HHS Public Access

Author manuscript

J Biomed Inform. Author manuscript; available in PMC 2018 March 01.

Published in final edited form as:

J Biomed Inform. 2017 March ; 67: 90–100. doi:10.1016/j.jbi.2017.02.009.

Modeling and Validating HL7 FHIR Profiles Using Semantic Web Shape Expressions (ShEx)

Harold R. Solbrig[1], **Eric Prud'hommeaux**[2,3], **Grahame Grieve**[4], **Lloyd McKenzie**[5], **Joshua C. Mandel**[6], **Deepak K. Sharma**[1], and **Guoqian Jiang**[1]

[1]Department of Health Sciences Research, Mayo Clinic, Rochester, MN, USA

[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institutes of Technology, Cambridge, MA, USA

[3]World Wide Web Consortium (W3C), Cambridge, MA, USA

[4]Health Intersections Pty Ltd, Victoria, Australia

[5]Gevity Consulting Inc, Vancouver BC, CA

[6]Harvard Medical School Department of Biomedical Informatics, Boston, MA

Abstract

Background—HL7 Fast Healthcare Interoperability Resources (FHIR) is an emerging open standard for the exchange of electronic healthcare information. FHIR resources are defined in a specialized modeling language. FHIR instances can currently be represented in either XML or JSON. The FHIR and Semantic Web communities are developing a third FHIR instance representation format in Resource Description Framework (RDF). Shape Expressions (ShEx), a formal RDF data constraint language, is a candidate for describing and validating the FHIR RDF representation.

Objective—Create a FHIR to ShEx model transformation and assess its ability to describe and validate FHIR RDF data.

Methods—We created the methods and tools that generate the ShEx schemas modeling the FHIR to RDF specification being developed by HL7 ITS/W3C RDF Task Force, and evaluated the applicability of ShEx in the description and validation of FHIR to RDF transformations.

Correspondence to: Guoqian Jiang, MD, PhD, Department of Health Sciences Research, Mayo Clinic, 200 First Street, SW, Rochester, MN 55905, USA, Telephone: 507-266-1327, Fax: 507-284-1516, Jiang.Guoqian@mayo.edu.

CONTRIBUTORS

H.S., E.P. and G.J. conceived and designed the study; H.S., E.P. and G.J. drafted the manuscript; H.S., E.P., and J.C.M led the transformation tool development and evaluation; all authors contributed expertise and edits.

COMPETING INTERESTS

The authors declare that no competing interests exist.

CONFLICT OF INTEREST STATEMENT

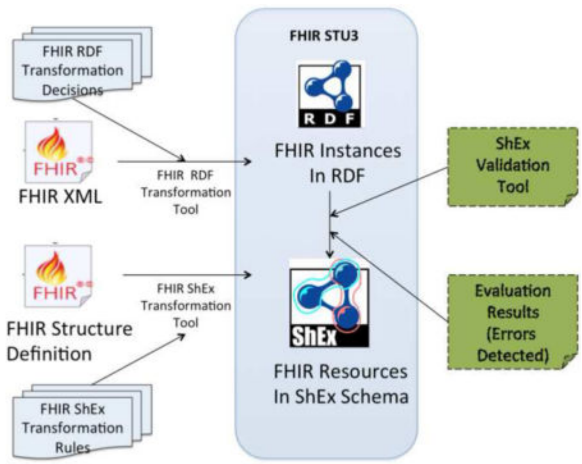
The authors declare that no competing interests exist.

Results—The ShEx models contributed significantly to workgroup consensus. Algorithmic transformations from the FHIR model to ShEx schemas and FHIR example data to RDF transformations were incorporated into the FHIR build process. ShEx schemas representing 109 FHIR resources were used to validate 511 FHIR RDF data examples from the Standards for Trial Use (STU 3) Ballot version. We were able to uncover unresolved issues in the FHIR to RDF specification and detect 10 types of errors and root causes in the actual implementation. The FHIR ShEx representations have been included in the official FHIR web pages for the STU 3 Ballot version since September 2016.

Discussion—ShEx can be used to define and validate the syntax of a FHIR resource, which is complementary to the use of RDF Schema (RDFS) and Web Ontology Language (OWL) for semantic validation.

Conclusion—ShEx proved useful for describing a standard model of FHIR RDF data. The combination of a formal model and a succinct format enabled comprehensive review and automated validation.

Graphical abstract



Keywords

HL7 Fast Healthcare Interoperability Resources (FHIR); Shape Expressions (ShEx); Resource Description Framework (RDF); Semantic Web Technology; Quality Assurance

1 INTRODUCTION

HL7 Fast Healthcare Interoperability Resources (FHIR) is an emerging open standard for the exchange of electronic healthcare information[1]. FHIR defines a collection of "resources" that "can easily be assembled into working systems that solve real world clinical and administrative problems at a fraction of the price of existing alternatives." This assembly process typically requires "profiling" – the adaptation of the FHIR base resources for use in particular contexts and use cases[2, 3]. The core FHIR resources are described by FHIR-specific schema structures using the *StructureDefinition* resource.[4] The FHIR specification also defines how resource instances should be represented in XML and JSON.

