Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

Terveysteknologia

2018

Jaakko Kouvonen

MIRTH INTEGRAATIO - TYÖKALUN HYÖDYNTÄMINEN HL7-VIESTINNÄSSÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka | Terveysteknologia

2018 | 49

Jaakko Kouvonen

MIRTH INTEGRAATIO -TYÖKALUN HYÖDYNTÄMINEN HL7-VIESTINNÄSSÄ

Sosiaali- ja terveydenhuollon tietojärjestelmät käyttävät HL7-standardin mukaisia sanomia tiedonvälitykseen, välittääkseen ja vastaanottamaan potilastietoja. Yleisesti sosiaali- ja terveydenhuollon tietojärjestelmät käyttävät HL7 V2.3 -standardia tai uudempaa HL7 V3 -standardia, jossa HL7-sanomien esitystapa pohjautuu XML:ään. Tulevan SOTE-uudistuksen myötä yhä useampien sosiaali- ja terveydenhuollon tietojärjestelmien tulee vaihtaa tietoja keskenään. Tietojärjestelmien ja ohjelmistojen integraatioissa on havaittu vakavia ongelmia. Sosiaali- ja terveydenhuollon tietojärjestelmät on rakennettu eri arkkitehtuureilla, tietojärjestelmät ovat vanhoja, lisäksi tietojärjestelmät eivät noudata yhtä ja samaa tiedonvälitysstandardia.

Opinnäytetyössä selvitettiin kahden yleisimmän HL7-standardin ominaisuudet sekä sitä, mihin standardeja tarvitaan sekä vertailtiin näitä kahta standardia keskenään. Lisäksi työssä perehdyttiin Mirth -integraatioalustaan, mihin se on tarkoitettu sekä mitä suuria ongelmia, joka liittyvät tietojärjestelmien ja ohjelmistojen välisiin integraatioihin Mirth -integraatioalustan on tarkoitus ratkaista maailmanlaajuisesti.

Osana tavoitteena oli selvittää, miten Mirth -integraatioalustaa voidaan käyttää osana HL7-sanomien välitystä. Tarkoituksena oli parsia tulevasta HL7-sanomasta tietyt halutut tiedot. Työn tavoitteena oli luoda Mirth -integraatioalustalla lähtevä HL7-sanoma siten, että sanomaan parsitaan tietyt tiedot tekstitiedostosta, sekä tavoitteena oli lähettää HL7-sanoma käyttämällä Mirth -integraatioalustaa.

Tuloksista kävi ilmi, että Mirth -integraatioalussa on haastavaa parsia tietoja, koska JavaScript-ohjelmointi Mirth -integraatioalustassa on haastavaa. Tekniikat, jotka helpottavat ohjelmointia, olivat kovin riittämättömät. Tulevaisuudessa on parempi käyttää ulkoista Java-kirjastoa, jossa on suoritettavat ohjelmakoodit, jolloin ohjelmointi helpottuu huomattavasti sekä ohjelmankoodien ylläpito ja versionhallinta ovat hallittavissa.

ASIASANAT:

HL7, HL7 V2.3 -standardi, Mirth -integraatioalusta, Järjestelmäintegraatio, Parsinta, JavaScript

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Health Informatics

2018 | 49

Jaakko Kouvonen

UTILIZING THE MIRTH INTEGRATION PLATFORM FOR HL7 COMMUNICATIONS

Social and health care information systems use messages in the HL7 standard to communicate, transmit and receive patient information. In general, social and health care information systems use the HL7 V2.3 or later HL7 V3 standard, where the HL7 message format is based on XML. With the upcoming service structure reform of social welfare and health care, more and more social and health care information systems will need to exchange information with each other.

Serious problems have been identified in the integration of information systems and software. Social and health care information systems are built in different architectures, the information systems are old, and the information systems do not comply with one and the same communication standard.

The thesis investigated the properties of the two most common HL7 standards and the standards for which they are needed, as well as comparing these two standards with each other. In addition, the Mirth Connect integration platform was explored in the work, as well as the major problems associated with integration between information systems and software. The Mirth Connect integration platform is intended to be solved globally.

The aim of the study was to find out how the Mirth Connect integration platform can be used as a part of HL7 messaging. The intention was to partake in the future HL7 message with certain desired information. The aim of the thesis was to create a HL7 message from the Mirth Connect Integration Platform to parse specific information from the text file and to send the HL7 message using the Mirth Connect Integration Platform.

The results showed that in Mirth Connect integration is challenging to parse information, since JavaScript programming in the Mirth Connect integration platform is challenging. Techniques that make programming easier were very inadequate. In the future, it is better to use an external Java library with executable program codes, which makes programming much easier and the program code maintenance and version control are manageable.

KEYWORDS:

HL7, HL7 V2.3 standard, Mirth integration platform, System integration, Parsing, JavaScript

SISÄLTÖ

KOODIT

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET

ACK	Acknowledgement. Signaali, joka kertoo lähettäjälle, jos lähetys onnistui ilman virheitä.
AIMD	Active implantable medical devices. Aktiivisia implantoitavia lääkinnällisiä laitteita koskeva jäsenvaltioiden lainsäädäntö (Ståhlberg 2018).
ASCII	American Standard Code for Information Interchange. Tapa muuttaa merkkejä numeeriseksi esitykseksi, jotta tietokoneet ymmärtävät merkit (AsciiTable).
DOM	Document Object Model. Ohjelmointirajapinta HTML- ja XML-dokumenteille. Tapa kuvata rakenteisen dokumentin rakenne puuna (MDN web docs).
EHR	Electronic Health Record. Sähköinen terveysrekisteri, johon kerätään sähköisesti terveystietoja yksittäisistä potilaista ja väestöryhmistä (Gunter & Terry 2005).
HL7 V2.x -standardi	Tiedonvälitysstandardi potilastietojärjestelmien välisessä tiedonsiirrossa.
HL7 V3	Uusi lähestymistapa kliinisen tiedonvaihtoon esittämällä sähköiset asiakirjat XML-syntaksilla.
HL7	Health Level Seven. Voittoa tavoittelematon ANSI-akkreditoitu standardiorganisaatio, joka tarjoaa kattavat standardit sähköisen terveydenhuollon tietojen vaihtoon, integroitumiseen, jakamiseen ja hakemiseen (Health Level Seven 2018).
HL7-sanoma	Sanoma, joiden avulla pystytään vaihtamaan potilastietoja eri sosiaali- ja terveydenhuollon tietojärjestelmien välillä.
IVD	In-vitro diagnostics. In vitro –diagnostiikkaan tarkoitetuista lääkinnällisten laitteiden Euroopan parlamentin ja neuvoston direktiivi (Ståhlberg 2018).
MD	Medical Devices. Lääkinnällisten laitteiden neuvoston antama direktiivi (Ståhlberg 2018).
MLLP	Minimal Lower Layer Protocol. Minimalistinen OSI-mallin viidenteen kerrokseen perustuva protokolla (Health Level Seven International 2018b). jonka avulla siirretään suurin osa HL7-sanomavälityksistä.
NACK	Negative-acknowledgement. Signaali, joka kertoo lähettäjälle, jos lähetys epäonnistui, tai ilmaisee jos lähetyksen aikana tapahtui jokin virhe.
NodeJS	Avoimeen lähdekoodiin perustuva palvelin viitekehitys.

OSI-Malli	ISO organisaation kehittämä malli, joka kuvaa miten eri tietoliikennejärjestelmät tulisi suunnitella (Tietotekniikan peruskurssi).
PDF	Portable Document Format. Tiedostomuoto, jota käytetään dokumenttien luotettavaan esittämiseen ja vaihtoon (Adobe 2018).
RIM	Reference Information Model. Tietomalli joka edustaa HL7-viestien tietojen mukana olevien yhteyksien välisiä yhteyksiä (Health Level Seven International 2018f).
SOTE	Sosiaali- ja terveydenhuolto.
SQL	Structured query language. Standardoitu kyselykieli tietojen tallentamiseen, hakemiseen ja muokkaamiseen tietokannassa (w3schools).

1 JOHDANTO

Sosiaali- ja terveydenhuollon tietojärjestelmät käyttävät HL7-standardin mukaisia sanomia välittääkseen ja vastaanottamaan potilastietoja. Asiakas- ja potilastietojärjestelmät ovat keskeinen osa potilaiden ja asiakkaiden palvelu- ja hoitoprosesseja, potilasturvallisuuden edellyttämää hoidon ja palveluiden dokumentointia sekä toiminnanohjausta (Jormalainen 2015). Terveydenhuoltoalalla yksi suurimmista ongelmista on tietojärjestelmien erilainen arkkitehtuuri ja niiden toteutus. Viime vuosina terveydenhuollon eri tietojärjestelmiä on pyritty integroimaan keskenään, jotta tietojärjestelmät keskustelisivat toisten järjestelmien kanssa ja vaihtaisivat tietoja moitteettomasti. Tämä on osoittautunut erittäin haasteelliseksi ratkaisuksi, koska tietojärjestelmät on tehty eri tekniikoilla. Lisäksi järjestelmät käyttävät eri protokollia toimiakseen sekä järjestelmät käyttävät eri HL7-standardeja. Järjestelmät eivät siis noudata yhtä ja samaa tehokasta tiedonvälitysstandardia.

SOTE-uudistuksen myötä yhä useampien sosiaali- ja terveydenhuollon tietojärjestelmien tulee pystyä kommunikoimaan ja vaihtamaan tietoja muiden sosiaali- ja terveydenhuollon tietojärjestelmien kanssa. Vuonna 2015 erityisesti Suomessa suurimpia ongelmia aiheutuu sosiaali- ja terveydenhuollon tietojärjestelmien yhteen toimivuus ongelmista. Perusterveydenhuollon, erikoissairaanhoidon ja yksityisen sektorin potilastietojärjestelmät eivät kykene vaihtamaan tietoa keskenään sekä kaikkien tarvittavien tietojen kokoaminen yhteen on vaivalloista, koska tiedot ovat hajallaan eri järjestelmissä ja tietokannoissa (Jormalainen 2015). Mirth -integraatioalusta, joka perustuu avoimeen lähdekoodiin, ensisijaisena tavoitteena on tarjota kestävä ratkaisu sosiaali- ja terveydenhuollon tietojärjestelmien tiedonvälitys ja integraatio-ongelmiin.

Tämä opinnäytetyö käsittelee PerkinElmer-teknologiakonserniin kuuluvan Wallac Oy:n tarvetta selvittää Mirth -Integraatiotyökalun käyttämistä HL7-sanomien välityksessä. Tavoitteena on selvittää, miten Mirth -integraatiotyökalulla pystytään vastaanottamaan HL7-sanoma, parsimaan tulevasta HL7-sanomasta tiettyjä tietoja sekä miten parsitut tiedot saadaan tallennettua tekstitiedostoon. Lisäksi tavoitteena on selvittää, miten Mirth -integraatiotyökalulla luodaan lähtevä HL7-sanoma, ja miten lähtevään sanomaan parsitaan tiedot tekstitiedostosta. HL7-sanomien parsintaa tutkitaan kahdella eri JavaScript-ohjelmalla ja saatujen tuloksien pohjalta arvioidaan, miten hyvin Mirth -integraatiotyökalu soveltuu HL7-sanomien parsintaan.

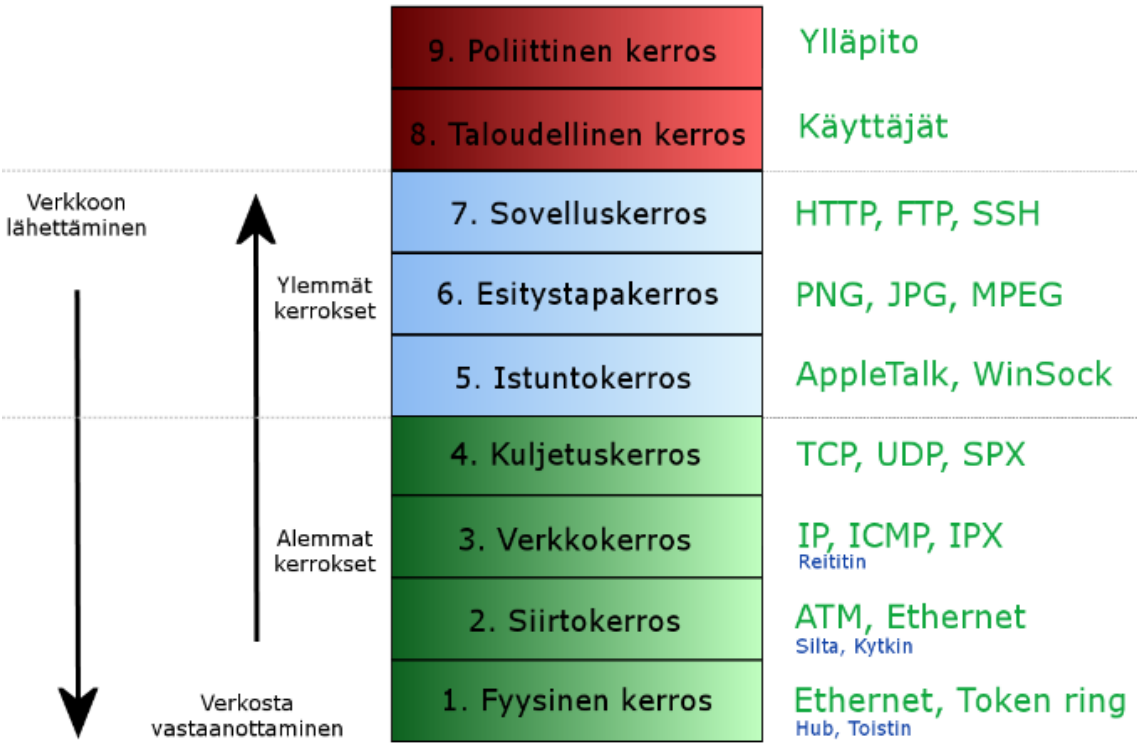
Mirth -integraatioalustasta on tehty muutama opinnäytetyö, joissa on tutkittu mikä Mirth -integraatioalusta on, miten sitä käytetään järjestelmäintegraatioissa, sekä miten sillä luodaan HL7 V3 -standardin mukaisia sanomia. Opinnäytetyöt on toteutettu hyvin yksinkertaisilla esimerkeillä, joiden avulla Mirth -integraatioalustan ominaisuuksia on analysoitu. Opinnäytetöissä ei kuitenkaan ole suoritettu JavaScript-ohjelmointia Mirth -ohjelmistossa, eikä tutkittu sitä, miten HL7 V2.3 -sanomista saadaan JavaScript-ohjelmoinnilla parsittua määrättyjä tietoja.

Opinnäytetyö jakautuu kahteen eri kokonaisuuteen, joista ensimmäisessä osassa käsitellään HL7-standardeja: mihin tarkoitukseen HL7-standardit on luotu sekä mikä organisaatio on vastuussa HL7-standardien kehityksestä. Ensimmäisessä osassa käsitellään myöskin erityisesti HL7 V2.x -standardia sekä hieman HL7 V3 -standardia ja miten nämä kaksi standardia eroavat toisistaan. Luvussa 3 kerrotaan Mirth -integraatioalustan tarkoituksesta. Mitä sillä voidaan tehdä, mitä ongelmia Mirth -integraatioalusta pyrkii ratkaisemaan.

Opinnäytetyön toinen kokonaisuus käsittelee Mirth -integraatioalustan soveltamista HL7-sanomien parsinnassa. Ensimmäiseksi luodaan HL7-sanoma, joka lähetetään Mirth -integraatioalustalle, minkä jälkeen parsitaan JavaScript-ohjelmalla HL7-sanomasta tiettyjä tietoja. Parsittavat kentät ja järjestys on määritelty asetustiedostossa ja parsitut tiedot tallennetaan tekstitiedostoon. Lopuksi kerrotaan, miten Mirth -integraatioalustalla luodaan sellainen lähtevä HL7-sanoma, jossa on parsittu JavaScript-ohjelmalla tekstitiedostosta tarvittavat tiedot ja lisätty tiedot lähtevään HL7-sanomaan. Tämän lisäksi saatuja johtopäätöksiä arvioidaan.

