TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rainer Randmaa 193468IABB

Äriarhetüüpidel põhinevate metamudelite ja FHIR ressursside semantilise koostalitluse hindamine

Bakalaureusetöö

Juhendajad: Igor Bossenko
Gunnar Piho

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Rainer Randmaa

18.05.2022

Annotatsioon

Infosüsteemide koostalitlus tervishoiuinformaatika valdkonnas on osutunud keerukaks probleemiks andmemudelite, sõnumiformaatide jm. semantilise heterogeensuse tõttu. Senised katsed probleemi lahendada tuginevad ühildamisel põhineval lähenemisel – arendatakse ja võetakse kasutusele ühiseid standardeid. Standardid, mis on töötatud välja erinevate tarnijate poolt, kipuvad aga olema semantilise heterogeensuse tulemusel koostalitlusvõimetud. Ühildamisel põhinevale koostalitluse lähenemisele võib kasulikuks täienduseks olla föderatsioonil põhinev koostalitluse lähenemine, mille kohaselt saab mudeleid dünaamiliselt käitusajal spetsifitseerida ning mudelid pole ette määratud. Lõputöö põhineb artiklil, mis on esitatud HEDA 2022 konverentsile. Lõputöö autor on artikli esimene autor. Artiklis esitatakse äriarhetüüpidel põhinevad metamudelid, mida on artikli autorite arvates võimalik kasutada terviseandmete föderatsioonil põhineva koostalitluse saavutamiseks. Illustreeritakse ja hinnatakse metamudelite koostalitlusvõimet FHIR ressurssidega, mis on HL7 poolt arendatud terviseandmete koostalitlusstandardi põhiosadeks. Metamudelite hindamiseks juurutatakse hajutatud test-keskkond, milles transformeeritakse andmeid FHIR'i ja äriarhetüüpide vahel. Sõnumivahetuse implementeerimiseks võetakse kasutusele NextGen Connect (Mirth Connect). Artikli autorid usuvad, et nende töö on oluline panus terviseinfosüsteemide *no-code/low-code* föderatsioonil põhineva koostalitluse võimaldamise suunas, mis suures pildis võimaldaks erinevaid protokolle, andmemudeleid ja nende versioone (näiteks FHIR, HL7 v2.x, openEHR jne.) kasutavatel terviseinfosüsteemidel andmeid semantiliselt sidusal viisil vahetada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 16 leheküljel, 8 peatükki, 6 joonist ja 4 lisa. Lisa 2 sisaldab artiklit, mille põhjal lõputöö koostatud on.

Abstract

Evaluating business meta-models for semantic interoperability with FHIR resources

Information systems interoperability in healthcare has proved itself a difficult challenge due to the semantic heterogeneity of models, data, messages, etc. So far, the attempts to solve this problem have involved a unified approach through the development and utilization of common standards. However, this approach also has issues in semantic heterogeneity because in the case the standardized data models and exchange protocols have been developed by different vendors, they normally tend to not be interoperable. A federated interoperability approach, where models can be dynamically specified at runtime rather than being pre-determined, might be a useful supplement to the unified interoperability approach. This thesis is based on a workshop paper submitted to the HEDA 2022 conference. The author of this thesis is the first author of the article. The authors of the article propose executable business meta-models, that according to their understanding, can be utilized as a shared ontology to achieve federated interoperability. These meta-models are evaluated by illustrating semantic interoperability with FHIR resources, the building blocks of the healthcare interoperability standard developed by HL7. For the evaluation, a distributed test environment is set up, to illustrate how the data is exchanged and transformed, back and forth. Here NextGen Connect (Mirth Connect) is used as a mapping engine between FHIR and the meta-models. The authors of the article believe their work is an important contribution towards seamless no-code/low-code federated interoperability of health information systems where different systems can exchange medical data in a semantically coherent way by using different protocols, data models and their versions, such as FHIR, HL7 v2.x, openEHR, etc.

The thesis is in Estonian and contains 16 pages of text, 8 chapters, 6 figures and 4 appendixes. Appendix 2 contains the article this thesis is based on.

Lühendite ja mõistete sõnastik

ABC4HEDA	Äriarhetüüpidel põhinevaid metamudeleid implementeeriv tarkvara
API	Application Programming Interface
FHIR	Fast Healthcare Interoperability Resources
GraphQL	API-ga kasutamiseks mõeldud päringukeel, API paradigma
HL7	Health Level 7
HTTP	Hypertext Transfer Protocol
HTTP otspunkt	Liides veebiteenusega suhtlemiseks HTTP protokolli kaudu
JSON	JavaScript Object Notation
LOINC	Regenstrief Institute arendatud meditsiiniterminoloogia koodisüsteem
No-code/low-code lahendused	Rakenduste arendamise võimalus visuaalsete liidestega, mis nõuavad vähe või üldse mitte koodi kirjutamist
REST	Representational State Transfer, API paradigma
RESTful	API paradigma, mis üldjoontes allub REST paradigmale
SNOMED	SNOMED International arendatud meditsiiniterminoloogia koodisüsteem
SQL	Structured Query Language
SUM	Single Underlying Model
TDD	Test-Driven Development
UML	Unified Modelling Language
XML	Extensible Markup Language

Sisukord

Jooniste loetelu

1 Sissejuhatus

Käesolev lõputöö koosneb ingliskeelsest artiklist (vaata Lisa 2) ja laiendatud eestikeelsest kokkuvõttest. Artikkel on esitatud HEDA 2022 konverentsile [1], mis leiab aset Bergenis 19-24 juuni 2022. Käesoleva töö autor on artikli esimene autor. Töö autori panus artiklisse on valminud prototüübi arendus ja artikli käsikiri. Käsikirja valmimist toetasid ka ülejäänud autorid. Artikkel on kirjutatud kasutades LaTeX-it ning kasutab Ceurart malli. HEDA 2022 nõudmiste alusel võis artikli maht olla kuni 10-14 lehekülge. Valminud käsikirja pikkuseks tuli 11 lehekülge. Lisa 1 kujutab artikli esmast versiooni, mille esitamise tähtajaks oli 30.04.2022. 16.05.2022 seisuga on artikkel täies mahus konverentsil esitamiseks ja publitseerimiseks vastu võetud. HEDA 2022 retsensendid esitasid otsuse vastu võtmise kuupäeval ka kaks retsensiooni, mille põhjal tehakse artiklile parandusi. Paranduste tulemusena valmib *camera ready* versioon publitseerimiseks. Retsensioonid on samuti esitatud töö lisadena (vaata Lisa 3 ja Lisa 4).

Käesolev laiendatud kokkuvõte annab ülevaate artiklis käsitletavast probleemist, artikli eesmärkidest, kasutatud metoodikast ja tuuakse esile tähtsam informatsioon uuritavate objektide (äriarhetüübid, HL7 FHIR, Mirth Connect) kohta. Lisaks esitatakse artikli tulemused ja järeldused.

1.1 Probleem

Artikkel käsitleb probleemi, et andmete ja infosüsteemide keerukus tervishoiuinformaatika valdkonnas pärsib terviseandmete kasutamist meditsiini, teaduse, rahva tervise ja majanduse arenguks. Tervishoius kasutatavad andmemudelid on semantiliselt heterogeensed, mis tähendab, et kuigi andmed kirjeldavad sama objekti, pole need koostalitlusvõimelised. Erinevad tarnijad arendavad oma tarkvara lähtudes erinevatest standarditest ja spetsifikatsioonidest. Seetõttu on andmevahetus tervishoiuasutuste süsteemide vahel keeruline. Senised katsed koostalitlusvõime saavutamiseks on hõlmanud ainult uute standardformaatide loomist ja kehtestamist või ühest mudelist teise teisendamist. Sellised lahendused tuginevad ühildamisel põhinevale

(*unified*) koositalitluse lähenemisele, mis pole kuigi skaleeritav. Idee on proovida föderatsioonil põhineva (*federated*) koostalitluse lähenemist, mis tähendab, et pole kindlaks määratud ühist formaati vaid erinevad mudelid suudavad üksteise suhtes jooksvalt kohalduda. [2] Üks võimalus sellist kohaldumist pakkuda on süsteem, mis võimaldab mudeleid käitusajal spetsifitseerida. Artikli autorid usuvad, et äriarhetüüpidel põhinevaid metamudeleid on võimalik kasutada terviseandmete ühiste ontoloogiate spetsifitseerimise keelena. Nendel metamudelitel põhinev süsteem võimaldaks saavutada föderatsioonil põhinevat koostalitlust.

1.2 Eesmärk

Artikli eesmärgiks on hinnata ja katsetada äriarhetüüpidel põhinevate metamudelite semantilist koostalitlust HL7 FHIR terviseandmete sõnumivahetusprotokolliga [3]. Illustreeritakse metamudelite hindamise ja valideerimise ning FHIR protokolli kasutavate sõnumivahetuskanalite arendamise protsessi. Artikli raames koostati kontseptsiooni tõestav prototüüp sõnumivahetuskanalist FHIR’i ja äriarhetüüpidel põhinevate metamudelite vahel. Lisaks on artikli eesmärk hinnata Mirth Connecti kui tööriista sõnumivahetuskanalite implementeerimisel.

2 Metoodika

Artikli eesmärgi saavutamiseks kasutatud metoodikaks on konstrueeritud ülesannete lahendamine. Arendati sõnumivahetuskanal HL7 FHIR'i ja äriarhetüüpidel põhinevate metamudelite vahel, mille jaoks disainiti integratsiooni arhitektuur kasutades ettevõtteintegratsiooni mustreid [4]. Semantilist teisendamist FHIR formaadi ja metamudelite vahel illustreeriti kasutades UML diagramme. Sõnumivahetuskanali disaini implementeerimiseks kasutati NextGen Connect (Mirth Connect) integratsioonimootorit. Mirth Connectis oli vajalik Java, JavaScript'i ja SQL'i kasutamine. Lisaks kasutati vabavaralist HAPI FHIR API-t [5], mis on HL7 FHIR'it implementeeriv Java API. Sõnumivahetuskanalite arendamisel lähtuti puhta koodi põhimõtetest [6] ning ametlikust Mirth Connecti API dokumentatsioonist [7]. Integratsiooni testimiseks juurutati metamudeleid implementeeriv tarkvara ABC4HEDA [8] ja selle talletuskihina kasutatav Microsoft SQL Server virtuaalmasinatesse kasutades Azure pilveteenuseid. Samuti juurutati sõnumikanali implementatsioon Mirth Connectis eraldi virtuaalmasinasse, et jäljendada hajusate süsteemide keskkonda. Seejärel saadeti HL7 poolt ametlikult avaldatud FHIR sõnumite näiteid läbi süsteemi, et valideerida integratsiooni implementatsiooni. Sõnumite saatamiseks kasutati Postmani [9].