The Semantic Web's Resource Description Framework (RDF)[5] offers another potential FHIR instance representation format. The Yosemite Manifesto[6] positions RDF as the "Universal Healthcare Exchange Language" and asserts that "existing standard healthcare vocabularies, data models and exchange languages should be leveraged by defining standard mappings to RDF, and any new standards should have RDF representations". The Manifesto also asserts that healthcare information should either have a "standard mapping to RDF" or should already be in an RDF format. A critical component in the realization of both the FHIR and Yosemite visions is the ability to specify a formal data model or "schema" for RDF – an equivalent of the combination of XML Schema and Schematron used to describe FHIR XML[7]. This RDF "schema" would need to specify which sets of triples must, may and may not be present for a given subject in a given context. It would need to be able to identify valid predicates, data types, minimum and maximum values, value sets, literal match patterns, and Boolean combinations thereof.

The RDF Shape Expressions language (ShEx)[8, 9], developed as an input to the W3C RDF Data Shapes task force[10], meets most of the above requirements. ShEx is a language for formally describing RDF structures and can serve the same role with RDF as that of XML schema to XML. ShEx also provides a "semantic action" capability that enables conformant RDF instance data to be transformed to RDF, XML, JSON, HTML, CSV, SQL, or other representations.

The objective of this study was to develop a FHIR to ShEx model transformation and use it to both document and validate the RDF representation of FHIR data instances. We analyzed the FHIR RDF representation specification produced by HL7 ITS/W3C RDF Task Force[11] and created a transformation from the FHIR metamodel to its ShEx equivalent. The result was evaluated for its usefulness in the documentation of workgroup decisions as well its capability to formally represent the various aspects of the FHIR metamodel. Note that while we focus on the FHIR RDF representation in this study, we believe that other clinical information models such as OpenEHR archetypes[12] and Clinical Element Models[13] can also be transformed into RDF for enabling semantic interoperability among systems that implement these information models, through leveraging the ShEx-based modeling and validation mechanisms developed in this study.

2 BACKGROUND

2.1 FHIR Resources

The FHIR specification defines a collection of "resources" which define the basic units of exchange for clinical information in the FHIR domain[1]. As of September 2016, the Standards for Trial Use (STU) 3 Ballot version of the FHIR core specification[14] included 109 resources for the exchange of information about care plans, risk assessments, patients, organizations, appointments, observations, etc. and three "infrastructure" resources: *Composition*, *Bundle* and *MessageHeader*. Figure 1 shows a portion of the FHIR model of the *Patient* resource.

FHIR resources are defined using the FHIR *StructureDefinition*[4] resource, which provides the metamodel for all FHIR resources including *StructureDefinition* itself. FHIR resource definitions are published in a human readable form through a set of custom FHIR-specific representations on the FHIR web site, including XML-like and JSON-like views and a Unified Modeling Language (UML)-like graphic format. A combination of XML Schema and Schematron provides an alternative representation of FHIR schema in XML. The FHIR *StructureDefinition* resource embodies the following four different model specification mechanisms:

1. *ElementDefinition* specifies model properties, types and cardinalities and defaults
2. *Constraints* provide rules about relationships and co-occurrence requirements between model elements
3. *Slicing* specifies uniqueness requirements across instance model elements and allows specific model instances to be referenced by value rather than name
4. *Extensions* allow additional elements to be added to existing models in the form of tag (uri)/value pairs.

2.2 FHIR Extensibility and Modifiers

The FHIR extensibility mechanism allows the cardinality, type or other aspects of a resource or any component thereof to be constrained. Optional attributes can be required or prohibited. The types or value ranges of elements can be restricted to a subset of those in the base resource. It is even possible to profile core data types, stating, as an example, that all integers in a given profile must be less than a certain value. The extensibility mechanism also allows new elements to be added to an existing resource. Instances of these added elements are represented as tag (url)/value pairs vs. "first class" named elements in the core model. As an example, the *us-core-direct* extension[15] adds a boolean flag indicating whether an e mail address can be used with the Direct protocol[16] for secure e-mail. Figure 2 shows that the difference in representation, with the core elements, "system", "value", "use", etc. are represented as JSON keys while the "direct" extension is represented using a url/valueBoolean pair.

The FHIR metamodel includes the notion of a "modifying element" – an attribute that "affects the interpretation of the element or resource that contains it." This attribute presents an interesting challenge as, along with the Open World Assumption (OWA) and the No Unique Name Assumption[17] it is assumed that each assertion (triple) on the semantic web is *monotonic* – it can stand on its own and its meaning won't change in the presence of another (possibly unstated) assertion. The FHIR notion of modification has the potential to violate this assumption, as a given statement may mean one thing by itself and something entirely different when accompanied by a modifier. As an example, the *Quantity* data type includes a numeric *value* and a *comparator* element. In the absence of *comparator*, *value* represents an actual value. When accompanied by a *comparator*, however, *value* represents the minimum or maximum of a value range.