2 HL7

Health Level Seven International (HL7) on yhdysvaltalainen vuonna 1987 perustettu voittoa tavoittelematon ANSI-akkreditoitu, standardointiorganisaatio, joka on sitoutunut tarjoamaan erittäin kattavia viitekehyksiä, standardeja terveydenhuollon sähköisten tietojen siirtämiseen, integroimiseen, jakamiseen, hakemiseen, terveyspalveluiden hallintaan, palveluiden toimittamiseen sekä palveluiden arviointiin. HL7:n toimintaa tukee yli 1 600 jäsentä yli 50 eri maasta, mukaan lukien yli 500 työntekijää, terveydenhuollon tarjoajia, valtion sidosryhmiä, maksajia, lääkeyhtiöitä, myyjiä, toimittajia ja konsulttiyrityksiä. ”Level Seven” viittaa kansainvälisen standardisoimisjärjestön ISO seitsenkerroksiseen viestintämalliin, OSI-malliin (Kuva 1). ja sen seitsemänteenkerrokseen, sovelluskerrokseen (Health Level Seven International 2018a). Sovelluskerroksen tarkoituksena on tarjota varsinaiset rajapinnat sovelluksille, jotta sovellukset voivat olla yhteyksissä toisiinsa (Koudata). Käytännössä sovelluskerros toimii linkkinä ohjelmaan, joka tarvitsee tiedonsiirtoa (Ala-Mutka ym. 2002)



Kuva 1. OSI-malli (Wikimedia commons 2018).

Suomessa toimiva paikallisyhdistys HL7 Finland ry perustettiin syksyllä vuonna 1995. Yhdistyksen tavoitteena on edistää järjestelmäintegraatioperiaatteella tapahtuvaa tietojärjestelmien kehitystyötä ja terveydenhuollon tietojärjestelmästandardien käyttöä. Yhdistyksellä on jäseninä nykyään jo yli 70 eri yritystä ja muita terveydenhuollon organisaatioita. Merkittävimmät jäsenet ovat käyttäjäorganisaatioita, kuten sairaanhoitopiirejä sekä tietojärjestelmien toimittajia.

Hl7 Finland paikallisyhdistyksellä on käytössään kattava rajapintakartta, jonka tavoitteena on tuottaa tukea ja kuvausta siitä, mitä standardeja ja avoimia rajapintamäärittelyjä on käytettävissä ja käytössä Suomessa eri tarkoituksiin ja integraatioihin sosiaali- ja terveydenhuollon tietojärjestelmiä varten (HL7 Finland ry 2018a).

2.1 Standardit

HL7 ja kaikki siihen kuuluvat paikallisyhdistykset tarjoavat viitekehykset ja niihin liittyvät standardit sähköisen terveydenhuollon tietojen vaihtoon, integrointiin, jakamiseen ja hakemiseen. Nämä standardit määrittelevät, miten tiedot on pakattu ja miten tiedot välitetään eri järjestelmien välillä. Standardit kertovat selkeästi, mitä kieltä, rakennetta ja tietotyyppejä käytetään, joita tarvitaan järjestelmien saumattomaan integraatioon. HL7-standardit tukevat niin kliinistä käytäntöä ja terveydenhuollon johtamista kuin myös toimitusta ja arviointia ja nämä standardit tunnetaan ja niitä käytetään maailmanlaajuisesti terveydenhuollon tietojärjestelmien välisessä kommunikaatiossa ja integraatiossa (Health Level Seven International 2018b).

HL7-standardit on ryhmitelty seitsemään ryhmään. Ensimmäisessä ryhmässä ovat kaikki ensisijaiset standardit. Ensisijaisia standardeja pidetään kaikkein suosituimpina standardeina, koska ne ovat olennaisessa osassa järjestelmän integroimisessa, järjestelmien yhteen toimivuudessa sekä yhteensopivuudessa. Standardit, jotka ovat useimmin käytetyt ja jotka ovat kaikkein vaativimpia, kuuluvat ensimmäiseen ryhmään. Toisessa ryhmässä ovat niin sanotut perustamisstandardit ja perustavanlaatuiset standardit, jotka määrittelevät perusratkaisut ja rakennusosat, joita tarvitaan standardien luomiseen, teknologian infrastruktuurin, jotka HL7-standardien käyttävien organisaatioiden on hallittava. Kolmannessa ryhmässä ovat Kliiniset ja hallinnolliset alueet. Organisaatiot, jotka ovat kliinisellä erikoisalalla, heille suunnatut viestintä- ja dokumenttistandardit kuuluvat kolmanteen ryhmään. Tämän ryhmän standardeja käytetään yleensä silloin, kun organi-

saatiossa on jo käytössä ensisijaiset standardit. Neljänteen ryhmään kuuluvat EHR-Profiilit. Nämä standardit tarjoavat toiminnallisia malleja ja profiileja, jotka mahdollistavat sähköisten terveystietojen hallinnan. Viidennessä ryhmässä ovat kaikki toteutusoppaat. Ryhmän standardit koskevat toteutustapoja tai tukitietoja, jotka on luotu käytettäväksi olemassa olevan standardin kanssa. Kaikki asiakirjat, jotka ovat tässä ryhmässä ovat lisä- ja täydennysstandardeja vanhemmille standardeille. Kuudennessa ryhmässä ovat kaikki standardit, jotka koskevat säännöksiä ja viitteitä. Kaikki tekniset määrittelyt, ohjelmointi struktuurit ja ohjeet, joita tarvitaan ohjelmistojen ja standardien kehittämiseen kuuluvat kuudenteen ryhmään. Viimeisessä, seitsemännessä ryhmässä eli koulutus ja tietoisuus ryhmässä ovat standardit, joita käytetään kokeiluhankkeissa, nykyiseille olemassa oleville hankkeille, sekä hyödyllisiä resursseja ja työkaluja, jotka täydentävät edelleen HL7-standardien ymmärtämistä ja hyväksymistä (Health Level Seven International 2018b).

HL7-organisaation kehittämät standardit ovat ensisijaisia standardeja sähköisten terveystietojen siirtämisessä eri järjestelmien välillä. Vaikka osa standardeista on jo kehitetty vuosikymmeniä sitten, standardit kuitenkin kehittyvät jatkuvasti teknologian ja tietojärjestelmien kehityksen mukana, joten HL7-standardeista on julkaistu useita eri versioita historian saatossa.

2.2 HL7 V2.x

HL7 Version 2 on viestintästandardi, jota käytetään sähköisten terveystietojen tiedonvälitykseen, mikä todennäköisesti tekeekin tästä standardista kaikkein laajimmin sovelletun terveydenhuollon standardin. Tämä viestintästandardi mahdollistaa kliinisten terveystietojen vaihdon eri järjestelmien välillä. Standardi on suunniteltu siten, että se tukee keskeisesti potilastietojärjestelmiä ja niiden välisiä tiedonsiirtoja. Standardi tukee myöskin hajautettua ympäristöjä, joissa tiedot sijaitsevat useissa eri järjestelmissä (Health Level Seven International 2018d). Vaikka HL7 Versio 2 -standardi on jo alun perin kehitetty viime vuosituhannella, sitä edelleenkin kehitetään ja parannetaan käyttäjien tarpeisiin. Merkittävin hyöty, joka saadaan käyttämällä HL7 Versio 2 -standardia, on siinä, että standardi tukee uusia tietoja, jotka voidaan lisätä tapausraporttiin. Standardi tukee myöskin kyselyitä jotka tulevat toiselta järjestelmältä ja standardi tukee myöskin sen, miten tulevaan kyselyyn vastataan. EHR-järjestelmästä voidaan pyytää ja vastaanottaa potilaan immunisaatiohistoria rekisteristä. Standardit on toteutettu siten, että ne ovat ”taaksepäin

yhteensopivia”. Järjestelmä, joka käyttää terveystietojen vastaanottamiseen ja lähettämiseen standardi versiota 2.5 pystyy myöskin vastaanottamaan ja lähettämään standardi versio 2.3 mukaisia viestejä. Tämä on tavoitteena standardien kehityksessä, mutta, että viestit vastaanotetaan ja lähetetään onnistuneesti eri järjestelmien välillä, jotka käyttävät eri standardiversiota ei kuitenkaan aina toteudu. Koska HL7-yhteisö on kasvanut historian saatossa kokeneemmaksi standardien kehittämisorganisaatioksi ja tekniikka on kehittynyt, HL7 on kehittänyt uuden standardimallin Versio 3, jota käytetään pääosin muualla maailmassa kuin Yhdysvalloissa, mikä on johtanut siihen, että standardin Versio 2:n kehitykseen on vaikuttanut myöskin standardi versio 3. (Public Health Informatics Institute 2016). Uusin versio standardista Versio 2 on HL7 Versio 2.8.2 (Health Level Seven International 2018d). Suomessa terveydenhuoltojärjestelmissä kuitenkin käytetään standardin versiota HL7 Versio 2.3.

HL7-sanoma lähetetään jonkin tietyn reaalimaailman tapahtuman tapahduttua. Tapahtumassa jokin järjestelmä lähettää tietoja jollekin laitteelle tai vastaanottaa tietoa. Potilas voi kirjautua vuodeosastojärjestelmään, joka toimii liipaisimena (trigger) sisään kirjaus sanomalle. HL7-sanoma näyttää pitkältä merkkijonolta, joista suurin osa merkeistä on kirjoittuvia ASCII-merkkejä, sanomilla on kuitenkin omat sisäiset rakenteensa. HL7-sanomat rakentuvat tietoryhmistä (segmenteistä) ja tietoryhmät puolestaan tietokentistä (kenttä). Riippuen tietokentän tietotyypistä, sanoma voi edelleen jakautua komponentteihin (components) ja vielä alikomponentteihin (subcomponent), tästä voidaan käyttää myöskin nimitystä alikenttä. HL7 Versio 2.3 -sanomien perusrakenne on kuvattu kuvassa (Kuva 2). Sanomilla on kolmikirjaiminen tunnus, esimerkiksi MSH (Message Header), josta selviää mm. kuka sanoman lähettää ja kenelle. Tietoryhmillä on myös kolmikirjaiminen tunnus, esimerkiksi PID (Person Identification), joka sisältää perustietoja potilaasta mm. nimi, ikä, syntymäaika ja osoite (HL7 Finland ry 2018b).

```
MSH|^~\&|AcmeHIS|StJohn|ADT|StJohn|20060307110111||ADT^A04|MSGID20060307110111|P|2.4
EVN|A04
PID|||12001||Jones^John||19670824|M|||123 West St.^^Denver^CO^80020^USA
PV1||O|OP^PAREG^||||2342^Jones^Bob|||OP|||||||2|||||||||||||||||||||||20060307110111|
AL1|1||3123^Penicillin||Produces hives~Rash~Loss of appetite
```

Kuva 2. HL7 V2.X -standardin mukainen sanoma (Corepointhealth 2018a).

HL7:n koodaussäännöt määräävät, miltä siirrettävä sanoma näyttää siirron aikana, kaikki sanoman osat (tietoryhmät, tietokentät, komponentit ja osakomponentit) (HL7 Finland ry 2018b).

2.3 HL7 V2.H3 sanomissa käytetyt tietoryhmät

HL7-sanomat muodostuvat erilaisista tietoryhmistä. Tietoryhmät voi esiintyä HL7-sanomassa vain yhdessä kohtaa sanomaa ja tietoryhmä voi olla pakollinen tai vapaaehtoinen ja se voi myös toistua (HL7 Finland ry 2018b), riippuen siitä, mitä tietoja halutaan lähettää ja kuinka paljon lähetettävää tietoa on. Tietoryhmä alkaa aina kolmikirjaimisella tunnuksella (segmentin tunnuksella), esimerkiksi PID. Tietoryhmät koostuvat tietokentistä, jotka erotetaan toisistaan tietokentän erotinmerkillä (”|”). Tietoryhmän tunnus ja ensimmäinen tietokenttä erotetaan myös toisistaan tietokentän erotinmerkillä (HL7 Finland ry 2018b). Tietoryhmät, joita käytetään Suomen terveydenhuoltojärjestelmien HL7-sanomissa ovat: MSH, PID, PV1, NTE, ORC ORB ja OBX.

MSH (Message Header) on pakollinen tietoryhmä HL7-viesteissä ja sitä käytetään sanoman lähettäjän, vastaanottajan ja sanomatyypin tunnistamiseen. Yleisesti voidaan sanoa, että siinä on tiedot, kuka lähettää HL7-sanomia, kenelle ja millä järjestelmällä. PID (Patient Identification Segment) käytetään ensisijaisena keinona potilaan tunnistetietojen siirtämiseen kaikkien järjestelmien välillä. PID sisältää kiinteiden perustunnistustietojen (nimi, ikä, osoite jne.) lisäksi sellaisia demografisia tietoja, jotka muuttuvat vain harvoin. PV1:ssä on hoitojakson tai käynnin tiedot (HL7 Finland ry 2018c). Tiedot ovat mm. hoitopaikan tiedot (missä potilas kävi hoidossa), vastaanottajan tietoja (kuka otti potilaan vastaan hoidossa) etunimi, sukunimi, osastontiedot, jossa potilas kävi. NTE-tietoryhmässä on kaikki huomautukset tai kommentit, jotka koskevat HL7-viestiä. Huomautuksessa voidaan myös ehdottaa jotain toimenpidettä, jos potilaalta löydetään jokin poikkeama. Esimerkiksi ei ole vasta-aineita HIV-1 vastaan, jolloin voidaan ehdottaa, että potilaalle kannattaa tehdä verikoe (Corepointhealth 2018b). ORC eli yleinen tilaus tietoryhmä sisältää tiedot, jos tilataan jostain jotain materiaalia. Tiedoissa käy ilmi mm. Tilaajan tilaustunniste, tilaajan yleiset tiedot, tilauksen vastaanottajan tiedot, tilausaika jne. OBR tietoryhmä on tutkimuspyyntö tietoryhmä. Siellä on tiedot mm. mitä halutaan tutkia, mitä tutkitaan, pyydetyt tutkimusajat, otettavat näytteet jne. OBX on tietoryhmä, jossa ovat kaikki kliiniset lisätiedot, jotka tulevat tutkimuspyynnön yhteydessä (Kuva 3). Tiedoissa voi olla mm. milloin potilaalta on tutkittu jotain tiettyä asiaa, potilaan tietoja, kuten paino, pituus jne. (HL7 Finland ry 2018c).

```
OBR|1|Lähetenumero||3270^U-Perust^LAB-KL-98|1||1998…
OBX|1|ST|Pyy19^Pyyntöindikaatio^PYL-ML2-HYKS|1|I Perustut…
OBX|2|ST|Min21^Minkälainen perustutkimus^PYL-ML2|1|k…
OBR|2|Lähetenumero2||3270^U-Perust^LAB-KL-98|1||1998…|
OBX|1|ST|Pyy19^Pyyntöindikaatio^PYL-ML2|2|I Perustut…
OBX|2|ST|Min21^Minkälainen perustutkimus^PYL-ML2|2|k…
OBX|3|ST|PYL34^Näytteenottotapa^PYL-ML2|2|PLV aamupa…
```

Kuva 3. Kliiniset lisätiedot (HL7 Finland ry 2018c).

OBX-tietoryhmiä voi olla useita allekkain ja ne erotetaan toisistaan laittamalla juokseva järjestysnumero ensimmäiseen kenttään. Kaikki OBX-tietoryhmät, jotka ovat OBR-tietoryhmän alla liittyvät ylimpään OBR-tietoryhmään, kunnes tulee joki uusi tietoryhmä tai HL7-viesti loppuu.

2.3.1 Tietokentät

Tietokenttä on HL7-viestissä vaihtelevan mittainen merkkijono. Tietokenttiä ei yleensä täytetä määrättyyn pituuteen, esimerkiksi välilyönneillä, vaan tietojärjestelmien, jotka kommunikoivat keskenään lähettämällä ja vastaanottamalla HL7-viestejä, tulee osata lukea vaihtelevan mittaisia HL7-viestejä. HL7-viestissä on kuitenkin erityisesti huomattava, että puuttuva tieto ja tyhjä tieto ovat eriasioita. Jos HL7-viestissä tieto puuttuu, niin viestissä tietokenttää edeltävä ja seuraava erotinmerkki ovat peräkkäin (||). Tässä tapauksessa vastaanottavassa järjestelmässä vastaava kenttä jätetään muuttamatta eli siinä säilyy mahdollinen aikaisempi arvo. HL7-viestissä tyhjä tieto esitetään kahdella peräkkäisellä lainausmerkillä, ””. Tyhjän tiedon tapauksessa vastaanottava järjestelmä nollaa vastaavan kentän eli mahdollinen aikaisempi tieto poistetaan. Jokainen tietokenttä on jotain tietotyyppiä, esimerkiksi merkkijono, jonka tietotyyppi on ST (string), tai esimerkiksi numeerinen, jonka tietotyyppi on NM (numeric), josta esimerkki näkyy kuvassa 4. (HL7 Finland ry 2018b).

```
MSH|^~\&|From||To||199809192228||ORM^O01|Sanomanumero||2.3|||NE|AL|FI|ASCII|
PID|1|070707-0707^^^From^HETU|343432^^^From^POTNUM||sukunimi^etunimi^nimi||19070707|2|
PV1|1|O|||||^VASTAANOTTAJA^ETUNIMI^^^TKL|||||||||||||||||||||||||||OSASTO|
ORC|NW|Lähetenumero|||||||199809291402|^VASTAANOTTAJA^NIMI^^^TKL|||||||osasto|
OBR|1|Lähetenumero||3270^U-Perust^LAB-KL-98|||199809300600||||O|||||||||||||||||||PORT|
```

Kuva 4. Pyyntöviesti esimerkki (HL7 Finland ry 2018c).

