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 15A, 12618 Tallinn, Estonia*
[2]*Department of Software Science, TalTech, Akadeemia 15A, 12618 Tallinn, Estonia*

Abstract

Information systems interoperability in health care has proved a difficult challenge due to the semantic heterogeneity of models, data, messages, etc. So far, attempts to solve this problem have involved a unified approach through the development and utilisation of common standards. However, this approach has issues in semantic heterogeneity; if the standardised data models and exchange protocols are developed by different vendors, they tend not to be interoperable. A federated interoperability approach where models can be dynamically specified at run-time rather than being predetermined might be a useful supplement to the unified interoperability approach. In this workshop paper, we propose executable business meta-models, which, according to our understanding, can be utilised 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 data are 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 to the seamless no-code/low-code federated interoperability of health information systems, whereby different systems can exchange medical data in a semantically coherent way using different protocols and data models and their versions, such as FHIR, HL7 v2.x, openEHR.

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

1. Introduction

The complexity of data and information systems in the field of healthcare informatics creates many challenges in using health data for advancing medicine, science, population health and the economy. Data exchange between healthcare systems is specifically challenging because of the semantic heterogeneity of data models. Different vendors in health care use different standards and specifications as predetermined models when developing their applications, making data incompatible between systems. A federated interoperability approach where models can be dynamically specified at run-time rather than being predetermined might be a useful supplement to the unified interoperability approach, meaning that there is a common meta-level structure

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)

across constituent models [1]. We believe that meta-models based on business archetypes can be utilised as a shared ontology to achieve federated interoperability. A system using these meta-models allows specification of models during run-time and transformations between them. A means to exchange health data without affecting their 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 newer standardised communication protocols in health care and is constantly gaining popularity. This paper aims to illustrate and explain the process of validating these meta-models as well as integrating the systems developed using FHIR models by implementing proof-of-concept data exchange channels between FHIR and the meta-models.

2. Methodology

For developing the proof-of-concept integration between HL7 FHIR and business archetype-based meta-models, the integration architecture is designed using enterprise integration patterns and principles [2]. Semantic mapping between FHIR models and the meta-models is performed and displayed using UML diagrams. NextGen Connect (Mirth Connect) is used as an integration engine to implement the architecture and mapping by aggregating Mirth Connect channels to achieve 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. With the help of HAPI, FHIR messages can be serialised into Java objects, which makes the mapping code more reliable and easier to write [4]. To test the integration, an instance of the meta-models based software, ABC4HEDA, is deployed on a virtual machine with a Microsoft SQL Server as the persistence layer. Mirth Connect is deployed on a separate virtual machine. This is done using Microsoft Azure cloud services and mimics a distributed environment. A set of example resources is provided by HL7 on the official FHIR documentation website [5], which are used as messages when testing the integration implementation. Messages are sent using Postman [6]. Critical retrospective analysis is then performed on the process of developing this proof-of-concept.

3. Archetype pattern-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. The meta-models discussed in this chapter build on previous work regarding archetypes based meta-modeling for healthcare [9].

One of the archetype patterns we need to utilise 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 the parties involved in business activities, such as a person or an organisation. 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 us to describe how these parties relate to one another, e.g. a client-supplier relationship.