2.3 Representation of FHIR resource instances in RDF

Resource Description Format (RDF) provides another possible representation of FHIR resource instances[18]. RDF's native graph representation allows information from disparate sources and structures to be combined into a single, federated collection of "linked open data". RDF provides a simple structure that can represent data, metadata, terminology and ontology as a single unified whole. RDF reduces the technological "impedance mismatch"[6], allowing developers to focus on domain problem space rather than on formatting and syntax issues. RDF is designed for "distributed extensibility"[19], allowing a consumer to cherry-pick comprehensible information without necessarily understanding it in totality. Adoption of RDF also allows one to use conventional RDF tooling to integrate knowledge from FHIR information models with healthcare ontologies such as SNOMED-CT[20], the Disease Ontology (DO)[21], the Foundational Model of Anatomy (FMA)[22], etc.

2.4 Shape Expressions (ShEx)

Shape Expressions (ShEx) is a concise, formal modeling language used to define possible sets of RDF triples. A collection of RDF triples (a "graph" or "dataset") can be tested against a given ShEx definition (schema) to determine whether the collection meets the requirements defined in the schema. Figure 3 shows a "nodes and arcs" diagram illustrating a RDF graph (a collection of RDF triples) in which an observation in some example schema has a status of "final" and a subject who in turn has a name and a birthdate.

Such data can be constrained by a schema in order to ensure data integrity. A ShEx schema contains one or more *shape* definitions. Validating an RDF node against a shape tests the adjacent nodes against the constraints in the shape. The constraints can be atomic triple constraints or boolean combinations thereof. A triple constraint identifies a specific predicate and describes the possible object values for that predicate. Object values may be restricted to a specific type, range, pattern or list (value set), or conformance to another shape, e.g. that an observation's subject conforms to a PatientShape as shown in Figure 4.

Figure 4 shows a ShEx schema defined for validating the trivial observation and patient schema introduced in Figure 3. There are two shapes specified in the schema. The first shape (ObservationShape) requires:

- Exactly one :status with a value in the value set ("prelim", "final");
- Exactly one :subject which references a user, here expressed as a value shape referencing PatientShape.

The second shape (PatientShape) states:

- Exactly one :name with a datatype of xsd:string;
- One or more :birthdate with a datatype of xsd:date.

This schema in Figure 4 can be used to detect violations in data. For instance, validating the data in Figure 5 against the schema in Figure 4 reveals two schema violations:

- Obs1's *:status* is not in the value set ["prelim" "final"];
- Patient2's *:birthdate* is expected to be an *xsd:date*, not an *xsd:dateTime*.

ShEx also allows a sequence of semantic actions to be associated with a schema, shape or triple constraint. These actions may be directly embedded in the schema and may be used to trigger external functions. Semantic actions serve two purposes – the first is to extend the expressivity of ShEx to compare two date fields in a data structure. The second purpose of a semantic action is similar to that of a parser such as YACC or ANTLR – to perform functions or emit output as an RDF graph is validated.

A UML diagram of the abstract syntax gives a reasonable idea of the expressivity of ShEx. Figure 6 shows an abbreviated UML model of the ShEx abstract syntax.

3 MATERIALS AND METHODS

3.1 Materials

3.1.1 FHIR StructureDefinition—The FHIR *StructureDefinition* resource is the metamodel for FHIR resource definitions. This means that other FHIR resources, such as Patient, are formally defined using an instance of *StructureDefinition* that declares elements like “Patient.name” and “Patient.birthDate”. Our goal is to transform an arbitrary *StructureDefinition* instance into a corresponding ShEx schema such that the RDF transformation rules summarized in section 4 are formalized and verifiable. The salient fields in the FHIR *StructureDefinition* resource include the resource url, which becomes its rdf:type, the name (e.g. *Observation*, *MedicationOrder*), the definition *kind*, and the ordered list of snapshot *ElementDefinitions.* Figure 7 shows the *StructureDefinition* resource definition in UML-like format.

StructureDefinition contains both a *differential* and *snapshot* collection of *ElementDefinitions*. The *differential* branch carries only the elements that have changed from the base definition, while the *snapshot* carries the result of the application of the differential to the parent. While FHIR resource definitions can have a differential and/or a snapshot definition, the specification states that “*Structure Definition* resources used in operational systems should always have the snapshot view populated.[23]” For this reason, ShEx definitions derive from the snapshot branch.

Each *ElementDefinition* entry contains a path, which becomes the RDF predicate, minimum and maximum cardinality, and either a type definition or a reference to a definition in another element. The *type.code(s)* identify the possible type(s) of particular element.

3.1.2 HL7 ITS/W3C RDF Task Force Decisions—RDF is not a “format” but is an abstract model that can be realized in a variety of actual formats including XML[24], Turtle[25], N3[26], etc. This document uses the RDF Turtle format to represent various examples with the understanding that the actual RDF content representation is an implementation decision.