Tietokenttä voi myöskin toistua. Tässä tapauksessa tietokentän peräkkäiset toistot erotetaan toisistaan toistoerottimella (~). Komponenttien ja osakomponenttien erotinmerkit lasketaan mukaan pituuteen, mutta ei toistoerotinta, sillä viestin maksimipituus koskee

vain yhtä toistumaa. Viestin maksimi pituus on vain ohjeellinen arvo, joka toimii tavallaan oletuksena. Organisaatiokohtaisesti voidaan kuitenkin viestin maksimipituudelle sopia jokin muukin arvo, kuin mikä on standardissa määritelty (HL7 Finland 2018b).

2.3.2 Tietotyypit

HL7-viesteissä on mukana myöskin tietotyyppejä, jotka tuovat mukanaan tietokentän hienorakenteen. Tietokenttä voi silloin jakautua pienempii osiin (komponentteihin ja osakomponentteihin). Tietotyyppi voi olla ns. perustietotyyppi tai johdettu (koostettu) tietotyyppi. Kaikki johdetut tietotyypin komponentit koostuvat perustietotyypeistä. Jos perustietotyyppi pitää sisällään komponentteja, niin johdetussa tietotyypissä niistä tulee osakomponentteja. Yksi esimerkki tällaisesta on esim. koostettu hinta (CP), jossa hinnan (MO) komponenteista (määrä ja rahayksikkö) tulee osakomponentteja (HL7 Finland 2018b.) Komponentteihin jakautuvasta tietotyypistä yksinkertainen esimerkki voisi olla: |200^EUR|, jossa 200 on määrä, EUR puolestaan vastaa sitä, mitä rahayksikköä määrä tarkoittaa. Osakomponentteihin jakautuva on tietorakenne (Kuva 5), jossa lääkäri perii 300 euroa ensimmäiseltä 15 minuutilta ja lisäksi 100 euroa, jos vastaanottoon kuluu yli 15minuuttia, mutta alle tunnin. Toimistomaksu on 100 euroa (HL7 Finland ry 2018b).

```
|300&EUR^UP^0^15^min^P~100&EUR^UP^16^60^min^P~100&EUR^AP|
```

Kuva 5. Osakomponentteihin jakautuva tietorakenne (HL7 Finland ry 2018b).

HL7 ei yleensä pyri määrittelemään komponenteille ja osakomponenteille pituuksia (HL7 Finland ry 2018b).

2.3.3 Erotinmerkit

Tietokentillä on HL7-viestissä useita erotinmerkkejä, jotka ovat HL7-standardin ainoita kenttiä, joiden täytyy olla tietyssä kohtaa viestiä. Niissä tieto löydetään laskemalla merkkejä viestin alusta, kun taas normaalissa tilanteessa tieto sijaitsee heti tietoryhmän tunnisteen MSH (Message Header) jälkeen positiossa 4. Positiossa 4 oleva merkki toimii samalla viestin kenttäerottimen määrittelynä ja kenttäerottimena tietoryhmän tunnisteen ja erotinmerkin tietokentän välillä (HL7 Finland ry 2018c). HL7-viestissä viestin eri osat,

tietoryhmät, tietokentät, komponentit ja osakomponentit on erotettu toisistaan erotinmerkeillä. Erotinmerkit kerrotaan HL7-viestin aloittavassa MSH-tietoryhmässä (HL7 Finland ry 2018b). Alla olevassa taulukossa on esiteltynä HL7-viestin erotinmerkit, erotinmerkkien ASCII-koodit, sekä erotinmerkkien sijainti tietoryhmä tunnisteessa (MSH).

Taulukko 1. HL7-viestin erotinmerkit (HL7 Finland ry 2018b).

Erotinmerkin nimi	erotin-merkki	ASCII-koodi	sijainti
Komponentin erotin (component separator)	^	94	MSH-2 positio 1
toistoerotin (repetition separator)	~	126	MSH-2 positio 2
vapautusmerkki (escape character)	\	92	MSH-2 positio 3
osakomponentin erotin (subcomponent separator)	&	38	MSH-2 positio 4
tietokenttäerotin (field separator)	|	124	MSH-1 positio 1

Oletusarvoisista erotinmerkeistä poikkeamista tulisi välttää. Toistoerottimella erotetaan toisistaan toistuvat tietokentät. Tietoryhmätasolla toistuma havaitaan tietoryhmän esiintymisestä useamman kerran, erotinmerkeistä sitä ei pysty havaitsemaan. Useimmissa tapauksissa, kun tietoryhmä toistuu HL7-viestissä useamman kerran, tietoryhmän ensimmäinen kenttä on varattu toistuman järjestysnumerolle. Esimerkiksi OBX|1|, OBX|2|.

Suomen oloissa tietokenttäerotin ”|” aiheuttaa hankaluuksia, koska tietokenttäerotin on 7-bittisessä ASCII-merkistössä sama kuin pieni ”ö”. Jos viestissä on siis pieni ”ö” -kirjain, viesti menee siitä kohtaan poikki ja kaikki loput kentät menevät sekaisin, koska tietoryhmään syntyy uusi tietokenttä. Ongelma saadaan kuitenkin ratkaisua käyttämällä varsinaisessa tiedossa 8-bittistä Latin-1 merkistöä (ISO 8859-1). Suomen HL7 yhdistys suosittaakin, että ”Käytetään varsinaisen datan siirtoon 8-bittistä Latin- 1 merkistöä (ISO 8859-1)”. Tällöin täytyy kuitenkin huomata, että koska standardin mukaisesti oletusmerkistä on ASCII, pitää merkistö 8859- ilmoittaa erikseen tietokentässä MSH-18 merkistö koodilla 8859/1 (HL7 Finland ry 2018b).

2.4 HL7 V3

HL7 Versio 3 edustaa uutta lähestymistapaa kliiniseen tiedonvaihdokseen. Se perustuu mallikäyttöön perustuvaan menetelmään, joka tuottaa HL7-sanomia ja sähköisiä asiakirjoja esittämällä ne XML-syntaksilla (Kuva 6). (Health Level Seven International 2018e). Versio 3:n ensisijaisena tarkoituksena on tukea kaikkea terveydenhuollon tiedonkulkua, tukea kaikkia laitteita sekä ennen kaikkea tehostaa entisestään Versio 2 standardin prosesseja ja lopputuloksia. Uusia ominaisuuksia, joita Versio 3 -standardi tarjoaa verrattuna Versio 2 -standardiin ovat mm. laajamittainen tuki integraatioille, uudelleenkäytön ja yhteen toimivuuden ratkaiseminen useilla eri toimialueilla. Versio 3 standardia on mahdollista soveltaa muihinkin terveysaloihin mukaan lukien yhteisön lääketiede, epidemiologia, eläinlääketieteet, kliininen genomiikka sekä turvallisuuteen (Health Level Seven International 2018f).

Kuva 6. Esimerkki HL7 Versio 3 -viestistä (Oracle 2011).

Versio 3 -standardi keskittyy semanttiseen yhteen toimivuuteen täsmentämällä, että tiedot esitetään täydellisessä kliinisessä yhteydessä, joka puolestaan varmistaa sen, että HL7-viestien lähettävät ja vastaanottavat järjestelmät jakavat vaihdettavan tiedon merkityksen (semantiikan). Standardi on suunniteltu siten, että sillä voi olla mahdollisimman

laajoja globaalisia vaikutuksia, mutta ne ovat myös mukautettavissa paikallisten ja alueellisten vaatimusten mukaisiksi. Standardi myöskin sallii toteuttajien hyödyntää milloin tahansa uusimpia ja tehokkaimpia käyttöönottotekniikoita. Standardi varmistaa johdonmukaisen kehityksen ja kyvyn tallentaa ja muuttaa määriteltyjä tietoja tietovarastoihin (Health Level Seven International 2018e).

HL7 Versio 3 -standardi tyypillisesti otetaan käyttöön sovellusalueilla, joissa se ei ole ollut ollenkaan laajassa käytössä. Käyttöönotto on kuitenkin osoittautunut hitaaksi, koska HL7 Versio 3 -sanomanvälityksen implementointi on koettu vaikeaksi ja se vaatii monien uusien asioiden opiskelua (HL7 Finland ry 2018d).

Suomessa Versio 3:n käyttö on aloitettu ajanvarauksista. Kansallistetut ajanvarausmääritykset hyväksyttiin kesällä 2007. Samoin on hyväksytty Versio 3:n kuolinilmoitussanoma. Laajamittaiseen käyttöön Versio 3 -standardi tuli E-reseptien käyttöönoton myötä vuonna 2008 (HL7 Finland ry 2018d).

2.5 RIM

Tietomalli RIM (Reference Information model) on HL7 Versio 3:n -standardin kehitysprosessin kulmakivi. Versiossa 3 luodun esimallin mukaan RIM on kliinisten tietojen, kuten verkkotunnusten kuvauksellinen esitystapa, joka tunnistaa niiden tapahtumien elinkaaren, jotka viestissä tai niihin liittyvissä viesteissä on. RIM on yhteinen malli kaikkien verkkotunnusten kesken ja on itsessään malli, josta kaikki verkkotunnukset luovat viestinsä. Yksinkertaistettuna RIM edustaa HL7-viestien tietojen mukana olevien yhteyksien välisiä yhteyksiä ja RIM on pääasiallinen edellytys jatkuvalle kehitykselle, tarkkuuden lisäämiselle sekä toteutuskustannuksien pienentämiseksi (Health Level Seven International 2018f).

3 MIRTH -INTEGRAATIOALUSTA

Mirth -integraatiotyökalu tarjoaa suurimmille ja arvostetuimmille terveydenhuollon organisaatioille ratkaisuja tehostamaan hoidonhallinnon prosesseja sekä vastaamaan säänteleväin ja kilpailukykyisen terveydenhuollon alan vaatimuksia. Mirth ratkaisu on ilmainen ja joustava ohjelmisto, jotta terveydenhuollon alan yritykset pystyvät helposti ja kustannustehokkaasti hallinnoimaan pienistä potilastiedoista aina suuriin potilastietoihin. Mirth on avoimeen lähdekoodiin perustuva integraatioalusta, jonka omistaa yhdysvaltalainen terveysteknologian ohjelmistoalan yritys NextGen Healthcare. (NextGen Healthcare 2018).

Usein potilastietoja välitetään eri terveydenhuollon tietojärjestelmien välillä. Lääkäri lähettää potilaan tutkimustietoja sairaalaan, terveyskeskus lähettää reseptipyynnön apteekille. Järjestelmien välinen kommunikaatio ja tiedonvaihto eivät kuitenkaan ole täysin varmaa. Tietojen siirtämisessä on viiveitä, tietoja voi jopa kadota ja tietojärjestelmien välinen tietoturva ei aina ole vakuuttavaa, joka tekee liiketoiminnasta vähemmän tehokasta ja liiketoiminta ei ole niin luotettavalla tasolla, kuin sen olisi mahdollista olla. Tekijät, jotka vaikuttavat tietojärjestelmien väliseen kommunikointiin ja tiedonvaihtoon (NextGen Healthcare 2018).

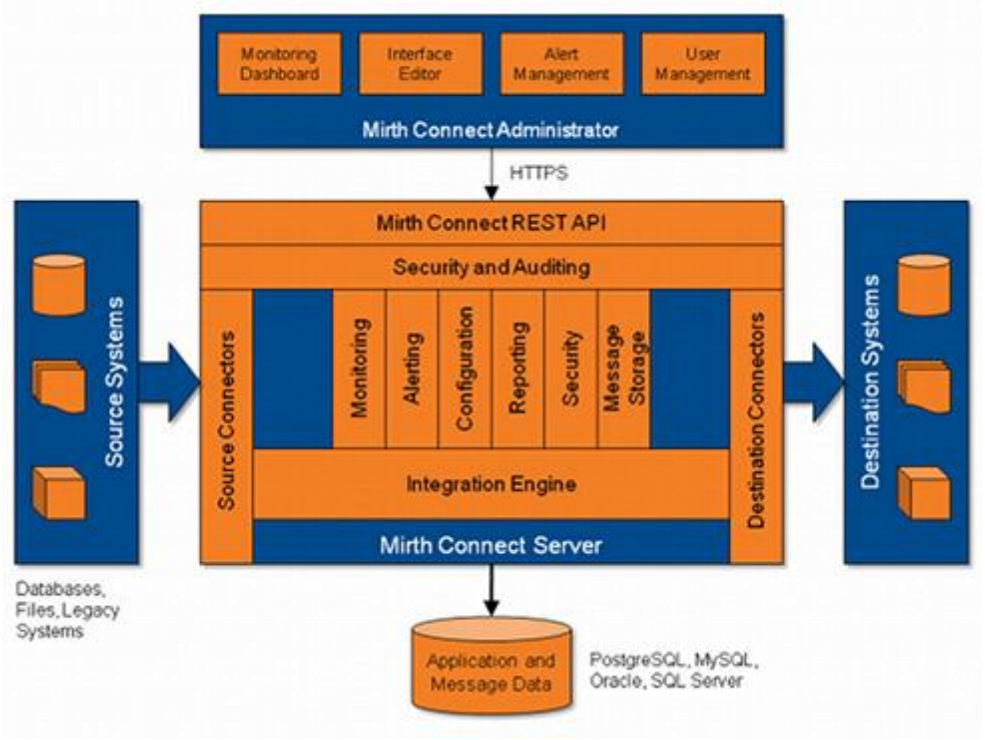
- Tietojärjestelmät käyttävät eri protokollia, jolloin näiden kesken syntyy ristiriita.
- Potilaantietojen siirtämiseen käytettävien ohjelmistojen yhteen sopimattomuus.
- Ohjelmistojen lisenssit ovat erittäin kalliita.
- Potilastietojen tietoturva on usein puutteellista.
- Yhteen sopimattomat tiedot, jotka johtuvat monimuotoisista ohjelmistoista ja ohjelmistojen välisestä kommunikaation menetelmästä.
- Ohjelmistojen käyttö ei ole hallittua ja joustavaa.

Mirth on joustava ratkaisu näihin ongelmiin. Mirth -ohjelmistolla pystytään hoitamaan useita eri tiedonsiirtoja paikasta toiseen tehokkaasti. Esimerkiksi, laboratorio lähettää HL7-viestin Mirth -ohjelmistolle käyttämällä MLLP protokollaa. Mirth siirtää tiedot EHR -tietokantaan, luo PDF-tiedoston ja lähettää viestin vastaanottajalle sähköpostiviestin, johon on automaattisesti liitetty luotu PDF-tiedosto. Mirth pystyy myöskin lukemaan potilastietoja EHR:stä käyttämällä hyödyksi standardoitua SQL kyselykieltä. Kaikki elementit, jotka Mirth hakee EHR:stä siirretään omaan kanavaan, Mirth luo HL7-viestin ja lähettää viestin MLLP:llä lääkärille, joka saa tarvittavat potilastiedot. Viesti voidaan lähettää

useammalle eri henkilölle, luoda PDF-tiedosto tai lähettää tiedot lääkärin sähköpostiin riippuen siitä, miten kanavan asetukset on luotu (NextGen Healthcare 2018).

Mirth -ohjelmistolla pystytään välittämään HL7-standardien mukaisia sanomia, suodattamaan sanomia tai muuttamaan sanomia käyttämällä hyödyksi erilaisia tiedonsiirtoprotokollia. Mirth -ohjelmistoon on saatavilla useita erilaisia maksullisia laajennoksia, joilla pystytään entistä tehokkaammin hallinnoimaan sanomien suoraviivaisia yhteyksiä, turvaamaan viestintäyhteydet ja sanomat. Ulkopuolisen järjestelmän lähettäessä viestin Mirth -ohjelmistolle Mirth kääntää viestistandardit muotoon, jota järjestelmät ymmärtävät. Viestin suodattamisessa Mirth lukee viestin parametreja ja siirtää viestin eteenpäin tai pysäyttää viestin etenemisen, kun viesti on menossa siirtymävaiheeseen. Viestin muutoksessa, Mirth muuntaa tulevan viestistandardin toiseen standardiin. Esimerkiksi HL7-standardin mukainen viestin muunnetaan XML-standardin mukaiseksi tulevaksi viestiksi. Mirth pystyy myös ottamaan tietoja tai laittamaan tietoja tietokantoihin. Viestien reitityksessä Mirth varmistaa, että lähetetyt viestin saapuvat juuri niihin kohteisiin, joihin lähettäjä on viestit osoittanut. (NextGen Healthcare 2018).