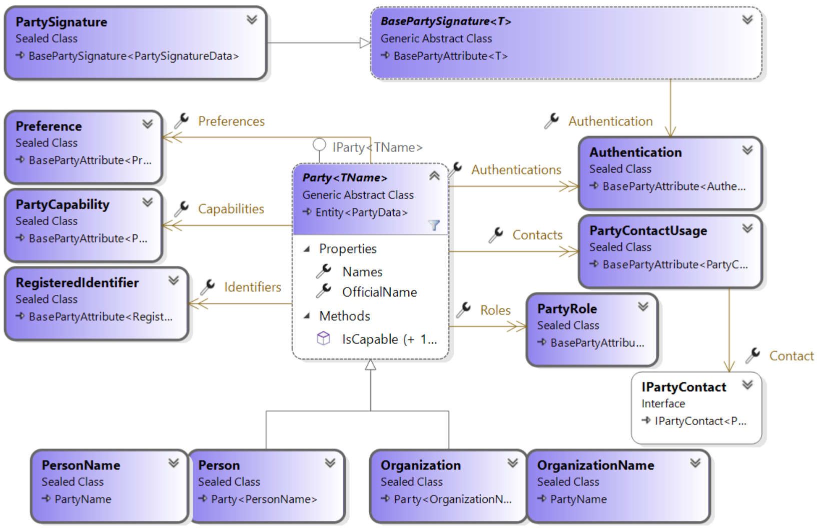


Figure 1: Party archetype.

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. This figure also illustrates the use of the *Item Description* pattern[10], which is used throughout the meta-models. The

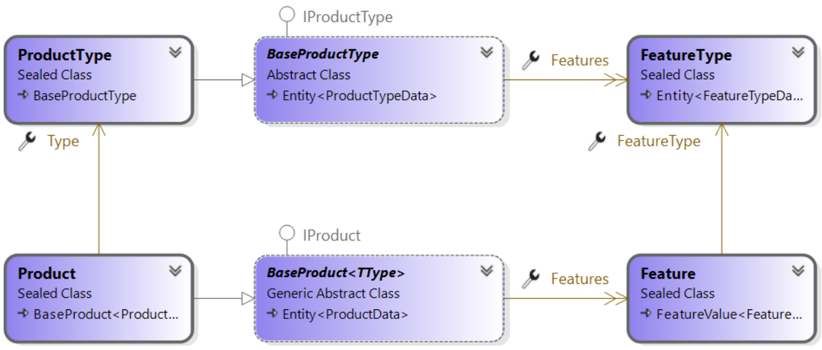


Figure 2: Product archetype.

Product archetype pattern is used to describe any product or service a business provides or uses to provide another product or service. In other words, a product can be thought of as any item or service used in business processes. The *ProductType* archetype represents a product's generic information, 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 catalogue or leaflet, and the *Product* represents the actual thing you walk out the door with. In the context of health care, a product might be a blood sample in storage, medicine ready to be administered or a consultation booked.

In addition to the aforementioned archetype patterns, the meta-models also implement *Or-*

der, *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[11]. We believe that these archetype patterns can be used as a Single Underlying Model (SUM) [12] for healthcare data models. The important aspect to highlight about these meta-models is that data and knowledge of the data are 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 contains an administrator UI component developed with ASP.NET. Of the 120K lines of source code, *ca* 45% are automated unit tests and UI acceptance tests, resulting in the 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 standardised manner [13, 14]. 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 *Organisation* and the *Specimen* resource. The *Organisation* resource is used to describe a formally or informally recognised grouping of people or organisations formed for the purpose of achieving 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 the origin of the specimen [5].

5. Integration using Mirth Connect

NextGen Connect (Mirth Connect) is an open source healthcare integration engine, for which enterprise extensions that require a subscription are available. The integration unit between a sending app and a receiving app in Mirth Connect is called a channel. In other words, one-to-many abstract unidirectional pipes to decouple components from one another 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 [15]. A channel consists of a *Source connector*, a *Destination connector* and a *Response connector*. Each has their own set of configurable filters and transformers. Channels can also be used in conjunction for integration problems with complex requirements.

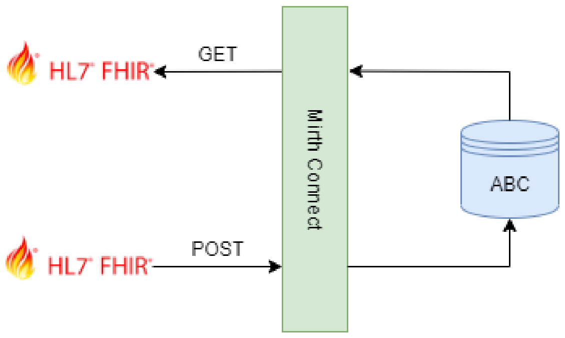


Figure 3: FHIR interface for ABC4HEDA.

