelmästä toiseen saumattomasti. Työn helpottamiseksi haluttaisiin, että nämä tiedot olisivat saatavilla heti myös toisessa järjestelmässä. Suurin syy järjestelmäintegraatioiden vaikeuteen piilee perinteisissä tietojärjestelmissä, jotka hyödyntävät vanhemman kehitysvaiheen edelleen käyttökelpoista tekniikkaa, joka on kuitenkin kehityksessä jäänyt taka-alalle.

Opinnäytetyön toimeksiantona oli suunnitella ja totetuttaa järjestelmäintegraatio Mirth Connect -nimisellä integraatioalustalla. Tässä opinnäytetyössä tuodaan esille järjestelmäintegraatiossakin tärkeitä arkkitehtuurimalleja ja esitellään HL7-standardia, koska se on terveydenhuoltoalan tärkein sanomastandardi. HL7-standardista tuodaan tärkeimpänä kokonaisuutena HL7 v2.x -sanoma ja sen rakenne. Rakennetta pyritään käsittelmään yksityiskohtaisesti, koska rakenne on helpompi tuoda esille, kun sanoma puretaan yksityiskohtaisesti pieniin osiin.

Mirth Connect -integraatioalustalla toteutetun järjestelmäintegraation on tarkoitus toimia selvitystyönä siihen, miten Mirth Connect selviytyy terveydenhuoltoalan integraatioalustana. Mirth Connect on tarkoitettu nimeomaan terveydenhuoltoalan HL7-standardin järjestelmäintegraatioalustaksi. Ensin tutustuttiin Mirth Connectiin lukemalla ohjeita ja uutissivustoja, jonka jälkeen sovellus asennettiin omalle palvelimelleen. Asennuksen jälkeen perehdyttiin itse järjestelmään suorittamalla ohjattuja tehtäviä, joiden tarkoituksena oli opettaa järjestelmän käyttöä ja sen ominaisuuksia. Lyhyen tutustumisvaiheen jälkeen ruvettiin suorittamaan

toteutusosiota, jossa pyrittiin tekemään Mirth Connectilla järjestelmäintegraatio yhdelle toimeksiantajan tuotteelle.

Opinnäytetyössä kuvataan myös projektin aikataulutukseen liittyviä asioita ja tekniikoita, joilla pystytään hieman tieteellisemmällä tavalla arvioimaan työmääriä ja aikatauluja. Aikataulun suunnitteleminen oli myös olennainen osa opinnäytetyötä, sillä toimeksiannon jälkeen oli vastuu projektin toteutumisesta ja sen tuloksista.

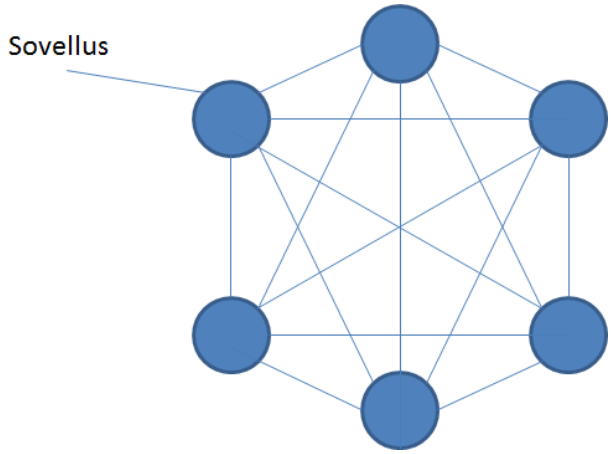
2 JÄRJESTELMÄINTEGRAATIO

Järjestelmäintegraatioilla tarkoitetaan yleisesti tietojärjestelmien kytkeytymistä toisiinsa. Selkokielisesti sanottuna voidaan puhua tietojärjestelmien välisestä kommunikaatiosta, sillä tietojärjestelmäintegraatiot tehdään yleisesti ottaen vain muutaman ohjelmistokomponentin välille eikä suinkaan kokonaisten järjestelmien välille. Monessa tapauksessa parhaiten toteutettu integraatio on toteutettu siten, että käyttäjä ei huomaa käyttävänsä jonkun toisen tietojärjestelmän toimintakomponentteja. [1]

Järjestelmäintegraatioita tarvitaan, kun esimerkiksi kaksi yritystä tekee yhteistyötä tai harjoittaa liiketoimintaa keskenään. Kysymyksessä voi olla esimerkiksi myyntitiedon lähettäminen toisen organisaation varasto-ohjelmistolle, joka tarkistaa onko tuotteita saatavilla varastosta. Tarkoituksena on varmistaa, että tieto kulkee saumattomasti järjestelmästä toiseen ilman yhteensopivuus ongelmia. Oikeastaan järjestelmäintegraatio luo yhteensopivuuksia tietojärjestelmien välille. Tämä tarkoittaa sitä, että tiedon pitää olla häviämätöntä ja muuttumatonta jokaisessa tiedonsiirtovaiheessa. [1]

2.1 Point-to-point-integraatio

Point-to-point-integraatio saa nimensä kirjaimellisesti kahden pisteen välisestä tiukasta yhteydestä. Kuvassa 1 nähdään, kuinka jokaisesta pisteestä lähtee ainoastaan yksi yhteys kuhunkin pisteeseen. Poin-to-point-integraatiossa ei tarvitse välttämättä olla useita yhteyksiä, kuten kuvassa näkyy. Parhaimmassa tapauksessa siinä voi olla kaksi pistettä joiden välillä on vain yksi yhteys. Point-to-point-integraation etuudet on helppo huomata, koska sillä on yksinkertainen arkkitehtuurirakenne, sekä se on nopea ja tehokas. Yksinkertaisuutensa takia sitä on helppo hyödyntää pienissä integraatioratkaisuissa. [2]



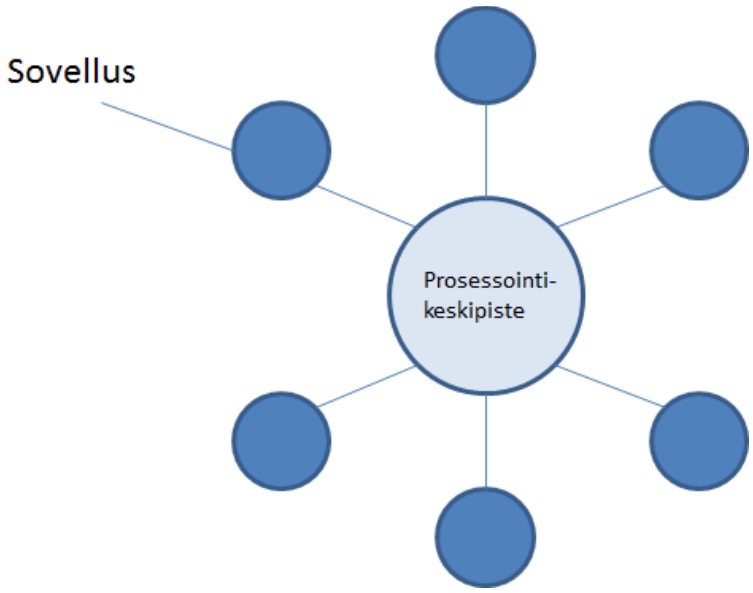
Kuva 1. Point-to-point-integraatioarkkitehtuuri

Haittapuoliakin point-to-point-integraation yksinkertaisuudessa on. Sen suoraviivaisuus muuttuu hyödystä haitaksi samassa suhteessa, kun sovelluspisteiden määrä kasvaa. Siitä syystä järjestelmän kokonaiskuva saattaa olla äärimmäisen monimutkainen, ja sen joustavuus heikentyy yhteyspisteiden lisääntyessä, mitkä aiheuttavat yleensä suuria ylläpitokustannuksia yrityksille. [2]

2.2 Hub-and-spoke-integraatio

Hub-and-spoke-integraatiossa on kyse siitä, että kaikki sisääntuleva informaatio käsitellään yhdessä keskitetyssä palvelinpisteessä. Tämän keskitetyn palvelinpisteen tehtävä on vastaanottaa kaikki sanomat, käsitellä sanomat ja reitittää ne oikeisiin sovelluspisteisiin. Hub-and-spoke-integraatiot erottelee lähettävät ja vastaanottavat sovellukset toisin, kuin point-to-point-integraatiot. [2]

Kuvassa 2 nähdään, kuinka jokainen sovelluspiste on kiinni ainoastaan yhdessä keskitetyssä pisteessä. Keskitetyn palvelinpisteen ansiosta integraatioarkkitehtuuri ei ole yhtä monimutkainen kuin point-to-point integraatioarkkitehtuuri (useilla yhteyspisteillä). Hub-and-spoke-integraatio on varsin suosittu keskikokoisissa integraatioprojekteissa sen suhteellisen helpon ylläpidon vuoksi. [2]



Kuva 2. Hub-and-spoke integraatioarkkitehtuuri

Hub-and-spoke-integraatioiden haittapuolena on kaksisuuntaisten yhteyspisteiden hankaluus. Keskitetyn palvelinpisteen täytyy koordinoida sanomia eri sovelluksien välillä saumattomasti. Tästä syystä lähettävillä ja vastaanottavilla sovelluksilla pitää olla jotain tietoa toisistaan, jotta sanomien prosessointi olisi mahdollista. Sovelluspisteiden pitää siis omistaa pientä tietoa kaikista mahdollisista sovelluksista, joiden kanssa ne ovat yhteistyössä. Tämä ei välttämättä ole ongelma yrityksen sisäisissä integraatioprojekteissa, mutta yritysten välisissä integraatioprojekteissa tämä saattaa olla iso kynnys. Vaikka hub-and-spoke-integraatiossa on helppo yhdistää sovelluksia keskitettyyn palvelinpisteeseen, on siitä huolimatta suurin ongelma sovelluspisteiden tiedonmuruset keskitetyn palvelinpisteen muista sovelluspisteistä. Tämä tarkoittaa sitä, että yritystenvälisissä integraatioprojekteissa, joissa on päätetty käyttää hub-and-spoke-integraatioarkkitehtuuri ratkaisua, suurin työ kohdistuu sovelluspisteiden muutostyöhön. [2]

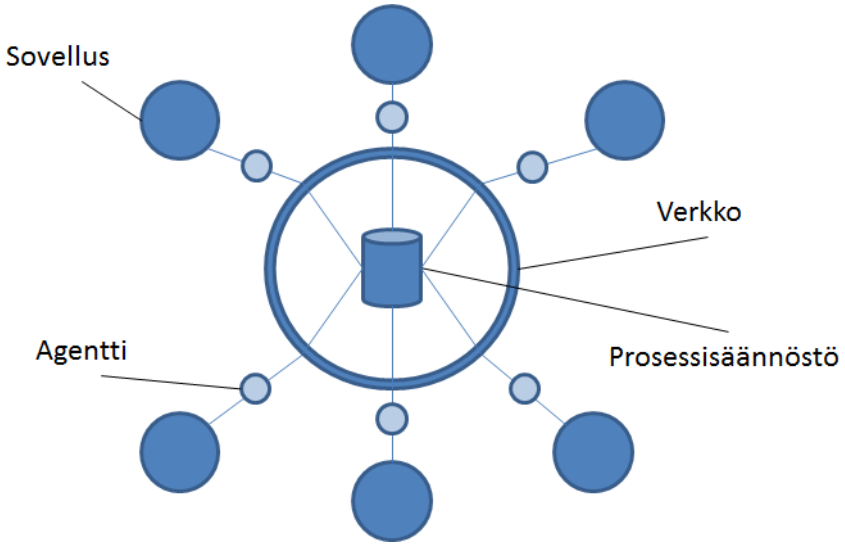
Prosessoinninkeskipiste joutuu suureen rasitteeseen, koska se hoitaa kaiken sanomien käsittelystä sanomien uudelleen lähettämiseen. Tästä syystä keskitetty palvelinpiste tarvitsee paljon tehoja ja levytilaa. Palvelin on siis

suuressa rasituksessa ja tämän takia prosessoinnin hoitavan prosessointikeskipisteen eli palvelimen on oltava äärimmäisen tietoturvallinen ja haavoittuvuuksia prosessointikeskipisteeseen ei saa olla. Haavoittuvuuksien vähentämiseksi sovelluspisteiden tunnistautuminen on erittäin yksityiskohtaisella tasolla. Esimerkiksi, jos sovelluksen tunnistautumisparametristä puuttuu yksikin merkki, ei sanomayhteyttä sallita. Tämä saattaa aiheuttaa integraatioalustan järjestelmänvalvojalle ongelmia, sillä varsinkin sairaalamaailmassa nämä tunnistautumiseen käytetyt parametrit saattavat muuttua riippuen käyttäjän tekemistä valinnoista lähettävän sovelluksen puolella. Tällöin nämä kaikki vaihtoehdot pitää olla otettuna huomioon, jotta integraatioyhteys toimisi saumattomasti. [2][3]

2.3 Hajautettu integraatio

Hajautettu integraatio on tarkoitettu isoille ratkaisuille, joissa skaalautuvuus on tärkeää. Hajautettu integraatio tekee sanomakäännökset, reititykset, sanoman muokkaukset ja parsimisen lähempänä kohdesovelluksia käyttäen pienempiä tietokoneita, joita kutsutaan agenteiksi. Agenttitietokoneet ovat yhteydessä vain yhteen järjestelmään, jotta prosessointiaika olisi mahdollisimman pieni. Agentit ottavat tietoa sovelluksilta, joihin ne on yhdistetty. Kerätty tieto prosessoidaan, jonka jälkeen se lähetetään kohdesovelluksille. [2]

Hajautettua integraatioarkkitehtuuria sovelletaan keskittämällä säännöt ja vaatimukset yhteen paikkaan, kuten kuvasta 3 nähdään. Suurin prosessointitehokkuus saavutetaan jakamalla työmäärä dedikoiduille suorittimille. Skaalautuvuuden ansiosta hajautettulla integraatioarkkitehtuurilla on selkeä etu verrattuna aikaisemmin mainittuihin point-to-point- ja hub-and-spoke-integraatioarkkitehtuureihin. [2]



Kuva 3. Hajautettu integraatioarkkitehtuuri

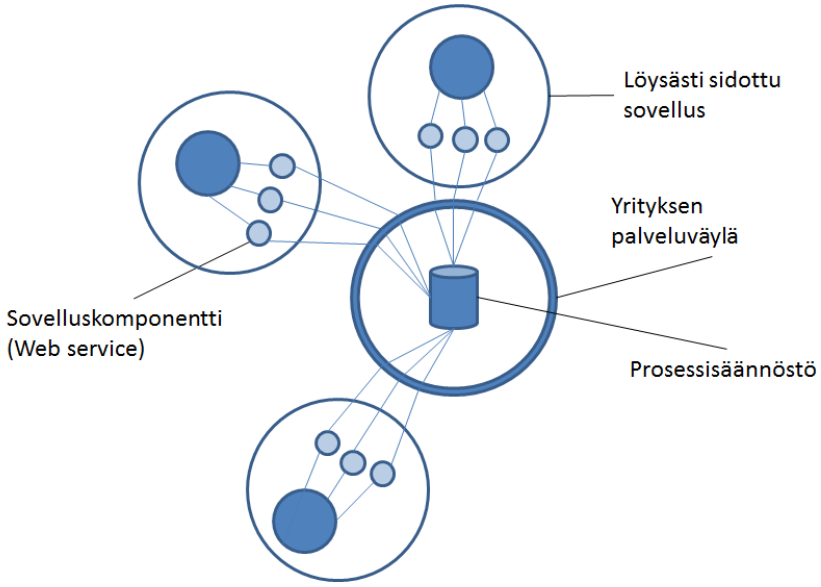
Monilla organisaatioilla on käytössä sekoitus erilaisia sovellusalustoja ja käyttöjärjestelmiä. Aina ei ole mahdollista vain valita yhtä teknologiaa jota halutaan käyttää, koska organisaatiolla voi olla käytössä useita eri sovellualustoja. Useiden erilaisten sovellusalustojen seurauksena yhden teknologian valitseminen kattaisi vain osan koko järjestelmästä. Tämä on selvä haitta käytettäessä hajautettua integraatioarkkitehtuuria, koska yritysten välisissä integraatioprojekteissa ei voida ennakkoon tietää toisten yritysten sovellusalustoista mitään. [2]

2.4 Palvelukeskeinen integraatio (SOA)

Palvelukeskeisessa integraatioarkkitehtuurissa tuodaan tieto saataville tarjoamalla sitä yleisen käyttörajapinnan kautta. Sen tavoitteena on luoda uudelleen käytettävä ja hyvin skaalautuva integraatiojärjestelmä. [4]