Both the JSON and XML FHIR data representations are fully “round trippable”, meaning that an instance of a FHIR resource can be serialized in XML or JSON and then re-parsed into the same FHIR model from which it was derived without losing, adding or changing any information or its representation. The RDF representation of FHIR data is expected to meet the same requirement – it must be possible to convert any FHIR resource instance into RDF format and back to the FHIR abstract representation unchanged.

The various aspects of the HL7 ITS/W3C RDF Task force decisions are summarized below. (see: supplementary material S1 for detail)

- **RDF header** – the namespaces used in the RDF representation of FHIR, e.g., “fhir:” and “xsd:”.
- **Resource subject** - resource subject, type and document level identifier. The RDF representation of a FHIR resource as identified by the *StructureDefinition kind* code, “resource”, is specified by the RDF working group.
- **Resource elements** - representation of *ElementDefinition* paths and types as predicates and objects. A FHIR resource instance consists of one or more “elements”, in the form of a tag/value pair, where the tag is a simple identifier and the value can either be a primitive (atomic) value, a nested set of tag/value pairs or an ordered list of values. Because tag name are not unique within a FHIR resource or data type in which they appear, the corresponding RDF property names reflect the nesting of each property, for instance, *Observation.code* vs. *Observation.component.code*. While RDF provides the ability to nest structures, the identity of an RDF predicate is global. This means that the predicates themselves must be fully qualified. For n-ary elements, the RDF working group decided not to use RDF collections to represent ordered lists in FHIR and, instead, used an integer index that provided a relative ordering.
- **Primitive types** - representation of the 16 FHIR primitive types. Each of these types is mapped to a corresponding XML and JSON representation. The RDF working group defined the corresponding data types used in the RDF representation.
- **Choices** - representation of FHIR choice elements. The type of many FHIR elements can be chosen from two or more alternatives. For example, in the MedicationOrder resource, the value of the *dose* element can either be a *Range* or a *SimpleQuantity*. FHIR resources can reference instances of other resources. The RDF working group specified the corresponding RDF representations.
- **Resource references** - references to other resources by name and URI. References can be constrained to a single resource type (e.g. the *patient* tag in the *MedicationOrder* resource must reference a resource of type *Patient*) or a list of possible types (e.g. the *reporter* tag in a FHIR *AllergyIntolerance* resource can reference a resource of type *Patient, RelatedPerson* or *Practitioner*). The RDF working group specified the corresponding RDF representations.
- **Concept code references** – references to terminology codes. The FHIR *CodeableConcept* and *Coding* data types represent a second area where the “traditional” modeling approaches potentially collide with the notion of “linked data paradigm” embodied in the RDF approach to representation. RDF uses the URI as the fundamental mechanism to associate (link) information. When it

comes to codes, however, FHIR uses a compound structure that includes of the URI of the coding system and a separate field for the actual concept code. This representational form, by itself, does not contain enough information to generate a concept URI without relying on outside knowledge. As an example, the FHIR representation SNOMED CT concept of *appendicitis* would have a code system URI of http://snomed.info/sct and a concept code of *74400008*. The corresponding SNOMED CT URI, as defined by the SNOMED publishers would be http://snomed.info/id/74400008. The RDF working group addressed this disparity by allowing an optional *fhir:concept* attribute that contains the complete IRI(s) of the referenced concepts. Implementations that are intended for use in the linked data space are encouraged to add this attribute when the actual URIs are known.

3.2 Methods

Figure 8 shows the components of our FHIR ShEx transformation approach used in this study. Specifically, we first define the FHIR ShEx transformation rules through examining the FHIR RDF decisions documents created by the HL7 ITS/W3C RDF Task Force. We then implement the FHIR ShEx transformation tools and evaluate the utility of the FHIR ShEx schemas leveraging the ShEx validation tools publicly available in the ShEx community.

3.2.1 Define the FHIR ShEx transformation rules—We reviewed the existing FHIR RDF decisions documents available from the HL7 ITS/W3C RDF Task Force. Based on the decisions, we reviewed the elements of both *StructureDefinition* and *ElementDefinition*, and created a minimal set of elements and defined the FHIR ShEx transformation rules for the elements.

3.2.2 Implement the FHIR ShEx transformation tools—We began from a prototype FHIR to RDF conversion developed by EP and JCM[27] and extended it to generate both FHIR instance data in RDF and FHIR ShEx schemas[28]. The result of this prototype was used to guide and validate the discussions of the HCLS FHIR/RDF task force. Once an initial approach was established, the RDF and ShEx generation processes were incorporated directly into the FHIR build process[29, 30].

3.2.3 Evaluation Design—We took the set of core STU3 FHIR example instances (as of September, 2016), and validated them against the corresponding FHIR ShEx schemas using the ShEx node.js REST validation service[31]. We collected and analyzed the error reports, using a web-based ShEx validation service[32] to pinpoint the exact cause, which could be derived from:

- FHIR to RDF conversion errors or misinterpretations
- FHIR *StructureDefinition* to ShEx conversion errors or misinterpretations
- Ambiguities or incorrect statements in the task force documentation

We corrected errors as needed and repeated this procedure until no errors remained.