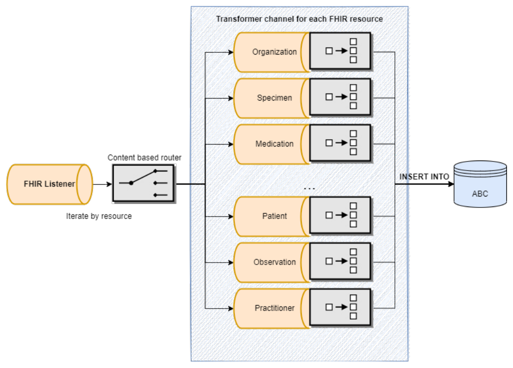
Mirth Connect offers a few ways to implement mappings between two message formats. The low-code version uses message templates. This is the preferred option for a non-developer individual, but to our understanding, there is no straight-forward way to obtain consistent templates for FHIR messages. The 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 the problem of accessibility. A graphical FHIR resource builder is provided, 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 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, as shown in Figure 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, this is not much different than communicating with a RESTful API where each database table is often allocated its own resource endpoint.

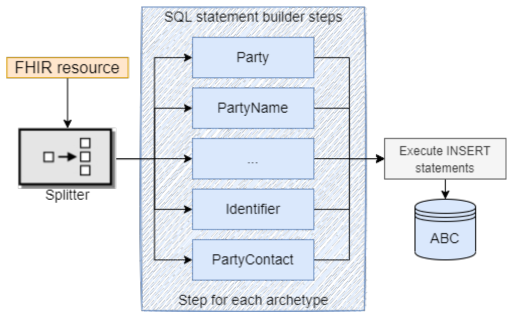
5.1. From FHIR to meta-models

For inserting data in FHIR format to the ABC4HEDA database, an endpoint should be able to listen for an incoming FHIR message and then split it into instances of resources 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 them into the database. The integration architecture for inserting data is illustrated in Figure 4a.

A resource transformer channel will deserialize the FHIR message into a HAPI FHIR [3] Java class instance and then break the resource down into 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 4b



(a) FHIR to Mirth to ABC4HEDA.

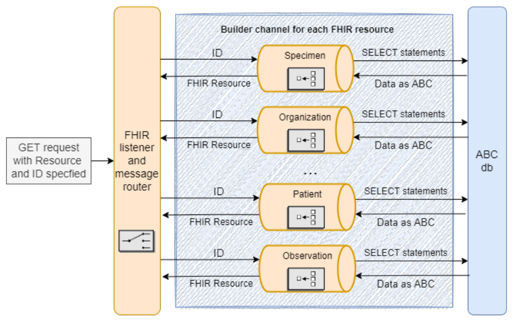


(b) FHIR resource to business archetype transformer channel.

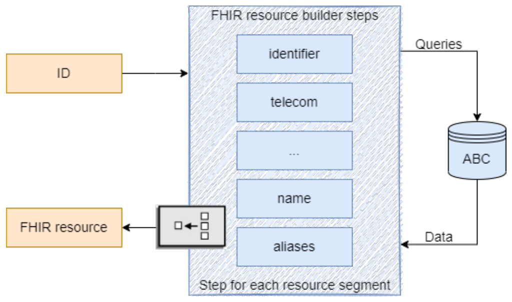
Figure 4: FHIR to ABC4HEDA architecture.

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 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 5a. A resource builder channel creates



(a) ABC4HEDA to Mirth to FHIR.



(b) Business archetype to FHIR resource builder channel.

Figure 5: ABC4HEDA to FHIR architecture.

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 5b.

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 compatibility of meta-models with the FHIR specification. This process gives insight into not only 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 inside the orange background and business archetypes are outside.