3 Äriarhetüüpidel põhinevad metamudelid

Semantilist koostalitlust võimaldavad metamudelid põhinevad Arlow ja Neustadt'i äriarhetüüpidel [10] ja terviseandmetele rakendatud Zachmani raamistikul [11]. Artiklis tulid kasutusele eelkõige *Party* (osapoole), *PartyRole* (osapoole rolli) ja *Product* (toote) arhetüübimustrid, kuid metamudelid implementeerivad nendele lisaks *Order* (tellimuse), *Inventory* (inventari), *Rule* (reegli), *Money* (raha) ja *Quantity* (kvantiteedi) arhetüübimustreid. Ühtlasi on metamudelitele juurde ehitatud ka *Process* (protsessi) arhetüübimuster, mille arendas välja Gunnar Piho oma doktoritöö raames [12].

Artikli autorid usuvad, et need metamudelid moodustavad terviseandmetele ühtse alusmudeli (SUM) [13] ning iga tervishoiuinformaatika standardi mudel on vaade sellest alusmudelist. Oluline aspekt nende metamudelite olemuses on see, et andmed ja teadmised andmetest on üksteisest lahutatud. Kuidas konkreetsed arhetüübid terviseandmete konteksti sobituvad on välja toodud artikli kolmandas peatükis.

Välja pakutavaid metamudeleid implementeeriv tarkvara kannab nimetust ABC4HEDA. Tarkvara koosneb andmemudelist, valdkonnamudelist, repositooriumi mustrit [14] implementeerivast infrastruktuuri kihist ning kasutajaliidest implementeerivatest kihtidest. Käesoleva töö autor on olnud osa ABC4HEDA arendusmeeskonnast 2021. aasta algusest saati. Hetkese seisuga on tarkvara lähtekoodis 120 tuhat rida koodi, millest umbes 55 tuhat rida moodustavad automaatsed ühiktestid ja vastuvõtutestid. Selle tulemusena on lähtekoodi testidega kaetavus lähedal 100%-le. Meeskonnas kasutatav arendusmetoodika põhineb ekstreemprogrammeerimisel [15] ning TDD-l [16]. ABC4HEDA on arendatud kasutades värskeimat .NET versiooni ja C# programmeerimiskeelt [17]. Objekt-relatsioonilise teisendajana on kasutusel Entity Framework [18]. Kasutajaliides on arendatud kasutades ASP.NET raamistikku. Lisaks eelmainitud repositooriumi mustrile on ABC4HEDA arenduses kasutatud veel mitmeid tarkvara disainimustreid nagu *Item Description* muster [19], *Abstract Factory* muster [20], *Pure Fabrication* muster [20] ning suurel hulgal teisi.

4 HL7 FHIR

Fast Healthcare Interoperability Resources (FHIR) on HL7 perekonda kuuluv terviseandmete koostalitlusstandard, mis defineerib lihtsad, *RESTful* põhimõtetel koostatud, struktuurid. FHIR spetsifikatsioon koosneb laiendatavatest ja kohandatavatest ressurssidest JSON või XML kujul. FHIR'i struktuur põhineb kaasaegsetes veebirakendustes kasutatavatel andmevahetusstandarditel, mistõttu on see paljude arendajate jaoks intuitiivne ja lihtsasti omandatav. [21], [22], [23] FHIR'i pidevalt kasvav populaarsus on põhiline faktor, mis suunas artikli autoreid just seda standardit uurima.

FHIR on pidevas arenduses. Üks arendustsükkel kestab umbes 18-24 kuud, peale mida avaldatakse spetsifikatsiooni uus versioon. Igale versioonile vastab enda unikaalne versiooninumber. [3] Artiklis võeti vaatluse alla kaks ressurssi FHIR R4 mudelist ning kasutati eksklusiivselt JSON formaati. Nendeks olid $Organization$ ja $Specimen$. $Organization$ ressurssi kasutatakse selleks, et kirjeldada mingisuguse ühise eesmärgi saavutamiseks ametlikult või mitteametlikult moodustatud inimgruppi. $Specimen$ ressurss kirjeldab mistahes materjalist proovi, mida analüüsitakse. $Specimen$ ressurss katab ka analüüsimise ja proovi kogumise toimingud.

Konkreetsete ressursside valikul võeti arvesse seda, et ressursid võiksid olla võimalikult erinevad. $Organization$ ressurss paistis silma sellega, et see sisaldas palju keerukaid FHIR andmetüüpe, mis tegi selle teisendamise väljakutsuvaks. $Specimen$ ressurss kirjeldab kliinilist sisu, mis on oluline äriarhetüüpidel põhinevate metamudelite valideerimisel terviseandmete ühtse alusmudelina.

5 Integratsiooni disain ja implementatsioon

5.1 Mirth Connect

NextGen Connect (Mirth Connect) on vabavaraline tervishoiu-integratsioonimootor. Mirth Connecti tarkvarapakett koosneb serverist ja administraatori töölauarakendusest. Administraatori rakendust kasutatakse sõnumivahetuskanalite arendamiseks ja haldamiseks. Mirth Connecti põhielemendiks on kanal (*channel*), mis ühendab sõnumi saatjat ja saajat. Kanali arhitektuur võimaldab sõnumi töötlemist jagada väiksemateks iseseisvateks sammudeks. Kanal koosneb allika pistikust (*source connector*), sihtkoha pistikust (*destination connector*) ja vastuse pistikust (*response connector*). Sõnum liigub allikast sihtkohta ning saab sihtkohalt vastuse, mis saadetakse tagasi allikale. Mitut kanalit saab omavahel järjestikku ühendada, et lahendada keerukamate nõuetega integratsiooniprobleeme. [24]

Igas Mirth Connecti kanali pistikus on võimalik implementeerida eraldiseisvad filtrid ja teisendajad (*transformer*). Teisendajaid kasutatakse kahe formaadi vahelise teisendamise implementeerimiseks. Artikli käigus arendatud prototüübi seisukohalt on teisendajad kõige olulisemad lülid – kõige enam aega kulus just teisendajate skriptide kirjutamiseks.

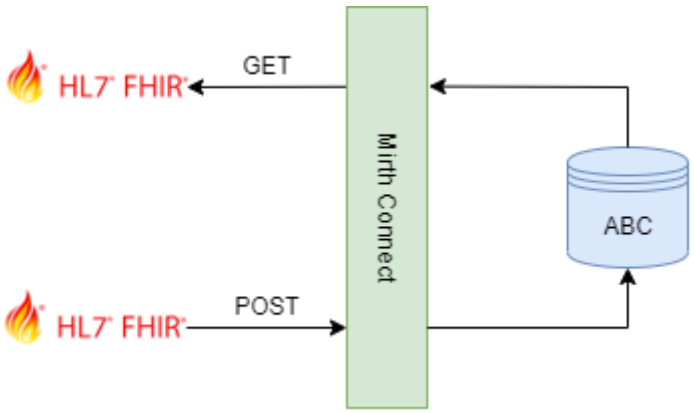
Teisendajate implementeerimiseks pakub Mirth Connect mitut võimalust. Üks võimalustest on graafiline *low-code* liides sõnumiformaatide segmentide omavaheliseks sobitamiseks, mille jaoks on vaja sissetuleva ja väljamineva sõnumi malle. Selline viis sõnumivahetuse arendamiseks sobiks eelkõige analüütikule, kellel pole programmeerija tausta. Artikli autorite arvates pole aga FHIR formaadi tõlkimiseks seda graafilist liidest võimalik kasutada. FHIR sõnumite kuju on oma olemuselt laiendatav ja kohandatav, mis tähendab, et iga süsteem, mis FHIR'it kasutab, implementeerib seda oma äranägemise järgi. FHIR sõnumite kuju pole piisavalt järjepidev, et saaks kasutada mallidepõhist lähenemist. Teiseks võimaluseks on implementeerida sõnumit teisendav loogika koodis, mida ka selle töö raames kasutati.

Mirth Connecti jaoks on ka olemas ametlik FHIR laiendus. Huvitaval kombel ei paistnud see lahendavat probleemi, et FHIR'iga on mõistlik ümber käia ainult koodi kasutades.

Graafiline FHIR ressursiehitamise liides on küll olemas, kuid sellega on võimalik ainult ressursse kokku panna. Ressursse lahti lammutada sellega võimalik ei ole. Ühtlasi on liideses keeruline töökindlalt itereerida üle kollektsioonide, mida FHIR sõnumites esineb küllaltki tihti.

5.2 Integratsiooni arhitektuur

Esialgse prototüübi arendamisel jäeti skoop kokkuvõtlikuks. Eesmärgiks oli arendada üks HTTP otspunkt, mille kaudu saab FHIR sõnumi abil ABC4HEDA andmebaasi andmeid sisestada ning teine otspunkt, mille kaudu saab ABC4HEDA andmebaasilt andmeid pärida nii, et vastu saaks sõnumi FHIR kujul (vaata Joonis 1).



Joonis 1. Prototüübi kavand.

Tasub ka mainida seda, et artikli kirjutamise hetkel puudus ABC4HEDA tarkvaral igasugune veebiliides. Seetõttu pidi sõnumikanal suhtlema otse ABC4HEDA andmebaasiga. Prototüübi kontekstis ei erine see väga *RESTful* API-ga suhtlemisest, sest *RESTful* põhimõtete järgi vastab tihti igale andmebaasi tabelile üks API ressurss.

Kummagi arendatava otspunkti jaoks disainiti enne implementeerima hakkamist arhitektuur, lähtudes „*Enterprise Integration Patterns*“ raamatus esitatud mustritest. Kasutust leidsid *Content base router* muster, *Splitter* muster, *Message Router* muster ja *Aggregator* muster. [4] Arhitektuuri joonised koos seletustega on esitatud artikli viiendas peatükis.