Kuten kuvasta 4 nähdään, löysästi sidottu sovellus tarkoittaa sitä, että sovelluksella on avoimia toimintoja ja prosesseja, jotka on suunniteltu toimimaan itsenäisinä, avoimina ja joustavina palveluina. Nämä verkkopalvelut (engl. web service) ovat yhteydessä yrityksen palveluväylään, joka puolestaan yhdistää verkkopalvelut lopulta yrityksen prosessisäännöstöön. Palvelukeskeisen integraatioarkkitehtuurin keskeisin ajatus on nimenomaan

verkkopalveluissa, joita tulisi aina pystyä käyttämään avoimien standardien rajapintojen kautta. Avoimia ja standardisoituja rajapintoja käyttämällä pystytään aikaansaamaan tietojärjestelmien joustava ja järjestelmäriippumaton vuorovaikutus. [4]



Kuva 4. Palvelukeskeinen integraatioarkkitehtuuri

Palvelukeskeisen integraatioarkkitehtuurin avulla pystytään saavuttamaan etuja sisäisissä palveluissa niin kuin myös organisaatioiden välisessä vuorovaikutuksessa. Tämä etu näkyy parhaiten siinä, että palvelukeskeisen integraatioarkkitehtuurin verkkopalvelut toimivat sovelluskehitystekniikoista riippumatta. Nämä verkkopalvelut ovat läpinäkyviä siten, että vaikka niiden sisäiseen toimintaan tehtäisiin muutoksia, niiden käyttö tulisi edelleen säilyä entisellään. Avoimien rajapintojen standardit takaavat muutoksien yhdenmukaisuuden ja käytön säilymisen ennallaan. [4]

Kuten muissakin arkkitehtuuriratkaisuissa, myös SOA-arkkitehtuurissa on omat haasteensa. SOA-arkkitehtuuri parantaa johdonmukaisuutta ja hallittavuutta. Tämä aiheuttaa samalla myös negatiivisen ilmiön, sillä SOA-arkkitehtuuri vaatii paljon yhteyksiä prosessisäännöstön ja sovelluskomponenttien välillä. Suuri määrä yhteyksiä aiheuttaa ongelmia siinä vaiheessa, kun yhdellä sovelluksella

alkaa olemaan useita eri sovelluskomponentteja yhteydessä prosessisäännöstöön. Vasteajat siis suurenevat, mitä enemmän sovelluskomponentteja on yhteydessä prosessisäännöstöön. [5]

SOA-arkkitehtuuri perustuu yleensä XML:ssä liikutettavaan tietoon. SOA-arkkitehtuurin yksi suurimmista heikkouksista onkin XML:n heikkoudet. XML tulee sanoista EXtensible Markup Language ja tarkoittaa merkintäkieltä, kuten esimerkiksi HTML (Hyper Text Markup Language, www-sivujen merkintäkieli). HTML on toiminallinen merkintäkieli, kun taas XML:ää käytetään ainoastaan kuvaamaan rakennetta ja säilyttämään tietoa. XML:stä johtuen SOA-arkkitehtuuria käytettäessä kulutetaan paljon resursseja, kuten esimerkiksi kaistaa. XML:n tietoturva ei myöskään ole paras mahdollinen, sillä HTTP GET- ja POST-pyyntöihin pystytään lisäämään helposti SQL injektioita. SQL-injektio on eräänlainen hyökkäystekniikka, jossa syötetään pätkä SQL:ää esimerkiksi sovelluksessa sijaitsevaan tietokenttään. Suoritettaessa palvelin päässä POST- tai GET-pyyntöä, suoritetaan samalla tämä pätkä SQL:ää ja pahimmassa tapauksessa tietokanta saattaa korruptoitua tai täyttyä massiivisista määristä virheellistä tietoa. [5][6]

3 HL7

Health Level Seven (HL7) on rahallista hyötyä tavoittelematon organisaatio, joka on omistautunut tuottamaan kattavat puitteet terveydenhuoltoalan standardeihin, tiedon vaihtoon, integraatioon ja elektronisten terveystietojen jakamiseen. HL7:stä yleisesti puhuttaessa, viitataan sillä HL7 organisaation julkaisemiin standardeihin. HL7 organisaation tavoite on luoda laajasti käytettyjä terveydenhuoltoalan standardeja. [7]

HL7-lyhenteen loppuosa ”Level 7” viittaa ylimmäiseen tasoon OSI-mallin seitsemän portaisessa järjestelmässä, nimeltään sovellustaso (application layer). Sovellustaso on OSI-mallin tasoista lähimpänä loppukäyttäjää, joka tarkoittaa, että molemmat käyttäjä, sekä sovellustaso ovat suorassa

vuorovaikutusessa käytettävään sovellukseen. Sovellustason tyypillisiin toimintoihin kuuluu kommunikointiosapuolien identiteetin tunnistaminen sovellukselle, resurssien saatavuuden määrittely ja kommunikoinnin synkronisointi. Kommunikointi osapuolia tunnistettaessa sovellustaso määrittää osapuolien identiteetin ja saatavuuden sovellukselle, johon tietoa välitetään. Resurssien saatavuuden määrittelyn yhteydessä sovellustason on päätettävä, että onko tieto riittävää, tai ylipäätään pyydetty verkko olemassa. [8]

3.1 HL7-standardit

Sairaaloilla ja muilla terveydenhuoltoalan toimijoilla on tyypillisesti monenlaisia tietojärjestelmiä, joita käytetään potilastietojen ylläpitoon. Kaikkien näiden järjestelmien tulisi kommunikoida keskenään saumattomasti ja ilman minkäänlaisia katkoja. Nykypäivän terveydenhuollon tietojärjestelmät on rakennettu niin, etteivät kaikkien yksittäisten tietojärjestelmien rajapinnat kykene kommunikoimaan toistensa kanssa. HL7 määrittää useita joustavia standardeja, suuntauksia ja metodeja, joiden avulla useiden terveydenhuoltoalan tietojärjestelmien on helpompi kommunikoida keskenään. Nämä suuntaukset ja standardit mahdollistavat tietojen käsittelyn ja jakamisen yhdenmukaisesti ja järjestelmällisesti. [9]

HL7 kehittää käsitteellisiä standardeja (HL7 RIM), dokumenttistandardeja (HL7 CDA), sovellusstandardeja (HL7 CCOW) ja sanomastandardeja (HL7 v2.x ja v3.0). Tässä opinnäytetyössä keskitytään enimmäkseen sanomastandardeihin sillä, sanomissa voidaan määrittää, kuinka tieto pakataan ja toimitetaan yhdeltä osapuolelta toiselle. Kyseiset standardit määrittävät kielen, rakenteen ja tietotyypit, jotka tarvitaan saumattomaan tiedonvaihtoon järjestelmästä toiseen. Sanomat ovat siis järjestelmäintegraation perusta. [9]

3.2 HL7 v2.x

HL7 versio 2.x:n sanomastandardi on elektronisen tiedonvaihdon työhevonen, joka on eniten implementoitu standardi terveydenhuoltoalalla. Muitakin terveydenhuollon standardeja löytyy. Esimerkkinä ovat seuraavassa luvussa mainittavat HL7 v3.0 -standardi ja Suomessa laajalti käytössä oleva CDA R2 -standardi. Tässä opinnäytetyössä keskitytään HL7 v2.x sanomastandardiin, sillä myöhemmin tulevassa toteutusosiossa käydään läpi integraatiota nimenomaan HL7 v2.x:n avulla. [10]

HL7-standardi kattaa sanomat, jotka vaihtavat tietoja seuraavilla osa-alueilla:

- potilaan väestötiedot
- potilaan laskutukset ja kirjanpitotiedot
- potilaan vakuutukset ja takaaja
- kliiniset havainnot
- kohtaamiset kuten, ilmoittautuminen, sisäänpääsy, kotiuttaminen ja sairaalasiirrot
- kliinisten yksiköiden tilaukset (kokeet, toimenpiteet, apteekki, ruokavalio ja tarvikkeet)
- havainnointiraportit, joihin sisältyvät testitulokset
- master-tiedostojen synkronisointi järjestelmien välillä
- potilastieto asiakirjojen hallinnointi
- resurssien ja potilaskäyntien aikataulutus
- potilas viittaukset
- potilaan hoito, ongelmasuuntautuneet asiakirjat. [10]

3.3 HL7 v3.0

Toisin kuin HL7 v2.x versioissa, HL7 v3.0 perustuu suurelta osin yhteen muodolliseen malliin nimeltään RIM (Reference Information Model). RIMin tavoite on vähentää HL7-ratkaisujen toteutuskustannuksia ja yhdenmukaistaa HL7 viestinnän määrittelyjä terveydenhuollon järjestelmien välillä. [11]

HL7 v3.0 on täysin uusi määritelmä HL7 standardista. Sen tavoitteena on ratkaista muutamia ongelmia nykyisessä standardissa. Versio 3 ei muuta ainoastaan sanomien sisältöä ja kenttiä, sekä myös koodaus säännöstöjä, alemman tason protokollien (LLP) perustaisia datatyyppejä ja jopa HL7 kommunikointiin liittyvien ohjelmistojen rooleja. XML on suunniteltu HL7 sanomien vaihdon välikappaleeksi sen sijaan, että käytettäisiin nykyistä ASCII merkistökoodauksella tuotettua omin välimerkein putkitettua tekstiä. [11]

HL7 v3.0:n pitkästä kehityskaaresta huolimatta, versio 3.0:n standardia ei ole vieläkään tarkasti määritelty. Tästä johtuen hyvin harva terveydenhuoltoalan toimilaitos on siirtynyt käyttämään HL7 v3.0:aa. Tähän yksi olennainen syy saattaa olla se, että HL7 v3.0 ei ole yhteensopiva HL7 v2.x:n kanssa. Tästä johtuen ohjelmistojen tulisi antaa tukea molemmille sanomamuodoille. Sellaisen käyttöliittymän kehittäminen molemmille sanoma muodoille olisi äärimmäisen kallista ja vaikeaa. Tämä on suurin syy, minkä takia HL7 v2.x on edelleen hallitseva standardi terveydenhuoltoalalla, kuten myös terveydenhuoltoalan palvelujen tarjoajilla. [11]

3.4 HL7-sanomat

HL7 v2.x versio on hyvin tuettu terveydenhuoltoalan tietojärjestelmissä, minkä takia sitä käytettiin tässä opinnäytetyössä. Tästä eteenpäin mainittaessa HL7-lyhenne, tarkoitetaan sillä HL7 v2.x versiota.

HL7 on käyttännössä ryhmä segmenttejä ennalta määrätyssä järjestyksessä. Nämä järjestyksessä olevat segmentit toimivat HL7-sanoman rakennusosina. Jokaisella segmentillä on oma tietty tarkoituksensa, ja niillä ryhmitellään osia viestin tuottamasta kokonaisinformaatiosta. HL7-sanomassa on yli 100 ennalta määritettyä segmenttiä, joista jokainen pystytään tunnistamaan kolmikirjaimisella kirjainlyhenteellä, joista esimerkkinä sanoman otsikko-osa (MSH), potilaan tunnistetiedot (PID) ja sukulaistiedot (NK1). Yksi edellä mainitun kaltainen segmentti voi taas koostua erilaisista kentistä, jotka voivat

sisältää yksinkertaisia datatyyppejä, kuten merkkijonoja tai numeroita. Kentät voivat sisältää myös monimutkaisempia tietoja, kuten omia osakenttiä, tai osakenttien alaosakenttiä.

Segmentit, kentät ja osakentät tunnistetaan erotinmerkeillä. Erotinmerkit kertovat, mitä notaatiota käytetään kyseisessä sanomassa ja auttavat ymmärtämään järjestyksen, tiedon merkityksen. Erotinmerkit ja muut erityistiedot sijaitsevat MSH-segmentissä. MSH-segmentti toimii kuten oikea kirjeen kuori. Kuori sisältää tiedot siitä, että kuka lähetti sanoman, minne se on menossa, milloin se lähetettiin ja mitä voidaan odottaa sen sisällöltä. [12]

3.4.1 HL7 v2.x-sanoman rakenneosat

HL7-sanomat ovat ennalta määrätyillä merkeillä jaoteltuja, ASCII-muodossa olevia sanomia, minkä takia sanomia saattaa olla äärimmäisen vaikea tulkita. Syy löytyy suuresta määrästä putkimerkkejä ja HL7-standardin omista viestitunnuksista. HL7-sanomaa pystyy kuitenkin ensisilmäyksellä tulkitsemaan maalaisjärjellä, mikäli sanoma sisältää tuttuja sanoja tai esimerkiksi nimiä. [13]

Jokainen segmentti koostuu yhdestä tai useammasta kentästä. Putkimerkkiä käytetään erottamaan kentät toisistaan, kentät voivat myös sisältää alakenttiä, jotka erotellaan ^-merkillä. Segmenttien nimet kerrotaan aina segmentin ensimmäisessä kentässä ja sen nimi on aina 3 merkkiä pitkä. Kuten ohjelmakoodi 1:ssä nähdään, tässä esimerkkisanomassa on seuraavat segmentit: MSH, SCH, PID, PV1, PV2, OBX, DG1, PR1 ja AIL. Sanomaa on helpompi ymmärtää, jos tiedetään segmenttien nimien merkitykset. Kyseisen sanoman segmentit ja niiden selitykset: [14]

- MSH on sanoman ylätunniste.
- SCH sisältää aikataulutiedot.
- PID sisältää henkilötiedot.
- PV1 sisältää potilaskäynnin yksityiskohtia.
- PV2 sisältää potilaskäynnin yksiskohdista lisätietoa.

- OBX sisältää observaatio tiedot ja tulokset.
- DG1 sisältää diagnoositiedot.
- PR1 sisältää toimenpidetiedot.
- AIL sisältää sovitun tapaamisen tiedot. [15]

Ohjelmakoodi 1. HL7-sanoma

```
MSH|^~\&|SENDER||RECEIVER||20090403130015||SIU^S15||D|2.3||||||8859/1|
SCH|956152000041778822|0902029|||111085000011111111||||0|min|^^^00010101000000
^00010101000000|||||||||^^^^^^^^HENKILÖ^^^^TÄYTTI|||||6^Cancelled^^2^Ei   mahdu
leikkauslistalle
PID||140366-1234^^^OHJELMA^HETU|15910^^^SOFT^POTNUM||Testaaja^Teppo
^^^^^D^A||19700224|1
PV1|||20O|||||||20O|||||||||||||||||||||||||||||4302 Leikkausyksikkö II|
PV2|||||||||||||2201^2201                        -                      Ortopedian
osasto^^^^^^^OHJELMA^^^^LAI||||||||||^^^^^OHJELMA^LAI||3|
OBX|1|TX|
DG1|1|ICD10|M17.1^Muu prim.polven nivelrikko^ICD10|||||||||||2|
PR1|1||NGB^Polven     risti-/sivusidekorjaus/kiinnitys||||||||||1||ZXA00^Oikea
puoli
AIL|1||||4302^Leikkausyksikkö II|
```

Ohjelmakoodi 2:ssa nähdään esimerkkisanomasta otettu PID-segmentti. Segmentin nimi on ensimmäisenä ja kolmikirjaiminen, kuten pitääkin (PID). Ensimmäisen putkimerkin jälkeen ei olekaan mitään tietoa. Tämä johtuu siitä, että kyseinen kenttä on tyhjä. Muutaman putkimerkin jälkeen on ilmoitettuna potilaan sosiaaliturvatunnus, joka on ihan normaalissa muodossa, missä se on totuttu näkemäänkin. Sosiaaliturvatunnus on sijoitettu alakenttään, tästä kertoo putkimerkkien sisällä olevat ^-merkit. Kuten ohjelmakoodi 2:sta nähdään on samassa tietokentässä missä sosiaaliturvatunnus sijaitsee, 5 alakenttää ja ainoastaan 3:ssa niistä sijaitsee tietoa. [16]

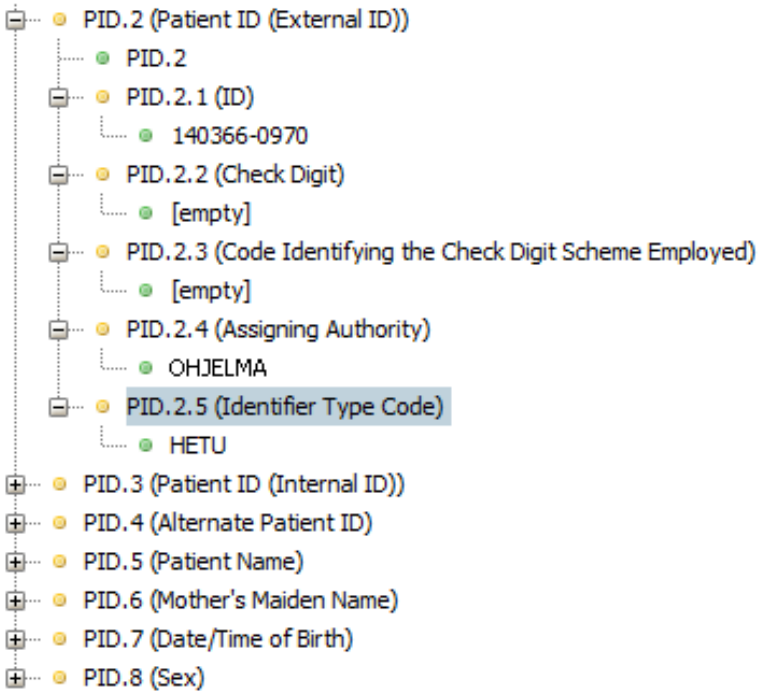
Ohjelmakoodi 2. Esimerkkisanoman PID-elementti

```
PID||140366-1234^^^OHJELMA^HETU|15910^^^SOFT^POTNUM||Testaaja^Teppo^^^^^D^A|
|19700224|1
```

Kuva 5 esittää ohjelmakoodi 2:ssa PID-elementtiä puurakennenäkymässä. Kuvassa 5 on avattu tietokenttä, josta ylempänä puhuttiin, eli

sosiaaliturvatunnuskenttä. Kuvasta nähdään selvästi kuinka kyseinen tietokenttä sisältää 5 alakenttää ja joista 3:ssa on tietoa syötettynä.