6.1. Organisation

An FHIR *Organisation* resource is an instance of the *Organisation* business archetype, which is a subclass of the *Party* archetype. The *Organisation* resource has a type, which maps to the *OrganisationType* archetype, as illustrated in Figure 6. The *Organisation* resource has a

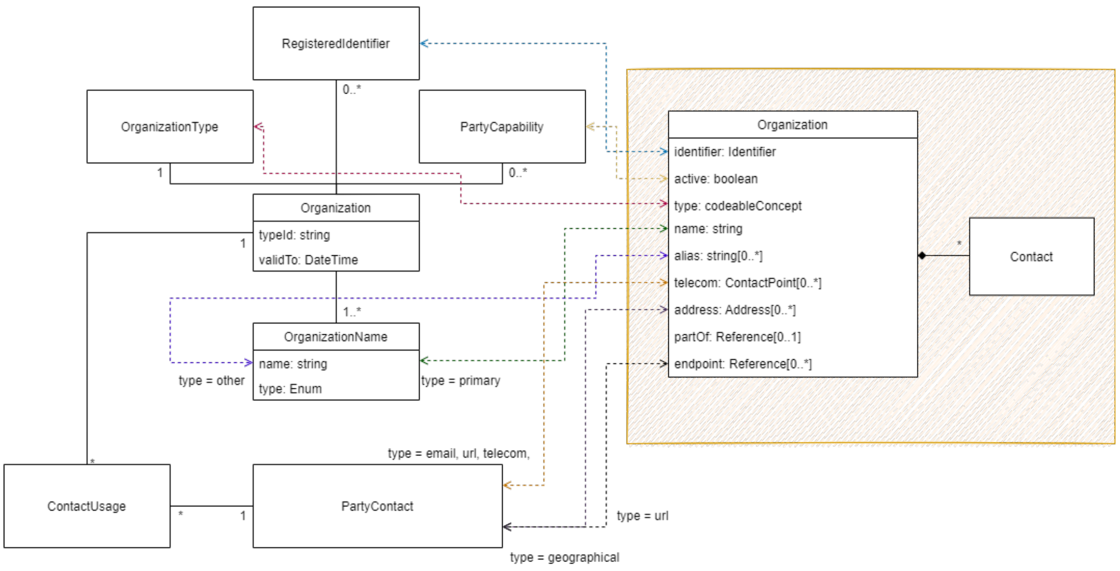


Figure 6: Organisation resource and archetype mapping diagram. FHIR resource segments highlighted with orange.

boolean field to represent whether it is active or not. Being an active organisation is semantically equivalent to having a capability that is a prerequisite for doing anything, so this is an instance of the *PartyCapability* archetype. The *Organisation* resource has a name and aliases, both of which map to the *OrganisationName* archetype with the name types of **primary** and **other**, respectively. The *Organisation* resource has *identifier* fields, which are business identifiers, meaning that they 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 7. The *Organisation* resource's telecoms, addresses and

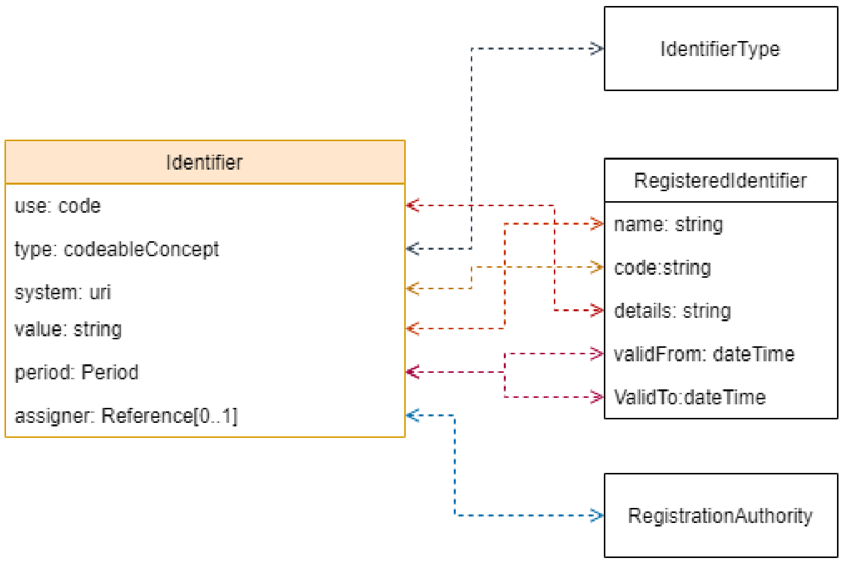


Figure 7: FHIR Identifier data type and RegisteredIdentifier archetype mapping. FHIR data type highlighted with orange.