6 FHIR ressursside ja äriarhetüüpide vaheline teisendamine

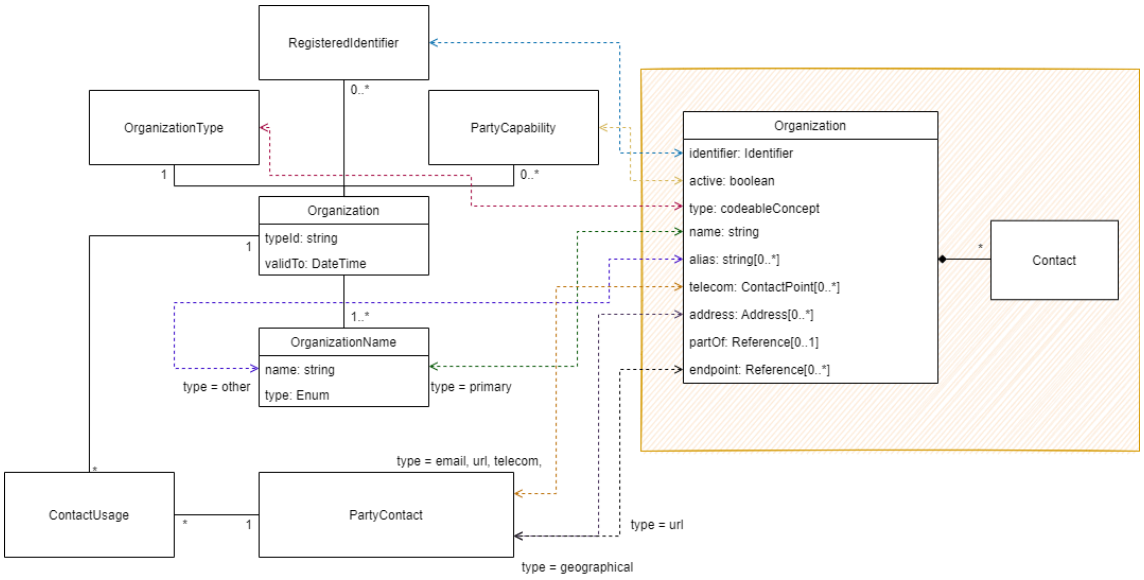
Semantiline teisendamine FHIR ressursside ja äriarhetüüpide vahel osutus väga heaks viisiks metamudelite ja FHIR spetsifikatsiooni semantilise sobivuse valideerimiseks. Teisendamise implementeerimise protsessi käigus tulid hästi välja kas ja millised puudujäägid metamudelitel FHIR mudeli suhtes on.

FHIR ressursi segmendile vastava äriarhetüübi leidmiseks tuleb lähtuda konseptsioonist, mida see kirjeldab. Siinkohal toetuti ametlikule FHIR dokumentatsioonile [3]. Dokumentatsioonis esitatud kirjeldust üldistades oli võimalik leida vastav äriarhetüüp või selle osa, toetudes Arlow ja Neustadt'i tööle [10]. Neid samme järgides oli võimalik koostada teisendamist illustreerivad UML diagrammid, mis täies mahus koos seletustega on välja toodud artikli kuuendas peatükis.

Järgnevalt on kirjeldatud mõttekäigud FHIR ressurssidele vastavate äriarhetüüpide leidmiseks.

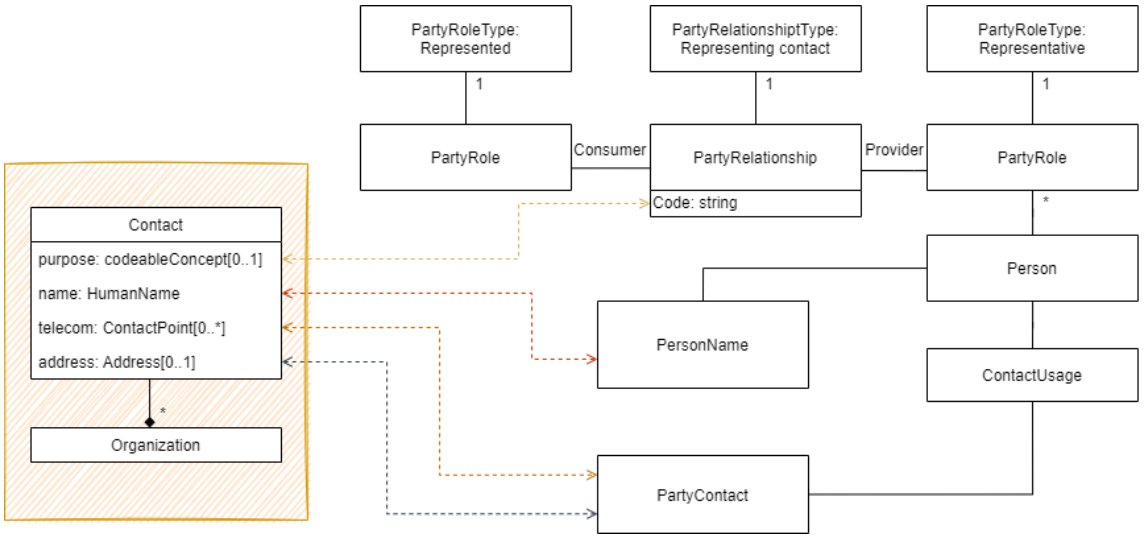
6.1 *Organization* FHIR ressurss

$Organization$ ressursile vastab äriarhetüüpidest $Organization$ arhetüüp (üks $Party$ arhetüübi alamtüüp). $Organization$ ressursi väljadele oli $Party$ arhetüübimustri seast vastavate teisenduste leidmine küllaltki elementaarne. $Organization$ ressursi nimede ja aliaste väljad teisenduvad mõlemad $PartyName$ arhetüübiks. Ressursi $identifier$ väli teisendub $RegisteredIdentifier$ arhetüübiks. $Organization$ ressursi $telecom$, $address$ ja $endpoint$ väljad teisenduvad kõik $PartyContact$ arhetüüpideks. $Organization$ ressursil on tõeväärtusega väli $active$. See teisendub $PartyCapability$ arhetüübiks, sest organisatsiooni aktiivne olemine on eeldus kõikide teiste võimekuste olemasoluks. Seni välja toodud teisendusi illustreerib Joonis 2. Ressursi väli $partOf$ sisaldab viidet organisatsiooni vanemstruktuurile (näiteks emaettevõte). Äriarhetüüpide kaudu modelleeritakse see seos kasutades $PartyRelationship$ arhetüüpimustrit.



Joonis 2. *Organization* ressursi teisendusdiagramm.

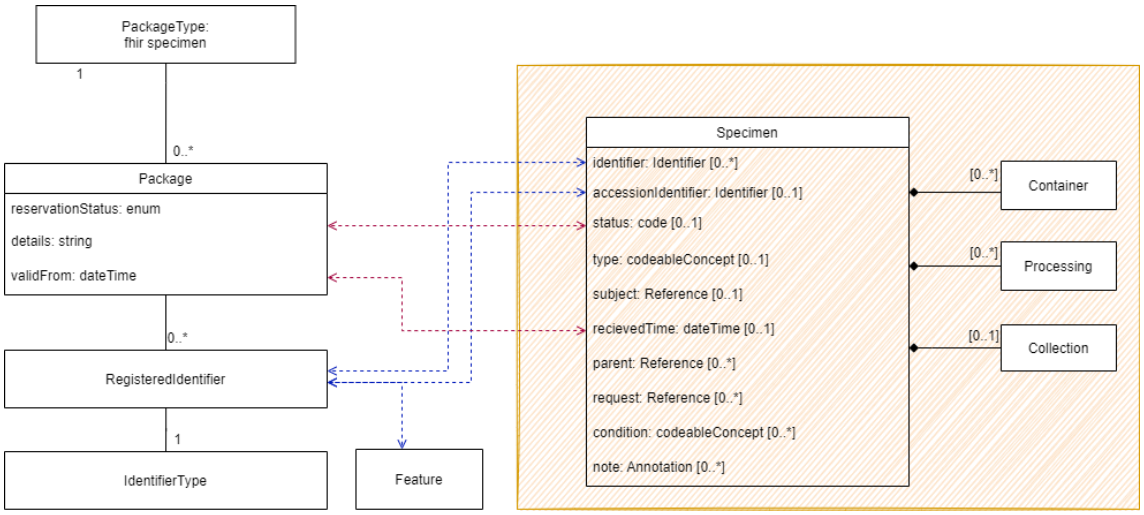
Organization ressurss sisaldab ka *Contact* komponenti, mis kirjeldab inimestest osapooli, kes seda organisatsiooni esindavad. *Contact* komponent teisendub *Person* arhetüübiks (samuti *Party* arhetüübi alamtüüp) ning komponendi *name*, *address* ja *telecom* väljad teisenduvad täpselt samamoodi nagu *Organization* ressursi enda puhul. Seoseid *Organization* ressursi enda ja *Contact* komponentides kirjeldatud inimeste vahel modelleeritakse *PartyRelationship* arhetüübimustriga. (vaata Joonis 3)



Joonis 3. *Contact* komponendi teisendusdiagramm.

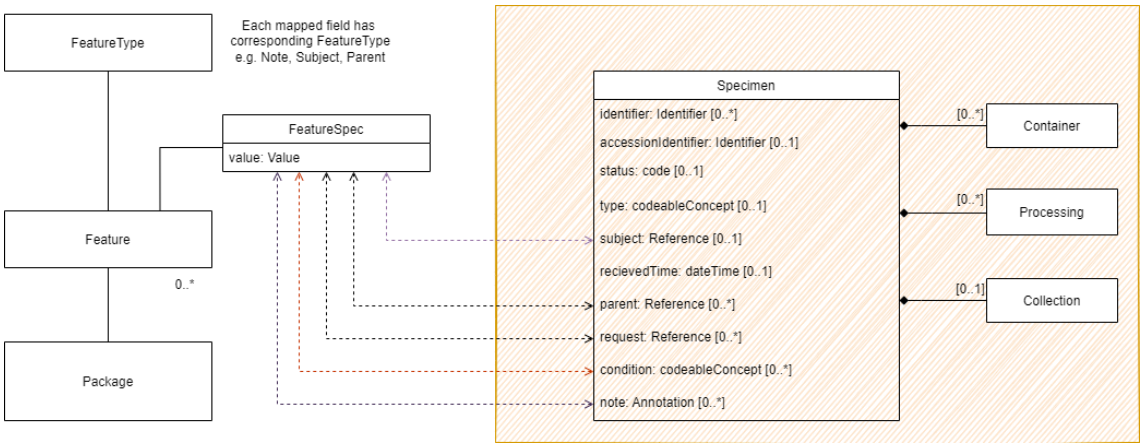
6.2 *Specimen* FHIR ressurss

Specimen ressursi teisendus äriarhetüüpideks on *Organization* ressursiga võrreldes veidi keerukam. Kuna *Specimen* ressurss on analüüsitav proov, võib öelda, et *Specimen* on mingisugune asi, mida tervishoiasutus kasutab tervishoiuteenuse pakkumiseks. Seetõttu teisendub *Specimen* ressurss *Specimen* tüüpi *Product* arhetüübiks. Täpsemalt teisendub *Specimen* ressurss *Package* arhetüübiks (vaata Joonis 4), mis sisaldab *Container* arhetüüpide kollektsiooni, kusjuures *Specimen* ressursi *Container* komponendid vastavad nendele *Container* arhetüüpidele. Iga *Container* arhetüüp sisaldab omakorda mingisugust kogust *Specimen* ressursi väljas *type* täpsustatud proovi. Kõik *Container* arhetüübid sisaldavad sama tüüpi proovi, kuid proovide kogused on erinevad.