4. RESULTS

4.1 ShEx representation of FHIR *StructureDefinition* Resource

Table 1 lists the components of *StructureDefinition* and *ElementDefinition* that describe the actual content of a resource instance how they are defined using ShEx. The *Impl?* column identifies which aspects were implemented for the current evaluation. Note that the "facets" referred to in the table refer to the standard XML Schema facets[33] as used in the ShEx specification. The element names in ShEx are being selected by a HL7/W3C standardization process so people are expected to be compatible with these choices as they are being published in as part of the FHIR standard.

The notes in Table 1 reflecting the transformation decisions in ShEx are summarized as follows.

Note 1: A *path* is a dot ('.') separated list of identifiers. Paths that form the root of nested structures do not have an associated *type* entry (e.g. "*Observation*", "*Substance*", "*Observation.referenceRange*") and, as a consequence, are ignored by the ShEx constraint process. Root paths represent ShEx shapes – *Observation* is the Observation shape, *Observation.referenceRange* the Observation.referenceRange shape. While not strictly necessary, the ShEx definition of nested shapes are named instead of being represented as anonymous inner shapes.

Note 2: Slicing imposes additional constraints over the values of a collection. The slicing constraint usually assigns a *discriminator* - an attribute or set of attributes that must be unique within the context of the collection and includes assertions about the collection ordering, mandatory and optional contents.

Note 3: Elements with no *type* or *contentReference* property are ignored. A single *type* entry produces a simple tripleConstraint whose target is the referenced shape. Choice elements, which are identified by an "[x]" on the end of the path are expanded as an anonymous inner shape, with each unique predicate (generated by replacing the '[x]' with the name of the target type) object combination represented as an alternative.

Note 4: The *profile* attribute establishes additional constraints on the possible values of the attribute. As an example, the *SimpleQuantity* profile adds the constraint that there is no comparator ('<', '>=') by changing the max cardinality of *Quantity.comparator* to "0". The "snapshot" part of a *StructureDefinition* contains the result of the application of any constraints to the underlying type. As a result, *profile* constraints are automatically incorporated into the resultant ShEx definitions.

Note 5: The *condition* attribute references an optional collection of invariants (constraints). The actual constraints are expressed in XPath and, optionally, FluentPath, which allows the expression of arbitrarily complex path based invariants. As of September 2016, there were 448 invariants asserted in the core FHIR model, but only seven of these involved the comparison of two different paths within the constraint itself. The remainder of the invariants involved combinations of existential and content restrictions, which could be readily translated into the equivalent logical combinations of ShEx sub-shapes.

Note 6: FHIR "binds" coded elements to "value sets", which are lists of allowed (or suggested) values. This binding is accompanied by a *binding strength* attribute that states how the value set should be interpreted. FHIR currently defines four binding strengths: *Required, Extensible, Preferred* and *Example*. Of these, *Required* is the only strength that has implications for validation as the remaining types actually assert that *any* code could be valid[*]. The ShEx generator that was included in STU3 assumed that all *Required* bindings were of type *code*. This turned out to not be the case as there were seven resource definitions that had one or more *Required* value set bindings to *Coding* or *CodeableConcept*. We manually edited those resource definitions with the correct bindings before we ran the final test. The FHIR *code* data type consists of a single string. This could be directly validated against a list of possible strings in ShEx. The *Coding* data type, however, carries a complex structure. The RDF mapping includes the capability to represent this structure as a single URI, meaning that it would be possible to define validation value sets as lists of URI's. An issue, however, is that the URI is optional and would not be used when the prefix is unknown. If this is taken into account, ShEx value set validation may become considerably more verbose.

4.2 Transformation Tools

The FHIR to RDF conversion, TurtleParser.java,[29] implements the decisions of the FHIR task force and published the FHIR examples on the FHIR web site. As an example, the RDF rendering of the FHIR medication order that includes the dosage of a prescription in text can be found at http://hl7.org/fhir/2016Sep/medicationorderexample2.ttl

The ShEx generator, ShExGenerator.java[30], generates the corresponding ShEx definitions. As an example, the corresponding definition for the FHIR medication order can be found at http://hl7.org/fhir/2016Sep/medicationorder.shex.html

The RDF validation was accomplished using the ShEx validation REST server[31] and, when specific issues needed resolution, version 2 of the online ShEx workbench[32] (see Figure 9).

4.3 Evaluation Results

The draft FHIR Standard for Trial Use (STU3) was used for the evaluation. As of September 2016, there were 112 core FHIR resource definitions and 511 examples. The resource definitions were converted to their ShEx equivalent. Two of these (*DomainResource* and *Resource*) were abstract definitions that had no direct instances and one, *CompartmentDefinition*, had no examples. All of the 109 remaining resources had at least one example, with 43 resources having exactly one and *MedicationOrder* having the most (n=32). Figure 10 shows the distribution of the number of examples bythe FHIR resources. FHIR profiles, such as the US core and extension definitions were not evaluated at this time, as the RDF group was still working on these definitions.