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 similar), e-mail addresses or web addresses and endpoints are a special type of web address 6. The *Address* and *ContactPoint* FHIR data types both map to the *PartyContact* arhcetype, as seen in Figures 8 and 9. The *Organisation* resource has a *partOf* reference field to use when the organisation is a part of a larger whole. In business archetypes, this is modelled with the *PartyRelationship* archetype pattern, where the parent organisation is in the consumer role and the child organisation in the provider role.

Finally, the FHIR *Organisation* resource has a *Contact* component, which is a human who is a contact for the organisation 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 organisation and the contact person is modelled with the *PartyRelationship* archetype pattern, where the organisation is in the role of the **represented** and the person 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 before. The *name* field is of the FHIR *HumanName* data type, which maps to the *PersonName* archetype.

6.2. Specimen

The FHIR *Specimen* resource can be thought of as an item used by a healthcare institution to provide medical care. Perhaps it is analysed to give a diagnosis and assign proper treatment to a 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

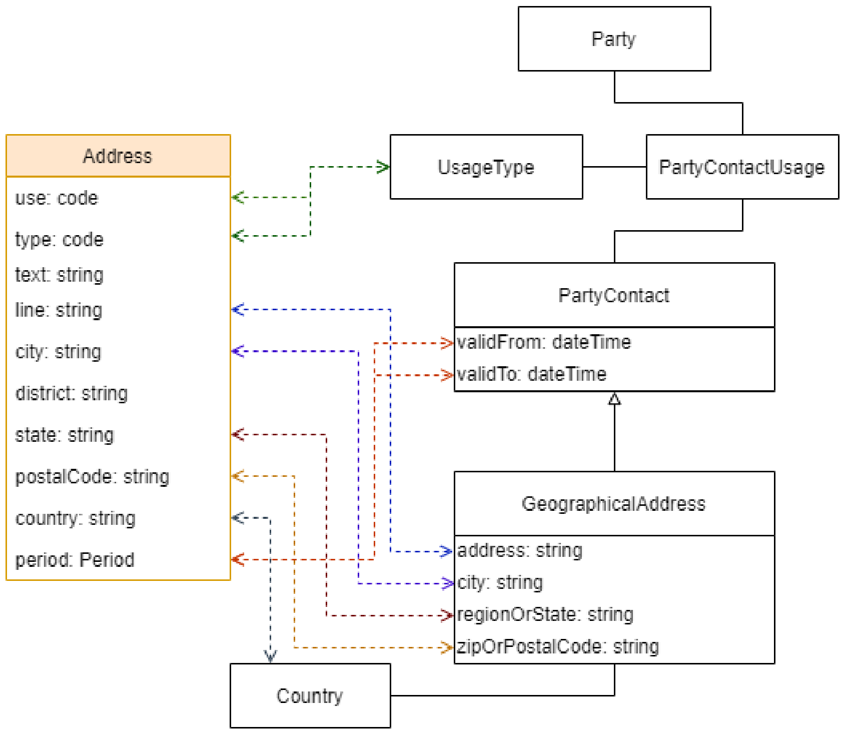


Figure 8: FHIR Address data type and PartyContact archetype mapping. FHIR data type highlighted with orange.