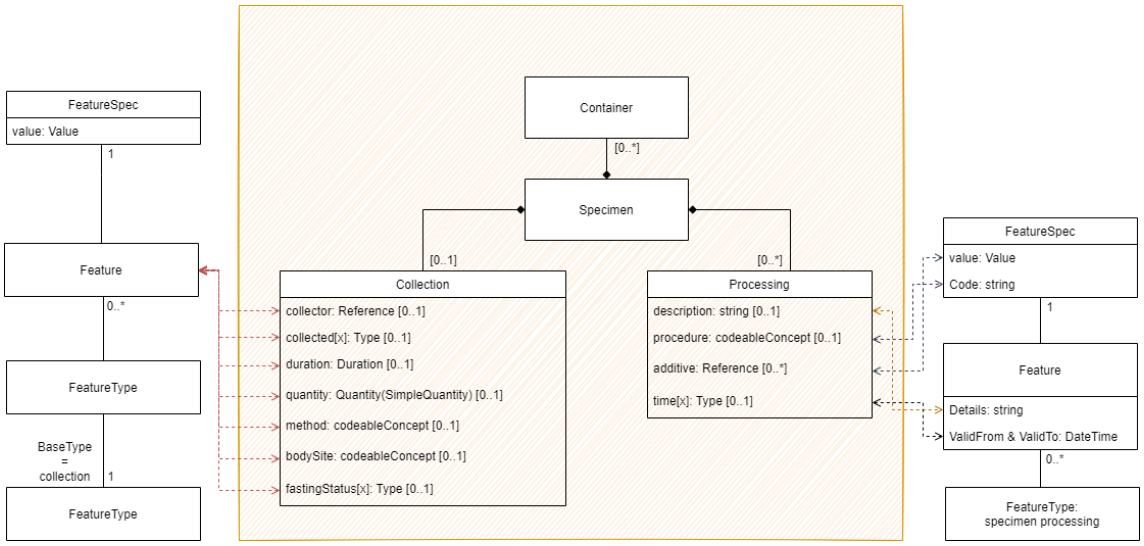


Joonis 4. *Specimen* ressursi teisendusdiagramm.

Suur hulk *Specimen* ressursi väljasid teisenduvad *Feature* arhetüübiks. Nendeks on *subject*, *parent*, *request*, *condition* ja *note* (vaata Joonis 5). *Feature* arhetüübiks teisendub ka ressursi *Container* komponendi *capacity* väli. Ka *Specimen* ressursi *Collection* ja *Processing* komponendid teisenduvad *Feature* arhetüüpideks – *Collection* komponent on *Feature* arhetüüpide kogum ja *Processing* komponendid teisenduvad igaüks üheks *Feature* arhetüübiks (vaata Joonis 6).



Joonis 5. *Specimen* ressursi väljade teisendumine *Feature* arhetüüpideks.



Joonis 6. Collection ja Processing komponentide teisendumine *Feature* arhetüüpideks.

7 Tulemused

7.1 Semantilise koostalitluse hinnang

Esialgsed tulemused näitasid, et suure tõenäosusega on terve HL7 FHIR spetsifikatsioon äriarhetüüpidel põhinevate metamudelitega koostaltitlusvõimeline, kuigi enne enamuste FHIR ressursside analüüsi ei saa seda kindlalt väita. Juba kaht ressurssi kasutades tuli esile mõningaid kitsaskohti, mis metamudelites esineda võivad. Näiteks tekkis vajadus võimaldada seost $RegisteredIdentifier$ arhetüübiga lisaks $Party$ arhetüübile ka $Product$ arhetüübiga.

Teisendamise implementeerimisel esinenud probleemid tulenesid suuresti sellest, et FHIR kasutab keerukaid andmetüüpe nagu *Identifier*, *Reference* ja *CodableConcept*, samas kui metamudelid kasutavad sama informatsiooni kirjeldamiseks objekt-relatsioonilisi mustreid. Keeruka andmetüübi sobitamisel metamudelitesse tuleb lähtuda sellest, milline arhetüüp vastab ressursi väljale, mis seda andmetüüpi sisaldab. Sõltuvalt ressursist tulenevast kontekstist võib sama FHIR andmetüüp võtta erinevate arhetüüpide kuju. Näiteks $Organization$ ressursi puhul vastab $type$ väljas olev $CodableConcept$ andmetüüp $OrganizationType$ arhetüübile. Samas $Specimen$ ressursi $type$ väljas olev $CodableConcept$ vastab $PackageType$ arhetüübile ning $condition$ väljas olev $CodableConcept$ vastab üldse $Feature$ arhetüübile.

Artikli üheksandas peatükis on näited FHIR sõnumitest. *Listing 1* on ametlik $Specimen$ ressursi näide HL7 poolt. Kui see sõnum saata juurutatud sõnumikanalisse, teisendatakse see sõnum äriarhetüüpide kujule ja sisestatakse ABC4HEDA andmebaasi. Süsteemist pärimisel küsitakse ABC4HEDA andmebaasist vajalikud kirjed ja konstrueeritakse FHIR ressurss. Vastuseks saadakse sõnum, mida kujutab *Listing 2*. Erinevused sõnumite vahel on artiklis analüüsitud ja põhjendatud. Kaduma läinud sõnumisegmendid on triviaalsed või süsteemipõhised. See tähendab, et integratsiooniga ei kaasne andmekadu.

7.2 Mirth Connect integratsioonimootorina

Mirth Connect osutus väga võimsaks tööriistaks. Selle põhilisteks eelisteks olid integreeritavate süsteemide ja integratsiooni enda lahutatus ning Mirth Connecti laiendatavus. Integratsioonimootoris on võimalik kirjutada enda korduvkasutatavaid koodi mooduleid ning samuti on võimalus kasutada välist Java koodi *.jar* laiendiga failide kaudu. Artikli autorite hinnangul on aga Mirth Connect eelkõige tööriist tarkvaraarendajale. $Organization$ ressursi integratsiooni implementatsioon nõudis üle 600 rea koodi ning $Specimen$ ressursi integratsiooni implementatsioon nõudis natuke alla 800 rea. See pole realistlikult analüütikule jõukohane.

Lisaks eeldab Mirth Connect'is mõistlik arendamine korralike väliste tööriistade olemasolu. Eelmainitud HAPI FHIR API kasutamine tegi kogu sõnumikanali ehitamise FHIR poole pealt oluliselt lihtsamaks. Mingisugusest metamudelite poolsest tööriistast oli tunda suurt puudust. Samal moel nagu prototüüp arendati, poleks terve FHIR spetsifikatsiooni integreerimine mingil juhul mõeldav.

Artikli autorite hinnangul tooks ABC4HEDA-le suurt kasu GraphQL API arendamine. GraphQL abil muutuks integratsiooni kavandamine lihtsamaks, sest igale FHIR ressursile tuleks disainida ainult üks vastav päring. Samuti nõuaks integratsiooni implementatsioon sellisel juhul vähem koodi.

Lisaks vajaks ABC4HEDA veebiteenust meditsiiniliste klassifikaatorite haldamiseks. Föderatsioonil põhineva koostalitluse lahenduse kasutamisel ei saa eeldada, et mingisuguse standardi kood (näiteks LOINC [25] kood) on igas olukorras unikaalne identifikaator. Selleks, et integratsioonikanalis arhetüüpide vahel korrektseid seoseid luua, on vaja võimalust küsida ja valideerida, mis on mingisuguse klassifikaatori identifikaator ABC4HEDA andmebaasis. Näiteks peaks olema võimalik saata teenusele päring sisuga SNOMED [26] kood „119364003" uusim versioon kuupäeva 01.05.2022 seisuga, vastusena tagastataks konkreetse kirje identifikaator, et luua valideeritud seos arhetüübi ja klassifikaatori vahel.

Artikli autorite hinnangul oleks suuremas mastaabis integratsiooni arendamise korral võimalik veel lisaks suur osa koodi eraldada kordukasutatavasse moodulisse. Prototüübi mahu ja ressursside valiku tõttu korduvkasutatavaid elemente piisavalt ei esinenud. Samas on alust arvata, et terve standardi sõnumikanali arendamisel tekib selliseid

olukordi piisavalt. Korduvkasutatava koodi mooduli eraldamisel võiks ühe ressursi integreerimiseks kirjutatud koodi maht oluliselt väheneda.

7.3 Töö võimalikud edasiarendused

Artikli autorite arvates on mitu suunda, kuhu tehtud töö põhjal edasi võiks areneda. Lisaks FHIR'ile on tulevikus vaja uurida ka teisi terviseandmeid käsitlevaid standardeid nagu HL7 v2.x, HL7 v3.x ja openEHR. Artikli näitel oleks võimalik sarnase metoodikaga uurida nende spetsifikatsioonide koostalitluse olemust ja sõnumikanalite arendamise võimalusi.

Samuti oleks kasulik panustada äriarhetüüpide transformatsioone kirjeldava metakeele välja töötamisse. Metakeel oleks analüütikule mõeldud tööriist, mis peaks võimaldama implementatsiooni 600 ja 800 koodirida vastavalt 60 ja 80 rea koodiga kirjutada. Sellise tööriista vastu on meditsiinitarkvara turul suur huvi. Tuleks ka arendada prototüübid välja pakutud tööriistadest, mis toetaksid ABC4HEDA tööd. Näiteks võiks ehitada mingisuguse minimaalse võimaliku skoobiga GraphQL API ning artiklis ehitatud integratsioon ümber kirjutada, et siis tulemusi võrrelda ja lahendus valideerida.

Lisaks eelnevale tuleb FHIR spetsifikatsiooni mudelid äriarhetüüpidel põhinevates metamudelites formaliseerida. Kui käesolev töö keskendus konkreetsetele andmetele ja nende vahetusele, siis teine osa tööst on kirja panna meditsiiniteadmised nende andmete kohta. Formaliseerimine hõlmab näiteks FHIR klassifikaatorite, tüüpide ja väärtushulkade esitamist äriarhetüüpide kujul.