Puunäkymästä pystyy hahmottamaan hyvin sanoman elementit, kun HL7-sanoma parsitaan, kuten esimerkiksi kuvassa 5. on tehty. Tällä tavalla parsittu tieto on paljon visuaalisempaa ja varsinkin puunäkymä auttaa sanoman hahmottamisessa, koska alakenttiä ja erilaisia elementtejä pystytään aukaisemaan ja sulkemaan napin painalluksella. Useimmissa HL7-sanoman parsintaohjelmissa on myös kenttien nimet kenttien vieressä. Tämä on todella suuri lisäapu, mikäli tekstimuotoisen HL7-sanoman lukeminen tuottaa vaikeuksia.



Kuva 5. Esimerkkisanoman PID-elementti puurakennenäkymä (Mirth Connect)

3.4.2 HL7-sanoman erotinmerkit

Kuten aikaisemmin on jo mainittu, erotinmerkkejä käytetään erottamaan tietokenttiä segmentin sisällä toisistaan, tai vaihtoehtoisesti erottamaan alakenttiä toisistaan. Taulukossa 1 on esitetty HL7-sanoman oletuserotinmerkit.

Taulukko 1. HL7-sanoman oletuserotinmerkit [17]

Merkki	**Tarkoitus**
0x0D	Merkitsee jokaisen segmentin lopun.
|	Tietokenttien erotinmerkki.
^	Alatietokentän erotinmerkki.
&	Ala-alatietokentän erotinmerkki.
~	Erottaa toistuvat kentät toisistaan.
\	Ohitusmerkki.

Mikäli HL7-sanoma sisältää erotinmerkkejä osana sen ydinsisältöä, voidaan niiden sijasta käyttää ohitusmerkkejä, jotta niitä ei luultaisi erotinmerkeiksi. Ohitusmerkkien avulla eri tietojärjestelmät pystyvät erottamaan merkin, joka kuuluu ydinsisältöön ja merkin, joka on erotinmerkki. Jokaiselle erotinmerkille on oma ohitusmerkkijärjestys. Tämä ohitusjärjestys alkaa ja päättyy \-merkillä, näiden merkkien sisään tulee uniikki iso alkukirjain. Taulukkossa 2 kerrotaan jokaisen erotinmerkin ohitusjärjestys. [18]

Taulukko 2. Erotinmerkkien ohitusjärjestys [18]

Merkki	**Tarkoitus**
&	\T\
^	\S\
|	\F\
~	\R\
\	\E\
Heksadesimaalinen xx-merkki	\Xxx..\

Heksadesimaamaalisessa ohitusjärjestyksessä, xx on 2-numeroinen heksadesimaaliluku, joka edustaa merkin ASCII arvoa. [18]

Taulukossa 1 mainitut erotinmerkit ovat oletuksena käytössä, kun vaihdetaan tietoja HL7-sanomamuodossa tietojärjestelmien välillä. HL7-sanomaan voidaan kuitenkin valita eri erotinmerkit, jos standardin mukaiset erotinmerkit eivät sovellu kyseiseen käyttöön. [19]

Jokaisessa HL7-sanomassa erotinmerkit ilmoitetaan sanoman MSH-segmentin ensimmäisessä tietokentässä. Mikäli oletuserotinmerkit ovat käytössä alkaa MSH-segmentti seuraavasti, MSH|^~\&. Tässä vaiheessa on hyvä jälleen muistaa, että HL7-sanomassa on oltava MSH-segmentti ja sanoman on aina alettava MSH-segmentillä. MSH-segmentin pakollisuus johtuu siitä, että MSH-segmentissä annetaan tarkoitus, lähettäjä ja vastaanottaja sanomalle. Ilman näitä tietoja ei voida kirjeitäkään lähettää postilla, joten kirjekuori on varsin sopiva kuvaus MSH-segmentistä. [19]

Erotinmerkkien järjestyksellä on merkitystä, sillä järjestys kertoo, mikä on määriteltyjen erotinmerkkien tarkoitus. Tämä tulee siis ottaa huomioon, mikäli ei käytetä standardin mukaisia erotinmerkkejä. Erotinmerkkien järjestys on seuraava:

1. erotinmerkki, joka erottaa tietokentän toisesta (oletus: |)
2. erotinmerkki, joka erottaa alatietokentän toisesta (oletus: ^)
3. erotinmerkki, joka erottaa toistuvat tietokentät (oletus: ~)
4. erotinmerkki, joka määrittää alun ja lopun ohitusjärjestyksestä (oletus: \)
5. erotinmerkki, joka erottaa ala-alatietokentän toisesta (oletus: &). [17]

Esimerkkinä MSH-segmentin alku: MSH#^~\&, jossa putkimerkki ei olekaan merkki, joka erottaa tietokentät toisistaan, vaan tässä tapauksessa se on #-merkki. [17]

3.4.3 HL7-sanoman tyyppi

Kuten aikaisemmin mainittu, jokaisella HL7-sanomalla on oma tyyppinsä. Sanoman tyyppi kertoo, mitä terveydenhuoltoalaan liittyvää tietoa sanoma sisältää. Tyypin avulla tiedetään myös minkälaisia segmenttejä sanoma voi sisältää. Sanoman MSH-segmentin 9:s tietokenttä on yleensä tyyppikenttä. [20]

Ohjelmakoodi 3:ssa olevan MSH-segmentin kentästä 9 voidaan nähdä, että, että aikaisemmin käytetyn esimerkkisanoman sanomatyyppi on SIU S15. SIU S15 tarkoittaa ajanvarauksen perumista, joten todennäköisesti tätä sanomaa on käytetty potilastapaamisen perumiseen, käyttäen potilaan tietoja hyväksi. HL7-sanomien tyyppejä on useita satoja, joka kertoo siitä, kuinka monipuolinen HL7-sanoma voi olla. Sanoman tyyppi kertoo jo itsessään hyvin paljon siitä, minkälaista tietoa sanoman tulee sisältää.

Ohjelmakoodi 3. HL7-sanoman tyyppi

```
MSH|^~\&|SENDER||RECEIVER||20090403130015|| SIU^S15||D|2.3||||||8859/1|
```

3.4.4 HL7-sanoman segmentit

Jotkin sanomatyypit sallivat segmentin toistamisen yhden tai useamman kerran. Tämä voi olla erittäin hyödyllistä esimerkiksi silloin, kun tiedoista täytyy tulla ilmi useampia yhteystietoja. NK1-segmentti tarkoittaa segmenttiä, johon syötetään tietoja lähiomaisista. Toistamalla NK1-segmenttiä useita kertoja HL7-sanomassa, saadaan näkymään useita lähiomaisia, joille voidaan ilmoittaa hätätilanteessa. [21]

Segmentin toistamisen lisäksi joissain sanomatyypeissä on mahdollisuus määrittää segmentti valinnaiseksi. Tämä tarkoittaa sitä, että segmentti pystytään määrittämään voimassa olevaksi, mikäli siihen on syötetty tietoja. Esimerkiksi AL1-segmentti (Potilaan allergiatiedot) saattaa sisältää tietoja, mikäli potilaalla on allergioita. AL1-segmentti ei ole pakollinen, joten AL1-segmenttiä ei sisällytetä sanomaan, jos AL1-segmenttiin kuuluvaa tietoa ei ole annettu. [21]

HL7-sanomiin voidaan syöttää tietoa, mikä ei sovi varsinaisesti mihinkään valmiiksi määritettyyn segmenttiin. Räätälöidylle tiedolle voidaan luoda oma segmentti, jonka avulla tieto voidaan lähettää. Räätälöidyn tiedon omia segmenttejä kutsutaan yleisesti Z-segmenteiksi, koska kaikki räätälöidyn tiedon segmentit alkavat aina Z-kirjaimella. ZPID-segmentti voisi olla esimerkiksi räätälöityjen potilashenkilötietojen kenttä. Z-segmenttejä tulisi kuitenkin käyttää vain viimeisenä vaihtoehtona, sillä yleensä vastaanottavia järjestelmiä ei ole asetettu huomioimaan Z-segmenttejä. Näin ollen niitä tulisi käyttää ainoastaan silloin, kun tiedetään, että vastaanottava tietojärjestelmä on asetettu huomioimaan myös Z-segmentit.

Monet organisaatiot käyttävät melko yleisesti Z-segmenttejä toimittaessaan tietoja vastapuolen organisaatiolle. Z-segmenttiä käytetään usein järjetelmäintegraation toteuttajan puutteellisista tiedoista johtuen esittämään tietoja, joka voitaisiin esittää myös standardin mukaisesti. HL7-sanoman tietorakenne menettää täysin tarkoituksensa, kun Z-segmenttiä käytetään liian yleisesti. Tiedonkäsittely vaikeutuu ja tästä johtuen HL7-sanoman tehokkuus laskee. Näiden syiden takia Z-segmenttiä tulisi käyttää vain ainoastaan niissä harvoissa tilanteissa, jossa tieto on liian yksityiskohtaista ja tiedolle ei löydy paikkaa HL7-sanoman ennaltamäärätyistä segmenteistä. [22]

Segmenttejä pystytään myös ryhmittelemään, joka tarkoittaa sitä, että samannimisiä segmenttejä on useita ja ne erotellaan ainoastaan järjestysnumerolla. Segmenttiryhmiä kutsutaan kokoelmiksi ja nämä kokoelmat voivat olla vaihtoehtoisia, tai toistuvia. Esimerkiksi laboratoriotuloksia voi olla tulosryhmässä 1 tai useampia. Laboratoriotulosryhmän laboratoriotestejä voi olla otettu esimerkiksi 2 ja molemmissa on useampia, kuin 1 laboratoriotulos. [23]

3.5 ACK–tunnistautumisprotokolla

HL7-standardin yksi tärkeimmistä ominaisuuksista on ACK-tunnistautumisprotokolla. ACK-tunnistautumisprotokollan tarkoituksena on lähettää viestin onnistuneen vastaanoton varmistava ACK-sanoma takaisin järjestelmälle, josta HL7-sanoma vastaanotettiin. Lähettävän järjestelmän tulee säilyttää alkuperäinen HL7-sanoma, kunnes ACK-sanoma on vastaanotettu. [24]

ACK-sanoma koostuu 2:sta segmentistä:

- MSH-segmentistä, joka sisältää lähettävän ja vastaanottavan sovelluksen tiedot, sekä sisältää alkuperäisen HL7-sanoman tunnuksen,
- MSA-segmentistä, joka osoittaa tiedon HL7-sanoman hylkäämisestä, tai hyväksymisestä. [24]

ACK-sanomaa ei lähetetä ennen kuin vastaanottava sovellus on lukenut ja käsitellyt HL7-sanoman. HL7-sanoman tunnus on MSH-segmentin 10:nnessä kentässä ja on aina ainutlaatuinen. ACK-sanoman tunnus on sama, kuin alkuperäisessä HL7-sanomassa, joka lähetettiin toiseen järjestelmään. Tällä tavalla järjetelmät pysyvät ajantasalla siitä, että mitkä HL7-sanomat on tunnistettu ja hyväksytty. ACK-sanoman MSA-segmentistä löytyy HL7-sanoman tilanne: [24]

- AA = Positiivinen tunnistautuminen, HL7-sanoma on käsitelty oikein.
- Sovellusvirhe, HL7-sanoman käsittelyssä tapahtui virhe. Lähettävän järjestelmä on täten korjattava virhe, joka sattui HL7-sanoman käsittelyssä.
- Sovellushylkäys, vastaanottava järjestelmästä johtuva virhe, joka ei liity HL7-sanoman sisältöön tai sen rakenteeseen. [24]

ACK-tunnistautumisprotollaa on suositeltavaa käyttää, mutta on otettava huomioon, että jokainen tietojärjestelmä ei välttämättä käytä ACK-tunnistautumisprotokollaa. Vastaanottavat tietojärjestelmät saattavat lähettää HL7-sanoman, mutta eivä jää odottamaan ACK-tunnitautumisprotokollan

vastausta. Tässä tapauksessa on hyvä huomioda, että ACK-sanomat eivät mene lävitse kohdejärjestelmään. ACK-tunnistautumisprotokollan hyöty katoaa, kun toinen vuorovaikutuksessa olevista tietojärjestelmistä ei käsittele ACK-sanomia oikein tai järjestelmää ei ole konfiguroitu vastaanottamaan ACK-sanomia. [25]

3.6 Avoimet ja suljetut tietojärjestelmät

Standardien noudattaminen on aina kannattavaa. Kannattavaa standardien noudattamisesta tekee se, että kyseistä standardia tukeva järjestelmä toimii samoilla asetuksilla ja konfiguraatiolla nyt ja tulevaisuudessa. Internet-selaimet ovat äärimmäisen hyvä esimerkki standardeista. Esimerkiksi alkuperäinen Netscape Internet-selain pystyy vieläkin yhdistämään Internetiin käyttäen samoja HTTP- ja HTML-standardeja. [26]

Avoimella tietojärjestelmällä tarkoitetaan sitä, että kuka tahansa voi liittyä avoimen tietojärjestelmän rajapintoihin oikeanlaisilla protokollilla. HL7-standardin tapauksessa tämä tarkoittaa sitä, että HL7-standardin tietojärjestelmärajapinta sallii yksittäisiin syötteisiin useiden eri tietojärjestelmien liittymisen. Tämän etuna on se, että uusien tietojärjestelmien ei tarvitse muokata alkuperäistä lähdejärjestelmää liittyäkseen jo esimerkkinä annetun HL7-standardin tietojärjestelmärajapintaan. [26]

Suljetuilla tietojärjestelmillä tarkoitetaan tietojärjestelmiä, jotka ovat nimensä mukaan piilotettu näkyvistä ja niiden avoin ja ilmainen käyttö on estetty. Suljetut tietojärjestelmät ovat luettavampia, mutta niiden riippuvaisuus toisista sovelluksista ja teknologioista on paljon suurempi kuin avoimilla tietojärjestelmillä. Avoimien tietojärjestelmien käyttöönotto saattaa olla paljon kustannustehokkaampaa, kuin suljettujen tietojärjestelmien käyttöönotto, mutta ilman kuluja ei pääse avoimillakaan tietojärjestelmillä. Avoimien tietojärjestelmien käyttöönotto on aina yrityksen omalla vastuulla ja yrityksestä itsestään täytyy löytyä tietotaitoa käyttöönottaakseen kyseinen avoin

tietojärjestelmä. Suljetuissa tietojärjestelmissä joudutaan yleensä investoimaan aluksi suuria summia, mutta itse käyttöönotto vaihe tapahtuu jonkun muun osalta, kuin oman yrityksen. Näitä asioita tulisi vertailla tarkkaan, ennen kuin tehdään päätöksiä tietojärjestelmä hankinnoista. Trendi on ollut avoimien tietojärjestelmien suuntainen, koska läpinäkyvät kustannukset ovat huomattavasti pienemmät avoimissa tietojärjestelmissä, kuin suljetuissa tietojärjestelmissä. [26]

3.7 HL7-standardin vaikea käyttöönotto

HL7 standardia ei ole helppo ottaa käyttöön. Organisaatiot ovat totttuneet HL7 standardissa hieman erilaisiin käytäntöihin. Tuloksena tästä aiheutuu ongelmia, jotka tarkoittavat sitä, että huonoimmassa tapauksessa joudutaan muokkaamaan HL7-sanomien jäsentelyä hieman erilaiseksi jokaisella asiakkaalle. HL7-sanomien jäsentelyn muokkaaminen voi myös aiheuttaa sen, että myös sovellusta joudutaan muokkaamaan yhteensopivaksi HL7-sanomien jäsentelylle. [27]

Taulukossa 3 on kuvattuna useimmat syyt, jotka aiheuttavat ongelmia HL7-standardin käyttöönotossa.