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 10. The FHIR *Identifier* data type mapping is exactly the same as that of the *Organisation* resource 7. 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 a system. Since these fields are not instances of other resources, the *Feature* archetype is the perfect fit. The mapping of these fields to the *Feature* archetype can be seen in Figure 11. Each *Container* 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

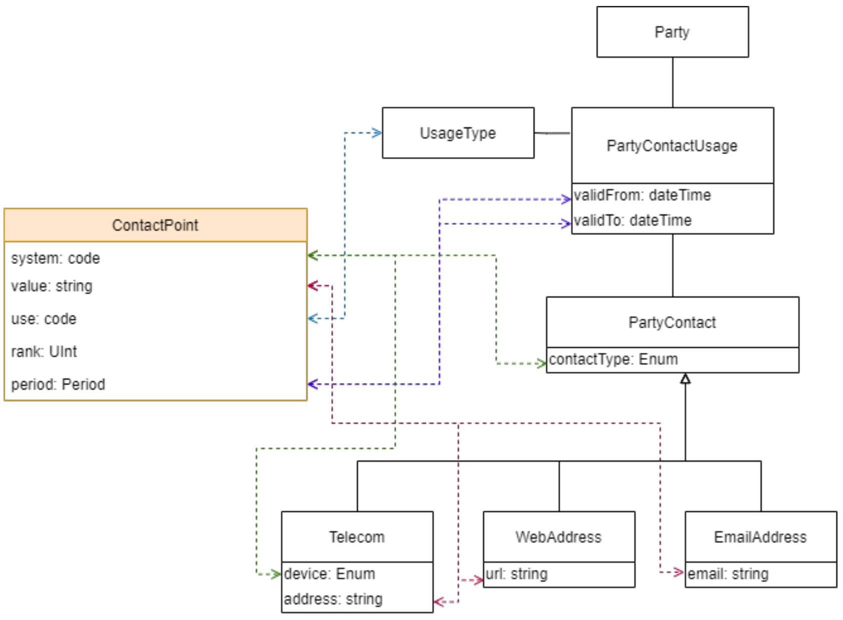


Figure 9: FHIR ContactPoint data type and PartyContact archetype mapping. FHIR data type highlighted with orange.

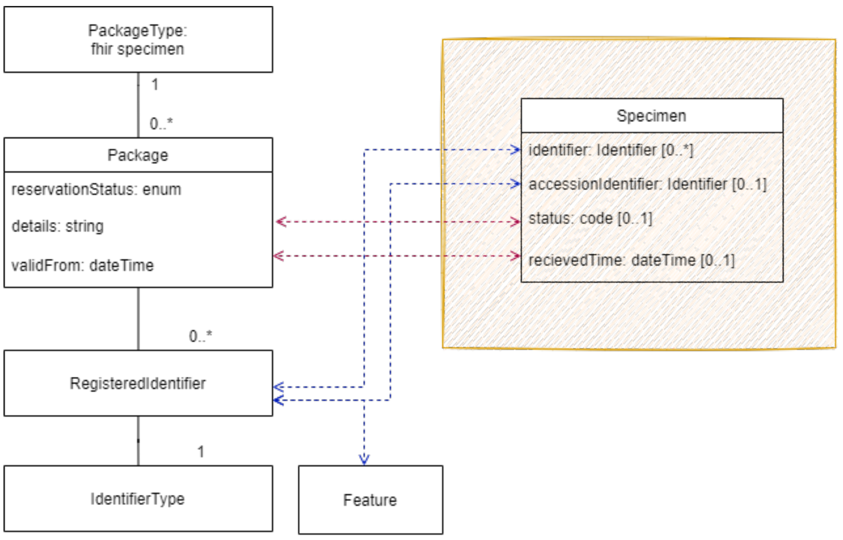


Figure 10: FHIR Specimen resource and Package archetype mapping. FHIR resource segment highlighted with orange.

holds information about the sample inside it. This is mapped to the *MeasuredProduct* archetype, another subclass of the *Product* archetype. The type of *MeasuredProduct* is specified in the *type* field of the *Specimen* resource because this is the type of sample. All of this is illustrated in Figure 12. The *Processing* components and the *Collection* component of the *Specimen* resource can be thought of as specifications of the specimen. Each *Processing* component maps to a single