8 Kokkuvõte

Tehtud töö eesmärgiks oli hinnata ja katsetada äriarhetüüpidel põhinevate metamudelite semantilist koostalitlust HL7 FHIR terviseandmete sõnumivahetusprotokolliga. Üldise järeldusena võib välja tuua selle, et FHIR'i koostalitlus äriarhetüüpidel põhinevate metamudelitega on võimalik ja implementeeritav. Integratsioonikanali arendamise lihtsustamiseks oleks vajalik arendada välja toetavad tööriistad nagu metakeel, GraphQL liides ning meditsiiniteadmiste ja -klassifikaatorite veebiteenus. Lisaks oli eesmärgiks hinnata Mirth Connect integratsioonimootorit FHIR'iga integratsiooni implementeerimisel. Mirth Connect on võimas tööriist sõnumikanalite arendamiseks, kuigi eelkõige on see mõeldud programmeerimisvõimekusega inimesele. Artikli autorid usuvad, et tehtud töö on oluline panus tervishoiusüsteemide semantilise föderatsioonil põhineva koostalitluse suunas.

Kasutatud kirjandus

[1] HEDA, [Võrgumaterjal]. Available: https://cs.ttu.ee/events/heda-2022/. [Kasutatud 17 mai 2022].

[2] D. Chen, G. Doumeingts ja F. Vernadat, „Architectures for enterprise integration and interoperability: Past, present and future," *Computers in Industry,* nr 0166-3615, 2008.

[3] HL7, [Võrgumaterjal]. Available: http://hl7.org/fhir/. [Kasutatud 6 aprill 2022].

[4] G. Hohpe ja B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] HAPI FHIR, [Võrgumaterjal]. Available: https://hapifhir.io/hapi-fhir/docs/. [Kasutatud 6 aprill 2022].

[6] R. C. Martin ja J. O. Coplien, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2009.

[7] MirthCorp, [Võrgumaterjal]. Available: http://javadocs.mirthcorp.com/connect/3.6.0/user-api/. [Kasutatud 6 aprill 2022].

[8] K. M. Raavel, T. Sõerd ja R. Randmaa, „ABC4HEDA: Centry ABC projekti dokumentatsioon," 2021.

[9] Postman, Inc., [Võrgumaterjal]. Available: https://learning.postman.com/docs/. [Kasutatud 6 aprill 2022].

[10] J. Arlow ja I. Neustadt, Enterprise patterns and MDA: building better software with archetype patterns and UML, Addison-Wesley, 2003.

[11] J. Zachman, „A Framework for Information Systems Architecture," *IBM Systems Journal,* 1999.

[12] G. Piho, *Archetypes based techniques for development of domains, requirements and software: towards LIMS software factory,* Tallinn Technical University Press, 2011.

[13] C. Atkinson, „Single Underlying Models for Projectional, Multi-View Environments," 2019.

[14] E. Evans, Domain-Driven Design, Addison-Wesley, 2003.

[15] K. Beck, Extreme Programming Explained, 2nd edition, Addison-Wesley, 2004.

[16] K. Beck, Test-Driven Development by Example, Addison-Wesley Professional, 2002.

[17] Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/. [Kasutatud 6 aprill 2022].

[18] Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/. [Kasutatud 6 aprill 2022].

[19] P. Coad, „Object-oriented patterns," *Communications of the ACM,* 1992.

[20] GoF, Design Patterns: Elements of Reusable Object-Oriented Software, 1994.

[21] S. Nizamov, Unofficial Developer’s Guide to FHIR on Mirth Connect, 2016.

[22] F. O. a. R. Snelick, Healthcare Interoperability Standards Compliance Handbook, Springer, 2016.

[23] M. Braunstein, Health Informatics on FHIR: How HL7's New API is Transforming Healthcare, Springer, 2018.

[24] S. Nizamov, Unofficial Mirth Connect v3.12 Developer’s Guide, 2021.

[25] Regenstrief Institute, Inc., [Võrgumaterjal]. Available: https://loinc.org/. [Kasutatud 17 mai 2022].

[26] SNOMED International, [Võrgumaterjal]. Available: https://www.snomed.org/. [Kasutatud 17 mai 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Rainer Randmaa

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Äriarhetüüpidel põhinevate metamudelite ja FHIR ressursside semantilise koostalitluse hindamine“, mille juhendajad on Igor Bossenko ja Gunnar Piho
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

18.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Artikkel HEDA 2022 konverentsile

Evaluating business meta-models for semantic interoperability with FHIR resources

Rainer Randmaa[2], Igor Bossenko[1], Toomas Klementi[1], Gunnar Piho[2] and Peeter Ross[1]

[1]Department of Health Technologies, TalTech, Akadeemia Str 15A, 12618 Tallinn, Estonia
[2]Department of Software Science, TalTech, Akadeemia Str 15A, 12618 Tallinn, Estonia

Abstract

Information systems interoperability in healthcare has proved itself a difficult challenge due to the semantic heterogeneity of models, data, messages, etc. So far, the attempts to solve this problem have involved a unified approach through the development and utilization of common standards. However, this approach also has issues in semantic heterogeneity because in the case the standardized data models and exchange protocols have been developed by different vendors, they normally tend to not be interoperable. A federated interoperability approach, where models can be dynamically specified at runtime rather than being pre-determined, might be a useful supplement to the unified interoperability approach. In this workshop paper, we propose executable business meta-models, that according to our understanding, can be utilized as a shared ontology to achieve federated interoperability. We evaluate these meta-models by illustrating semantic interoperability with FHIR resources, the building blocks of the healthcare interoperability standard developed by HL7. For the evaluation, a distributed test environment is set up, to illustrate how the data is exchanged and transformed, back and forth. Here NextGen Connect (Mirth Connect) is used as a mapping engine between FHIR and the meta-models. We believe our work is an important contribution towards seamless no-code/low-code federated interoperability of health information systems where different systems can exchange medical data in a semantically coherent way by using different protocols, data models and their versions, such as FHIR, HL7 v2.x, openEHR, etc.

Keywords – FHIR, Mirth Connect, Interoperability, EHR, Meta-model

1. Introduction

The complexity of data and information systems in the field of healthcare informatics create many challenges for using health data for advancing medicine, science, population health and the economy. Data exchange between different healthcare systems is specifically challenging because of the semantic heterogenity of data models. Different vendors in healthcare use different standards and specifications as pre-determined models when developing their applications, making the data incompatible between systems. A federated interoperability approach, where models can be dynamically specified at runtime rather than being pre-determined, might be a useful supplement to the unified interoperability approach, meaning there is a common meta-level structure across constituent models.[1] We believe that meta-models based on business archetypes can be utilized as a shared ontology to achieve federated interoperability. A system using these meta-models would allow specifying models during runtime and allow transformations between them. A means to exchange health data without affecting its quality is critical to the secondary use of health data.

These meta-models need to be evaluated and tested with existing healthcare information models. HL7 FHIR is one of the more modern standardized communication protocols in healthcare, constantly gaining in popularity. This paper aims to illustrate and understand the process of validating these meta-models as well as integrating systems developed using the FHIR models by implementing proof-of-concept data exchange channels between FHIR and the meta-models.

2. Methodology

For developing a proof-of-concept integration between HL7 FHIR and business archetypes based meta-models, the integration architecture will be designed using enterprise integration patterns and principles[2]. Semantic mapping between FHIR models and the meta-models will be performed and displayed using UML diagrams. NextGen Connect (Mirth Connect) will be used as an integration engine to implement the architecture and mapping by aggregating Mirth Connect channels to reach the desired result. Channels in Mirth connect are developed using a mixture of JavaScript, Java and SQL. The mapping implementation also makes use of the HAPI FHIR API[3], an open source HL7 FHIR API written in Java.

HEDA-2022: The International Health Data Workshop, June 19-24, 2022, Bergen, Norway
rainer.randmaa@taltech.ee (R. Randmaa); igor.bossenko@taltech.ee (I. Bossenko); toomas.klementi@taltech.ee (T. Klementi); gunnar.piho@taltech.ee (G. Piho); peeter.ross@taltech.ee (P. Ross)
taltech.ee/en/emed-lab (R. Randmaa)
0000-0000-0000-0000 (R. Randmaa); 0000-0003-1163-5522 (I. Bossenko); 0000-0002-8260-526X (T. Klementi); 0000-0003-4488-3389 (G. Piho); 0000-0003-1072-7249 (P. Ross)

With the help of HAPI, FHIR messages can be serialized into Java objects, which makes the mapping code much safer and easier to write.[4] To test the integration, an instance of the meta-models based software, ABC4HEDA, will be deployed on a virtual machine with a Microsoft SQL Server as the persistence layer. Mirth Connect will be deployed on a separate virtual machine. This will be done using Microsoft Azure cloud services and will mimic a distributed environment. A set of example resources is provided by HL7 on the official FHIR documentation website[5], which will be used as messages when testing the integration implementation. Messages will be sent using Postman[6]. Critical retrospective analysis will then be performed on the process of developing this proof-of-concept.

3. Archetype patterns based meta-model

The proposed business meta-models for semantic interoperability are based on business archetype patterns[7] in combination with the Zachman Framework[8] applied to health data.

One of the archetype patterns we will need to utilize in the context of this paper is the *Party* archetype pattern. The central part of this pattern is the *Party* archetype 1. The *Party* archetype pattern is used to describe different

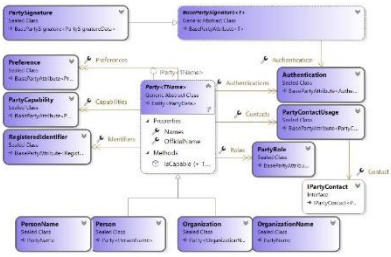


Figure 1: The party archetype

parties involved in business activities, such as a person or an organization. Parties have many attributes, such as names, contacts, identifiers, capabilities and preferences. The *Party* archetype pattern is complemented by the *PartyRelationship* archetype pattern, which allows to describe how these parties relate to one another, a client-supplier relationship for example.