There were a total of 511 examples that were converted and validated against the corresponding ShEx definitions using the shex.js REST validation service. None of the 511

[*]If the validation process were extended to include warnings, it might be useful to issue a warning about an *Extensible* code that wasn't in the supplied list, but it couldn't be flagged as an error.

examples passed the initial validation pass but after several iterations, we were able to address all of the conversion errors and arrive at interim solutions to all of the remaining issues. Table 2 shows a list of errors identified by validating the FHIR examples (n=511) against FHIR ShEx schemas (n = 109). In total, we detected 10 types of errors. We also analyzed the causes for each error type and provided solutions for all but 2 types of errors – Resource Reference errors (n=18) and ShEx EXTRA errors (n=1).

4.4 Additional Observations

The introduction of the ShEx models into the standardization process resulted in significant, positive changes to the group's progress. ShEx was first introduced on a "FHIR Constellations[34]" page[†], where either a simplified FHIR model or the ShEx equivalent could be viewed alongside sample RDF instances. The ShEx exposed many of the group decisions about cardinality, optionality and type variants – decisions that were otherwise difficult to represent through simple text and collections of RDF samples. It also allowed the group to see the complete FHIR model and RDF options in a succinct, side-by-side format. As the results, the value of ShEx has been demonstrated, and it has been decided that the FHIR ShEx representations are included in the official FHIR web pages for the STU 3 Ballot version since September 2016, which have then become the primary source for ShEx representations.

5. DISCUSSION

In this study, we demonstrated that ShEx is a good candidate for formally describing the FHIR metamodel when it is realized as RDF data instances. The combination of the RDF data model and ShEx schema has the potential to combine the disparate FHIR core, extension, slicing and constraint modeling mechanisms under a single, unified model.

5.1 Why impose constraints in RDF rather than XML?

The FHIR/RDF task force's decisions for the RDF representation can be documented in an ad-hoc way through a series of examples, much as has been shown above. The challenges with this approach include:

- Completeness – have the examples covered all possible cases? Are there representational structures or combinations thereof that may not have been addressed using selected examples?
- Verifiability – how does one determine whether an independently developed toolset meets the requirements specified in the examples above? How does one verify that the generated RDF is valid?

The first challenge can be addressed on the metamodel level, using FHIR models mapped to format-specific schema languages. FHIR models are defined using instances of FHIR *Structure Definition* resource. In the case of the XML representation, examples are accompanied by corresponding XML Schema and Schematron definitions that formally

[†]Note that the "Constellations" web page has not been maintained since the ShEx representation was moved to the official FHIR pages. Some of the information on this page is no longer current.

describe the rules for an XML representation of a given resource instance. Likewise, the FHIR/RDF examples are accompanied by and verified by ShEX schemas mechanically transformed from the FHIR *Structure Definition*.

When it comes to verifiability, one approach would be to demonstrate that data can be round-tripped by a single software tool, e.g. from FHIR/XML to RDF and back, and then show that the starting and end results are identical. The problem with this, however, is that any number of RDF representations could meet this challenge as long as the same developer was responsible for both transformation and interpretation. There is no guarantee that two developers would interpret the same RDF in the same fashion and, even if they did, that the specification was precise enough that subsequent tooling developers would also interpret them in exactly the same way.

In the case of the XML representation of FHIR, the verifiability can be addressed more directly – if one can demonstrate that a bidirectional mapping exists and this demonstration is accompanied by a formal (XML Schema in the case of XML) definition of the target model, then tooling developers only need demonstrate that (a) the transformation rules from FHIR to XML have been followed and (b) that the target XML can be validated by the accompanying XML Schema. ShEx provides the equivalent direct verifiability for the RDF representation.

5.2 Benefits of native RDF constraints using ShEx?

RDFS and OWL are ontology languages. Both of these languages provide inference, e.g. "If Fido is a dog and all dogs are mammals, then Fido is a mammal." These languages, however, are the *interpretation* of a set of statements, not the form of the statements themselves. As an example, the RDFS assertion that the range of the predicate "*drives*" is "*automobile*", when applied to the statement "*Jack drives an orange*", results in the conclusion that "*orange is an automobile*" not an error condition. Similarly, the assertion "*Person has exactly one father*" in OWL, when applied to the statements, "*Sally has father Joe*" and "*Sally has father Sam*" would result in a conclusion that Joe and Sam were the same person, and would not result in a cardinality violation[‡].

ShEx is <u>not</u> an ontology language. It makes assertions about statements themselves, not their interpretations. The ShEx assertion "*<Person> { :has Father . }"* states that a given subject must appear in exactly one triple whose predicate is "*:hasFather*". It says *nothing* about persons, fathers, or their relationships. ShEx is used to define and validate the syntax of a FHIR resource. RDFS and OWL are used to define and validate its semantics.

As described in the background section FHIR uses (at least) four different modeling paradigms - *ElementDefinition, Constraints, Slicing and Extensions.* While we focused on the core *StructureDefinition*/*ElementDefinition* model in this study, we believe that ShEx can be used to represent all of four paradigms in a single, (relatively) easy to understand grammar. The current implementation status is shown in Table 1.