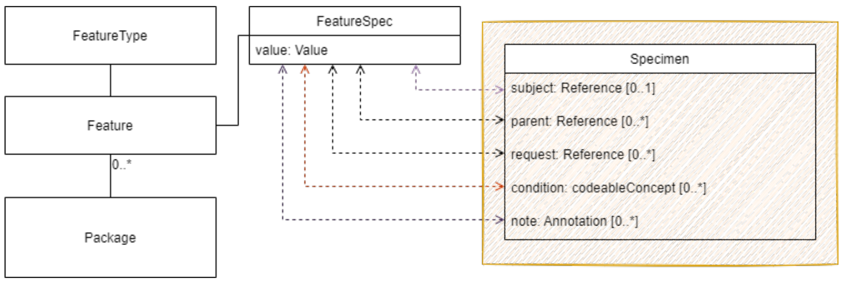


Figure 11: FHIR Specimen resource and Feature archetype mapping. FHIR resource segment highlighted with orange.

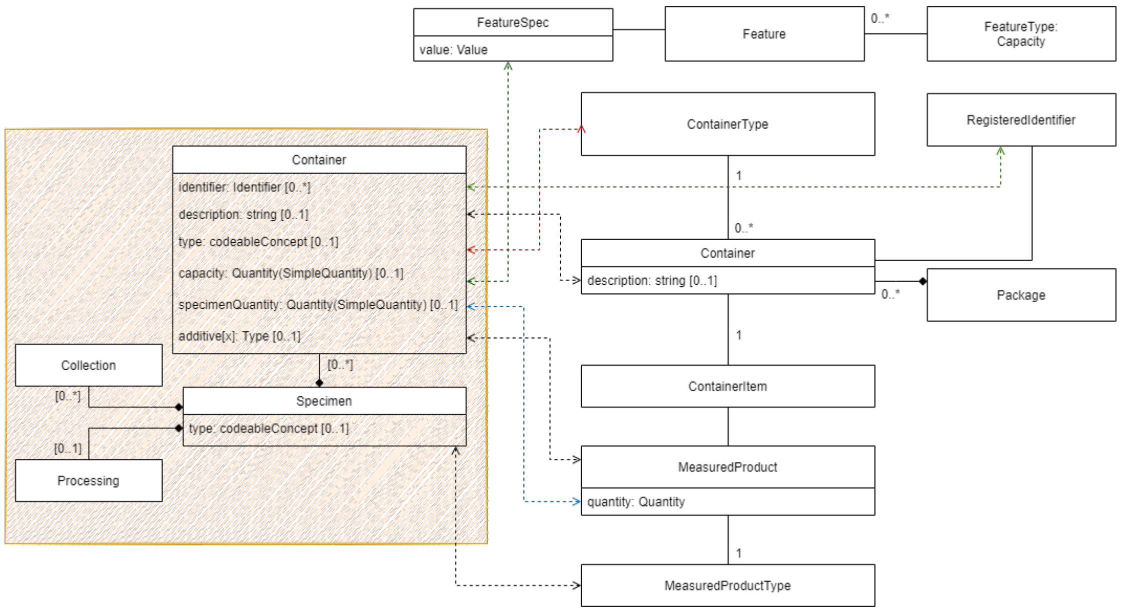


Figure 12: FHIR Specimen resource Container component and Container archetype mapping. FHIR resource segment highlighted with orange.

Feature archetype instance and the *Collection* component maps to a set of *Feature* archetypes.

7. Evaluation and discussion

7.1. Business archetypes as FHIR meta-models

The 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, the possible shortcomings of the abstraction 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 suitable archetype in the context of the resource's matching archetype pattern; there is no one-fits-all matching business archetype for a complex FHIR data type in each case.

When sending example messages 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 that contains all meaningful data from the inserted message. Differences in the processed FHIR message only include content that is maintained by the infrastructure or content that can be generated from data already included in the message.