Another archetype pattern we will encounter in this paper is the *Product* archetype pattern, at the centre of which is the *Product* archetype, illustrated in figure 2. The figure also illustrates the use of the *Item Description* pattern[9], which is used throughout many places in the meta-models. The *Product* archetype pattern is used to de-

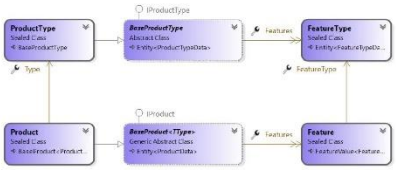


Figure 2: The product archetype

scribe any product or service a business provides, or uses to provide another product or service. In other words, a product could be thought of as any item or service used in business processes. The *ProductType* archetype represents the generic information about a product, and the *Product* is a specific physical instance of the type. When you go into a shop to buy a product, the *ProductType* represents the information the salesperson gives you verbally or in a catalog or leaflet, and the *Product* represents the actual thing you walk out the door with. In the context of healthcare, a product could be a blood sample in storage, medicine ready to be administered or a booked consultation.

In addition to the aforementioned archetype patterns, the meta-models also contain the *Order*, *Inventory*, *Rule*, *Money* and *Quantity* archetype patterns with the inclusion of the *Process* archetype pattern developed by Gunnar Piho by abstracting the *Customer Relationship Management* archetype pattern[10]. We believe that these archetype patterns can be used as a Single Underlying Model[11] (SUM) for healthcare data models. The important aspect to highlight about these meta-models is that data and the knowledge about the data is decoupled.

The software implementing these models is ABC4HEDA. It is written using the latest .NET framework. In addition to executable domain models, data models using pure POCOs (Plain Old CLR Objects) and an infrastructure layer, ABC4HEDA also contains an administrator UI component developed with ASP.NET. Out of the 120K lines of source code, about 45% are automated unit tests and UI acceptance tests, resulting in high reliability and test coverage of the code base.

4. HL7 FHIR

Fast Healthcare Interoperability Resources (FHIR) is a healthcare interoperability standard in the HL7 family that defines simple structures for a RESTful data exchange protocol. The FHIR model consists of modular

resources based on simple XML or JSON structures that can be extended beyond the base model in a standardized manner.[12, 13] As FHIR seems to be growing in popularity, the standard's semantic interoperability with the proposed meta-model needs to be validated and an integration solution with systems based on FHIR needs to be researched and developed.

FHIR is an evolving standard. New versions of FHIR are published on a release cycle of approximately 18-24 months. Each new release is assigned a unique version number. The proof-of-concept built in this paper uses two resources from the FHIR Release 4 (R4) model, the *Organization* and the *Specimen* resource. The *Organization* resource is used to describe a formally or informally recognized grouping of people or organizations formed for the purpose of achieving some form of collective action.[5] The *Specimen* resource is a clinical concept. It is used to describe any material sample to be used for analysis. The *Specimen* resource covers substances used for diagnostic and environmental testing. The focus of the *Specimen* resource is the process for gathering, maintaining and processing the specimen as well as where the specimen originated from.[5]

5. Integration using Mirth Connect

NextGen Connect (Mirth Connect) is an open source healthcare integration engine, with some enterprise extensions available that require a subscription. The integration unit between a sending app and a receiving app in Mirth Connect is called a channel. It can be seen as one-to-many abstract unidirectional pipes to decouple components from each other to transfer healthcare data between two or more applications. The channel architecture implemented in Mirth Connect can divide a large message processing task into a sequence of smaller independent steps.[14] A channel consists of a *Source connector*, *Destination connector* and a *Response connector*. Each of those has their own set of configurable filters and transformers. Channels can also be used in conjunction for integration problems with complex requirements.

Mirth Connect offers a few different ways to implement mappings between two message formats. The low-code version is using message templates. This would be the preferred option for a non-developer individual, but to our understanding, there is no straight forward way of obtaining consistent templates for FHIR messages. The reason for this is the extendibility and flexibility of the FHIR format. Because of this, the majority of the channels must be implemented using custom scripts written by a programmer. It is interesting to note, that Mirth Connect does have an official FHIR extension, but to our knowledge it does not fully address this problem of accessibility. A graphical FHIR resource builder is provided,

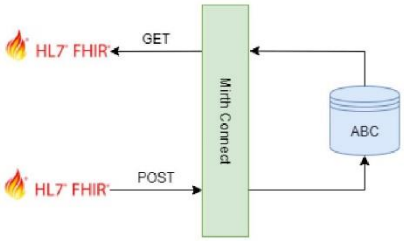


Figure 3: FHIR interface for ABC4HEDA

but it can only be used to construct resources and not the other way around. Frankly, it is also quite clunky to use when iterating through collections. The official extension also provides a FHIR endpoint and FHIR resource validation, though, which can be very useful.

As the initial proof-of-concept, the integration will be as simple as possible. Our system will have two endpoints, one for inserting data in FHIR format and one for requesting data in FHIR format 3. A notable constraint with developing this integration is that the ABC4HEDA software currently has no web API, so the database will need to be queried directly. In the context of a prototype though, this is not much different than communicating with a RESTful API where each database table is often allocated its own endpoint anyway.

5.1. From FHIR to meta-models

For inserting data in the FHIR format to the ABC4HEDA database, an endpoint should be able to listen for an incoming FHIR message and then split it by resource using the *Splitter* pattern[2]. Each resource gets routed to its corresponding transformer channel, using the *Content Based Router* pattern[2]. Each transformer channel is responsible for mapping the incoming data to business archetypes and inserting it into the database. The integration architecture for inserting data is illustrated in figure 4. A resource transformer channel will deserialize the FHIR message into a HAPI FHIR[3] Java class instance and then break the resource down to business archetypes, once again using the *Splitter* pattern [2]. Each step of the transformer channel generates an SQL INSERT statement for each instance of the corresponding business archetype. These SQL statements are then executed as a transaction. This is illustrated in figure 5

5.2. From meta-models to FHIR

For requesting a resource from ABC4HEDA in FHIR, we need to have another FHIR endpoint listening for a GET

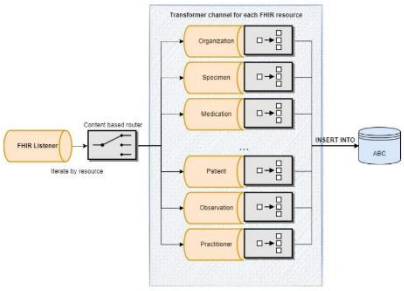


Figure 4: FHIR to Mirth to ABC4HEDA

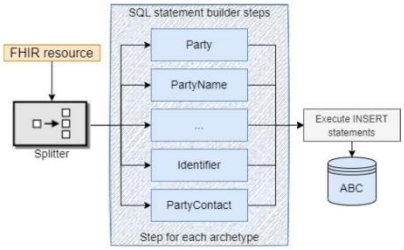


Figure 5: FHIR resource to business archetypes transformer channel

request with the resource type and ID specified. This ID is then routed to the correct FHIR resource builder channel using the *Message Router* pattern [2]. The resource builder channel then queries the database, builds the FHIR resource and sends it back as a response to the request. This architecture is illustrated in figure 6. A

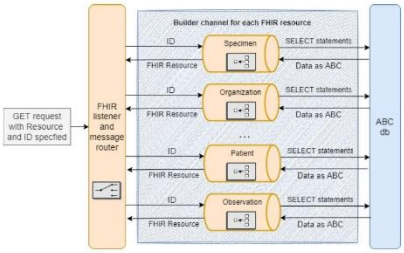


Figure 6: ABC4HEDA to Mirth to FHIR

resource builder channel creates an empty HAPI FHIR[3] Java instance of the resource. Based on the received ID, each builder step makes a series of SELECT queries to populate the FHIR resource segments, resembling the *Aggregator* pattern [2]. The Java object is then serialized into JSON and sent as a response. This is illustrated in figure 7.

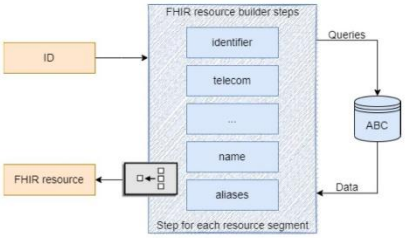


Figure 7: Business archetypes to FHIR resource builder channel

6. Mapping FHIR resources and business archetypes

Semantic mapping between the FHIR R4 model and business archetypes is a way to assess and validate the meta-models' compatibility with the FHIR specification. This process gives insight into the how the FHIR model should be specified in the meta-models, but also the potential shortcomings of business archetypes in the context of healthcare. Mapping a segment of a FHIR resource to the meta-models can be achieved by abstracting the concept it represents and matching it to a business archetype or a segment of an archetype, relying on definitions from the FHIR documentation[5] and Arlow and Neustadt's work[7]. On the mapping diagrams in the following subsections, FHIR resource segments are on the inside and business archetypes are on outside of the orange background.

6.1. Organization

A FHIR *Organization* resource is an instance of the *Organization* business archetype, which is a subclass of the *Party* archetype. The *Organization* resource has a type, which maps to the *OrganizationType* archetype, illustrated in figure 8. The *Organization* resource has a boolean field to represent if its active or not. Being an active organization is semantically equivalent to having such a capability that is the prerequisite to doing anything, so this is an instance of the *PartyCapability*

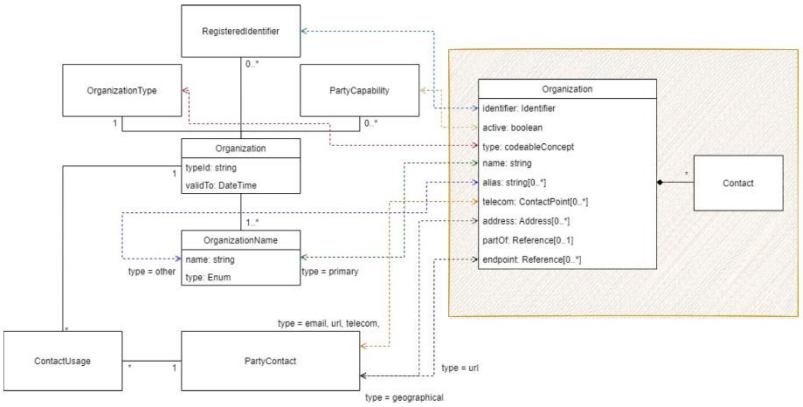


Figure 8: Organization resource and archetype mapping diagram 1

archetype with a boolean value attached. The *Organization* resource has a name and aliases, both of which map to the *OrganizationName* archetype, respectively with the name type of **primary** and **other**. The *Organization* resource has *identifier* fields, which are business identifiers, meaning these extend beyond the context of a single system. These fields are also of a complex FHIR *Identifier* data type. In business archetypes, these identifiers are represented by the *RegisteredIdentifier* archetype. In fact the entire FHIR data type seamlessly maps to this archetype, as illustrated in figure 9. The *Organization* resource's telecoms, addresses and endpoints are all instances of the *PartyContact* archetype, which represents the information, that can be used to contact a party. The resource's address fields represent geographical contact information, telecoms are either phone numbers (or the likes), e-mail addresses or web addresses and endpoints are a special type of web address 8. The *Address* and *ContactPoint* FHIR data types both map to the *PartyContact* arhcetype, as seen in figure 10 and 11. The *Organization* resource has a *partOf* reference field to use when the organization is a part of a bigger whole. In business archetypes, this is modelled with the *PartyRelationship* archetype pattern where the parent organization is in the