[‡]Adding the assertion that Joe and Sam were different individuals would cause the OWL reasoner to conclude that the ontology was inconsistent, not that a specific cardinality constraint had been violated.

6. CONCLUSION

ShEx provides an effective way to formally specify how (FHIR) data should be represented in RDF. The FHIR *StructureDefinition* to ShEx transformations assured that all the transformation options had been covered, and the ability to validate sample data instances against their corresponding ShEx definitions helped uncover multiple issues and errors in the transformation process. The FHIR ShEx representations have been included in the official FHIR web pages for the STU 3 Ballot version since September 2016. The script used to evaluate the ShEx definitions can be found at: https://github.com/caCDE-QA/FHIRShExTest.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

The manuscript is an expanded version of a podium abstract presented in the AMIA Clinical Research Informatics (CRI) 2016 conference. We are thankful for the input from the HL7 ITS/W3C RDF Task Force.

FUNDING

This work has been supported in part by funding from caCDE-QA (U01 CA180940).

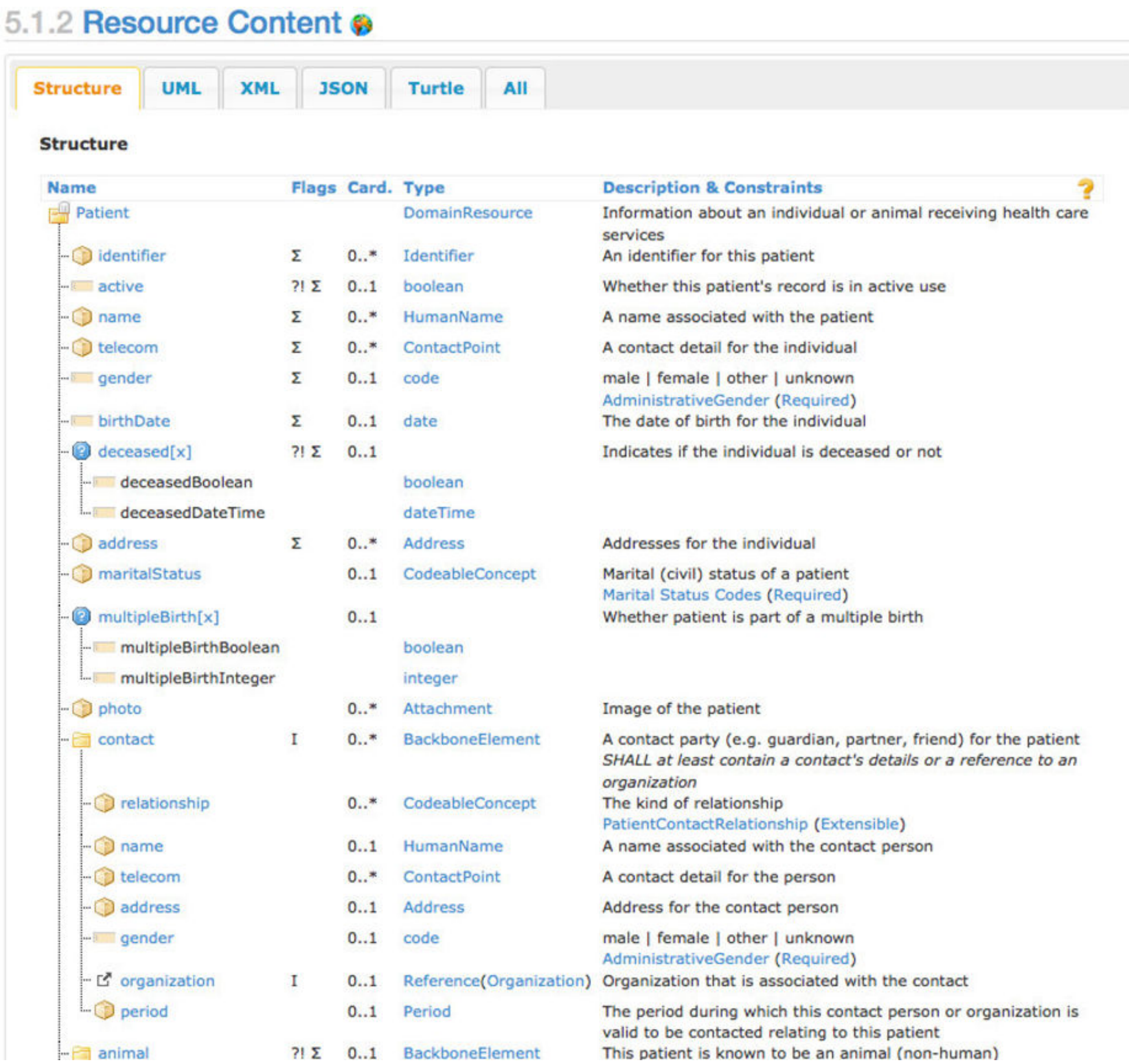
References

1. HL7 FHIR DSTU 2. 2016. [Available from: http://hl7.org/fhir/index.html
2. Alterovitz G, Warner J, Zhang P, Chen Y, Ullman-Cullere M, Kreda D, et al. SMART on FHIR Genomics: Facilitating standardized clinico-genomic apps. Journal of the American Medical Informatics Association : JAMIA. 2015
3. Warner JL, Rioth MJ, Mandl KD, Mandel JC, Kreda DA, Kohane IS, et al. SMART precision cancer medicine: a FHIR-based app to provide genomic information at the point of care. Journal of the American Medical Informatics Association : JAMIA. 2016
4. FHIR StructureDefintion. 2016. [Available from: http://hl7.org/fhir/structuredefinition.html
5. W3C Resource Description Framework (RDF). 2016. [Available from: http://www.w3.org/RDF/
6. Yosemite Manifesto. 2016. [Available from: http://yosemitemanifesto.org/
7. XML Schematron. 2016. [Available from: http://www.schematron.com/
8. Shape Expressions Langauge. 2016. [Available from: http://shex.io/
9. Staworko, S., Boneva, I., Prud'Hommeaux, E., Hym, S., Gayo, JEL., Solbrig, HR. Expressiveness and Complexity of ShEx for RD. EDBT/ICDT 2015 Joint Conference; March 23–27, 2015; Brussels, Belgium.
10. W3C RDF Data Shapes Working Group. 2016. [Available from: http://www.w3.org/2014/data-shapes/wiki/Main_Page
11. HL7 ITS/W3C HCLS RDF Workgroup. 2016. [Available from: http://wiki.hl7.org/index.php?title=RDF_for_Semantic_Interoperability
12. OpenEHR. 2017. [Available from: http://www.openehr.org/