Mirth -ohjelmiston arkkitehtuuri koostuu neljästä eri komponentista (Kuva 7). Mirth Connect Server, Mirth Connect Administrator, Mirth Connect Server Manager ja Mirth Connect Command Line Interface. Mirth Connect Server on tärkein komponentti, joka pitää sisällään- loppupään hallintorajapinnan, integrointimoottorin komponentit, joiden tehtävänä on suorittaa viestin suodattamisen, muutoksen ja viestin lähettämisen vastaanottajalle. Mirth Connect Administrator on graafinen käyttöliittymä, jonka tehtävänä on yhdistää käyttäjä Mirth Connect Serveriin ja sallia käyttäjän konfiguroida rajapintaa, valvoa käyttöliittymän aktiivisuutta ja jolla voidaan selata viestejä viestien tallennuspaikasta. Mirth Connect Server Manager on toinen graafinen käyttöliittymä, joka sisältyy Mirth Connectin arkkitehtuuriin. Sen tehtävänä on hallita Mirth Connect palvelua, näyttää käyttäjälle lokitiedostoja, joita Mirth Connect automaattisesti luo. Mirth Connect palvelussa on myös muita asetuksia, joilla käyttäjä pystyy konfiguroimaan Mirth Connect Serveriä. Mirth Connect Command Line Interface (CLI) on komentokehotetyökalu, joka sallii yhdistämisen Mirth Connect Serveriin, jonka jälkeen käyttäjä voi hallita kanavia kuten, lähettää, tuoda tai viedä tietoja kanavissa eteenpäin. Käyttäjä voi suorittaa myös muita hallinnollisia tehtäviä, joita kanavien hallinta mahdollisesti vaatii (Brauer 2011).



Kuva 7. Mirth Connect -arkkitehtuuri (Richard & Christopher 2015).

Mirth Connect on lisensoitu Mozilla Public License (MPL) 1.1 alle. Koska Mirth -ohjelmisto perustuu avoimeen lähdekoodiin, asiakkaiden saama etu on siinä, että suuri joukko alan ammattilaisia on testannut ja myötävaikuttanut Mirth -ohjelmiston kehitykseen. Monet virheet, joita Mirth -ohjelmisto pitää sisällään, korjataan erittäin nopeasti ja yhteisön antama palaute on saanut aikaan sen, että Mirth on yhä enemmän hyödyllinen ja käyttäjäystävällinen ohjelmisto terveydenhuolto alalla. Mirth -ohjelmiston käyttöön on mahdollista saada ilmaista verkkoneuvontaa.

Mirth Connect tukee lukuisia eri rajapintastandardeja, jolloin pystytään välittämään tietoja eri sosiaali- ja terveydenhuolto järjestelmien käyttämien standardien välillä. Rajapintastandardit, joita Mirth Connectin tukee ovat

- HL7 v2.x
- HL7 v3.x
- Delimited Text (CSV, tab-delimited, fixed-width)
- DICOM (Digital imaging and Comminications in Medicine)
- EDI / X12 (Electronic data interchange)
- XML (Extensible Markup Language)
- JSON (JavaScript Object Notation)

- NCPDP (National Council for Prescription Drug Programs)
- Raw (Kaikki tietoformaatit)
- ASTM E1392 (Kaupallinen laajennos)

Mirth -ohjelmisto tukee erittäin kattavasti erilaisia tiedonsiirtoprotokollia, joiden avulla pystytään luomaan yhteydet laitteiden ja ohjelmistojen välille. Tiedonsiirtoprotokollat, joita Mirth Connect tukee ovat:

- MLLP (v1 ja v2) (Minimum Layer Protocol)
- TCP/IP (Transmission Control Protocol / Internet Protocol)
- HTTP (Hypertext Transfer Protocol)
- Flat Files Databases
- SQL (Structured Query Language)
- SFTP (Secure File Transfer Protocol)
- FTP (File Transfer Protocol)
- SMB (Server Message Block)
- WebDAV (Web Distributed Authoring and Versioning)
- IMAP / POP3 (Internet Message Access Protocol / Post Ofiice Protocol , käytetään Sähköpostiviestien välitykseen)
- SMTP (Simple Mail Transfer Protocol, käytetään Sähköpostiviestien välitykseen)
- DICOM (Digital Imaging and Communications in Medicine)
- JMS (Java Message Service)
- SOAP Web Services (Simple Object Access Protocol)
- PDF/RTF Documents (Portable Document Format / Rich Text Format)
- Custom Java and JavaScript
- ASTM E1381 (Standard Specification for Low-Level protocol to Transfer Messages Between Clinical Laboratory Instruments and Computer Systems) (ASTM International 2018).
- Serial / RS-232 (Serial / Recommended Standard 232)

3.1 E4X Standardi

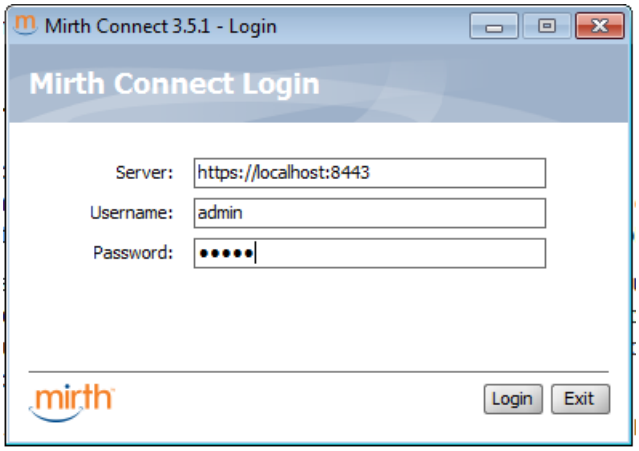
Mirth Connect käyttää XML-viestien muodostamiseen hyödyksi JavaScript laajennosta ECMAScript (E4X). E4X on standardoinut Ecma International, standardissa ECAM-357, (ISO/IEC 22537:2006). Viimeisin versio, joka standardista on voimassa, on toinen painos, joulukuulta 2005. E4X lisää alkuperäisiä XML-tietotyyppejä ECMASripti skriptikieleen (komentokieleen), laajentaa tuttujen ECMASripti-operaattorien semantiikkaa manipuloimalla XML-tietoja ja lisää pienen joukon uusia operaattoreita yleisiin XML-operaatioihin, kuten hakuun ja suodatukseen. E4X myös tukee XML-kirjastoja, nimiavaruuksia ja muita mekanismeja XML-tietojen käsittelyn helpottamiseksi (Ecma International 2018).

Standardi lisää XML-tuen JavaScript-ohjelmointikielelle. E4X on suunniteltu yksinkertaistamaan JavaScriptin-ohjelmakoodin kirjoittamista XML:lle. Se on hyödyllinen ja tehokas työkalu, jota voidaan käyttää XML-solmuja ja attribuuttien välisessä vuorovaikutuksessa. E4X:n ensisijaisena tavoitteena on antaa JavaScript kehittäjille selkeä ja tehokas tapa manipuloida XML-asiakirjaa käyttämättä hyödyksi DOM rajapintoja. JavaScript-ohjelmointikieli käyttää E4X -standardia antamalla uusia ominaisuuksia globaaleille olioille. XML-objektilla puolestaan on useita ominaisuuksia, jotka ovat käyttökelpoisia E4X-sarjalistauksessa ja jäsentelyfunktioissa. E4X kierrättää useita olemassa olevia JavaScript-operaattoreita ja käyttää näitä hyödyksi XML:n luomiseen, manipulointiin ja navigointiin. E4X:ää käyttämällä kehitystyöhön käytettävä aika vähenee ja E4X:n käyttö on nopea oppia (Grace 2008).

Mirth Connect käyttää E4X:n lisäksi avoimeen lähdekoodiin perustuvaa Rhino JavaScript-laajennosta, joka on yleensä sulautettu osaksi JavaScript-sovelluksiin tarjoamaan ohjelmakoodit loppukäyttäjille (SphinxKnight.) E4X-standardi on vanhentunut, joten se on vaiheittain poistettu käytöstä selaimista kuten Firefoxista ja Google Chromesta (jmnater). Mirth Connect kuitenkin edelleen käyttää tätä ratkaisua hyödyksi XML-tietojen muodostamisessa. Jos Mirth -ohjelmistoon halutaan nykyaikaisempi ja kestävä ratkaisu, standardoitu ratkaisu on erittäin vaikeaa muuttaa. E4X standardi on erittäin syvällä Mirth Connectin arkkitehtuuria, joten standardin vaihtaminen toiseen kestävään ratkaisuun on erittäin haasteellista, koska koko XML-tietojen muodostaminen perustuu tähän standardiin. Internetkirjoituksissa onkin pohdittu sitä, että tulevaisuudessa, jos Mirth Connect yhä käyttää E4X -standardia hyödyksi, tämä voi olla jopa vaarantaa koko Mirth -ohjelmiston tulevaisuuden (Vjeux 2013).

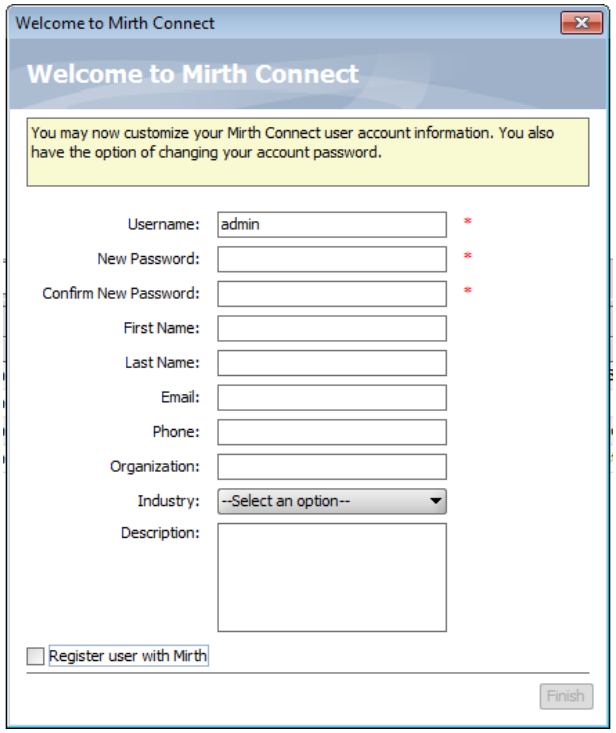
3.2 Mirth -ohjelmiston asentaminen

Mirth Connect (versio 3.5.1) asennettiin tietokoneelle, joka täyttää Mirth -ohjelmiston laitteistovaatimukset. Versio, joka asennettiin, oli julkaistu vain kolme päivää aikaisemmin ja tämä versio piti sisällään 15 eri virheiden korjausta. Ohjelmisto ladattiin Mirthin omilta kotisivuilta latausosiosta. Asennuksen aikana määriteltiin kaikki tarvittavat ominaisuudet, joita tarvitaan Mirthin tehokkaaseen käyttämiseen. Määriteltiin salasanavaatimukset, jonka pitää olla vähintään kahdeksan merkkiä pitkä-, ja sen täytyy pitää sisällään vähintään kaksi- isoa ja pientä kirjainta, numeroa sekä erikoismerkkiä. Ohjelmiston asentaminen suoritettiin muuten, kuin salasanaosio osalta oletusasetuksin, koska tarvetta asetuksien muuttamiseen erilaiseksi ei ollut. Kun asennus oli saatu suoritettua loppuun, ohjelma kysyy käyttäjätunnusta ja salasanaa, jotka ovat oletuksena ”admin**”** (Kuva 8).



Kuva 8. Mirth Connect Login.

Tämän jälkeen ohjelma pyytää käyttäjää vaihtamaan käyttäjätilin oletus käyttäjätunnuksen ja salasanan (Kuva 9). joiden tulee täyttää salasana vaatimukset, jotka käyttäjä ohjelmiston asentamisen aikana on määritellyt. Käyttäjätilin tietojen muokkaamisessa voi myöskin antaa enemmän tietoja, kuin pelkän käyttäjätunnuksen ja salasanan.



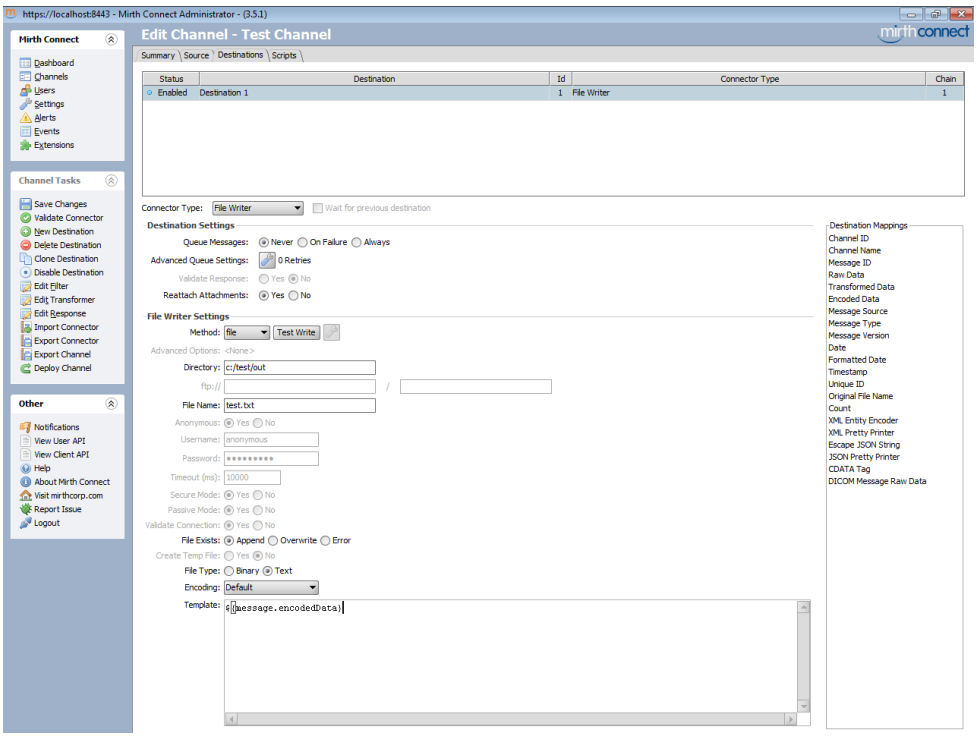
Kuva 9. Mirth Connect user account information.

Näiden asetuksien muuttamisen jälkeen käyttäjä pääsee kirjautumaan Mirth Connect Administraatioon, käyttämällä Mirth Connect palvelua. Mirth Connect Administraation käyttöliittymään on myöskin mahdollista kirjautua nettiselaimen avulla kirjoittamalla selaimen osoitekenttään osoite: http://localhost:8080.

3.3 Mirth testaaminen

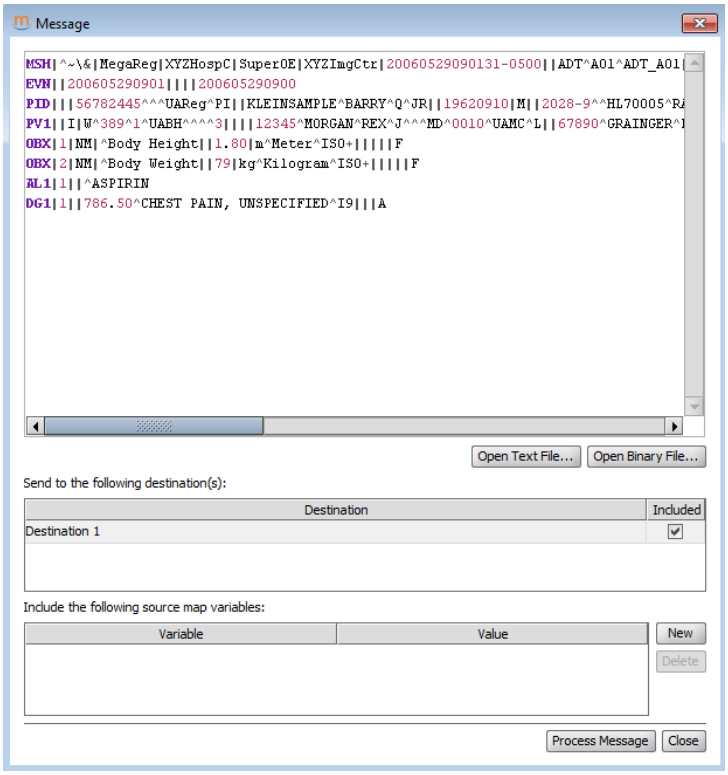
Mirth -ohjelmiston testaaminen suoritettiin katsomalla Mirthin kotisivuilta olevalta opetusvideolta, yksinkertainen esimerkki siitä, miten HL7-viesti lähetetään kanavalta ja miten lähetetty HL7-viesti tallennetaan tekstitiedostoon paikallisesti. Tämän jälkeen testattiin Mirth -ohjelmistoa videon ohjeiden mukaisesti. Ensimmäiseksi täytyi luoda kanava, joka nimettiin ”Test channel”. Tämän jälkeen valittiin lähde välilehdestä TCP Listener yhdistämiseen, jota käytetään viestin välittämiseen. Lähde tarkoittaa, mistä tiedot luetaan, jotka lähetetään eteenpäin. Seuraavaksi valittiin ”Kohde” välilehdestä kohteen, joka puolestaan tarkoittaa sitä, että mihin tiedot lähetetään ja miten ne lähetetään. Yhdistämiseen valittiin ”File Writer”, koska lähetetty HL7-viesti haluttiin tallentaa tekstitiedostoon. Tämän jälkeen valittiin tiedostopolku,

johon lähetetty HL7-viesti lähetetään ja annettiin tekstitiedoston nimeksi test.txt, johon viesti kirjoitetaan. Lopuksi siirrettiin oikealta olevasta ”Destination Mappings” valikosta ”Encoded Data” alla olevaan ”Template” laatikkoon (Kuva 10).