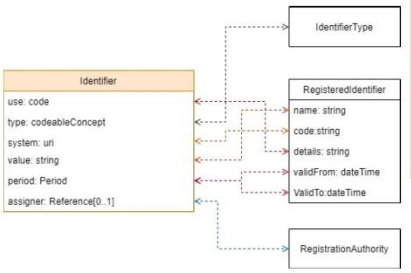


Figure 9: FHIR Identifier data type and RegisteredIdentifier archetype mapping

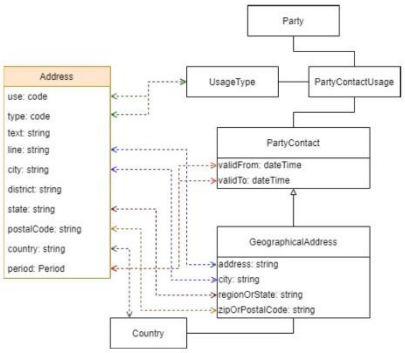


Figure 10: FHIR Address data type and PartyContact archetype mapping

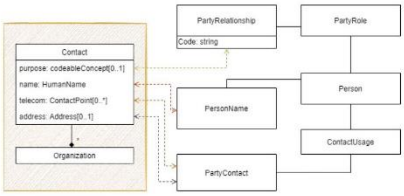


Figure 11: FHIR ContactPoint data type and PartyContact archetype mapping

consumer role and the child organization in the provider role 12. Finally, the FHIR *Organization* resource has a

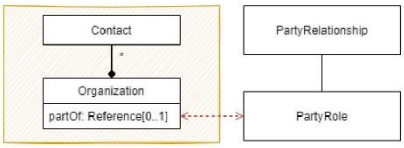


Figure 12: FHIR ContactPoint data type and PartyContact archetype mapping

Contact component, which is a human, who is a contact for the organization for a certain purpose. In terms of business archetypes, this is an instance of the *Person* archetype, another subclass of the *Party* archetype. The connection between the organization and the contact person is modelled with the *PartyRelationship* archetype pattern, where the organization is in the role of the **represented** and the person in the role of the **representative**. The *purpose* field is also mapped to the relationship. The *Contact* resource component has its own set of telecoms and addresses, mapped in the same manner as previously. The mapping is shown in figure 13. The *name* field is

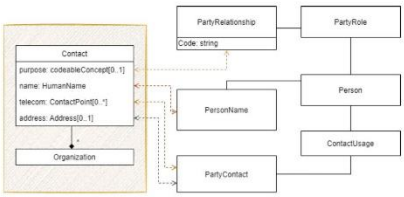


Figure 13: FHIR Organization Contact and Person archetype mapping

of the FHIR *HumanName* data type, which maps to the *PersonName* archetype as shown in figure 14.

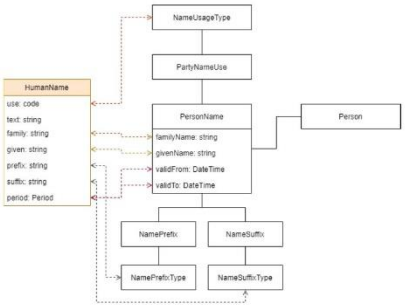


Figure 14: FHIR HumanName data type and PersonName archetype mapping

6.2. Specimen

The FHIR *Specimen* resource can be thought of as an item a healthcare institution uses to provide medical care. Perhaps it is analyzed to give a diagnosis and give proper treatment to some patient. Taking that into consideration, as a business archetype, a Specimen resource is an instance of a *Product* of the **specimen type**. In FHIR, the *Specimen* resource is a composition of containers that all contain the same type of sample. This can be modelled with business archetypes in the following manner. The specimen is an instance of the *Package* archetype (a subclass of the *Product* archetype) that contains instances of the *Container* archetype. Each *Container* contains a specified amount of the sample that is described using the *MeasuredProduct* archetype. The parent object in this construct is the *Package* archetype. The *identifiers* and *accessionIdentifier* fields of the *Specimen* resource are tied to the instance of the *Package* archetype with the *RegisteredIdentifier* archetype. The *accessionIdentifier* is singled out from the other identifiers using a *Feature* archetype connected to the *Package*. The *status* field of the *Specimen* resource is mapped to the *ReservationStatus* value of the *Package* archetype. The *receivedTime* field of the *Specimen* resource maps to the *ValidFrom* value of the *Package* archetype. This is illustrated in figure 15. The FHIR *Identifier* data type mapping is exactly the same as with the *Organization* resource 9. The *Specimen* resource contains many fields that can be abstracted to specifications about the *Package*. With business archetypes, these specifications are modelled with the *Feature* archetype. A *Feature* has a type and a value. The segments of the *Specimen* resource that can be mapped to *Features* are the *subject*, *parent*, *request*, *condition* and *note* fields. The first three are *Reference* data types, which simply point to another resource in some system. Since these fields are

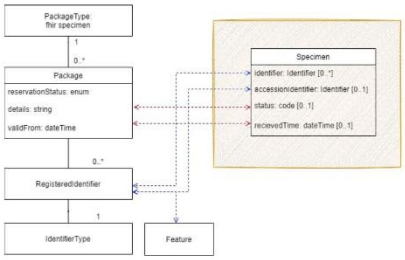


Figure 15: FHIR Specimen resource and Package archetype mapping

not instances of other resources, the *Feature* archetype is a perfect fit. The mapping of these fields to the *Feature* archetype can be seen in figure 16. Each *Container*

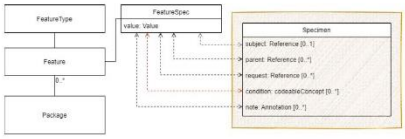


Figure 16: FHIR Specimen resource and Feature archetype mapping

component of the *Specimen* resource is an instance of the *Container* archetype, which is also a subclass of the *Product* archetype. The *Container* components type maps to the *ContainerType* archetype. The *Container* components capacity field is a *Feature*. The *Container* component also holds information about the sample inside it. This is mapped to the *MeasuredProduct* archetype, another subclass of the *Product* archetype. The type of the *MeasuredProduct* is specified in the *type* field of the *Specimen* resource because this is the type of the sample. All of this is illustrated in figure 17. The *Processing* components and the *Collection* component of the *Specimen* resource can also be thought of as specifications about the specimen. Each *Processing* component maps to a single *Feature* archetype instance and the *Collection* component maps to a set of *Feature* archetypes, as illustrated in figure 18.

7. Evaluation and discussions

7.1. Business archetypes as FHIR meta-models

Information gathered from the mapping performed during the development of the initial proof-of-concept yields that it is most likely possible to map the entire HL7 FHIR specification to the business archetypes meta-models, though more resources should be examined to say this with certainty.

This activity is a great way to validate the business archetype patterns as a SUM for FHIR models. Even with only two resources, some possible shortcomings of the abstraction that the meta-models provide were highlighted. The troubles were mainly caused by FHIR's use of complex data types (e.g *Identifier*, *Reference* and *CodeableConcept*), while the meta-models rely on object-relational patterns instead. A data type needs to be deconstructed and mapped to a semantically fitting archetype in the context of the resource's matching archetype pattern, so there is not a one-fits-all matching business archetype for a complex FHIR data type in each case.

Listing 1 is an example of a *Specimen* FHIR resource message provided by HL7[5]. When we send this message to the deployed FHIR endpoint, it is transformed and inserted into the ABC4HEDA database. When requesting the data back from ABC4HEDA in the FHIR format, we are given a response as shown in Listing 2. Notable differences in the processed FHIR message are the missing *text* and *meta* fields. The *text* field is generated using data already included in the message and is not essential. The *meta* field contains content that is maintained by the infrastructure, so this is system specific and not necessary. This is also the case with the *id* field of the resource. The final thing to note is that the *coding* instances in both the *type* and the *Container* component *type* field are missing the *system* and *display* fields. This is in the scope of formalizing medical classifiers and the FHIR specification in the meta-models and does not concern this paper. Given SNOMED and FHIR terminology are specified in the meta-models, these fields would not be empty. However, sometimes a *CodableConcept* instance contains multiple *Codes* and there are cases where business archetypes do not provide the means for this one-to-many relation. Further analysis is required to assess if such cases can be solved with organizing classifiers into a hierarchy or by using some other technique.

7.2. Mirth Connect as an integration engine

Mirth Connect's value as a tool mainly resides in the ability to decouple the integration from the systems themselves. Another great feature is the extendibility of the engine. Custom code libraries can be written in the Mirth Connect Administrator application, but it's also possible to import Java libraries via *.jar* files. That said, it is evident that Mirth Connect is mostly a tool for a developer not an analyst and a substantial amount of outside tooling is required to make integrating systems a decent

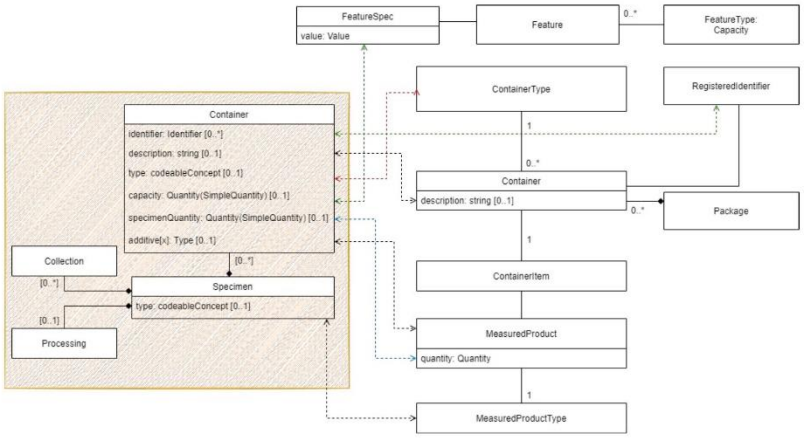


Figure 17: FHIR Specimen resource Container Component and Container archetype mapping

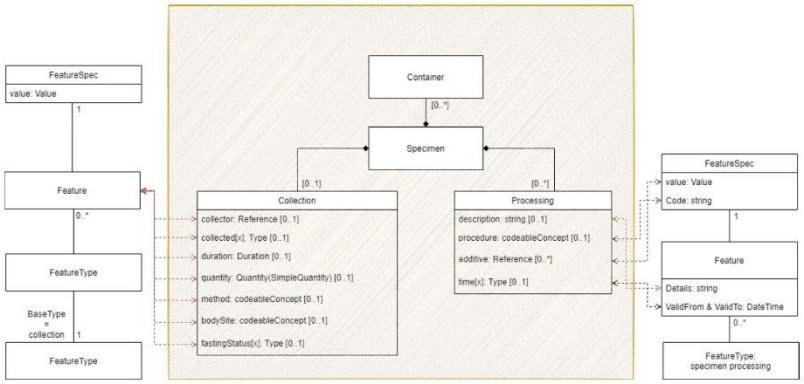


Figure 18: FHIR Specimen resource Collection and Processing Component and Feature archetype mapping

experience. The use of the open source HAPI FHIR API Java packages made the development of this proof-of-concept considerably easier from the FHIR side of things. As the meta-models lack a web programming interface of any kind, the development was quite cumbersome and mapping the whole FHIR standard in this manner is not feasible. The integration for the *Organization* resource took a little over 600 lines of code and the *Specimen* resource a little under 800 lines of code. Clean Code [15] paradigms were mostly adhered to.