Taulukko 3. Käyttöönotto ongelmat ja niiden kuvaukset. [27]

Syy	**Kuvaus**
Puuttuvat kentät	Jotkin organisaatiot jättävät HL7-sanomista tiettyjä kenttiä pois lähettävästä sanomasta, joiden pitäisi standardin mukaan löytyä sanomasta.
Samaa tietoa useissa eri kentissä	Tämä ei ole ole standardin vastaista, ja joissain tilanteissa tämä on sallittua, kuten aikaisemmin on kerrottu. Useimmiten tämä kuitenkin haittaa sanoman tulkintaa, kun tiettyä sanomatyyppiä otetaan käyttöön järjestelmään.
Samaa tietoa useissa eri muodoissa	Jotkin organisaatiot tuottavat esimerkiksi aikaleiman väärässä muodossa. Aikaleima saattaa olla muodossa, jossa käytetään alakenttiä

	(20121231^100000.000), kun se pitäisi olla standardin mukaan muodossa, jossa ei ole alakenttiä käytettynä (20121231100000.000).
Eri versiot	Lähettävällä ja vastaanottavalla organisaatiolla saattaa olla eri versio käyttössä HL7-standardista.
Puuttuvat arvo (pakolliset kentät)	Vaikka HL7-standardista 95 % on vaihtoehtoisia kenttiä, niin jotkin organisaatiot välittävät näitä vaihtoehtoisia kenttiä pakollisina.
Kelpaamaton segmenttien kielioppi	Joissain tapauksissa on havaittu piittaamattomuutta segmenttien kielioppiin, mikä on vaadittua HL7-standardissa. Jotkut kentät saattavat olla poissa, vaikka ne ovat vaadittuja ja samalla toisia kenttiä on ilmestynyt sanomaan, vaikka niitä ei pitäisi edes olla olemassa.

4 TOIMEKSIANTO

Opinnäytetyön toimeksiantaja on turkulainen terveydenhuollon tietojärjestelmätoimittaja BCB Medical Oy. BCB Medical kehittää tietojärjestelmiä hoitopolkujen toiminnanohjaukseen ja tehostamiseen. Järjestelmät ovat lääkärien, sekä hoitohenkilökunnan työkalu jokapäiväisessä työssä.

Yrityksellä oli tarve saada integraatiot toimimaan keskitetymmin ja hallittavammin. Nykytilanteessa uusien integraatioiden käyttöönotto on äärimmäisen hidasta ja vaikeaa, johtuen siitä integraatiot on toteutettu alusta asti itse ja integraatioarkkitehtuuri kuvaa parhaiten point-to-point-mallia.

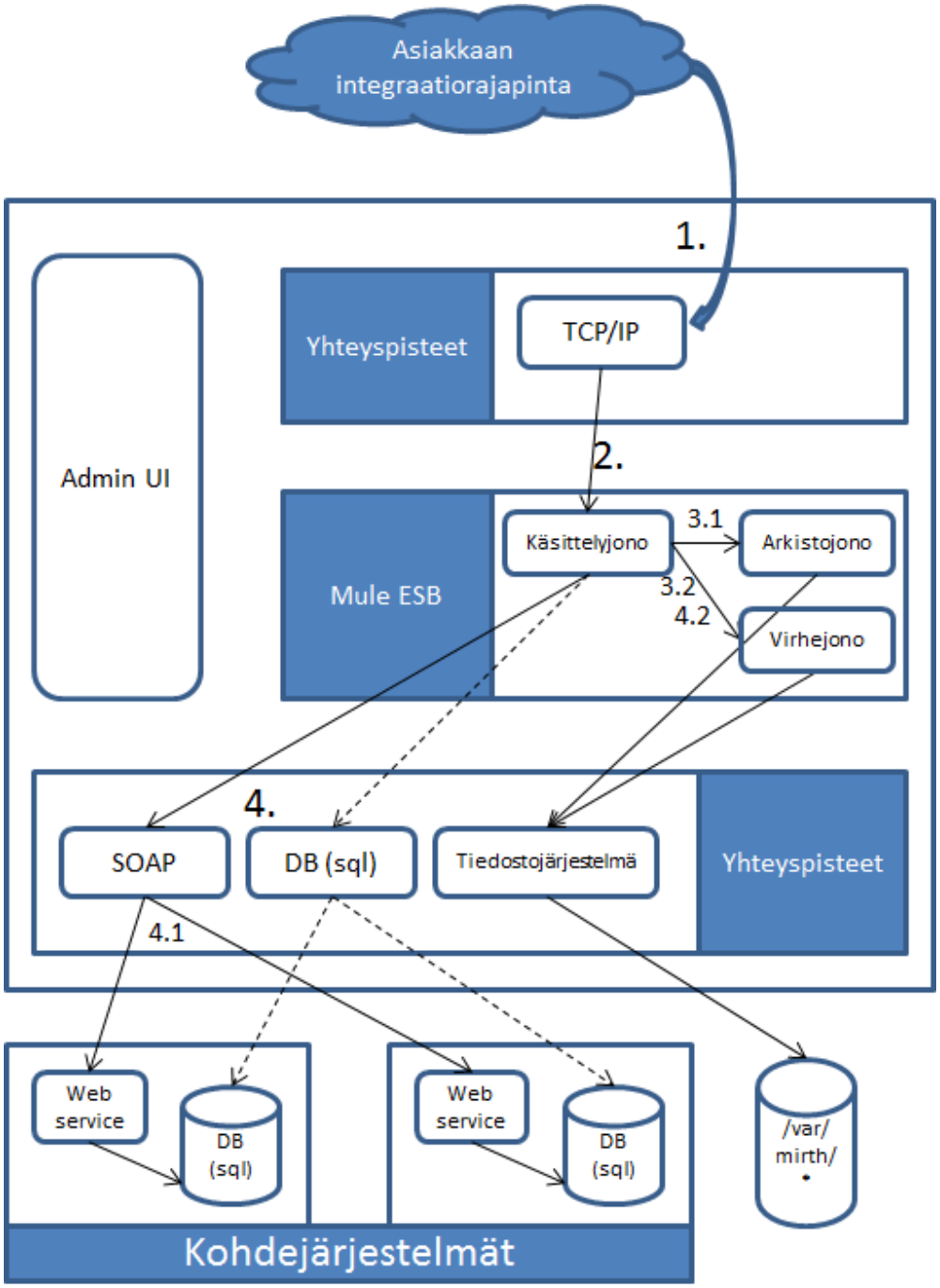
Toimeksiannon tavoitteena oli tutustua Mirth Connect avoimen lähdekoodin integraatioalustaan. Tutustumisen jälkeen tarkoitus oli muodostaa esimerkkikanavat yhden tuotteen HL7-sanomien käsittelijöiden pohjalta Mirth

Connect sovelluksella. Tarkoitus oli siis arvioida, olisiko Mirth:stä ratkaisu yrityksen nykyisiin integraatio-ongelmiin.

4.1 Tavoitteiden määrittely

Integraatioprojektiin annettiin valtuudet suunnitella aikataulu projektin toteutukselle, sekä mitä kaikkia yksityiskohtia se pitäisi sisällään. Aikataululle annettiin määräaika, joka sisälsi noin 2 kuukauden työmäärän.

Toimeksiannon aikana esiteltiin suunnitelma, joka olisi alustavasti tulevan integraationratkaisun integraatioarkkitehtuurimalli. Kuvassa 6 esitellään alustavan arkkitehtuurimallin rakenne.



Kuva 6. Alustava integraatioarkkitehtuuri

Kuvan 6 mukaiset arkkitehtuurikuvauksen vaiheet:

1. Asiakkaan integraatioalusta lähettää viestin kohti suunniteltua integraatioalustaa. Mirth ottaa viestin vastaan ja välittää viestin prosessointijonoon.
2. Viestin saapumisajankohta, sekä itse viesti tallennetaan lokiin. Viesti välitetään eteenpäin vallitsevan valintasäännön avulla (esim. lähettäjä litettynä viestityyppiin).
3. Mirth käsittelee ja muuntaa viestin kohdejärjestelmälle sopivaksi, eli viesti prosessoidaan kohdejärjestelmän tarpeiden mukaan.

   Onnistuneen viestinkäsittelyn jälkeen lähetetään ACK-vastine viestin lähettäneelle asiakkaalle, jolloin viesti on onnistuneesti loppuunkäsitelty. Mikäli viestiä ei voida onnistuneesti käsitellä (esimerkiksi ei voida vastaanottaa, koska Mirth on tilapäisesti alhaalla) lähetetään viestin lähettäneelle asiakkaan integraatioalustalle NACK-vastine.

   Jos viesti on rikki, tai sen tyyppiä ei voida ei käsitellä laitetaan viesti error-kansioon ”filesystem” connectorin avulla. (Kuvassa 6. nähdään kohdat 3.1, 3.2)

4. Viesti siirretään vastaanottavalle sovellukselle, tai kohdejärjestelmälle, suoraan tietokantaan välitauluun, tai SOAP-palvelun avulla. Mikäli kohdejärjelmä ei vastaa (Kuvassa 6. kohta 4.1) siirretään viesti jonoon, kunnes sovellus vastaa, tai maksimi pyynnöt täyttyvät.

   Mikäli esimerkiksi SOAP-palvelu tai jokin muu yhteyspiste ei vastaa siirretään sanoma virhejonon kautta tiedostojärjestelmään tallentaen koko sanoma levylle (Kuvassa 6. kohta 4.2).

Tavoitemäärittelyn aikana todettiin, että edellä esitelty integraatioarkkitehtuuri toimii alkuperäisenä mallina, kun suunnitellaan sanomien käsittelijöitä Mirth:iin. Mirth:in sanomien käsittelijöiden tulisi pohjautua vanhaan integraatioalustaan tavoin, jolla se olisi helppo sulauttaa yrityksen käyttöön lopullisesti.

Päätavoite oli suorittaa sanomien käsittelijät seuraaville sanomatyypeille:

- SIU – S12, S14 (ajanvaraus ja ajanvarauksen muokkaus)
- SIU – S15 (peruutussanoma)
- ADT – A31 (henkilötietojen päivityssanoma)

4.2 Lähtökohta

Toimeksiantajan sen hetkiset sanomienkäsittelijät olivat toteutettu Perl-kielellä. Perl-moduulit oli toteutettu point-to-point arkkitehtuuriratkaisun päälle ja kuten aikaisemmin mainittu, niin tästä kankeasta arkkitehtuurimallista haluttiin päästä pitkällä tähtäimellä eroon. Perl-moduuleissa ei sinällään ollut mitään ongelmaa, mutta niiden hallinta ja muokkaaminen oli erittäin vaikeaa. Mikäli oltaisiin haluttu tehdä muutoksia, niin muutostöihin oltaisiin jouduttu tuhlaamaan useita tunteja, vaikka muutos olisikin ollut pieni.

Perl on skriptikieli, jonka alkuperäinen tarkoitus oli toimia Unix-käyttöjärjestelmien skriptikielenä. Myöhemmin Perliä on alettu käyttämään yhä laajemmin eri käyttötarkoituksiin, kuten esimerkiksi www-sivujen kehitykseen. Alkuperäinen integraatioalusta oli toteutettu Perlin HL7-daemon –kirjaston avulla. HL7-daemon muodostaa varsin monipuoliset puitteet HL7-sanomien käsittelyyn. Aikaisemmin mainittu arkkitehtuuriratkaisu olikin suurin ongelma HL7-daemon –kirjaston avulla rakennettujen Perl-moduulien hallitsemisessa. Lisäksi visuaalisen käyttöliittyminen puuttuminen hitauttaa entisestään integraatioiden sanomienkäsittelijöiden hallintaa. [28][29]

5 SUUNNITTELU JA AIKATAULUTUS

5.1 PERT-menetelmä ja työmäärät

Projekti aloitettiin aikatauluttamalla koko projekti. Aikataulutuksen vaiheisiin kuului myös yksityiskohtaisten vaiheiden määrittely. Aikataulutusta lähdettiin toteuttamaan käyttämällä PERT-menetelmää. PERT-menetelmä valittiin siksi, että saataisiin kokemattomalle työresurssille mahdollisimman tarkka aikatauluarvio. [30]

PERT-menetelmällä pystytään suunnittelemaan aikataulu monimutkaiseen projektiin, josta on etukäteen mahdotonta arvioida tarkkaa aikaa. PER-menetelmä auttaa suunnittelemaan nimenomaan työmääriä, jonka pohjalta pystytään suunnittelmaan projektin eri vaiheiden pituudet. Kyseinen menetelmä perustuu kolmeen eri työmääräarvioon: [30]

1. Optimistinen työmääräarvio
2. Pessimistinen työmääräarvio
3. Realistinen työmääräarvio

Kaavasta 1 nähdään PERT-menetelmän kaava. Kirjain-a edustaa optimistista työmääräarviota, kirjain-b edustaa pessimististä työmäräääarviota ja c-tarkoittaa realistista työmääräarviota. Positiivinen työmääräarvio lisätään pessimistiseen työmääräarvioon, sekä realistinen työmäräarvio kerrottuna 4:llä lisätään edellä mainittuihin. Tästä laskusta saatu lukumäärä jaetaan vielä 6:lla, josta saadaan PERT-menetelmällä laskettu arvio työmäärästä. [30]

Kaava 1. PERT-menetelmän kaava

$$t^0 = \frac{a + 4c + b}{6}$$

5.2 Työmäärien liittäminen aikatauluun

Taulukosta 4 saadaan selville teoreettinen maksimi tehollisille työpäiville, sekä arvio tehollisten työpäivien todellisesta osuudesta.

Taulukko 4. Tehollisettyöpäivät [30][31]

Päiviä vuodessa	365
Viikonloput	-104
Pyhäpäivät, jotka osuvat viikolle	-6
Lomat	-30
Koulutukset	-10
Kokoukset, tapahtumat ja muut tilaisuudet	-15
Teoreettinen maksimi	**200**
Rutiinit ja muut työtehtävät	-20…-30
Sairaslomat ja muut poissaolot	-1
Joulupyhät	-5
Matkat	-5
Tehollisia työpäiviä vuodessa	**150 - 160**

Kuten taulukosta 4 nähdään arvioita ja aikatauluja ei voi suoraan liittää yhteen, koska aikatauluja ei voida suunnitella sillä perusteella, että viikossa on 5 työpäivää. Työmäärä arvioissa otettiin myös huomioon, että tehollinen työaika viikossa ei ole 5 päivää viikossa. Vuodessa on keskimäärin noin 150 – 160 tehollista työpäivää. Täten viikkokeskiarvo on 3 – 4 tehollista työpäivää viikossa. Teoreettinen maksimi työpäiville vuodessa on 200 päivää ja todellinen arvo on aikaisemmin todettu noin 160 päivää, niin 200 päivää jaettuna 160 päivällä antaa vakiokertoimen (1,25) työnkesto-laskelmiin. [30]

Todellinen työnkesto laskettiin käyttämällä edellisessä kappaleessa todettua vakiokerrointa, PERT-menetelmällä laskettua työnkestoa, henkilönresurssin käytettävyysprosenttia sekä henkilöresurssin osaamiseen liittyvää kerrointa. Henkilöresurssin osaamiseen liittyvä kerroin valitaan seuraavasta listasta: [30]

- harjoittelija (3,5 – 4,0)

- jonkin verran kokemusta (2,0 – 3,0)
- ammattilainen (1,0 – 1,5)
- asiantuntija (0,5 – 0,8).