Kuva 10. Kanavan asetukset.

Näiden asetuksien jälkeen testaukseen tarvittavat kanavan asetukset olivat kunnossa, lopuksi kanavan tallennettiin ja valittiin hiiren oikealla kanavan päältä valinta ”Deploy Channel”, joka ottaa luodun kanavan käyttöön. Seuraavaksi valittiin ”Dashboard” valikosta luotu kanava, jonka ilmestyi sinne, kun kanava otettiin käyttöön. Hiiren oikealla valittiin kohta ”Send Message”, johon kopioitiin esimerkki HL7-viesti ja lopuksi lähetettiin viesti painamalla ”Process Message” (Kuva 11).



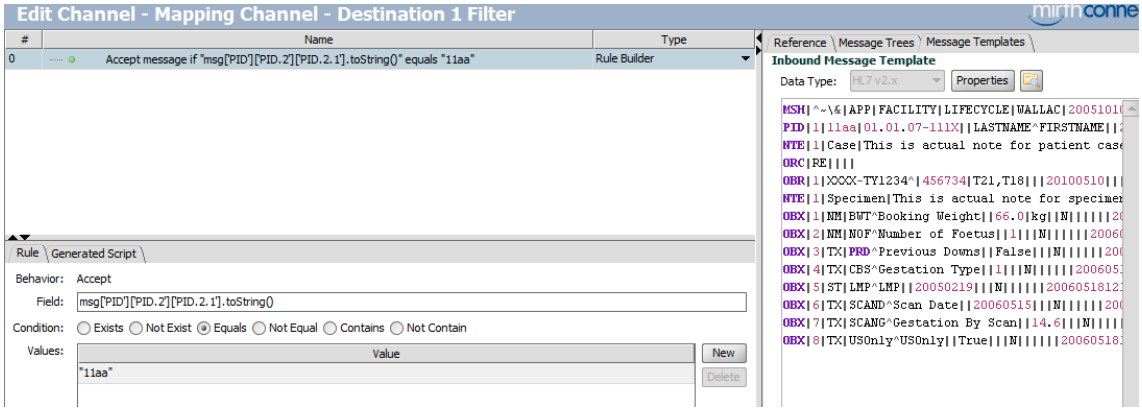
Kuva 11. HL7-viestin lähettäminen kanavaan.

Mirth Connect oli onnistuneesti lähettänyt ja kirjoittanut esimerkki HL7-viestin tiedostopolun tekstitiedostoon**,** joka ei näyttänyt samalta, kuin lähetetty HL7-viesti. Viesti oli tekstitiedostossa kirjoitettu yhteen. Tämä johtui siitä, koska tekstitiedosto avattiin muistio-ohjelmalla, joka ei laittanut rivinvaihtoja segmenttien loppuun. Kun tekstitiedosto avattiin WordPad-tekstinkäsittely ohjelmalla, lähetetty HL7-viesti oli ohjelmassa täysin samassa muodossa, kuin lähetetty viesti.

4 MIRTH-SOVELTAMINEN

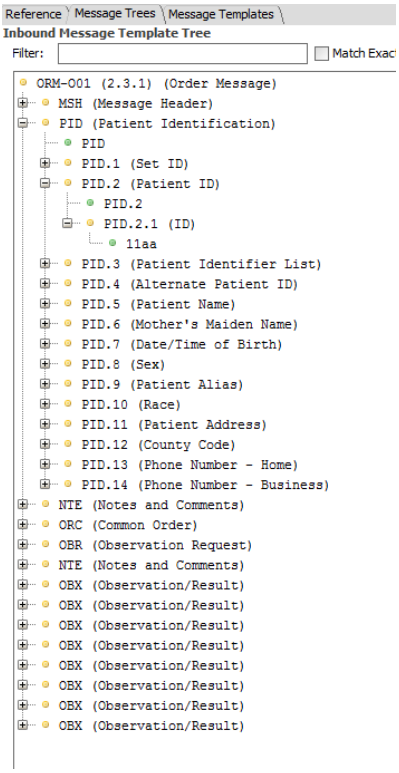
Mirth -integraatioalustaa testattiin yhtenä osana laajempaa kokonaisuutta. Mirth -integraatioalustasta selvitettiin ennen kaikkea sitä, miten Mirth Connect soveltuu HL7-sanomien parsintaa. Tämän lisäksi selvitettiin HL7 V2.3 -sanomien muodostamista Mirth -integraatioalustalla. Selvitys suoritettiin paikallisesti tietokoneella, koska kaikkia tarvittavia vaatimuksia ei ole vielä olemassa ja niitä ei ole täysin määritelty. Näin ollen Mirth Connectia ei pystytty käyttämään osana koko laajempaa kokonaisuutta. Tulevaisuudessa kaikki tarvittavat rajapinnat ja asetustiedostot tulevat internetpalvelimelta, mutta koska tätä internetpalvelin ratkaisua ei myöskään ole vielä olemassa. Tarvittava asetustiedoston sisältö on määritelty Mirthissä. Kaikki tarvittavat ohjelmat ohjelmointiin JavaScript-ohjelmointikielellä. Ohjelmointivaiheessa turvauduttiin myöskin Visual Studio 2015 -ohjelmistoon**,** NodeJSään sekä käytettiin Rhino JavaScript-laajennosta, koska ohjelmointi Mirth -integraatioalustassa oli haastavaa puutuvien ominaisuuksien johdosta.

Työssä käytettiin toimeksiantajan antamaa esimerkki HL7-sanomaa, sekä määriteltyä asetustiedostoa, josta ilmeni, mitä arvoja Mirth Connectissa pitää parsia HL7-sanomasta. Ensimmäiseksi työssä asetettiin kaikki tarvittavat kanavien asetukset mukaan lukien, mistä hakemistosta HL7-sanoma luetaan sekä mihin hakemistoon parsitut tiedot tallennetaan. Kaikkien tarvittavien asetuksien jälkeen pyrittiin siihen, että Mirth Connect kirjoittaa tekstitiedostoon HL7-sanomassa olevan segmentin PID.2.1 arvon. Tätä varten jouduttiin vielä lisäämään esimerkki HL7-sanoma asetuksiin ”Message Template” –kohtaan (Kuva 12).



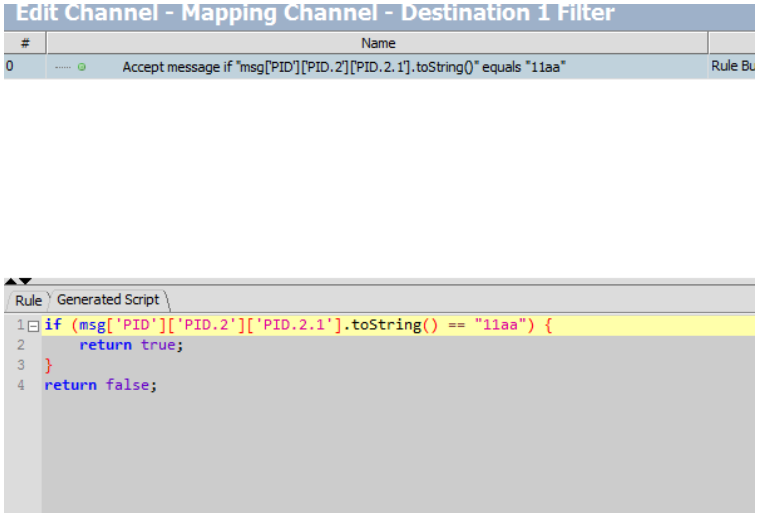
Kuva 12. Message Template valikko.

Esimerkki HL7-sanomat asettamisen jälkeen, Mirth Connect loi automaattisesti esimerkki HL7-sanomasta viestipuun (Kuva 13), josta pystyttiin, näkemään helposti mikä arvo on missäkin segmentissä.



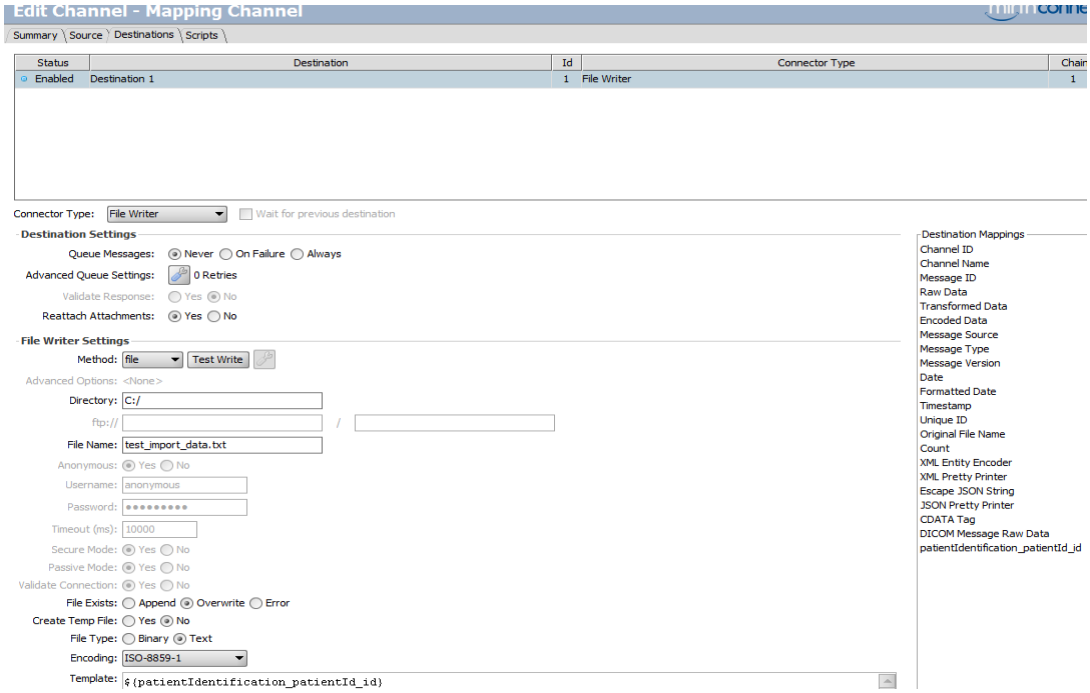
Kuva 13. HL7-esimerkkisanomasta muodostuva viestipuurakenne.

Tämän jälkeen siirrettiin PID.2.1 (ID) -kohdassa olevan 11aa -arvo Kuvassa ”message template” olevaan ”Field” -kohtaan. Mirth -integraatioalusta loi itse JavaScript-ohjelman, joka katsoo, että onko PID2.1 arvo sama kuin 11aa. Jos tämä ehto on tosi, ohjelma palauttaa ”true”, jos taas PID2.1 arvo on jokin muu kuin 11aa, ohjelma palauttaa ”false” (Kuva 14).



Kuva 14. Mirth Connectin luoma JavaScript-ohjelma.

Tämän jälkeen Mirth Connect kysyy vielä käyttäjältä, että luodaanko uusi tai jo olemassa oleva iteraatio kyseiselle ohjelmalle. Tätä iteraatio vaihetta ei tarvittu missään vaiheessa opinnäytetyötä. Lopuksi vielä täytyi mennä takaisin kanavan asetusnäkymään, jolloin ”Destination Mappings” –valikkoon oli ilmestynyt ”patientIdentification_patientId_Id”. Tämä siirretiin alla olevaan ”Template” kohtaan, jolloin arvo saatiin lähetettyä kanavalle (Kuva 15). Lopuksi kaikki muutokset tallennettiin ja kanava otettiin käyttöön.



Kuva 15. Kanavan asetusnäkymä.

Kaiken tämän jälkeen Mirth -integraatioalusta kirjoitti onnistuneesti PID.2.1 olevan ”11aa” arvon hakemistoon C:/test_import_data.txt. Mirth kuitenkin lähetti tämän joka viiden sekunnin välein, koska asetuksissa oli ”File Exists” asetus ”Append”. Tämä ongelma korjaantui kuitenkin helposti vaihtamalla asetus kohtaan ”Overwrite”, jolloin Mirth Connect päällekirjoittaa aikaisemman tiedoston. Yksittäisiä arvoja Mirth -integraatioalustasta saadaan varsin helposti kirjoitettua tekstitiedostoon, jos segmentit, joista arvot halutaan pysyvät aina samana. Jos segmentit muuttuvat, käyttäjän tulee itse ohjelmoida ohjelma, jolla pystytään automaattisesti parsimaan juuri oikeat arvot.

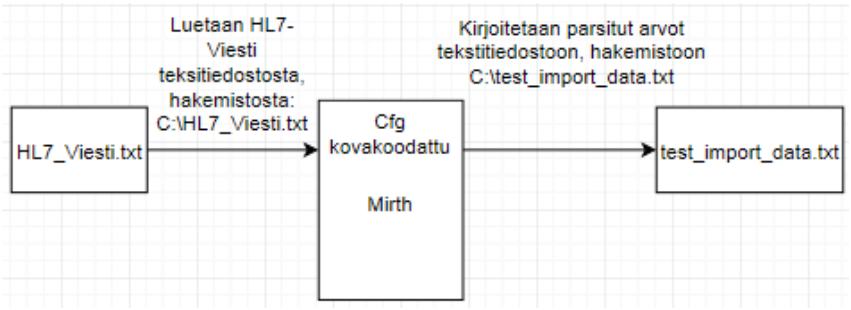
4.1 Ohjelmointi JavaScriptillä

Mirth -integraatioalustassa kanaville voidaan tehdä ohjelmoitnia käyttämällä JavaScript-ohjelmointikieltä. Ohjelmoinnin tarkoituksena on usein se, että tuleville ja lähteville HL7-viesteille tarvitaan muunnoksia, viestien tiedot tallennetaan tietokantaan tai tiedot lähetetään sähköpostilla vastaanottajalle. Yleisimmät ohjelmoinnit, joita joudutaan tekemään käyttämällä JavaScriptiä liittyvät HL7-viesteissä segmentteihin, kenttiin ja tietojen talteen ottamiseen viestistä. Ohjelmoinnissa voidaan käyttää jo olemassa olevia JavaScript-luokkia tai jopa itse luotuja JavaScript-kirjastoja ja luokkia. Ohjelmaan voidaan lisätä myös ulkoisia Java tai JavaScript-kirjastoja, jolloin käyttäjä voi yksinkertaisesti sisällyttää nämä ulkoiset kirjastot Mirth -integraatioalustaan. Näin ollen käyttäjän ohjelmointi Mirth -integraatioalustassa vähenee huomattavasti.

Mirth -integraatioalustassa on käytössä JavaScript-editori ominaisuus. Tämän avulla voidaan helposti esimerkiksi muuttaa jokin muuttuja toiseksi nimeksi, jolloin editori muuttaa muuttujan nimen kaikkialla ohjelmassa. Editorin yksi tarkoitus on minimoida käyttäjän turha edestakainen hyppiminen JavaScript-editorin ja käyttäjärajapinnan välillä. Editorin sivulle tulee kaikki ohjelman muuttujat, luokat, funktiot, jolloin käyttäjä näkee ne kaikki yhdestä näkymästä.