However, reasonable requirements for the tools needed for any kind of integration with the meta-models were gathered in the making of this proof-of-concept. For example, the meta-models would greatly benefit from having a GraphQL API. The integration process would go from programming a set of SQL statements to design-

ing a single GraphQL query for a FHIR resource, which could be reused for both requesting and mutating data. Another necessary element to enabling data exchange between models would be a web service to query classifier identifiers. An example use case would be sending the service a request with the SNOMED code "119364003" and the response would contain the unique system identifier of this classifier in the ABC4HEDA database, allowing to then create valid object relations. We also believe that a lot of integration code could be extracted to a common library of utility methods regarding the business archetypes. At scale, this could help reduce the code needed for integrating any single FHIR resource by a reasonable amount.

7.3. Further work

Formalizing the FHIR models and value sets themselves in the meta-models is another critical step in validating and integrating the specification with business archetypes. The model formalization is the medical knowledge of the specification, while transforming information between the formats regards only the medical data.

Further work also includes prototyping and validating the tooling mentioned in the previous subsection. Perhaps a minimal GraphQL implementation spanning parts of the *Product* archetype pattern could be developed to rewrite the *Specimen* resource mapping and compared with this proof-of-concept. Ideally, the logical mapping between specifications of health data should also be possible to be done by an analyst in a low-code/no-code manner. Although the business archetypes themselves are a kind of mapping language, it needs to have an intuitive way to interface with. In their current state, the meta-models are not sufficient in this regard. A GraphQL API for business archetypes could enable the development or use of an analyst-friendly mapping language and source code generators for the implementation of the integration.

This work could also be used as a basis for developing similar knowledge of integrating and formalizing other standardized healthcare data models, such as ones used by HL7 v2.x, HL7 v3.x and openEHR.

8. Conclusion

We think mapping the entire HL7 FHIR specification to the business archetypes meta-models is possible. Thus, the semantic interoperability between the meta-models and FHIR-based systems is entirely implementable, although some additional tooling would make the development process much easier. In addition, Mirth Connect is a great candidate for implementing integrations beyond this proof-of-concept, although it would most likely have to be done by a software developer, not an analyst.

This paper can be used as a starting point to implement integration tooling for ABC4HEDA, formalizing the FHIR specification in business archetypes and researching other healthcare data model standards.

Authors' contribution

Rained Randmaa wrote the manuscript with support from Igor Bossenko, Toomas Klementi and Gunnar Piho. All authors contributed to the final version of the manuscript. Gunnar Piho and Peeter Ross supervised the project.

Acknowledgments

This work in the project "ICT programme" was supported by the European Union through European Social Fund.

9. Listings

Listing 1: Serum example in

```
{
    "resourceType": "Specimen",
    "id": "sst",
    "text": {
      "status": "generated",
      "div": "..."
    },
    "accessionIdentifier": {
      "system": "http://acme.com/labs/
          accession-ids",
      "value": "20150816-00124"
    },
    "type": {
      "coding": [
        {
          "system": "http://snomed.info/sct",
          "code": "119364003",
          "display": "Serum sample"
        }
      ]
    },
    "subject": {
      "reference": "Patient/pat2"
    },
    "request": [
      {
        "reference": "ServiceRequest/ft4"
      }
    ],
    "collection": {
      "collector": {
        "reference": "Practitioner/f202"
      },
      "collectedDateTime": "2015-08-16T06
          :40:17Z"
    },
    "container": [
      {
        "type": {
```

```
            "coding": [
              {
                "system": "http://acme.com/labs
                    ",
                "code": "SST",
                "display": "Serum Separator
                    Tube"
              }
            ]
          }
        }
      ],
      "meta": {
        "tag": [
          {
            "system": "http://terminology.hl7.
                org/CodeSystem/v3-ActReason",
            "code": "HTEST",
            "display": "test health data"
          }
        ]
      }
    }
```

Listing 2: Serum example out

```
{
    "resourceType": "Specimen",
    "id": "49cb178c-69b8-42de-a76e-
        d3bf69d600ce",
    "accessionIdentifier": {
        "system": "http://acme.com/labs/
            accession-ids",
        "value": "20150816-00124"
    },
    "type": {
        "coding": [
            {
                "code": "119364003"
            }
        ]
    },
    "subject": {
        "reference": "Patient/pat2"
    },
    "request": [
        {
            "reference": "ServiceRequest/ft4"
        }
    ],
    "collection": {
        "collector": {
            "reference": "Practitioner/f202"
        },
        "collectedDateTime": "2015-08-16T06
            :40:17Z"
    },
    "container": [
        {
            "type": {
                "coding": [
                    {
                        "code": "SST"
                    }
                ]
            }
        }
    ]
}
```

References

[1] D. Chen, G. Doumeingts, F. Vernadat, Architectures for enterprise integration and interoperability: Past, present and future, Computers in Industry (2008). doi:https://doi.org/10.1016/j.compind.2007.12.016, enterprise Integration and Interoperability in Manufacturing Systems.

[2] G. Hohpe, B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Longman Publishing Co., Inc., 2003.

[3] HAPIFHIR, Hapi fhir is a complete implementation of the hl7 fhir standard for healthcare interoperability in java., https://hapifhir.io/, 2022. Accessed: 2022-04-06.

[4] S. Nizamov, Unofficial Developer's Guide to FHIR on Mirth Connect, 2016.

[5] HL7, Fhir is a standard for health care data exchange, published by hl7®., http://hl7.org/fhir/, 2022. Accessed: 2022-04-06.

[6] Postman, Postman is an api platform for building and using apis., https://www.postman.com/, 2022. Accessed: 2022-04-27.

[7] J. Arlow, I. Neustadt, Enterprise patterns and MDA: building better software with archetype patterns and UML, Addison-Wesley, 2003.

[8] J. Zachman, A framework for information systems architecture, IBM Systems Journal 38 (1999). doi:10.1147/sj.382.0454.

[9] P. Coad, Object-oriented patterns, Communications of the ACM 35 (1992) 152–159. URL: doi.org/10.1145/130994.131006.

[10] G. Piho, Archetypes based techniques for development of domains, requirements and software: towards LIMS software factory, Ph.D. thesis, Tallinn University of Technology, 2011. URL: digi.lib.ttu.ee/i/?636.

[11] J. Meier, H. Klare, C. Tunjic, C. Atkinson, E. Burger, R. Reussner, A. Winter, Single underlying models for projectional, multi-view environments, 2019.

[12] F. Oemig, R. Snelick, Healthcare Interoperability Standards Compliance Handbook, 2016. doi:10.1007/978-3-319-44839-8.

[13] M. Braunstein, Health Informatics on FHIR: How HL7's New API is Transforming Healthcare, 2018. doi:10.1007/978-3-319-93414-3.

[14] S. Nizamov, Unofficial Mirth Connect v3.12 Developer's Guide, 2021.
[15] R. C. Martin, J. O. Coplien, Clean code: a handbook of agile software craftsmanship, Prentice Hall, 2009. URL: https://www.amazon.de/gp/product/0132350882/ref=oh_details_o00_s00_i00.

Lisa 3 – HEDA 2022 artikli retsensioon 1

The paper presents an architecture to handle the interoperability issue of healthcare information. The authors proposed to use executable business meta-models that can be utilized as a shared ontology to achieve federated interoperability. This proposed technique is interesting as it provides robust solution to dynamic changes of domain models in the healthcare. The approach exploits the use of HL7 FHIR as standard but also incorporates archetypes to complement the meta-model for software interoperability. The proposed architecture is illustrated which gives an overview of the approach. The authors mentioned that the approach is evaluated by a distributed test environment which shows how healthcare information is exchanged and transformed, back and forth. The approach uses NextGen Connect (Mirth Connect) as a mapping engine between FHIR and the meta-models. However, there are other techniques for model transformations which could have been worth mentioning.

The references for archetype patters are very old references. Perhaps the author should also give reference to their previous work on archetype patters in [a]. Is the Party archetype patters in Figure 1 is an extended model of party archetype pattern presented in [a]?

The paper mentions briefly how GraphQL could be utilized for exchanging information but it might be worth mentioning similar works that exists in [b,c] which addresses the interoperability issue using GraphQL.

[a] https://www.sciencedirect.com/science/article/pii/S1877050914010345

[b] https://www.sciencedirect.com/science/article/pii/S187705091931782X

[c] https://pubmed.ncbi.nlm.nih.gov/34524029/

Lisa 4 – HEDA 2022 artikli retsensioon 2

This paper studies the inter-operability problem in the healthcare domain. As it seems unlikely that the whole industry agrees on a common standard, the mediation between different standards standards becomes more important. This paper suggests to use executable business meta-models as an intermediate format. To evaluate this idea the paper implements a prototype using ABC4HEDA as a business model engine and individual information encoded in HL7 FHIR for encodings of information.

The paper is well written and has interesting ideas. The described solution sounds convincing. However, there is not a lot of discussion in the paper, meaning that alternative solutions are not thoroughly investigated. As such, the contribution shows one solution but whether the solution is preferable to potential other solutions, if they exist, it is not obvious from this paper. Nevertheless I suggest acceptance of this contribution.