5.3 Resurssi ja aikataulusuunnittelu

Edellisessä luvussa esitetty kerroin on valittava jokaiselle resurssille jokaiseen työtehtävään erikseen, koska osaaminen on todennäköisesti eritasoista eri työtehtävissä. Itselleen kertoimien valitseminen ei ole helppoa, koska ihminen valitsee luontaisesti itselleen liian positiivisen arvon. Juuri siitä syystä on helppo todeta, että kertoimen tulisi valita joku ulkopuolinen. Liian positivisen arvon valitseminen vaikuttaa tottakai lopulliseen aikatauluun, joka voi jossain tapauksissa myöhästyttää projektin todellista aikataulua. Työn todellisen keston kaavassa kuitenkin otetaan huomioon se, että ihminen ei pysty valitsemaan absoluuttista totuutta subjektiivisella valintaprosessilla. [30]

Kuten kaavasta 2 nähdään, kaavassa käytetään aikaisemmin todettua vakiokerrointa, PERT-menetelmää (t^0) ja resurssin osaamiskerrointa. Henkilön x käytettävyysprosentti kuvaa lukua, joka vastaa prosenttimäärää työajasta, jonka kyseinen henkilö pystyy työskentelemään määrätyn työtehtävän parissa. Liite 1 esitettää tähän opinnäytetyön työosuuteen lasketun työnkeston. Työnkeston laskemiseen käytettiin kaavaa 2, jossa käytettiin myös hyödyksi aikaisemmin mainittua PERT-menetelmää. [30]

Kaava 2. Työn todellinen kesto (t^d)

Normaalisti projekteissa olisi tarpeellista etsiä riippuvuksia tehtävien kesken. Seuraavat riippuvuussuhteet tulisi ottaa aina huomioon aikataulusuunnittelussa:

- loogiset tehtävät voidaan tehdä vain tietyssä järjestyksessä

- viive, tehtävän aloittaminen riippuu toisen tehtävän vaiheesta
- lykkäys, tehtävä voidaan aloittaa vasta, kun toisen tehtävän päättymisestä on kulunut tietty aika
- resursseihin liittyvät tehtävät voitaisiin toteuttaa samaan aikaan, mutta sama resurssi tarvitaan molempiin tehtäviin, joten toinen näistä tehtävistä voidaan suorittaa jälkeenpäin
- kalenteri-tehtävät voidaan aloittaa, tai lopettaa vain tiettynä ajankohtana. [30]

Työtehtävien ja niiden riippuvuuksien pohjalta luodaan yleensä kartta tai piirros, joka esittää kriittisen polun. Kriittinen polku on projektin kannalta tärkein ja kriittisin polku. Kriitinen polku kertoo yksinkertaistettuna työtehtävät, jotka eivät voi myöhästyä, tai niissä on erittäin vähän liikkumavaraa. [30]

Integraatioprojektissamme ei ollut tarvetta määrittää kriittistä polkua, sillä työtehtävät oli määritelty suoraviivaisesti alkamaan toistensa päätyttyä. Työtehtävien suunnittelu peräkkäisiksi johtui täysin siitä, että projektissa ei ollut muita henkilöresursseja käytettävissä. Projektille luotiin GANT-kaavio ja työtehtävälistaus Openproj-nimisellä ohjelmalla. GANT-kaavioita käytetään projektinhallinnassa projektien aikataulun ja resurssien riippuvuuksien yleiseen kuvaukseen. Työtehtävät näytetään GANT-kaaviossa listassa ja listassa näytetään pylväsdiagrammissa suunniteltujen resurssien ajankäyttö ja riippuvuudet. Liite 3:sta nähdään, kuinka tehtävälistaus on paljon laajempi, kuin laskettu työnkesto liitteessä. Tämä johtuu siitä, että projektin edetessä jouduttiin laajentamaan tehtävälistaa ja jakamaan tehtäviä pienempiin osiin. Liitteessä 3 työtehtävät on esitelty liian korkealla tasolla, tästä johtuen tehtävien kestoja jouduttiin muuttamaan. [32]

6 TOTEUTUS

Projektinhallinan ohella alettiin tekemään varsinaista toteutusosuutta, johon kuului tutustuminen Mirth Connect -integraatioalustaan ja nykyisten sanomien käsittelijöiden uudelleen tekeminen Mirth Connectiin. Samalla oli hyödyllistä tutustua järjestelmäintegraatioihin yleisellä tasolla. Tätä pystyttiin hyödyntämään, kun etsittiin vastaavanlaisia integraatioprojekteja, joista voisi saada hyötyä.

6.1 Mirth Connect -integraatiomoottori

Mirth Connect mainostaa itseään terveydenhuoltoalan monitoimityökaluksi Mirth Corp organisaation sivuilla. Tämä on varmasti totta, sillä ainakin ominaisuuslista on vaikuttavan pitkä.

Mirth Connect on avoimen lähdekoodin integraatioalusta, joka on hyvin vahvasti keskittynyt terveydenhuoltoalaan. Vaikka Mirth Connect on avoimen lähdekoodin ohjelmisto, tarjoaa Mirth Corp Mirth Connectista maksullista tukea. Mirth Connectia kehittää pääasiassa Mirth Corporatio, koska se on Mirth Connect -projektin pääsponsori. Tämän takia Mirth Corporation on myös Mirth Connectin pääkehittäjä, mutta ei kuitenkaan ainoa kehittäjä. Avoimen lähdekoodin projektille ominaista on, että projektilla on useita kehittäjiä. Mirth Connectilla on laaja yhteisö käyttäjiä, joista useat ovat myös Mirth Connectin kehittäjiä. Käyttäjäyhteisö voi myös antaa vikaraportteja, tehdä lähdekoodipaikkauksia ja antaa sekä saada ilmaista verkkotukea Mirthin keskustelufoorumin kautta. [33]

Mirth Connect luotiin, koska oli tarve saada kommunikaatioväylä, jossa kulkee HL7-sanomia. Ei ollut joustavaa työkalua, joka pystyisi käsittelemään ja joustamaan tapauksissa, joissa järjestelmillä on erityyppisiä dataformaatteja, täysin erilaiset vastaanotto- ja lähetysprotokollat. [33]

Kuten aikaisemmin mainittiin, Mirth Connect on terveydenhuoltoalalle suunniteltu integraatioalusta, integraatiomoottori ja integraatiohallintatyökalu. Mirth Connectia pystytään käyttämään millä tahansa järjestelmällä, joka pystyy suorittamaan Javaa. Mirth Connectin graafiset hallintatyökalut toimivat kanavapohjaisella arkkitehtuurilla, joten sitä on helppo ymmärtää ja hallita. Tässä arkkitehtuurimallissa on helppo tehdä kanavien konfiguraatiot ja hallita viestien välitystä. [33]

Järjestelmällä on erittäin pieni käyttöönottokynnys, koska Mirth Connect on avoimen lähdekoodin projekti. Sillä ei ole lisenssimaksuja ja se on vapaasti muokattavissa käyttäjän omiin tarpeisiin Java-luokilla sekä Javascript-koodilla. Kuten aikaisemmin mainittu, Mirth Corporatio tarjoaa kuitenkin maksullista tukea. Tähän maksulliseen tukeen kuuluuvat:

- asiakaskohtainen ja nopea henkilötuki
- koulutukset
- konsultointi
- pääsy välittömiin vikakorjauksiin ja muutamiin lisäosiin. [33]

6.2 Asennus ja konfigurointi

Tiedon etsinnän jälkeen aloitettiin Mirth Connectin asennus. Projektin käyttöön annettiin virtutaalipalvelin, johon oli asennettu Ubuntu 8.04.4 LTS. Kehitysympäristö oli täysin puhdas ja hetki sitten asennettu, koska käyttötarkoituksena oli kokeilla uutta Mirth Connectia. Mirth Connectin asentaminen Windows-ympäristöön on myös äärimmäisen helppoa. Asennus käytännössä tapahtuu painamalla muutaman kerran ”Seuraava”-nappia, ja asennus on valmis. Linuxin vakauden, suorituskyvyn ja monipuolisuuden takia asennus suoritettiin Linux-ympäristöön.

Mirthin asentaminen alkoi hakemalla asennuspaketti Mirth Corporation Internet sivuilta. Kyseinen asennuspaketti siirrettiin FTP:n ylitse suoraan käyttäjän kotihakemistoon uudelle Ubuntu-palvelimelle. Ensimmäinen varsinainen

asennusvaihe oli kuitenkin Javan asentaminen. Java asennettiin seuraavalla käskyllä:

- sudo apt-get install openjdk-jdk

Tämän jälkeen Java piti päivittää käyttäen seuraavaa käskyä:

- sudo update-alternatives --config java

Järjestelmän antaessa erilaisia vaihtoehtoja, valittiin Sun-vaihtoehdot (samoin tehtiin myös javaw:lle). Javan päivitysten jälkeen voitiin käynnistää Mirth Connect -asennuspaketti. Asennuspaketille annettiin kirjoitusoikeudet ja asennus suoritettiin seuraavilla käskyillä:

- chmod +x mirth_connect_paketin_nimi.sh
- sudo ./mirth_connect_paketin_nimi.sh

Asennuksen jälkeen Mirth Connect on käytännnössä käyttövalmis. Tämä kokoonpano riittää kehitykseen hyvin, mutta tulevaisuutta varten on hyvä tehdä vielä muutamia muutoksia, jotta Mirth Connect olisi valmis tuotantokäyttöön.

On hyvä määrittää alias Mirth Connect –asennuskansioon. Tämä helpottaa ja ennen kaikkea nopeuttaa työskentelyä palvelinpäässä. Aliaksen määrittäminen tarkoittaa sitä, että jollekin Linux-komennolle määritellään lyhyt, tai yksinkertainen komento, jolla suoritetaan varsinainen komento. Ennen aliaksen määrittämistä on hyvä luoda oma käyttäjä Mirth-palvelulle. Esimerkiksi mirth-käyttäjän luominen on hyvä vaihtoehto, koska palvelun nimi on Mirth Connect. Käyttäjä luodaan yksinomaan Mirth-palvelulle, koska ei ole järkevää suorittaa komentoja järjestelmän pääkäyttäjänä (root). Tämä ei ole järkevää, koska pääkäyttäjän oikeuksia käyttämällä rohkaistaan ylikäyttämään oikeuksia ja näin mahdollistetaan virheiden tekeminen. mirth-käyttäjälle annetaan vielä kaikki tiedosto-oikeudet Mirth-palveluun seuraavalla komennolla:

- sudo chown -R mirth:mirth

Käyttäjän luomisen jälkeen tämän kotikansiossa .profile-tiedostoon voidaan määrittää alias Mirth-hakemistoon. Alias määritetään kirjoittamalla seuraava rivi .profile-tiedostoon: [34]

- alias mirth_home='cd /opt/Mirth'

Tämän jälkeen komentoriville kirjoitetaan mirth_home, komentorivi ohjaa suoraan Mirth Connect palvelun juureen.

Mirth Connectin sisäisen tietokannan vaihtaminen oletuksesta on myös suositeltavaa. Mirthin oletustietokanta on Derby. Derby-tietokanta sopii hyvin kehitykseen ja alkutestaukseen, mutta tuotantokäytössä Mirth Corporatio suosittelee vakaampaa tietokantaa, kuten MySQL:ää tai PostgreSQL:ää. Oletustietokannan vaihtaminen onnistuu mirth_home/conf/mirth.properties–tiedostosta. Mikäli muutoksia kyseiseen konfiguraatiotiedostoon tehdään, on Mirth Connect käynnistettävä uudelleen.

Opinnäytetyön aikana näitä konfiguraatioita ei tehty, koska ei tiedetty, että kyseiset konfiguraatiot olisi järkevä tehdä. Opinnäytetyössä käytettiin alkuperäistä Derby-tietokantaa ja mirth-käyttäjätunnus, sekä mirth-hakemiston alias jäivät luomatta. Kaikki komennot suoritettiin pääkäyttäjänä, koska vihje mirth-käyttäjän luomisesta löydettiin vasta opinnäytetyön loppuvaiheessa. Edellisessä kappaleessa mainitut konfiguraatiot olisi ollut hyvä tehdä, sillä tuotantoympäristössä niistä on todella hyötyä, mutta kuten todettua, kehityksessä ne eivät ole välttämättömiä. Nämä yllä mainitut konfiguraatiot on kuitenkin hyvä tehdä tuotantoympäristöön, koska ne nopeuttavat toimintaa huomattavasti.

6.3 Mirth Administrator -asiakasohjelma

Mirth Administrator on graafinen käyttöliittymä, jonka avulla hallitaan Mirth Connect -palvelimen sanoman käsittelijöitä. Mirth Administrator -ohjelmalla konfiguroidaan rajapintoja, valvotaan rajapintatapahtumia ja selataan

sanomavarastoa. Kaikki viestien käsittely, viestien ohjaus ja valvonta tapahtuvat Mirth Adminstrator -ohjelmassa. Sanomien käsittelijöitä eli kanavia voidaan hallita myös komentoriviltä, mutta kätevintä tämä on tehdä hallintaohjelmiston kautta.

Mirth Administrator voidaan aloittaa asennuksen jälkeen suoraan selaimesta syöttämällä oman Mirth Connect -palvelimen IP-osoite selaimen osoiteriville ja asettamalla portiksi 8080, esimerkiksi seuraavasti:

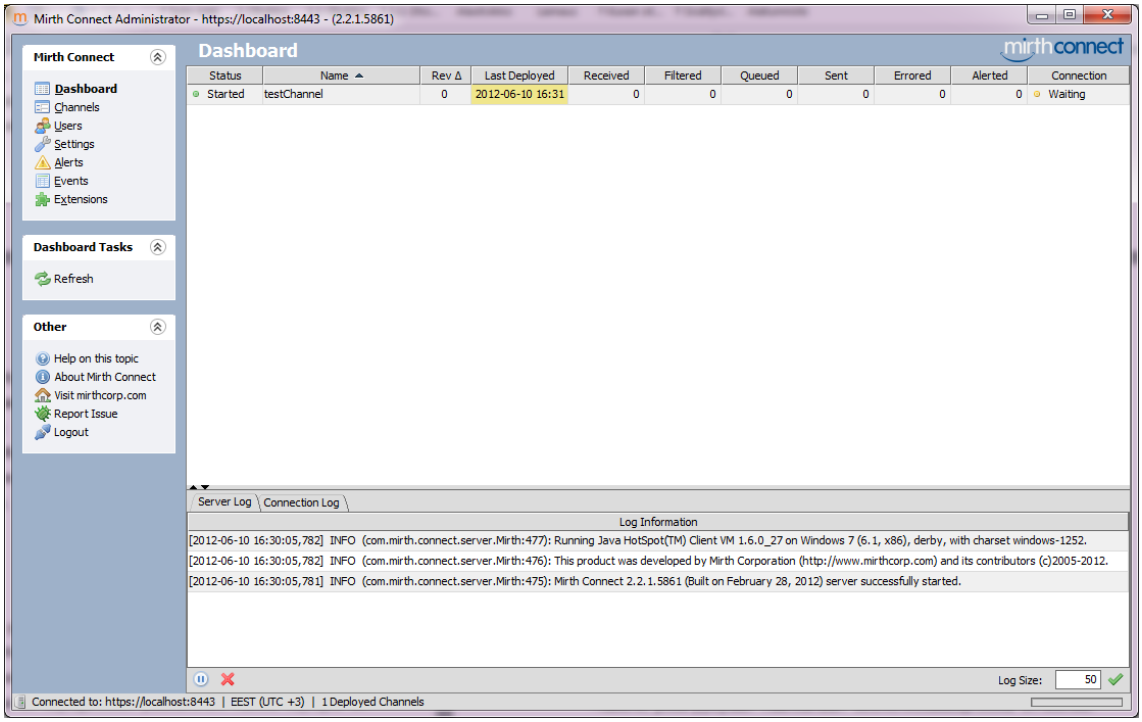
- http://192.168.1.15:8080

Mikäli Mirth Connect on asennettu samalle tietokoneelle, josta sitä aiotaan käyttää, voidaan IP-osoite korvata localhostilla. Opinnäytetyössä Mirth kuitenkin asennettiin virtuaalipalvelimelle, joten Mirth Administrator –ohjelma pystyttiin suorittamaan syöttämällä palvelimen IP-osoite ja 8080-porttinumero.

Kun oikea osoite on syötetty selaimen osoiteriville, avautuu eteen sivu, jossa on Mirthin logo ja vihreä painike, josta painamalla koneelle latautuu Mirth Connect Administrator asiakasohjelma. Mirth Connect Administrator ohjelmaan pääsee miltä tahansa tietokoneelta, joka on samassa verkossa kuin Mirth-palvelin.

Ohjelma kysyy ensimmäisen käynnistyksen yhteydessä käyttäjätunnusta ja salasanaa, jotka ovat admin ja admin. Kirjauduttua kysytään admin-käyttäjätunnuksen tiedot uudestaan, jotta saadaan uusi salasana käyttäjätunnukselle.

Kuvassa 8. nähdään Mirth Connect Administratorin -yleisnäkymä. Seuraavissa luvuissa käydään läpi kuvassa 8 näkyvät tärkeimmät valikot ja niiden ominaisuudet.