7.2. Mirth Connect as an integration engine

The value of Mirth Connect as a tool mainly resides in its ability to decouple the integration from the systems themselves. Another great feature is the extensibility of the engine. Custom code libraries can be written in the Mirth Connect Administrator application, but it is also possible to import Java libraries via *.jar* files. That being said, it is evident that Mirth Connect is mostly a tool for a developer, not an analyst, and substantial outside tooling is required to make integrating systems a decent experience. The use of open-source HAPI FHIR API Java packages makes the development of this proof-of-concept considerably easier from the FHIR side. As the meta-models lack a web programming interface of any kind, development was quite cumbersome, therefore mapping the entire FHIR standard in this manner is not feasible. The integration for the *Organisation* resource took a little over 600 lines of code and the *Specimen* resource a little under 800 lines of code. Clean Code [16] 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 course of this proof-of-concept. For example, the meta-models would greatly benefit from a GraphQL API. The integration process would go from programming a set of SQL statements to designing a single GraphQL query for a FHIR resource, which could then be reused for both requesting and mutating data. Combining GraphQL and FHIR has shown promising results in the past [17, 18]. Another necessary element to enabling data exchange between models is a web service to query classifier identifiers. An example use case is sending the service a request with the SNOMED code '119364003' and the response then containing the unique system identifier of this classifier in the ABC4HEDA database, thus allowing the creation of valid object relationships. We believe that a large volume 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 one FHIR resource substantially.

7.3. Further work

Formalising the FHIR models and value sets in the meta-models is another critical step in validating and integrating the specification with business archetypes. The model formalisation is the medical knowledge of the specification, while transforming information between the formats concerns 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 then compared with this proof-of-concept. Ideally, it would be possible for the logical mapping between specifications of health data to be done by an analyst in a low-code/no-code manner. 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, drawing inspiration from work regarding GraphQL and federated interoperability in [19].

This work could also be used as a basis for developing similar knowledge of integration and formalising other standardised healthcare data models, such as those used by HL7 v2.x, HL7 v3.x and openEHR. Alternative tools to Mirth Connect are also worth investigating.

8. Conclusion

We think mapping the entire HL7 FHIR specification to the business archetypes meta-models is possible. Therefore, semantic interoperability between the meta-models and FHIR-based systems is entirely implementable, though some additional tooling would make the development process much easier. In addition, Mirth Connect is a great candidate for implementing integration beyond this proof-of-concept, though 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, formalising the FHIR specification in business archetypes and researching other healthcare data model standards.

Authors' contribution

Rainer 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 the European Social Fund.

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] G. Piho, J. Tepandi, D. Thompson, T. Tammer, M. Parman, V. Puusep, Archetypes based meta-modeling towards evolutionary, dependable and interoperable healthcare information systems, Procedia Computer Science 37 (2014) 457–464. URL: https://www.sciencedirect.com/science/article/pii/S1877050914010345. doi:https://doi.org/10.1016/j.procs.2014.08.069.

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

[11] 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.

[12] J. Meier, H. Klare, C. Tunjic, C. Atkinson, E. Burger, R. Reussner, A. Winter, Single underlying models for projectional, multi-view environments, 2019. URL: http://dx.doi.org/10.5220/0007396401190130.

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

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

[15] S. Nizamov, Unofficial Mirth Connect v3.12 Developer's Guide, 2021.

[16] 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.

[17] S. K. Mukhiya, F. Rabbi, V. K. I Pun, A. Rutle, Y. Lamo, A graphql approach to healthcare information exchange with hl7 fhir, Procedia Computer Science 160 (2019) 338–345. URL: https://www.sciencedirect.com/science/article/pii/S187705091931782X. doi:https://doi.org/10.1016/j.procs.2019.11.082.

[18] S. K. Mukhiya, Y. Lamo, An hl7 fhir and graphql approach for interoperability between heterogeneous electronic health record systems, Health Informatics Journal 27 (2021) 14604582211043920.

[19] P. Stünkel, O. von Bargen, A. Rutle, Y. Lamo, Graphql federation: A model-based approach, Journal of Object Technology (2020). URL: http://www.jot.fm/contents/issue_2020_02/article18.html. doi:10.5381/jot.2020.19.2.a18.