4.2 Tulevan HL7-viestin parsinta

Tulevan HL7-sanoman parsinta aloitettiin määrittelemällä asetustiedosto, joka kertoo sen, mitä tietoja HL7-sanomasta halutaan parsia. Asetustiedostossa on määritelty segmentit, kentät sekä niiden järjestys. Parsimisen ideana oli yksinkertaisesti käydä For-toistorakenteella kaikki kentät läpi ja parsia ne määriteltyihin muuttujiin. Esimerkiksi PID.2.1 parsinnassa, segmenttimuuttujaan tuli arvoksi PID, kenttämuuttujaan 2 sekä alikenttämuuttujaan 1. Ohjelmassa määriteltiin myöskin ”fieldValue” muuttuja, johon otettiin jokaisella toistolausekkeen kierroksella HL7-esimerkkisanomasta arvo käyttämällä Mirth Connectin omaa ”msg” -objektia. Tämän jälkeen määriteltiin vielä muuttuja ”oneRow”, joka muuttui jokaisella toistorakenteen kierroksella siten, että muuttuja lisäsi uuden arvon entisen perään ilman, että aikaisemmat arvot päälle kirjoittuvat. Lopuksi vielä luotiin tiedosto ja kirjoitettiin oneRow -muuttujan arvo sinne. Koko prosessi, miten tulevan HL7-sanoman parsinta on toteutettu, on kuvattu alla olevassa kuvassa.



Kuva 16. Tulevan HL7-viestin parsinnan toteutus.

Tulevan HL7-viestin perustietojen parsinta osoittautui helpoksi. Ongelmia tuotti asetustiedoston lukeminen hakemistosta. Sopivia esimerkkejä oli vaikeaa löytää Mirthin kehitysfoorumeilta. Kokeilujen jälkeen asetustiedoston lukeminen onnistui, mutta tiedoston sisältämien tietojen lisääminen tuotti ongelmia, joten päädyttiin määrittelemään asetustiedoston arvot Mirthissä. Tämä ratkaisu oli kaikkein yksinkertaisin tapa, mutta tulevaisuudessa asetustiedostot tulee pystyä lukemaan ulkoisesta hakemistosta.

4.3 OBX-segmenttien parsinta tulevassa viestissä

Toinen vaihe, mikä parsinnassa tuli tehdä, oli OBX-segmenttien parsinta. Näiden segmenttien parsinta osoittautui huomattavasti haasteellisemmaksi kuin perustietojen parsinta, koska OBX-segmentit piti pilkkoa vielä pienempiin osiin. OBX-segmenttien parsinnan aikana jouduttiin käyttämään hyödyksi Visual Studio 2015 -ohjelmistoa, sekä siihen asennettavaa NodeJs -laajennosta, jotta pystyttiin paremmin ohjelmoimaan JavaScript-ohjelmointikielellä. Visual Studio 2015 on huomattavasti kehittyneempi ohjelmisto ohjelmointiin, joten se sisältää lukuisia hyödyllisiä tekniikoita ja ominaisuuksia, jotka helpottivat ohjelmointia huomattavasti.

OBX-segmenttien parsinnassa pystyttiin hyödyntämään perustietojen parsinta JavaScript-koodia. Hyödyllisyys tuli siinä heti alussa esille, koska jouduttiin myöskin OBX -segmenttien parsinnassa käymään asetustiedostossa olevat tiedot läpi toistorakenteella, mutta tätä koodia jouduttiin kuitenkin muokkaamaan. Koska OBX-segmenttien määrä vaihtelee (Koodi 1), parsintaan jouduttiin lisäämään suodatin, jonka tarkoituksena on se, että jos suodatin muuttuu, niin silloin kyseessä on eri OBX-segmentti.

```
OBX|1|NM|BWT^Booking Weight||66.0|kg||N|||||20060518121611
OBX|2|NM|NOF^Number of Foetus||1|||N|||||20060518121611
OBX|3|TX|PRD^Previous Downs||False|||N|||||20060518121611
OBX|4|TX|CBS^Gestation Type||1|||N|||||20060518121611
OBX|5|ST|LMP^LMP||20050219|||N|||||20060518121611
OBX|6|TX|SCAND^Scan Date||20060515|||N|||||20060518121611
OBX|7|TX|SCANG^Gestation By Scan||14.6|||N|||||20060518121611
OBX|8|TX|USOnly^USOnly||True|||N|||||20060518121611
```

Koodi 1. Esimerkki HL7-sanoman OBX-segmentit.

OBX-segmentti, jota jouduttiin parsimaan, oli määritelty asetustiedostossa seuraavanlaisesti: OBX.3[LMP^LMP].5.1. Parsintaa lähdettiin toteuttamaan siten, että suodatin, jonka avulla pystytään erottamaan, koska OBX-segmentti muuttuu, oli 3[LMP^LMP]. Näin ollen, jos edellä mainittuun esimerkkiin tulee jokin muutos, numero muuttuu tai LMP^LMP muuttuu, tarkoittaisi se silloin sitä, että OBX-segmenttikin on muuttunut. Asetustiedostossa voi olla määritelty kaksi kertaa sama OBX-segmentti. Tällöin se tarkoittaisi sitä, että samasta OBX-segmentistä pitää parsia useita arvoja. Esimerkiksi: OBX.3[LMP^LMP].5.1; OBX.3[LMP^LMP].8.1. Esimerkistä täytyisi ottaa samasta OBX -segmentistä arvot kohdista 5.1 ja 8.1. Yllä olevan Koodin mukaan arvot olisivat tässä tapauksessa: ”20050219” ja ”N”. OBX-segmenttien parsinnassa kaikki segmentit

asetettiin tyhjään taulukkoon. Lopuksi taulukosta haettiin kaikki segmentit, ja toistorakenteella laitettiin arvot oikeisiin segmentteihin. Ensimmäiseen segmenttiin tuli parsittu arvo 1, toiseen segmenttiin parsittu arvo 2 jne. Tämän jälkeen viesti lähetettiin samaa ideaa käyttäen, kuin perustietojen parsinnassa. Mirth Connect parsi onnistuneesti oikeat arvot esimerkki HL7-viestistä asetustiedoston määriteltyjen segmenttien mukaan ja kirjoitti ne onnistuneesti tekstitiedostoon oikeassa järjestyksessä (Koodi 2).

```
11aa;XXXX-TY1234;LASTNAME;FIRSTNAME;STREETADDRESS;20070825;T21,T18;20050219;66.0
```

Koodi 2. Parsitut arvot tekstitiedostossa.

Asetuksien määrittelyssä on yksi rajoitus. OBX-segmentit täytyy olla asetustiedostossa määritelty viimeiseksi. Perustietojen ja OBX-segmenttien parsinta tapahtuu omissa funktioissaan, joten jos asetustiedostossa on määritelty jokin perustieto segmentti esim. PID, ohjelma ei enää parsi tätä viimeistä PID-segmenttiä ollenkaan.

4.4 Lähtevän HL7-viestin parsinta HL7-viestiksi

Lähtevän viestin parsinnassa tarkoituksena oli laittaa parsitut arvot oikeaan kohtaan lähes tyhjään HL7-viesti malliin. Aluksi tyhjennettiin tulevasta HL7-viestistä kenttien arvot, sekä OBX-segmentit, jotta pystyttiin käyttämään samaa HL7-viestiä pohjana (Koodi 3). Näin saatiin HL7-viestin perusrakenne käyttöön, johon pystyttiin lisäämään segmenttejä ja kenttiin arvoja. Tämä tyhjä HL7-viesti täytyi myös lisätä Mirth Connectiin, jotta saatiin käyttöön Mirth Connectin ”tmp”-objekti, jota tarvitaan, kun HL7-viestiin halutaan lisätä tietoa.

```
MSH|^~\&|APP|FACILITY|LIFECYCLE|WALLAC|200510101144||ORM^O01|84|P|2.3.1
PID|1||||^||||||^^^^|||
NTE|1||
ORC|RE||||
OBR|1|^|||||||||||||||||||||||||^
NTE|1||
```

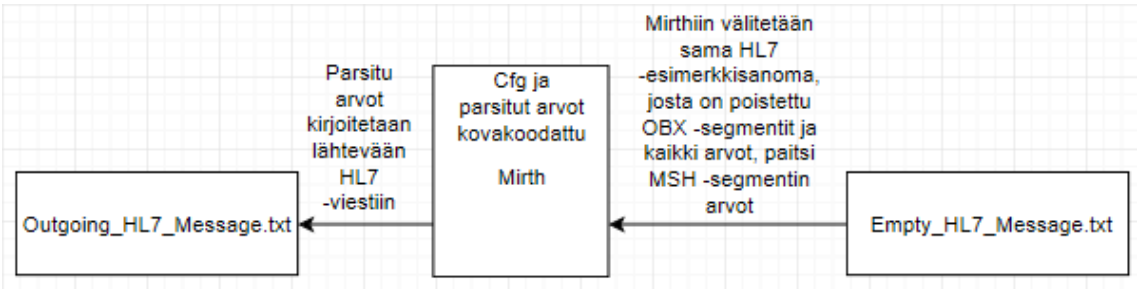
Koodi 3. Tyhjennetty HL7-viesti.

Lähtevän HL7-viestin parsinnassa pystyttiin hyödyntämään tulevan viestin parsinta JavaScript-ohjelmakoodia. Lähtevän viestin parsinta lähti siitä, että jouduttiin määrittelemään parsitut arvot ja asetustiedoston sisältö (Koodi 4).

```
var inputData = '11aa;XXXX-TY1234;LASTNAME;FIRSTNAME;S
var header = 'PID.2.1;OBR.2.1;PID.5.1;PID.5.2;PID.11.1
```

Koodi 4. Osa parsituista arvoista ja osa asetustiedostoa.

Yksinkertaisuuden vuoksi päätetiin, että myös asetustiedoston sisältö määritellään. Parsinnan idea oli lähes sama, kuin tulevan HL7-viestin parsinnassa. Kahdella eri For-toistorakenteella käydään läpi asetustiedoston sisältö ja toisella For-toistorakenteella parsitut arvot. Tämän jälkeen ohjelmassa käytettiin ”tmp”-objektia, jolla pystyttiin lisäämään tietoja HL7-viestiin. Perustiedot saatiin yksinkertaisesti parsittua ja asetettua lähtevään HL7-viestiin siten, että For-toistorakenteessa otettiin asetustiedoston ensimmäinen segmentti (PID.2.1) ja tämän arvoksi laitettiin parsituista arvoista ensimmäinen (11aa). Perustietojen laittaminen HL7-viestiin ja oikeiden arvojen asettaminen oli helppoa, koska molemmissa tiedostoissa (asetus- ja parsittujen arvojen), tiedot olivat oikeassa järjestyksessä. Segmentit PID.2.1 arvo oli 11aa, OBR.2.1 segmentin arvo oli XXXX-TY1234 (Koodi 4). Koko prosessi miten lähtevän viestin parsinta on toteutettu, on kuvattu alla olevassa kuvassa.



Kuva 17. Lähtevän HL7-viestin parsinta prosessi.

Perustietojen parsita (kaikkien muiden segmenttien parsinta, paitsi OBX-segmenttien) onnistui helposti ja nopeasti, koska vain hieman tulevan HL7-viestin parsinta koodia muokkaamalla sain perustiedot parsittua ja lisättyä lähtevään HL7-viestiin.

4.5 OBX-segmenttien parsinta lähtevässä viestissä

Lähtevän viestin parsinnassa OBX-segmentit tuli myös parsia, kuten tulevassakin viestissä. OBX-segmenttien parsinnassa aluksi täytyi selvittää, miten Mirth Connectissa voidaan JavaScript-ohjelmakoodilla luoda segmenttejä ja asettaa luotuihin segmentteihin arvoja. Segmenttien luomiseen on olemassa useita tapoja. Segmenttejä yritettiin luoda

käyttämällä Mirth Connectissa olevaa JavaScript-funktiota: ”GreateSegment”, jolla pystytään luomaan haluttu segmentti ja asettamaan mihin tahansa segmentin kenttään haluttu arvo. Luotu OBX-segmenttiä ja sille asetettua arvoa ei kuitenkaan saatu toimimaan. Luotu OBX-segmentti ei näkynyt lähtevässä HL7-viestissä. Syy miksi OBX-segmentti ei näkynyt johtui todennäköisesti siitä, että Mirth Connect päällekirjoitti tiedostoa aika-ajoin. Luotu OBX-segmentti ei olisi näkynyt kuin hetken. Syy on kuitenkin vain teoreettinen. OBX-segmenttien parsinnassakin pystyttiin hyödyntämään tulevan HL7-sanoman OBX-segmenttien parsintakoodia. Ohjelmaan lisättiin sama suodatin, kuin tulevassa viestissä. Ohjelma loi OBX-segmentin, joka oli määritelty asetustiedostossa: OBX.3[LMP^LMP].5.1, ja ohjelma laittoi parsitun arvon oikeaan kohtaan: 5.1. Tunnistetta: LMP^LMP, ohjelma ei laittanut ollenkaan kohtiin 3.1 ja 3.2. Tämä johtui siitä, että Mirth Connect käyttää XML:ää, Ohjelmoinnin aikana käytettiin Rhino JavaScript-laajennosta, jonka avulla pystyttiin komentoriviltä testaamaan ja näkemään, miltä viesti näyttää XML-muodossa. OBX-segmenttien luonti täytyi tapahtua siten, että ensin ohjelma tarkasti, onko OBX-segmenttejä olemassa. Jos ei ohjelma luo OBX-segmentin ja laittaa kaikki tarvittavat arvot oikeisiin kenttiin. Seuraavalla kierroksella ohjelma katsoo, että onko OBX-segmentin tunniste muuttunut, jos ei, niin ohjelma asettaa arvot edelliseen OBX-segmenttiin. Jos puolestaan tunniste muuttuu, silloin se tarkoittaa sitä, että ohjelman täytyy luoda uusi OBX-segmentti ja asettaa arvot jälleen oikeisiin kenttiin.

Alla olevassa koodissa (Koodi 5), on esitetty asetustiedostossa määritellyt OBX-segmentit. Kuvasta selviää, että asetustiedostossa on määritelty kaksi eri OBX-segmenttiä. Tunniste, jonka avulla pystytään havaitsemaan, onko OBX-segmentti muuttunut, on [] -merkkien välissä oleva tieto. Jos tämä tieto poikkeaa edellisen OBX-segmentin [] -merkkien välissä olevasta tiedosta, silloin kyseessä on eri OBX-segmentti ja ohjelman täytyy luoda uusi OBX-segmentti ja asettamaan arvot oikeisiin kenttiin.

```
OBX.3[LMP^LMP].5.1;OBX.3[BWT^Booking Weight].5.1'
```

Koodi 5. Asetustiedostossa määritellyt OBX-segmentit.

OBX-segmenttien luomisessa ja arvojen asettamisessa esille tullut ongelma oli se, että ohjelma päällekirjoitti OBX-segmentin, joten ohjelma loi sen OBX-segmentin oikein, mikä on määritelty asetustiedostossa viimeisenä. Monen testauksen ja ohjelmakoodin muokkaamisen jälkeen, OBX-segmentit ja niihin kuuluvat arvot saatiin asetettua täysin oikei-

siin kohtiin HL7-viestiä. OBX-segmenttien luomiseen täytyi lisätä vielä juokseva, numerointi, joka helpottaa HL7-sanoman ymmärtämistä. Sen avulla pystytään erottamaan OBX-segmentit toisistaan, kun jokaisella OBX-segmentillä on oma järjestysnumeronsa. OBX-segmenttien luomisessa rajoituksena on, että asetustiedostossa OBX-segmentit täytyy määritellä viimeisenä, samoin kuin tulevan viestin parsinnassa. Toinen rajoitus koskee samojen OBX-segmenttien järjestystä asetustiedostossa. Koodista (Koodi 6), selviää, rajoituksen tarkoitus. OBR-segmentti, joka on yksi perustietosegmentti, täytyy olla määritelty ennen OBX-segmenttejä. Sama OBX-segmentti, johon tulee kaksi eri arvoa (5.1 ja 7.1), täytyy olla juuri arvoa nostavassa numerojärjestyksessä (7.1 ei saa olla ennen 5.1). Kun kaikki samat OBX-segmentit on määritelty, voidaan määritellä uusi OBX-segmentti. Rajoituksena on myöskin se, että jos määritellään kaksi samaa OBX-segmenttiä, niiden välissä ei voi olla eri OBX-segmentin määritelmää (OBX.3[LMP^LMP].5.1) ei voi olla OBX.3[BWT^Booking weight].5.1 ja OBX.3[BWT^Booking weight].7.1 välissä määriteltynä.

```
OBR.4.1;OBX.3[BWT^Booking weight].5.1;OBX.3[BWT^Booking weight].7.1;OBX.3[LMP^LMP].5.1
```

Koodi 6. Segmenttien rajoitus asetustiedostossa.