Kuva 8. Mirth Connect Administrator -yleisnäkymä

6.3.1 Dashboard-valikko

Dashboard eli kojelautanäkymä näyttää kaikki kyseisellä hetkellä käyttöönotetut kanavat. Kojelautanäkymästä nähdään myös statistiikkaa jokaisen käyttöönotetun kanavan kohdalta. Statistiikoista pystytään näkemään esimerkiksi vastaanotettujen sanomien määrä, lähetettyjen sanomien määrä ja kyseisen kanavan sen hetkinen tila.

Halutun kanavan sanomia päästään katsomaan kaksoisklikkaamalla kyseistä kanavaa tai vaihtoehtoisesti valitsemalla kanava ja painamalla vasemalta puolelta löytyvästä Dashboard tasks -valikosta View Messages. Sanomien selausnäkymästä löytyy lista kanavan kautta kulkeneita sanomia. Näitä sanomia pystytään suodattamaan päivämäärän ja ajan mukaan. Tämä on erityisen kätevää, kun esimerkiksi korjataan kanavassa ilmentynyttä virhettä. Viesteistä pystytään näkemään alkuperä sekä se, mitä muutoksia sille on tehty.

Saman valikon alta löytyy muitakin ominaisuuksia. Dasboard tasks -valikosta pystytään lisäksi pysäyttämään valittu kanava tai keskeyttämään se kokonaan.

Opinnäytetyön aikana käytettiin paljon Send message -ominaisuutta, joka löytyy myös tämän valikon alta. Send message on ominaisuus, jolla pystytään lähettämään nimensä mukaisesti sanoma suoraan valitulle kanavalle. Tämä ominaisuus havaittiin käteväksi ominaisuudeksi kehitystilanteessa jossa huomattiin, että kyseinen kanava ei toimi oikein.

6.3.2 Channels-valikko

Channels-näkymä on tarkoitettu yleisnäkymäksi kanaville. Kanavalla tarkoitetaan yksinkertaisesti putkea, jota pitkin sanoma kulkee seuraavaan kohteeseensa. Tämä kohde voi olla toinen kanava, tietokanta, tai vaikka SOAP-palvelu, riippuen tietenkin kanavan liittimestä.

Kanavanäkymässä pystytään luomaan uusia kanavia ja ne pystytään ottamaan myös käyttöön tästä näkymästä. Kanavanäkymästä päästään yksittäisen kanavan asetuksiin käsiksi. Kaksoisklikkaamalla kyseistä kanavaa avautuu esille valitun kanavan asetukset. Tässä näkymässä on mahdollista valita lähdeliittimien lisäksi kohdeliitin ja niiden asetukset. Liittimien välissä molemmissa päissä (lähde, kohde) pystytään muokkaamaan sanomia transformer-nimisellä ominaisuudella. Lisäksi molempiin päihin (lähde, kohde) voidaan asettaa suodattimia, joilla pystytään säätelemään sanoma liikennettä kyseiseen kanavaan.

Jokaisen kanavaan tehdyn muutoksen jälkeen, kanava täytyy ottaa uudelleen käyttöön mikäli halutaan, että tehdyt muutokset tulevat voimaan. Tätä pystytään seuraamaan kojelautanäkymästä. Kojelautanäkymästä löydetään ”Rev” -kohta, joka ilmoittaa kuinka monta versiota kyseinen kanava on jäljessä. Tämä tarkoittaa käytännössä sitä, että kanavaa ei ole otettu käyttöön muutoksien teon jälkeen.

6.3.3 Users-valikko

Users-valikko on melko yksinkertainen ja itsensä selittävä. Täällä luodaan uusia käyttäjiä Mirth Administrator ohjelmaan. Käyttäjille voidaan syöttää lisätietoja, kuten oikeat nimet, puhelinnumero ja sähköposti. Suurin heikkous Mirth Administrator ohjelmassa on käyttäjäroolien puuttuminen. Kaikilla käyttäjillä on samat oikeudet eli käyttännössä kaikkien asioiden muokkaaminen onnistuu millä tahansa käyttäjällä.

Käyttäjäroolit ovat kuitenkin lisäosana, joka saadaan käyttöön ainoastaan, mikäli maksetaan asiakastukimaksua Mirth Corporatiolle. Mirth Connect versio on kuitenkin täysin sama kuin muillakin, mutta käyttäjäroolit ovat yksi niistä hyödyllisistä etuuksista, jonka saa käyttöönsä, jos maksetaan asiakastukimaksua.

6.3.4 Settings-valikko

Settings-valikosta pystytään säätämään asetuksia liittyen Mirth Connect –ohjelmistoon. Server-välilehdellä pystytään säätämään asetuksia liittyen Mirth-palvelimeen. Vaikka edelle mainittua kautta ei pysty kovinkaan montaa muutosta tekemään suoraan palvelimeen, niin muutama hyödyllinen kohta Server-välilehdeltä löytyy. Sähköposti-liittimen sähköpostiasetukset pystytään säätämään tältä välilehdeltä. Tämä on varsin hyödyllinen säätää heti ensimmäisenä, jos jossain kanavassa on aikeissa käyttää sähköpostiliitintä on näiden asetuksien oltava kunnossa.

Administrator-välilehdellä pystytään säätämään käyttäjätunnukseen liittyviä asetuksia. Valitettavasti tästä valikosta pystytään mukauttamaan ainoastaan viestien määrä, joka näytetään kojelaudalla, sekä vaihtoehtoisesti näytetäänkö sanomat XML-muodossa sanomaselaimessa.

Message pruner –välilehdellä pystytään muokkaamaan asetuksia, jotka määrittävät milloin viestejä karsitaan. Karsitut viestit riippuvat siitä, minkälaisia asetuksia kanaviin on asetettu.

Settings-valikosta löytyy vielä kaksi välilehteä, mikäli maksetaan asiakastukimaksua. Ensimmäinen näistä välilehdistä on aikaisemmin mainittu käyttäjäroolien hallitseminen (User Authorization). Käyttäjärooleille pystytään määrittämään käyttöoikeuksia ja hallitsemaaan, mitä näkymiä kyseiselle käyttäjäryhmälle on kytketty päälle. Toinen näistä välilehdistä on SSL Manager –välilehti, jossa pystytään määrittämään tietoverkkosalausprotollia eri liittimille.

6.3.5 Alerts-valikko

Alerts-valikko pitää sisällään hälytysasetukset. Hälytysasetuksia pystyään säätämään kanavakohtaisesti tai globaalisti. Tässä valikossa pystytään valitsemaan yksityiskohtaisesti tietynlainen virhetilannehälytys, jonka tarkoituksena on hälyttää käyttäjälle, kun valittu virhetilanne on tapahtunut. Hälytyksen järjestelmä lähettää sähköpostilla ennalta määrättyyn osoitteeseen. Sähköpostiviestille voidaan kirjoittaa pohja, johon voidaan kirjoittaa omaa tekstiä ja raahata käteviä muuttujia, kuten virhekoodi, virheviesti, tai vaikka päivämäärä.

6.3.6 Events-valikko

Events-valikosta nähdään tapahtumalokikirja. Tapahtumalokikirja kirjaa ylös kaikki tapahtumat, mitä Mirth Connectin sisällä tapahtuu. Tapahtumia pystytään suodattaamaan lisäasetuksilla, joten esimerkiksi tietyn päivämäärän tapahtumat saadaan helposti esille. Tapahtumalokista ei näe sanomiin liittyviä tapahtumia, vaan sanomiin liittyvät tapahtumat nähdään sanomaselaimesta. Tapahtumaloki näyttää tiedot, jos esimerkiksi tehdään muutoksia johonkin tiettyyn kanavaan tai

otetaan käyttöön jotain uusia kanavia. Lokista pystytään näkemään ajankohta ja muutoksien määrä, sekä mitä tehtiin.

6.3.7 Extension-valikko

Kaikki liittimet Mirth Connectissa ovat oikeasti lisäosia. Extension-valikon päätehtävä on näyttää kaikki Mirthiin asennetut lisäosat. Extension-valikko ei rajoitu ainoastaan näyttämään lisäosia, vaan niitä pystytään myös hallitsemaan täältä. Valmiiksi asennettuja tai kolmannenosapuolen lisäosia pystytään lisäämään tai poistamaan käytöstä Extension-valikon kautta. Itsetehtyjä lisäosia voidaan myös lisätä Extension-valikon kautta.

6.3.8 Tranformerit

Transformerit ovat Mirth Connectin tärkein työkalu. Mirth Connectin sanomien käsittely tapahtuu nimenomaan Transformereissa. Transformereita löytyy niin lähdeliittimistä, kuin myös kohteenliittimistä. Tämä tarkoittaa sitä, että sanomia pystytään muokkaamaan molemmissa päissä sanomien käsittelyä (sisääntuleva- ja ulosmeneväliikenne).

Transformerin rakenne perustuu askelmiin, joita pystytään luomaan lähes ääretön määrä transformereihin. Askelmat suoritetaan järjestyksessä, joten ei ole täysin yhdentekevää missä järjestyksessä sanoman eri muokkausosat tehdään. Tottakai lähestulkoon minkä tahansa sanoman käsittely pystytään suunnittelemaan siten, että edelliset askelmat eivät ole riippuvaisia seuraavasta tai toisinpäin.

Transformerien askelmia löytyy useita erilaisia. Oletusaskelman tyyppi on Mapper, joka tarkoittaa sanoman tietyn osan osoittamista muuttujaan. Tämä muuttuja pystytään sijoittamaan myöhemmin muunnettavaan sanomaan sopivaksi, tai vaihtoehtoisesti pistään ainoana parametrinä suoraan kohteelle (esim. tietokanta).

Mapper-askelmatyypillä pystytään sisääntulevan sanoman eri osia osoittamaan suoraan ulosmenävän sanoman pohjaan, jolloin muodostetaan täysin uutta sanomaa. Luonnollisesti tällekin askelmatyypille pystytään määrittämään oletusarvot, sekä merkkijono korvaukset.

Javascript-askelmatyyppi on ehkä kaikkein tärkein askelmatyyppi, koska sillä pystytään monipuolisesti muokkaamaan sanomia ilman rajoituksia. Javascript-askelmissa pystytään käyttämään myös Java-luokkia, mikäli ne on tuotu Mirth Connect-palvelimelle. Javascriptin yksinkertaisuus laskee kynnystä kokeilla sanomien muokkausta suoraan Javascript-koodilla. Mirth Connectin käyttöliittymä mahdollistaa myös esimerkkisanomien elementtiosien raahaamisen suoraan puunäkymästä, joka nopeuttaa pitkien elementtien kirjoittamista Javascriptillä. Tämän lisäksi Mirth Connectiin on määritetty paljon Javascript-funktioita, jotka on tarkoitettu yksityiskohtaisesti Mirth Connectin käyttöön. Näitä Javascript-funktioita pystytään helposti laajentamaan tekemällä omia Javascript-funktioita suoraan Mirth Connect Adminstrator käyttöliittymän kautta.

6.3.9 Suodattimet (fillterit)

Toinen ydinosa transformerien ohella on suodattimet. Suodattimilla pystytään suodattamaan sanomien sisältöä, tai koko sanoma, mikäli esimerkiksi sanomasta löytyy ennalta määrättyä tietoa, mitä ei haluta käsitellä.

Kuten tranformerit, niin myös suodattimet toimivat askelma-arkkitehtuurilla. Tämä tarkoittaa sitä, että jokainen suodatinehto, joka syötetään Mirth Connectiin suoritetaan juuri siinä järjestyksessä, kuten se löytyy käyttöliittymästäkin.

Suodattimien oletus tyyppi on ”Rule Builder”, eli vapaasti suomennettuna sääntöjen rakennuspalikka. Tämä on yksinkertaistettu näkymä, jossa pystytään lisäämään sisääntulevan sanoman tietty kenttä sääntöön ja lisätä tälle ehto seuraavista:

- On olemassa
- Ei ole olemassa
- On yhtäsuuri kuin
- Ei ole yhtäsuuri kuin
- Sisältää
- Ei sisällä

Yksikertaisetun ”Rule Builderin” sijasta, myös suodattimissa voidaan käyttää Javascript-askelmia. Samat hyödyt pätevät myös suodattimissa, sillä Javascript-askelmilla ei juurikaan ole rajoituksia.

6.4 Mirth Connectiin tutustuminen

Mirthiin tutustuminen aloitettiin hakemalla käyttöoppaita ja ohjeita Internetistä. Tämä osoittautui kuitenkin vaikeammaksi tehtäväksi, kuin olisi osannut odottaa. Mirth Connectista ei löydy virallista ohjekirjaa, vaikka kysesssä on erittäin suosittu integraatioalusta. Tämä johtuu käytännössä siitä, että Mirth Corporatio on pääsponsorina ja tarjoaa maksullista asiakastukea tähän avoimenlähdekoodin ohjelmistoon. Heidän tuellaan olisi hyvin vähän arvoa, jos jostain löytyisi täydellinen ohjekirja. Tämä ei kuitenkaan tarkoita sitä, etteikö apua löytyisi jostain. Mirth Connectista tarjotaan yleinen wiki, mutta se ei sisällä, kuin asennus ohjeet ja yleisen kuvauksen kaikista ominaisuuksista. Toinen virallinen tietolähde on Mirthin yhteisösivut. Yhteisösivuilla on yleensä useita ketjuja, joista saattaa löytyä ratkaisu haettuun ongelmaan. Osana yhteisösivuja ovat myös Mirth Corporaation työntekijät ja heidän lisäksi keskustelufoorumeilta löytyy useita muita ammattilaisia, joilta löytyy valtavan paljon tietoa Mirth Connectista, mikäli uskallat vain kysyä.

Ongelmana ei siis ole tiedon määrä vaan se, että tietoa ei löydy keskitetysti ja selkeästi yhdestä paikasta. Tieto on ripoteltuna ympäriinsä (wiki, yhteisö ja muutamat Mirthiä käsittelevät blogit) ja yksinkertaiseenkin asian varmistaminen

saattaa kestää, koska tieto pitää itse kaivaa esille tai kysyä keskustelufoorumilta.

Käyttöön liittyvän tiedonhaun lomassa havaittiin, että kokeilemalla saattaa löytyä myös vastauksia ongelmiin. Tästä lähdettiinkin tekemään muutamia harjoituksia, koska pääteltiin että se on nopein tapa oppia käyttämään Mirthiä. Ensimmäisenä kokeiltiin yksinkertaista sanomankäsittelijää, joka vastaanottaa ja tallentaa sanoman tiedostona jonkin käyttäjän kotihakemistoon. Ensimmäisen testikanavan luominen aloitettiin valitsemalle lähdeliittimeksi LLP ja kohdeliittimeksi File writer (tiedoston tallentaja). Kyseinen kanava ei tarvinnut oikeastaan minkäänlaista konfiguraatiota. Lähdeliittimelle (file writer) annettiin hakemisto, jonne tiedosto tallennetaan (käyttäjän kotihakemisto) ja tiedostonnimi. Sisällöksi tekstitiedostolle laitettiin kaikki sisääntuleva data, eli käytännössä lähetetty sanoma. Tähän liitettiin vielä myöhemmin toinen kohde, joka tallensi muutaman ennalta valitun kentän suoraan omaan testikantaan.

Seuraavaksi harjoiteltiin monimutkaisempaa toteutusta YouTube-palvelusta löydetyn Mirth Connect screencastin avulla. Mirth Connect screencastissä luotiin sovellus, joka lähettää sanoman Mirth Connectille, josta sanoma ohjataan uudelleen suoraan käyttäjän sähköpostiin. Tässä screencastissä näytettiin alusta loppuun asti, alle 5:ssä minuutissa, miten kaikki tapahtui. Kanavasta joka luotiin tätä harjoitusta varten, käytti liittimenään SOAP-kuuntelijaa, joka käytännössä perustaa oman yksinkertaisen Web servicen Mirth Connectiin. Kohdeliittimeksi valittiin sähköposti, jotta voitaisiin SOAP-sovelluksesta lähetetty viesti lähettää edelleen sähköpostiin.

Lähettävää sovellusta varten oli ladattava Microsoftin Visual Studio. Visual Studiolla saatiin ohjevideon avustuksella luotua helposti sovellus, josta löytyi tekstikenttä, sekä nappi jolla kirjoitettu viesti lähetetään. Nappiin liitettiin web-referenssi eli Mirthin SOAP-kuuntelijan antaman WSDL:n osoite. Yllättävää oli, että ongelmia ei harjoituksessa tullut Visual Studion parissa, vaan ainut ongelma, joka harjoituksessa havaittiin, oli autentikaatiovirhe Mirth Connectin päässä. Autentikaatiovirhe saatiin lopulta selvitettyä ja se johtui siitä, että ohjevideon mukaan sähköpostiliittimelle oli laitettu palvelimelle oma

autentikaatio. Tätä ei kuitenkaan tarvittu, koska sähköpostiosoitteen liittäminen sähköpostiliittimeen riitti. Tämän jälkeen kaikki alkoi toimia kuten pitikin ja Visual Studiolla luotu yksinkertainen sovellus lähetti sanoman Mirthiin, josta sanoman sisältö ohjattiin suoraan sähköpostiin.

6.5 Sanomakäsittelijöiden toteutus

Sanomakäsittelijöiden toteutuksessa lähdettiin siitä, että vanhat käytössä olevat sanomien käsittelijät otettiin pohjamalleiksi. Tämä tarkoitti sitä, että Perl-moduuleja oli tutkittava kooditasolla. Integraatioista oli tottakai toteutettu arkkitehtuurikuvaukset ja yksityiskohtaiset kuvaukset sanomien käsittelijöiden prossessikulusta, mutta jotta voisin käyttää aikaisempia käsittelijöitä pohjina, olisi minun käytettävä koodia hyödyksi. Sanomat, joille oli tarkoitus rakentaa käsittelijät olivat:

- SIU-S12, hoidonvaraussanoma
- SIU-S14, hoidonvarauksen muokkaussanoma
- SIU-S15, hoidon peruutussanoma
- ADT-A31, henkilötietojen muokkaussanoma

Sanomakäsittelijöiden toteutus aloitettiin käyttämällä Transformerin Rule Builder -ominaisuutta, koska ohjeissa ja oppaissa mainittiin, että kannattaa aloittaa mahdollisimman yksinkertaisimmasta askelmatapahtumasta. Rule Builder -ominaisuuden rajoittuneisuus huomattiin kuitenkin aikaisessa vaiheessa. Tietojenkäsittelyyn tarvittiin enemmän mahdollisuuksia muokata sanomien sisältöä. Rule Builder –askelmista siirryttiin nopeasti Javascript-askelmiin. Javascript-askelmissa on selvä etu verrattuna Rule Builder –askelmiin, sillä Javasript-askelmmissa ei ole käytännössä mitään rajoituksia. Rule Builder –askelmia voidaan käyttää kuitenkin yksinkertaisten totuusarvoaskelmien rakentamiseen.

Sanomakäsittelijöiden toteutuksen aikana huomattiin, että useissa eri askelmissa tarvittiin samoja arvoja. Ratkaisu tähän oli käyttää kanavalle

globaaleja muuttujia. Globaalimuuttuja pystyttiin asettamaan komennolla channelMap.put('muuttujan_nimi', arvo). Globaaliamuuttujaa oli helppo käyttää asettamisen jälkeen. Esimerkiksi channelMap –komennolla muuttuja asettaminen lähdelittimessä tarkoitti sitä, että asetettua arvoa pystyttiin käyttämään kaikissa tulevissa askelmissa, mukaan lukien kohdeliitin.

Code Templates on Mirth Connectissa osio, johon voi asettaa koodimallipohjia tai funktioita. Opinnäytetyön toteutusosion edetessä, siirrettiin usein käytettyjä pätkiä koodia Code Templatesiin. Koodimallipohjia tai funktioita kutsumalla vähennettiin toiston määrää, joka puolestaan pienensi virheiden määrää.

Javascriptin XML-syntaksi osoittautui olevan hieman oletettua vaikeammaksi ymmärtää. Ohjelmakoodi 4 kuvaa, kuinka sapluunapohjan (template) nimi-noodi haetaan muuttujaan.

Ohjelmakoodi 4. Sapluunapohjan nimi-noodi Javascript XML-syntaksissa

```
var nimi = tmp['ihminen']['nimi'];
```

Syntaksin ymmärtämistä auttoi Mirth Connectin raahaa ja pudota –ominaisuus. Raahaa ja pudota –ominaisuudella pystyttiin raahaamaan haluttu sanoman kohta Javascript koodiin. Ilman tätä ominaisuutta sanomien Javascript XML-syntaksin ymmärtäminen olisi ollut erittäin vaikeaa kokemattomalle resurssille. Raahaa ja pudota –ominaisuutta käyttämällä käyttäjä ymmärtää varmasti, mitä kohtaa sanomasta käytetään.

7 VAIHTOEHTOISET INTEGRAATIOALUSTAT

Toimeksiannon ulkopuolella toteutettiin vielä vaihtoehtoisten integraatioiden vertailu. Vertailu tehtiin siksi, että saataisiin yleiskuva integraatioalustoista, koska kokemusta oli vain yhdestä integraatioalustaratkaisusta. Tarkemmin suurennuslasin alle otettiin 3 ohjelmistoa. Näitä ohjelmistoja tarkasteltiin yleisesti tietolähteiden perusteella ilman, että kokeiltaisiin niitä käytännössä.

7.1 Iguana

Kuten Mirth Connect, myös Iguana keskittyy järjestelmäintegraatioalustana terveydenhuoltoalalle, tarjoamalla tiiviin tuen HL7 standardille. Toisin kuin Mirth Connect, Iguana ei ole avoimenlähdekoodin projekti. Tämä tarkoittaa sitä, että Iguana on täysin kaupallinen tuote, eikä siitä tarjota mitään ilmaiseksi. Ehdottomasti tässä suhteessa Mirth on vahvoilla, koska kustannustehokkuus on yksi tärkeimmistä hankintaperusteista, kun mietitään monipuolisen integraatioalustan hankkimista.

Iguanasta saa helposti raikkaan ja positiivisen mielikuvan, kun heidän sivuillaan vieraillaan. Sivuilta löytyy kuvankaappauksia Iguanasta ja muutama video, jotka ovat pääsääntöisesti markkinoinnin näkökulmasta tehtyjä. Olin yllättynyt, kuinka vähän vähän tarkkoja määrittelyitä Iguanasta sivuilta löytyy. Kuitenkin kun kaivaa hieman syvemmälle tulee esille Interfacewaren (Iguanan kehittäjäyritys) wiki-sivut. Nämä wiki-sivut pitävät sisällään käsittämättömän määrän tietoa Iguanasta ja positiivin yllätys oli videopohjaiset opetusohjelmat, jossa näytetään alusta alkaen, mitä Iguanalla pystytään tekemään.

Videoiden perusteella Iguanan käyttöliittymä näyttää paljon ilmeikkäämmältä ja miellyttävämmältä, kuin Mirth Connectin käyttöliittymä. Termit, joita Iguanassa käytetään, ovat lähestulkoon samoja kuin Mirth Connectissa. Tämä helpottaa käyttöä, jos on käyttänyt aikaisemmin Mirth Connectia.

Iguanan protokolla listaus on huomattavasti pienempi, kuin mitä Mirth Connectissa on. Iguanasta löytyy kuitenkin JSON-protokolla, jota ei Mirth Connectista löydy. JSON tulee sanoista JasvaScript Object Notation ja tarkoittaa kevyen tiedonvaihdon muotoa. JSONin käyttö olisi luontevaa varsikin sovelluskomponentteja (web service) käytettäessä, koska JSONia on helppo parsia ja sille tarkoitettuja ohjelmistokomponentteja on lukuisissa eri ohjelmistokielissä. [35]

7.2 Orion Rhapsody

Orion Rhapsody on nimenomaan terveydenhuoltoalaan ja HL7-standardiin keskittyvä integraatioalusta. Orion Rhapsody tarjoaa vedä ja pudota –tyylisen konfiguraatio-ominaisuuden, jolla hallitaan sanomien käsittelijöiden konfiguraatioita. [36]

Rhapsody tukee myös lukuisia eri protokollia. Mainostuksesta huolimatta Rhapsodystä ei luetella pitkää listaa protokollia, heidän sivuilllaan on mainittuna ainoastaan 3 protokollaa, (HL7, verkkopalvelut, FTP) jota Rhapsody tukee. Rhapsodystä ja Iguanasta jää samanlainen raikas tunne. Tämä saattaa johtua siitä, että heidän kotisivut on kuorrutettu markkinointipuheilla, joilla koitetaan ensisijaisesti myydä mielikuvilla. Kaiken kaikkiaan Orion Rhapsodystä on huono saada laajaa näkemystä, koska he eivät anna missään tarkkoja tietoja tuotteestaan. [36]

Mirth Connectia ja Orion Rhapsodyä on vaikea vertailla, sillä Orion Rhapsodystä on vaikea saada mitään vertailukohtaa. Rhapsodyn tutkiminen jätti paljon kysymysmerkkejä. HL7-standardin tuki on mainittu heidän sivuillaan, mutta Orion Rhapsody ei näytä tarjoavan mitään muuta järkevää liittyen terveydenhuollon järjestelmäintegraatioihin. Orion Rhapsody vaikuttaa paljon käytetyltä tuotteelta, sillä Orion Rhapsodyn parissa työskenneille henkilöille näyttäisi olevan runsaasti töitä tarjolla, tämä huomattiin etsiessä tietoa Orion Rhapsodystä. Orion Rhapsody on varmasti vaihtoehto, jota kannattaa harkita mikäli yritys on hankkimassa, tai vaihtamassa uuteen integraatioalustaan. Kannattaa kuitenkin pitää huolta siitä, että se täyttää kaikki yrityksen vaatimukset järjestelmäintegraatioalustalle.

7.3 BizTalk

BizTalk on Mircrosoftin oma integraatioalusta ratkaisu Windows käyttöjärjestelmille. Se on suunniteltu siten, että integraatioratkaisuksi olisi

helppo sulauttaa SOA-arkkitehtuuri. Sillä on myös etunaan hyvä integroitavuus Microsoftin muihin tuotteisiin (Visual Studio, SQL Server, Hyper-V). [37]

BizTalk ei ole suunniteltu ajatellen terveydenhuoltoalaa, vaan se on suunniteltu laajemmalle bisnesalueelle. Siitä syystä sen ominaisuudet ovat hieman laajemmat, eikä se ole rajoittunut vain tiettyyn toimialaan. Tämä ei tarkoita sitä ettei BizTalk soveltuisi terveydenhuoltoalalle. Nimittäin HUS (Helsingin ja Uudenmaan sairaanhoitopiiri) käyttää BizTalkkia heidän integraatioalustanaan. HUS on Suomen suurin sairaanhoitopiiri, joten heidän integraatioalustansa tulisi olla aina toimiva ja skaalautuva. [38]

Vertailevien käyttäjäkokemuksien mukaan BizTalk on äärimmäisen tehokas ja hyödyllinen integraatioalusta skaalautuvuutensa takia. HUSsin BizTalkin käyttöönotosta huolimatta, BizTalk on saanut moitteita sen HL7-sanomien implementaatio ominaisuuksista. Mirth Connectissa on lyhyempi oppimiskäyrä ja se on suunniteltu yksinomaan ajatellen HL7-standardia. Tästä syystä on hyvä pitää mielessä aina integraatioalustaa valittaessa, että mikä on ensisijainen standardi, jota integraatioissa tullaan käyttämään.

8 TULOSTEN TARKASTELU

Mirth Connectin asennus ja käyttöönotto oli yllätävän helppoa ensikertalaiselle. Asennus oli muutamassa minuutissa valmis ja ohjeiden mukainen konfiguraatio oli saatu valmiiksi noin 30 min:ssa. Asennuksen ja ohjeiden mukaisten konfigurointien jälkeen Mirth Administratorin käynnistäminen sujui helposti selaimesta. Käyttöliittymä oli hieman hankala ensisilmäyksillä, koska lähes jokaisen päävalikon alta löytyi saman nimiset alavalikot, joten oli hieman vaikeaa ymmärtää, miksi näin on ja miten näissä valikoissa tehdyt muutokset tulevat vaikuttamaan. Käyttöliittymän näyttäminen samalta on jokseenkin heikkous ainakin Mirthiä ensikertaa käyttävälle. Kokemuksen myötä valikoiden johdonmukaisuus avautuu käyttäjälle paremmin ja Mirth Administratorin käyttöliittymää on helpompi käyttää.

Integraatioiden hallitseminen jatkuvasti kehittyvällä alalla on nykyään erittäin haastavaa. Arkkitehtuurimallin vaihtaminen vallitsevasta arkkitehtuurista (Point-to-point) hajautettuun integraatioarkkitehtuurimalliin osoittautui huomattavasti tehokkaammaksi tavaksi hallita integraatioiden toimintaa. Hajautetulla arkkitehtuurimallilla saatiin tehokkuutta yksittäisten kohteiden hallintaan. Mirth Connectissa on myös helppo suunnitella minkälainen arkkitehtuuriratkaisu tahansa, koska muutokset näkyvät visuaalisesti käyttäjälle.

Yksittäisten muutosten hallinnan tehokkuus huomattiin, kun piti tehdä nopea muutos yksittäisen tietokentän käsittelyyn ja prosessointiin. Mirth Connectissa pystyttiin muutamalla klikkauksella muokkaaman yhden viestikentän prosessointia, kun taas Perl-moduulissa oli etsittävä oikea kohta koodin seasta ja tehtävä muutos sinne. Ymmärrettävästi Perl-moduuleihin tehtävät muutokset ovat huomattavasti hitaampia ja niiden testaus on huomattavasti vaikeampaa, koska koodin tehty muutos saattaa vaikuttaa moneen muuhunkin asiaan. Mirhiin tehty muutos on helppo testata Administratorissa sijaitsevalla ”Send message” –ominaisuudella.

Mirth Connectin tehokkuus paljastui sen transformereissa. Tranformereilla pystyttiin erittäin tehokkaasti Javascriptin avulla tekemään muutoksia HL7-

sanomaan ja muokkaamaan ulospäin lähetettävän sanoman ulkoasua ja sisältöä. Raahaa ja pudota –tyylinen sanomapohjien elementtien liikuttaminen helpotti työskentelyä. Raahaa ja pudota –tyylistä elementtien liikuttelua oli helppo käyttää, kun muokattiin Javascriptillä monimutkaisia osia sanomamallista. Lopputuloksia esitettäessä keskusteltiin paljon transformerien suodattimien toiminnasta ja ominaisuuksista. Yhdessä todettiin, että suodattimet ovat hyvä tapa ehkäistä niin sanottujen turhien viestien tulemista yrityksen järjestelmiin. Näiden turhien viestien prosessointi on ennen aiheuttanut vasteaikojen kasvua, koska resursseja on käytetty turhaan viestin osien tarkistamiseen, vaikka se ei olisi kuulunut yrityksen järjestelmiin. Mirthin suodattimien avulla sanomat pystytään hylkäämään ennen kuin varsinainen viestin prosessointi alkaa. Viestistä pystytään kätevästi tarkastamaan halutut kentät ja niiden arvot. Viesti voidaan hylätä ennen kuin viestiä on ruvettu prosessoimaan, esimerkiksi jos siitä puuttuu tietty arvo tietystä kentästä.

Kanavakäsittelijöiden toteutuksessa saatiin valmiiksi hoidon varaussanoman käsittelijä, hoidon peruutussanoman käsittelijä ja yhteys- ja kuolintietojen päivityssanoma. Suunnitellusta jäi siis yksi sanomakäsittelijä kesken, eli hoidon muokkaussanoma. Kaikki valmiiksi saadut kanavakäsittelijät olisivat olleet valmiita laajamittaiseen testaukseen, mutta suunniteltu testauskin jäi tekemättä. Laajan testauksen ja S14-sanoman kesken jäämisestä huolimatta projektin toteutusosio oli onnistunut, koska Mirth Connectista saatiin hyvää kokemusta ja hyvinkin yksityiskohtaista tietoa. Lisäksi opinnäytetyöprojektin aikana huomattiin, että käytettty tapa toimia eli suorien tietokantayhteyksien käyttäminen Mirth Connectista, ei välttämättä ole paras mahdollinen. Vaihtamalla arkkitehtuuria hieman enemmän SOA-arkkitehtuurin suuntaan saadaan suorat tietokantayhteydet Mirth Connectista pois, jolloin pystytään parantamaan tietoturvaa entisestään. Sovelluskomponenttien hoitaessa tiedon jakelun kohdejärjestelmälle pystytään varmistamaan tiedon eheys myös kohdesovelluksen päässä.

Opinnäytetyön toteutuksen aikana käytetty PERT-menetelmä osoittautui varsin toimivaksi työkaluksi projektinhallintaan. Suunnittelemani aikataulu tosin ei

täysin pitänyt, mutta se oli enemmänkin suunnittelijan kokemattomuutta eikä menetelmän heikkoutta. Menetelmä ei tietenkään ole täysin varma, mutta se antaa hyvän työkalun aikataulujen suunnitteluun. Kaavojen syöttäminen vaikka yksinkertaisesti Excel-ohjelmistoon antaa melko hyvän ja käytettävän työkalun projektipäällikölle. Kokenut projektipäällikkö pystyy varmasti käyttämään käyttettävyysprosenttia tehokkaasti ja osaa arvioida oman projektiryhmänsä jäsenten ominaisuuksia ulkopuolisena henkilönä hyvin. Kokeneen työnjohtajan käsissä PERT-menetelmä ja resurssi- ja aikataulusuunnitelmakaava on hyvä lisätyökalu, koska kokemuksen myötä työtehtävien arviointi on paljon kypsemmällä tasolla.