Alla olevassa koodissa (Koodi 7), on toimiva muodostunut lähtevä HL7-viesti, johon on parsittu Mirth -integraatioalussa JavaScript-ohjelmalla perustiedot ja OBX-segmentit ja asetettu arvot oikeisiin kohtiin siten, miten asetustiedostossa on määritelty, sekä lisätty OBX-segmenteille oma juokseva järjestysnumerointi.

```
MSH|^~\&|APP|FACILITY|LIFECYCLE|WALLAC|200510101144||ORM^O01|84
|P|2.3.1
PID|1|11aa|||LASTNAME^FIRSTNAME||20070825||||STREETADDRESS^^^^|||
NTE|1||
ORC|RE||||
OBR|1|XXXX-TY1234^||T21,T18|||||||||||||||||||||||^
NTE|1||
OBX|1||BWT^Booking weight||20050219||61
OBX|2||LMP^LMP||66.0|71
OBX|3||SCANG^Gestation By Scan||20122017
```

Koodi 7. Lähtevä HL7-viesti.

OBX-segmenttien ja arvojen asettaminen oikeaan kohtaan osoittautui lähtevässäkin viestissä haasteelliseksi, koska OBX-segmenttien luomisessa oli ongelmia, sekä miten XML- ja E4X indeksointi toimii Mirth -integraatioalustassa.

4.6 Mirthin hyödyt

Mirth -integraatioalustasta organisaatioiden saama hyöty on se, että Mirth Connect perustuu avoimeen lähdekoodiin, jolloin se on ilmainen ohjelmisto. Mirth Connectiin on kuitenkin saatavilla lukuisia kaupallisia laajennoksia, jolloin ohjelmiston käyttäminen tehostuu ja käytettävissä on yhä laajempi joukko asiantuntijoita ja pääsy ohjelman kehitysympäristöön mukaan. Yksi kaupallinen laajennos, joka olisi mahdollisesti ollut hyödyllinen tässä opinnäytetyössä on Message Generator -laajennos. Laajennoksella pystytään luomaan HL7 V2.x -standardin mukaisia viestejä. Laajennoksesta voidaan valita mitä Versio 2 -standardia käytetään viestin luomiseen. Laajennoksessa on myöskin vaihtoehtoja, mihin segmentteihin, kenttiin ja komponentteihin tietoja laitetaan (NextGen HealthCare 2018b).

Mirth -integraatioalusta on hyvin laaja ohjelmisto, joka tukee lukuisia eri viestintästandardeja ja HL7-standardeja. Mirth Connect:lla pystytään välittämään eri standardisia HL7-viestejä eri järjestelmien välillä, vaikka järjestelmät käyttävät eri standardeja, mikä tekee Mirth -ohjelmistosta entistä tehokkaamman, jolloin järjestelmäintegraatiosta tulee tehokas.

Mirth -ohjelmistolla pystytään toteuttamaan monenlaisia ja monitasoisia ratkaisuja HL7-sanomien välitykseen. Sanomia voidaan helposti muuttaa eri standardien välillä, ottaa HL7-sanomista yksittäisiä tietoja, käyttää hyödyksi lukuisia eri tiedonsiirtoprotokollia sekä lähettää sanomia tai niiden tietoja eri tietokantoihin ja niiden välillä. Sanomien tietoja on mahdollista lähettää käyttäjille jopa sähköpostitse. Mirth -ohjelmistoon on jo rakennettu paljon hyödyllisiä ominaisuuksia ja lukuisia eri toimintoja. Sanomien muokkaamisessa pystytään hyödyntämään näitä Mirth -ohjelmistossa jo olemassa olevia ominaisuuksia ja funktioita, mikä helpottaa ohjelmointia, koska käyttäjän ei itse tarvitse ohjelmoida tarvittavia funktioita.

4.7 Mirthin puutteet

Mirth -integraatioalusta suurin ongelma on se, että ohjelmistossa on haastavaa ohjelmoida, koska monet ohjelmointia helpottavat ominaisuudet puuttuvat. Merkittävin ominaisuus on ohjelmakoodin virheenkorjaus, jonka avulla pystytään helposti seuraamaan, mitä ohjelmakoodissa tapahtuu ja mitä arvoja muuttujat saavat. Virheenkorjauksen tarkoituksena on paikallistaa ohjelmakoodista virheitä. Jotta Mirthissä saadaan selville

muuttujien sisällöt, voidaan käyttää Mirthin omaa funktiota Logger.info(), jolloin () -merkkien sisään laittamalla muuttujan nimen saadaan muuttujan arvo tulostettua lokiin. Tätä voidaan käyttää virheenkorjauksen korvikkeena. Lokitiedot näkyvät kuitenkin eri näkymässä, kuin missä JavaScript-ohjelmakoodit ovat. Tämä aiheuttaa sen, että käyttäjän tulee valita uudelleen kanavat välilehti. Tämän jälkeen käyttäjän pitää valita oikea kanava, valita Edit Channel, valita välilehti, jossa JavaScript-ohjelmakoodia käytetään (Tulevassa vai lähtevässä). Lopuksi käyttäjän tulee valita ”Edit Transformer”, jolloin käyttäjä pääsee jälleen näkemään JavaScript-ohjelmakoodin. JavaScript-ohjelmakoodi on monen eri painikkeen takana eikä käyttäjä pysty näkemään JavaScript-ohjelmakoodia ja lokitietoja samaan aikaan, mikä on vakava puute.

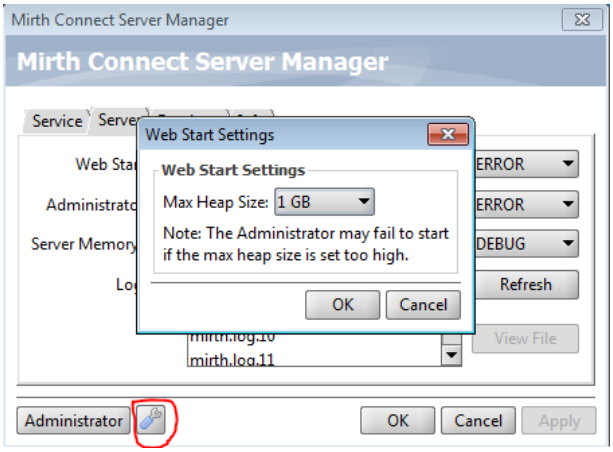
HL7-viestin parsinta osoittautui hyvin hankalaksi Mirthissä. Parsinnan kannalta tärkeää oli ymmärtää miten Mirthin omat ”msg ja tmp” -objektit toimivat. Näistä ei löytynyt kunnollista dokumentaatiota, joten niiden toiminta piti itse kokeilujen pohjalta hahmottaa ja selvittää. Yksi merkittävimmistä haitoista on se, että Mirth Connectissä käytetään XML-viestien muodostamiseen noin 10 vuotta sitten hylättyä JavaScript-laajennosta E4X-standardia. E4X-standardin kehitys- ja ylläpito on loppunut jo vuosia sitten, ja koska Mirth edelleen käyttää tätä standardoitua ratkaisua, sen muuttaminen on erittäin haasteellista.

Puutteita esiintyy silloin, kun kanava otetaan käyttöön. Tämän jälkeen JavaScript-ohjelmakoodissa ei enää pääse takaisin siihen tilanteeseen, missä koodi oli ennen kanavan käyttöönottoa. Jos käyttäjä tekee JavaScript-ohjelmaan muutoksia ja haluaa katsoa, miten se vaikuttaa ohjelman toimivuuteen, hän ei voi enää palata muokkaamattomaan koodiin ctrl+z näppäinyhdistelmällä. Käyttäjän täytyy muistaa missä tilassa koodi oli ennen kanavan käyttöönottoa. Käyttäjän täytyy muistaa, ottaa ylös tai laittaa kommentteihin alkuperäinen koodi, jotta se pysyy tallessa. Jos JavaScript-ohjelmassa on jokin virhe ja kanavan ottaa käyttöön, siitä ei tule käyttäjälle välttämättä virheilmoitusta. Tällöin käyttäjän pitää valita oikea kanava, ja valita ”view messages”, jolloin käyttäjä näkee mitä viestejä kanava lähettää tai mitä viestejä kanavaan tulee. Tämän jälkeen nähdään, onko kanavan lähtö- tai tulokohdassa sana ”Error”, jolloin sitä klikkaamalla käyttäjä pääsee näkemään mahdolliset virheilmoitukset. Jos virhe on selkeä ohjelmointivirhe, esimerkiksi rivin lopusta puuttuu ”;”, käyttäjä saa näkyvät virheilmoituksen, sekä sen millä rivillä kyseinen virhe ilmenee. Kanavan käyttöönotossa kestää n. 20 s, ennen kuin lokiin tulee tietoa, jos käyttäjä on ohjelmoinut ohjelman siten, että sinne täytyy tulla jotain tietoa. Tätä prosessia saa kuitenkin nopeutettua siten, että valitsee esimerkiksi ”Connection Log”, jonka jälkeen valitsee ”Server log”. Tämän jälkeen tiedot pitäisi näkyä.

Luodusta tiedostosta pitää käyttäjän katsoa ”Date modified” -aikaa, jolloin voidaan mahdollisesti huomata, tuleeko tiedostoon muutoksia. Esimerkiksi muuttuuko jokin tekstitiedostosta. Vaikka aika muuttuisikin, tämä ei tarkoita sitä, että muutoksia tiedostossa tapahtuu. Jos kuitenkin aika ei muutu, tämä tarkoittaa usein sitä, että JavaScript-ohjelma ei toimi ollenkaan.

Lokiin tulevissa tietojen esitystavassa on myös erikoinen poikkeus normaaleihin esitystapoihin. Jos käyttäjä on esimerkiksi JavaScript-koodissa määritellyt kaksi tietoa, jotka hän haluaa lokiin järjestyksessä: Logger.info(”segment”); ja Logger.info(”OBX”). Loki näyttää tiedot siten, että alimpana on sana segment ja ylimpänä OBX. Tiedot esitetään lokissa ”alhaalta ylös”, vaikka käyttäjä on ohjelmassa määritellyt ne juuri päin vastaiseen järjestykseen. Useissa ohjelmointityökaluissa tämä on usein juuri niin kuin sen pitäisikin olla, ensin tulee sana segment ja sen jälkeen sana OBX. Tämä voi aiheuttaa tulkintaongelmia, mutta pienen totuttelun jälkeen tämä ei osoittaudu suureksi ongelmaksi. Lähinnä tämä ongelma osoittautuu pieneksi häiriöksi, kun asia esitetään juuri päinvastaisessa järjestyksessä.

Yksi opinnäytetyössä ilmi tullut ongelma oli Mirth Connectin muistiongelma. Koska käytössä on useita kanavia ja ne lähettävät ja vastaanottavat viestejä, sekä käyttäjä voi samaan aikaan muokata JavaScript-ohjelmakoodeja, tämä voi aiheuttaa muistin loppumisen. Muistia voi kuitenkin lisätä käynnistämällä Mirth Connect Server Manager -työkalun ja painamalla Administrator -napin vierestä löytyvää asetukset kuvaketta, josta käyttävä voi suurentaa muistinkokoa (Kuva 18).



Kuva 18. Muisti kapasiteetin lisääminen.

Koko Mirth integraatio-ohjelma saattaa lakata toimimasta ja kaatua, jos JavaScript-ohjelmassa on joki vakava virhe. Jos JavaScript-koodissa on esimerkiksi ikuinen For-toistorakenne, ohjelmisto saattaa lakata toimimasta sekä antaa myös muistiongelma virheilmoituksen, jolloin koko ohjelma kaatuu. Kanavatkin saattavat tästä syystä lakata toimimasta, jolloin ainoa keino, jolloin kanavia pääsee taas muokkaamaan, on pysäyttää kanava. Tai jopa käynnistää koko tietokone uudelleen. Tämä ei kuitenkaan takaa, että kanavat toimivat, vaan ne saattavat uudelleen käynnistyksen jälkeen edelleen olla poissa käytöstä, jolloin keinona voi olla kanavan poistaminen tai jopa koko Mirth integraatio-ohjelmiston uudelleen asentaminen. Tästä syystä on erittäin suositeltavaa, että käyttäjä ottaa varmuuskopioita JavaScript-ohjelmakoodeista ja kanavan asetuksista, jolloin käyttäjä voi helposti päästä takaisin siihen, missä tilassa käyttäjä oli. Kanavia voi myös kloonata, jolloin ohjelma luo toisen täysin samanlaisen kanavan samoin asetuksin ja JavaScript-ohjelmiin. Tämä on yksi keino ottaa varmuuskopiota kanavasta, jos varsinainen kanava lakkaa toimimasta tai jää jumiin, kanavan voi yrittää kloonata ja poistaa, jolloin käyttäjä voi käyttää uutta kopioitua kanavaa ja jatkaa siitä mihin jäi.

5 LOPUKSI

Sosiaali- ja terveydenhuollon tietojärjestelmien integraatiossa on havaittu merkittäviä ongelmia johtuen tietojärjestelmien erilaisesta arkkitehtuurista ja niiden toteutuksesta. Tulevassa Sote-uudistuksessa digitalisaatio ja etenkin sosiaali- ja terveydenhuollon tietojärjestelmien välinen integraatio on keskeisessä osassa uudistusta. Teknologian, ohjelmistotuotannon ja ohjelmistojen nopea kehitys vaikuttavat oleellisesti myös sosiaali- ja terveydenhuollon käytössä oleviin tietojärjestelmiin ja niiden suunnitteluun.

Tässä opinnäytetyössä tutkittiin avoimeen lähdekoodiin perustuvaa Mirth -integraatioalustan soveltuvuutta sosiaali- ja terveydenhuolto alalla käytössä olevaan HL7-standardin mukaisten sanomien välitykseen. Tarkoituksena oli selvittää PerkinElmer-teknologiakonserniin kuuluvan Wallac Oy:n tarvetta, Mirth -integraatioalustan hyödyntämisestä osana laajempaa kokonaisuutta. Mirth -integraatioalustan hyödyllisyyttä arvioitiin saatujen tulosten perusteella. Selvityksen aikana esiin tulleiden tulosten avulla koottiin kokonaiskuva Mirth -integraatioalustan hyödyllisyyteen vaikuttavista tekijöistä HL7-sanomien parsinnassa ja välityksessä.

Opinnäytetyössä käytettiin toimeksiantajan määrittelemää esimerkki standardin HL7 V 2.3 mukaista HL7-sanomaa ja asetustiedostoa tietojen parsintaan. Työssä ohjelmoitiin Mirth -integraatioalustassa kaksi erillistä JavaScript-ohjelmaa. Toisella ohjelmalla parsittiin tulevasta HL7-sanomasta asetustiedoston määrittelemät tiedot ja tallennettiin parsitut tiedot tekstitiedostoon. Toisella ohjelmalla lähtevään tyhjään HL7-sanomaan parsittiin arvot tekstitiedostosta ja arvot asetettiin oikeisiin kohtiin HL7-sanomaa.

Selvityksen tuloksista selvisi, että Mirth -integraatioalustalla pystytään välittämään ja parsimaan HL7-sanomista tiettyjä osia. Mirth -integraatioalustassa toteutettavat JavaScript-ohjelmoinnit ovat erittäin haastavia, puuttuvien ohjelmointityökalujen, ominaisuuksien ja tekniikoiden puutoksen johdosta. JavaScript on ongelmallinen siinä mielessä, että JavaScript käyttää Mirthissä XML-syntaksia. Tämä johtaa siihen, että JavaScript-ohjelmaa ei pystytä testaamaan NodeJS:llä. HL7-sanomien parsinta osoittautui haastavaksi toteuttaa Mirth -integraatiotyökalussa, koska Mirthin käyttämien objektien ”msg” ja ”tmp” käyttö tulee tietää. Ohjelmoinnin aikana jouduttiin turvautumaan Microsoftin Visual Studio 2015 ohjelmistoon ja NodeJS:n käyttöön. Visual Studio 2015 helpotti ohjelmointia huomattavasti, koska se sisältää monia hyödyllisiä apuvälineitä ohjelmoin-

tiin, sekä Visual Studiossa pystyttiin simuloimaan Mirth Connectin ”msg”-objektia. JavaScript-ohjelmat toteutettiin suurimmaksi osaksi Microsoft Visual Studio 2015 ohjelmistossa, jossa pystyttiin hyödyntämään NodeJS:ää ohjelman testauksessa. Lähdekoodit siirretiin Visual Studiosta Mirth Connectiin, jossa puolestaan suoritettiin kaikki tarvittavat muutokset, jotta ohjelma toimisi Mirthissä. OBX-segmenttien parsinnan aikana jouduttiin turvautumaan Rhino JavaScript-laajennokseen, jolla pystyttiin testaamaan XML-syntaksia.