9 YHTEENVETO

Opinnäytetyössä tehtiin tutustuminen HL7-standardiin, arkkitehtuurimalleihin ja toteutettiin Mirth Connectilla määrättyjen sanomien, SIU-S12, SIU-S14, SIU-S15 ja ADT-A31 käsittely. Lisäksi työssä perehdyttiin aikatalulun suunnittelemiseen, koska aikataulu oli äärimmäisen tiukka ja siitä syystä aikataulun suunnitteluun ja laskemiseen tuli kiinnittää erityistä huomiota.

Uuden arkkitehtuurimallin valitseminen yhdistettynä Mirth Connectilla luotuihin käsittelijöiden toteutukseen uskotaan tuovan kaivattua hallittavuutta. Hallittavuuden parantuessa pystytään keskittymään paremmin tietoturvaan ja käsittelijöiden toiminnallisuuden parantamiseen. Opinnäytetyöprojektin aikana huomattiin, kuinka työlästä on käydä lävitse Perl-moduuleilla toteutettuja integraatioalustoja. Koodin seasta on vaikea löytää yksinkertaiselta vaikuttavia säännöstöjä, minkä vuoksi uusien säännöstöjen lisääminen tai muokkaaminen vie paljon aikaa.

Toteutetut sanomienkäsittelijät pyrittiin pitämään helposti irroitettavina kokonaisuuksina, jolloin pystytään parantamaan hallittavuutta entistä enemmän. Irroitettavien kokonaisuuksien eli sanomakäsittelijöiden yksittäiset funktiot (Mirth Connectissa ”Template functions”) pystytään ottamaan helposti käyttöön myös tulevissa integraatioprojekteissa, sillä lisäämisen jälkeen ne näkyvät raahaa ja

pudota –valikossa. Mirth Connectissa kaikki toimii taustalla XML-muotoisena, joten asetuksia tai kokonainen palvelinversio pystytään tallentamaan Mirth Connectista XML-muotoisena. XML-muotoisuus on helppo omaksua, ja pienikokoisia tiedostoja on helppo liikutella paikasta toiseen, joten esimerkiksi asetuksien vieminen versionhallintaan on äärimmäisen helppoa.

Mirthin tehokkuus huomattiin sen transformereissa ja suodattimissa. Näiden kahden eri ominaisuuksien tehokkuus oli äärimmäisen hyvä, ja niillä pystyttiin luomaan tehokkaasti säännöstö, jonka avulla sanomat tultaisiin käsittelemään. Sanomien käsittely niiden yksinkertaisimmilla tasollaan oli todella nopeaa ottaa käyttöön. Muutamassa minuutissa saatiin aikaan sanomankäsittelijä, jolla oli oikeasti järkeviä ominaisuuksia, Perl-moduulilla vastaavanlaisen käsittelijän käyttöönotto olisi kestänyt useita tunteja kauemmin.

LÄHTEET

[1] Web-opas, ”Mikä on järjestelmäintegraatio” [www-dokumentti]. Saatavilla: http://www.webopas.net/jintegraatio.html. (Luettu 24.10.2011)

[2] HubPages - Russel O’Brien, ”Integration Architecture Explained” [www-dokumentti]. Saatavilla: http://russellobrien.hubpages.com/hub/Integration-Architecture-Explained. (Luettu 22.4.2012)

[3] Gregor Hohpe, Bobby Woolf. 2003. Enterprise Integration Petterns: Designing, Building, and Deploying messaging solutions. Process Manager, s. 278.

[4] Wikipedia Foundation, ”Palvelukeskeinen arkkitehtuuri” [www-dokumnentti]. Saatavilla: http://fi.wikipedia.org/wiki/Palvelukeskeinen_arkkitehtuuri. (Luettu 22.4.2012)

[5] White paper – Lori MacVittie, “SOA: Challenges and Solutions” [pdf-dokumentti]. Saatavilla: http://www.f5.com/pdf/white-papers/soa-challenges-solutions-wp.pdf. (Luettu 22.4.2012)

[6] w3schools, ”Introduction to XML” [www-dokumentti] Saatavilla: http://www.w3schools.com/xml/xml_whatis.asp. (Luettu 10.6.2012)

[7] Health Level Seven International, “About HL7” [www-dokumentti]. Saatavilla: http://www.hl7.org/about/index.cfm. (Luettu 3.4.2011)

[8] Wikipedia Foundation, ”Layer 7: application layer” [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/OSI_model#Layer_7:_Application_Layer. (Luettu 4.4.2011)

[9] Health Level 7, ”HL7 standards” [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Health_Level_7. (Luettu 3.4.2011)

[10] Health Level Seven International, ”HL7 Standards – Master Grid” [www-dokumentti]. Saatavilla: http://www.hl7.org/implement/standards/v2messages.cfm. (Luettu 3.4.2011)

[11] Interfaceware, ”HL7 Version 3.0” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/v3.html. (Luettu 17.4.2011)

[12] YouTube, ”What does an HL7 message look like?” [video]. Saatavilla: http://www.youtube.com/watch?v=Wyp_4XgzP7A&feature=player_embedded#at=89. (Luettu 16.4.2011)

[13] Interfaceware, ”Example HL7 Message” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/example_hl7_message.html. (Luettu 19.4.2011)

[14] Interfaceware, “Segments” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/segments.html. (Luettu 26.4.2011)

[15] Interfaceware, ”HL7 Segments and Descriptions” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/hl7-standard/hl7-segments.html. (Luettu 26.4.2011)

[16] Interfaceware, ”HL7 Composites” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/composites.html. (Luettu 26.4.2011)

[17] Interfaceware, ”Delimiter Characters” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/delimiter_characters.html. (Luettu 28.4.2011)

[18] Interfaceware, ”Delimiter Escape Sequences” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/hl7_escape_protocol.html. (Luettu 28.4.2011)

[19] Interfaceware, ”Delimiter Redefinitions” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/hl7_delimiter_redefinition.html. (Luettu 28.4.2011)

[20] Interfaceware, ”Determining the HL7 Message Type” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/hl7_message_type.html. (Luettu 4.5.2011)

[21] Interfaceware, ”Repeating and Optional Segments” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/repeating_and_optional_segments.html. (Luettu 4.5.2011)

[22] Interfaceware, ”Custom Segments: HL7 Z-segments” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/z_segments.html. (Luettu 4.5.2011)

[23] Interfaceware, ”Segment Groups” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/segment_groups.html. (Luettu 14.2.2012)

[24] Interfaceware, ”The ACKnowledgement Protocol” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/ack_protocol.html. (Luettu 15.2.2012)

[25] HL7 Standards, ”ACK Message – original mode acknowledgement” [www-dokumentti]. Saatavilla: http://www.hl7standards.com/blog/2007/02/01/ack-message-original-mode-acknowledgement/. (Luettu 15.2.2012)

[26] Interfaceware, ”Open Systems vs. Closed Systems [www-dokumentti]. Saatavilla: http://www.interfaceware.com/open_systems_vs__closed_systems.html. (Luettu 6.3.2012)

[27] Interfaceware, ”Why is HL7 Not Plug-and-play?” [www-dokumentti]. Saatavilla: http://www.interfaceware.com/why_is_hl7_not_plug_and_play_.html. (Luettu 6.3.2012)

[28] Wikipedia Foundation, “Perl” [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Perl. (Luettu 10.6.2012)

[29] Perl HL7 Toolkit, ”What is it?” [www-dokumentti]. Saatavilla: http://hl7toolkit.sourceforge.net/. (Luettu 10.6.2012)

[30] Työmääräarviointi ja aikataulusuunnittelu IT-projekteissa, ”7.3 PERT-menetelmä, 9.1 Henkilöresurssit” [pdf-dokumentti]. Saatavilla: http://www.cs.uta.fi/research/thesis/masters/Helminen_Heli.pdf. (Luettu 10.6.2012)

[31] Elinkeinoelämän keskusliitto, ”Taulukko 1” [pdf-dokumentti]. Saatavilla: http://www.ek.fi/ek/fi/tutkimukset_julkaisut/2011/8_elo/tyoaikakatsaus2010.pdf. (Luettu 10.6.2012)

[32] TechTarget, “Gantt chart” [www-dokumentti]. Saatavilla: http://searchsoftwarequality.techtarget.com/definition/Gantt-chart. (Luettu 10.6.2012)

[33] Mirth Corporation, ”Mirth Connect” [www-dokumentti]. Saatavilla: http://www.mirthcorp.com/products/mirth-connect. (Luettu 31.5.2012)

[34] Lindesk, ”Customizing the Terminal: Create Useful Aliases” [www-dokumentti]. Saatavilla: http://lindesk.com/2009/03/customizing-the-terminal-create-useful-aliases/. (Luettu 3.6.2012)

[35] json.org, ”Introducing JSON” [www-dokumentti]. Saatavilla: http://www.json.org/. (Luettu 10.6.2012)

[36] hl7.com, ”HL7 Interface Engine” [www-dokumentti]. Saatavilla: http://www.hl7.com/interface_engine/. (Luettu 10.6.2012)

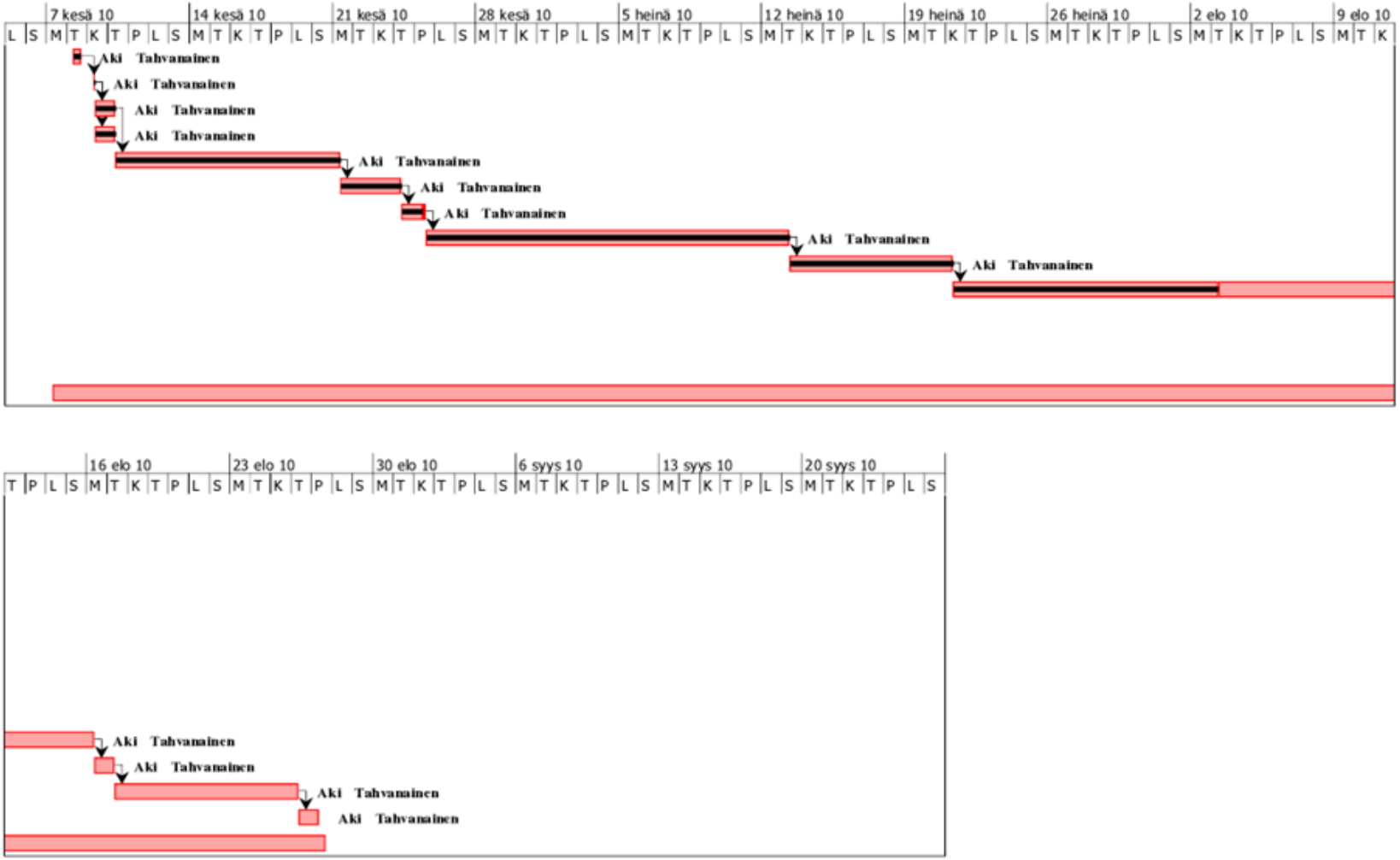
[37] Microsoft, ”What is BizTalk” [www-dokumentti]. Saatavilla: http://www.microsoft.com/biztalk/en/us/overview.aspx. (Luettu 17.6.2012)

[38] Microsoft, ”Hospital District of Helsinki and Uusimaa (HUS)” [www-dokumentti]. Saatavilla: http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=4000002764. (Luettu 17.6.2012)

Alkuperäinen laskettu työnkesto

Tehtävä	Resurssi	Työmäärä, t^0	Kerroin (1,25)	Kerroin (resurssin käytettävyys)	Kerroin (osaaminen)	Työn kesto, t^d, pv	Työn kesto pyöristetty
Tutustuminen vanhaan integraatioon	Aki	2	1,25	100	1,5	3,75	4
Mirth asennuksen suunnittelu	Aki	2	1,25	100	2	5	5
Mirth asennus	Aki	5	1,25	100	2	12,5	13
Mirth asetukset	Aki	8,7	1,25	100	1,9	20,6	21
Testauksen suunnittelu	Aki	2	1,25	100	1,5	3,75	4
Testaus	Aki	7	1,25	100	1,5	13,13	13
						58,70	60
						1,29	

Gant-kaavio


7 kesä 10
14 kesä 10
21 kesä 10
28 kesä 10
5 heinä 10
12 heinä 10
19 heinä 10
26 heinä 10
2 elo 10
9 elo 10
16 elo 10
23 elo 10
30 elo 10
6 syys 10
13 syys 10
20 syys 10
Aki Tahvanainen

Tehtävälistaus

	Nimi	Kesto	Aloitus	Lopetus	Edeltäjät	Resurssien nimet
1	Tutustuminen vanhaan integraatioon	1 päivä	8.6.2010 8:00	8.6.2010 17:00		Aki Tahvanainen
2	Mirth asennuksen suunnittelu	0,25 päivää	9.6.2010 8:00	9.6.2010 10:00	1	Aki Tahvanainen
3	Mirth asennus	1 päivä	9.6.2010 10:00	10.6.2010 10:00	2	Aki Tahvanainen
4	MYSQL asennus	1 päivä?	9.6.2010 10:00	10.6.2010 10:00	2	Aki Tahvanainen
5	Mirthympäristööntutustuminen	7 päivää?	10.6.2010 10:00	21.6.2010 10:00	3	Aki Tahvanainen
6	Tutustuminen vanhaan integraatioon 2	3 päivää?	21.6.2010 10:00	24.6.2010 10:00	5	Aki Tahvanainen
7	IDB ympäristön pystyttäminen testa...	1,5 päivää?	24.6.2010 10:00	25.6.2010 15:00	6	Aki Tahvanainen
8	IDB_S15 kanavan tekeminen	11,5 päivää?	25.6.2010 15:00	13.7.2010 10:00	7	Aki Tahvanainen
9	IDB_A31 kanavan tekeminen	6 päivää	13.7.2010 10:00	21.7.2010 10:00	8	Aki Tahvanainen
10	IDB_S12_S14 kanavantekeminen	18 päivää?	21.7.2010 10:00	16.8.2010 10:00	9	Aki Tahvanainen
11	Testauksen suunnittelu	1 päivä	16.8.2010 10:00	17.8.2010 10:00	10	Aki Tahvanainen
12	Testaus	7 päivää	17.8.2010 10:00	26.8.2010 10:00	11	Aki Tahvanainen
13	Muunnosohjelmien toteutus ja konfig...	1 päivä?	26.8.2010 10:00	27.8.2010 10:00	12	Aki Tahvanainen
14		60 päivää	7.6.2010 8:00	27.8.2010 17:00		