Mirth -integraatioalustan käyttö oli helppoa oppia informatiivisen graafisen käyttöliittymän johdosta, sekä lukuisten eri esimerkkien ja ohjelmoinnin kautta tulleiden kokemuksen pohjalta käyttö tehostui entisestään. Internetistä saatujen esimerkkien ja ohjeiden perusteella asioiden ymmärtäminen ja omaksuminen helpottuivat. Esimerkit rajoittuivat hyvin pitkälle uuteen standardiin HL7 V3 eikä juuri niihin asioihin, joita tehtiin, ei löytynyt kunnollisia esimerkkejä.

Johtuen laajasta standardien tuesta Mirth -integraatioalusta on kuitenkin tehokas ratkaisu sosiaali- ja terveydenhuollon tietojärjestelmien välisessä integraatiossa. Mirth -integraatioalusta on laajasti käytössä erityisesti Yhdysvalloissa sosiaali- ja terveydenhuollossa, joten tästä voidaan päätellä, että Mirth -ohjelmisto ratkaisu on tehokas ja kestävä ratkaisu.

Opinnäytetyön aikana tulleiden havaintojen ja kokemusten pohjalta Mirth -ohjelmistossa on sekä pieniä, että suuriakin ongelmia. Ohjelmoinnit ja tietojen parsinta olivat haastavia toteuttaa Mirthissä. Mirth Connectissa ei kuitenkaan tarvitse ohjelmoida, tai käyttää JavaScript-ohjelmakieltä, vaan ohjelmoinnit voidaan toteuttaa toisessa kehittyneemmässä ohjelmointiympäristössä ja sisällyttää ohjelmakoodit Mirthiin. Mirth voi kuitenkin havaintojen ja kokemusten kautta olla toimiva ratkaisu. Kokonaiskuvaa tulee, pilkkoa pienempii kokonaisuuksiin, jotta kaikkea ei tarvitse Mirth -ohjelmistossa toteuttaa. Mirth Connect yhteisöllä on laaja tuki eri organisaatioilta sekä lukuisia sosiaali- ja terveydenhuoltoalan asiantuntijoita. Näiden avulla tarkoitus on saada Mirth -integraatioalustasta yhä tehokkaampi ratkaisu HL7-sanomien välitykseen organisaatioiden välillä, että sosiaali- ja terveydenhuollon alan tietojärjestelmien integraatioiden välillä.

Mirth -integraatioalusta ratkaisussa on huomattavasti selvitettävää, jotta se voidaan ottaa tuotannolliseen käyttöön. Tulevaisuudessa Mirth -integraatiotyökalua tulisi ensisijaisesti testata osana laajempaa kokonaisuutta. Kestävän ratkaisun takia Mirth -integraa-

tioalustan tulisi voida lukea asetustiedostot ulkoisen rajapinnan kautta ja suorittaa parsinnat ulkoisessa Java-kirjastosta, jotta ohjelmistojen versiohallinta, päivittäminen ja hallittavuus tehostuvat huomattavasti. Tämä tekee Mirth Connectin käytön ohjelmistoteknisesti helpommaksi. Mirth Connect voisi vain vastaanottaa ja välittää HL7-sanomia organisaatioiden välillä. Mirth Connect voisi tulevaisuudessa hakea internetpalvelimelta asetustiedoston ja koodi-ikkunassa kutsutaan ulkoista Java-kirjastoa, jossa puolestaan suoritetaan viestien parsinnat.

Tässä opinnäytetyössä kaikki on toteutettu yksinomaa Mirth integraatio-ohjelmistossa. Tulevaisuudessa toteutusta tulisi pilkkoa pienempiin kokonaisuuksiin, jolloin saadaan enemmän rajapintoja, ohjelmistoja ja ohjelmia, joiden tarkoituksena on suorittaa erinäisiä tehtäviä. HL7-sanomien välityksessä ja tietojen hakemisesta internetpalvelimelta, suorituskyky nousee erittäin merkittäväksi tekijäksi. Tästä johtuen tärkeää on selvittää, miten järjestelmien välinen suorituskyky ja tietoliikenneyhteys saadaan toimivaksi ja, että se on kunnossa ja toimiva. Tilanteessa, jossa sovellus palvelee useampaa asiakasta tulisi selvittää, asetustiedostojen haku, parsinta ja suorituskyky.

Todellisuudessa joskus joudutaan tilanteeseen, jossa on useampi asiakas ja jotka lähettävät HL7-sanomia yhtä aikaa. Asiakkaat voivat lähettää useita eri HL7-viestejä eri tahtiin, jolloin yhden asiakkaan lähettämien HL7-sanomien väliin voi tulla jonkin toisen asiakkaan lähettämä HL7-sanoma. Tästä syystä on erityisen tärkeää tunnistaa, kuka viestit on lähettänyt ja minkälaiset asetukset ja parsinnat viesteihin täytyy tehdä.

Tilanteessa, jossa käytetään ulkoisia rajapintoja, joiden takana viestien validointi tapahtuu, täytyy tutkia, että onko suorituskyky riittävä. Viestin muodostamisessa tai lukemisessa voi tapahtua jokin poikkeavatapahtuma, joka täytyy pystyä hallitsemaan. Poikkeustilanne voi olla myöskin se, että Mirth -ohjelmisto ei ole syystä taikka toisesta käytössä. Käynnissä on huoltotilanne tai, että kanava johon tulevat HL7-sanomat tai lähtevät HL7-sanomat eivät ole käytössä. Jos tallaiseen tilanteeseen joudutaan, asiakkaan täytyy saada tästä jollain tapaa tieto, että HL7-sanomia ei tilapäisesti voida lähettää. Mirth Connect lähettäisi tästä tiedon asiakkaalle, käyttämällä NACK:ia**,** joka ilmoittaa asiakkaalle, että HL7-sanomia ei voida tällä hetkellä lähettää ja, että asiakas voi yrittää viestin lähetystä uudelleen tietyn määrätyn ajanjakson kuluttua. Kun asiakkaan lähettämät HL7-sanomat saapuvat onnistuneesti Mirth -ohjelmiston kanavaan, tästä lähtee tieto asiakkaalle tieto ACK:lla, joka ilmoittaa asiakkaalle, että HL7-sanomien lähetys on onnistunut ilman virheitä.

Mirth -ohjelmisto joudutaan asentamaan asiakkaiden tietokoneisiin ja asettamaan tietyt kanavat ja niihin tarvittavat asetukset, jotta asiakas pystyy lähettämään ja vastaanottamaan HL7-sanomia. Asetuksia voidaan joutua muuttamaan useaan kertaan. Ilman, että yrityksen työntekijän täytyisi mennä asiakkaan luokse asetukset muuttamaan, täytyisi selvittää, miten kanavien asetukset, määritykset ja ulkoiset kirjastot pystyttäisiin kopioimaan tiedostoon. Tämä tiedosto voidaan lähettää asiakkaalle, jolloin asiakas voisi kopioida kyseisen tiedoston Mirthiin, jolloin uudet asetukset astuvat voimaan. Näin saataisiin lisää tehokkuutta, koska asetukset pystyttäisiin lähettämään helposti useammalle asiakkaalle ilman asiakaskäyntejä tai, että asiakkaan täytyisi itse muuttaa asetukset.

Asiakkaiden lähettäessä HL7-sanomia on erityisen tärkeää pystyä todentamaan, että sanomat on muodostettu oikein ja, että sanoma on paikkansapitävä. Tästä syystä, kun asiakas lähettää HL7-sanoman yrityksen Mirth Connectin tiettyyn kanavaan, Mirth Connect tutkisi, että onko asiakkaalta tullut HL7-sanoma muodostettu oikein. Mirth huomaisi, jos viesti ei täyty kaikkia asetettuja kriteereitä, joiden avulla pystytään todentamaan, että HL7-sanoma on muodostettu oikein ja että sanoma on paikkansapitävä. Jotta tätä ratkaisua pystyttäisiin tulevaisuudessa hyödyntämään, täytyy selvittää, mitä ominaisuuksia Mirth -ohjelmistossa on virheiden käsittelyyn. Lokitiedot ovat yksi ominaisuus, mutta tämä ei yksin ole riittävä ja tehokas ratkaisu käsitellä virheitä, virheiden käsittelyyn tarvitaan muitakin ratkaisuja.

Sanomien lähettäessä asiakkailla voi olla erikoisvaatimuksia, jotka usein ovat melko harvinaisia. Mirth -ohjelmiston tulisi näin ollen taipua mahdollisiin erikoistilanteisiin, joita saattaa ilmetä, jos asiakkaalla on jokin erikoisvaatimus HL7-sanomassa. Tutkittavana onkin, kuinka Mirth Connect pystyy tulkitsemaan erikoistilanteita ja miten niitä voidaan tehokkaasti hallita Mirth -ohjelmistossa.

Potilastietoja voidaan joutua aika-ajoin muuttamaan, esimerkiksi asiakkaan muuttaessa joudutaan päivittämään asiakkaan osoitetiedot. Potilaantietojen muutos voisi tulevaisuudessa hoitua yksinkertaisella HL7-sanomalla. Potilastietojen muutokseen tulisi Mirth -ohjelmistossa olla tuki useammalle HL7-sanomatyypille. Tämä ratkaisu helpottaisi potilastietojen hallintaa ja toisi lisää tehokkuutta tietojen ylläpitämiseen.

Mirth -integraatioalustassa riittää paljon selvitettävää, jotta se voidaan ottaa tuotannolliseen käyttöön. Tulevat ja lähtevät HL7-sanomat tulee validoida. Sosiaali- ja terveydenhuolto alla on käytössä lukuista eri standardeja, direktiivejä, kuten AIMD-, MD ja IVD-

direktiivit, erityislakeja, asetuksia sekä maakohtaisia lainsäädäntöjä. HL7-sanomien välityksessä esiin nousee suuri kysymys tietoturvasta, joka on esillä lähes joka päivä uutisotsikoissa. Tietomurrot ja kyber aktivismi ovat kasvaneet viime vuosina huomattavasti. Asiakkaiden ja organisaatioiden välillä kulkee huomattava määrä potilastietoja, joten tietoturva ja tietosuoja täytyy olla tehokkaasti hallinnassa, jotta potilastiedot eivät vuoda kolmansille osapuolille. Tätä osaa täytyy myös tutkia, jotta tulevaisuudessa Mirth -ohjelmistoa voidaan käyttää tuotannolliseen tarkoitukseen.

LÄHTEET

Adobe 2018. PDF. Kolme kirjainta, jotka muuttivat maailmaa. Mikä PDF on? Viitattu 24. tammikuuta 2018 https://acrobat.adobe.com/fi/fi/acrobat/about-adobe-pdf.html

Ala-Mutka, K.; Rintala, M.; Savikko, V. & Palviainen, J. 2002. Tampere University of Technology. Tietotekniikan peruskurssi. 19.2 OSI-Malli. Viitattu 21. tammikuuta 2018 http://www.cs.tut.fi/eta-opetus/titepk/luku19/OSI.html.

AsciiTable 2010. ASCII Table and Description. Viitattu 24 tammikuuta 2018 http://www.asciitable.com/

ASTM International 2018. ASTM E1381 – 95. Viitattu 31. tammikuuta 2018 https://www.astm.org/DATABASE.CART/HISTORICAL/E1381-95.htm

Brauer, J. 2011. Mirth Connect FAQ. Viitattu 17. tammikuuta 2018 http://www.mirthcorp.com/community/wiki/display/mirth/Mirth+Connect+FAQ

Corepointhealth 2018a. The HL7 Evolution. Comparing HL7 Version 2 to Version 3, Including a History of Version 2. Viitattu 24 tammikuuta 2018

Corepointhealth 2018b. HL7 NTE (Notes and Comments) Segment. Viitattu 26 tammikuuta 2018 https://corepointhealth.com/resource-center/hl7-resources/hl7-nte-notes-comments

Ecma International 2018. ECAMScript for XML (E4X) Specification. Viitattu 18. tammikuuta 2018 https://www.ecma-international.org/publications/files/ECMA-ST-WITHDRAWN/Ecma-357.pdf

Grace, W. 2008. E4X: JavaScript on steroids. Viitattu 18. tammikuuta 2018 https://www.ibm.com/developerworks/library/x-javascript4x/index.html

Gunter, T. & Terry, N. 2005. The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions. Viitattu 24. tammukuuta 2018 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1550638/

Health Level Seven International 2018a. About HL7. Viitattu 14. tammikuuta 2018 http://www.hl7.org/about/index.cfm?ref=common

Health Level Seven International 2018b. Introduction to HL7 Standards. Viitattu 21. tammikuuta 2018 http://www.hl7.org/implement/standards/index.cfm?ref=nav

Health Level Seven International 2018c. Transport Specification: MLLP, Release 1. Minimal Lower Layer Protocol (MLLP). Viitattu 24. tammikuuta 2018 http://www.hl7.org/documentcenter/public_temp_868CC83C-1C23-BA17-0C5C3031E51B24A7/wg/inm/mllp_transport_specification.PDF

Health Level Seven International 2018d. HL7 Version 2 Product Suite. Viitattu 24 tammikuuta 2018

Health Level Seven International 2018e. HL7 Version 3 Normative Edition, 2017. Viitattu 29 tammikuuta 2018 https://www.hl7.org/implement/standards/product_brief.cfm?product_id=454

Health Level Seven International 2018f. HL7 Frequently Asked Questions. Viitattu 29 tammikuuta 2018 http://www.hl7.org/about/FAQs/index.cfm?ref=nav

HL7 Finland ry 2018a. Rajapintakartta. Viitattu 21. tammikuuta 2018 http://www.hl7.fi/hl7-rajapintakartta/

HL7 Finland ry 2018b. HL7-versio 2.3 paikallistettu dokumentaatio. Yleiskuvaus HL7-version 2.x perusperiaatteista. Viitattu 24 tammikuuta 2018 http://www.hl7.fi/hl7-rajapintakartta/hl7-versio-2-3-paikallistettu-dokumentaatio/

HL7 Finland ry 2018c. Rajapintakartta. HL7 v2.3 Laboratoriosanomat v3.1. Viitattu 25 tammikuuta 2018 http://www.hl7.fi/hl7-rajapintakartta/hl7-v2-3-laboratoriosanomat/

HL7 Finland ry 2018d. HL7 Finland – V3 messaging. Viitattu 29 tammikuuta 2018 http://www.hl7.fi/hl7-finland-%E2%80%93-v3-messaging/

Jmnater. 2017. E4X. Viitattu 18. tammikuuta 2018. https://developer.mozilla.org/en-US/docs/Archive/Web/E4X

Jormalainen, V. 2015. Terveydenhuollon tietojärjestelmät Suomessa nyt ja tulevaisuudessa. Aalto-yliopisto, Sähkötekniikan korkeakoulu. Viitattu 30 tammikuuta 2018 http://elec.aalto.fi/fi/midcom-serveattachmentguid-1e57d7f5ac168b07d7f11e589b0f5ad480af4e8f4e8/s1_jormanainen.pdf

Kodata 2018. OSI-Malli. Viitattu 21. tammikuuta 2018 http://www.koudata.fi/node/598

MDN web docs 2018. Introduction to the DOM. Viitattu 24. tammikuuta 2018 https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

NextGen Healthcare 2018. Mirth Connect 3.5 User Guide. Viitattu 10. tammikuuta 2018 https://bridge.nextgen.com/media/3244/mirth-data-sheet-mirth-connect-3-5-user-guide.pdf

NextGen HealthCare 2018b. Extensions. Message Generator. Viitattu 1. helmikuuta 2018 https://www.nextgen.com/Interoperability/Mirth-Solutions/Connect-Extensions?tab=true

Oracle 2011. Working With the Schematron HL7 V3 Sample Project. Viitattu 29 tammikuuta 2018 https://docs.oracle.com/cd/E21454_01/html/821-2671/giank.html

Public Health Informatics Institute 2016. HL7 Version 2 Messages. Viitattu 24. tammikuuta 2018 https://www.phii.org/sites/www.phii.org/files/resource/files/HL7%20V2%20Standards%20Overview.pdf

Richard, P. & Christopher, K. 2015. Case Study for Mirth Connect Investigation. Viitattu 17. tammikuuta 2018 https://www.slideshare.net/drighten/mirth-connect-as-interface-engine-case-study-v2

SphinxKnight. 2017. Rhino. Viitattu 18. tammikuuta 2018 https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino

Ståhlberg, T. 2018. Terveydenhuollon laitteiden lakisääteiset määräykset kansainvälisillä markkinoilla. Suomi ja EU fokuksessa. Business Finland

Vjeux. 2013. JSX: E4X The Good Parts. Viitattu 9. helmikuuta 2018 http://blog.vjeux.com/2013/javascript/jsx-e4x-the-good-parts.html

w3schools 2018. SQL Tutorial. Viitattu 24. tammikuuta 2018 https://www.w3schools.com/sql/

Wikimedia commons 2018. OSI-malli. Viitattu 21. tammikuuta 2018 https://commons.wikimedia.org/wiki/File:OSI-malli.png.