13. OpenCEM Browser. 2017. [Available from: http://www.opencem.org/-/
14. The STU 3 Version of FHIR Resources. 2016. [Available from: http://hl7.org/fhir/2016Sep/resourcelist.html
15. Extension: Email is a "direct" email. 2016. [Available from: http://hl7.org/fhir/2016Sep/extension-us-core-direct.html
16. The Direct Project. 2016. [Available from: http://directproject.org/

17. Open World Assumption. 2006. [Available from: http://www.cs.man.ac.uk/~drummond/presentations/OWA.pdf
18. FHIR RDF Representation. 2016. [Available from: http://hl7.org/fhir/2016Sep/rdf.html
19. RDF 1.1 Semantics - Extensions. 2016. [Available from: http://www.w3.org/TR/rdf11-mt/-extensions
20. SNOMED CT. 2016. [Available from: http://www.ihtsdo.org/snomed-ct
21. Disease Ontology. 2016. [Available from: http://disease-ontology.org/
22. Foundational Model of Anatomy (FMA). 2016. [Available from: http://si.washington.edu/projects/fma
23. FHIR StructureDefinitions Snapshot. 2016. [Available from: http://www.hl7.org/FHIR/profiling.html-snapshot
24. RDF 1.1 XML Syntax. 2016. [Available from: https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/
25. RDF 1.1 Turtle. 2016. [Available from: https://www.w3.org/TR/2014/REC-turtle-20140225/
26. RDF 1.1 N-Triples. 2016 Available from: RDF 1.1 N-Triples.
27. FHIR RDF Transformation By Josh Mandel. 2016. [Available from: http://github.com/jmandel/fhir-rdf
28. FHIR RDF/ShEx Transformation. 2016. [Available from: http://github.com/w3c/hcls-fhir-rdf
29. The FHIR to RDF Conversion Tool. 2016. [Available from: http://github.com/hl7-fhir/fhir-svn/blob/master/tools/java/org.hl7.fhir.dstu3/src/org/hl7/fhir/dstu3/elementmodel/TurtleParser.java
30. ShEx Generator. 2016. [Available from: http://github.com/hl7-fhir/fhir-svn/blob/master/tools/java/org.hl7.fhir.dstu3/src/org/hl7/fhir/dstu3/utils/ShExGenerator.java
31. ShEx Validation REST Server. 2016. [Available from: http://github.com/shexSpec/shex.js/tree/master/rest
32. FancyShExDemo. 2016. [Available from: http://github.com/shexSpec/shex.js/tree/master/rest
33. XML Schema Facet. 2016. [Available from: http://www.w3.org/TR/xmlschema-2/-facets
34. Constellations of related FHIR Resources. 2016. [Available from: http://www.w3.org/2016/FHIR-tutorial/Constellations

Highlights

- The ShEx models contributed significantly to workgroup consensus.
- Transformations from FHIR to ShEx were incorporated into the FHIR build process.
- 10 types of errors and root causes were detected in the actual implementation.
- The FHIR ShEx representations are included in the official FHIR STU 3 web pages.
- ShEx proved useful for describing a standard model of FHIR RDF data.



5.1.2 Resource Content

Structure | UML | XML | JSON | Turtle | All

Structure

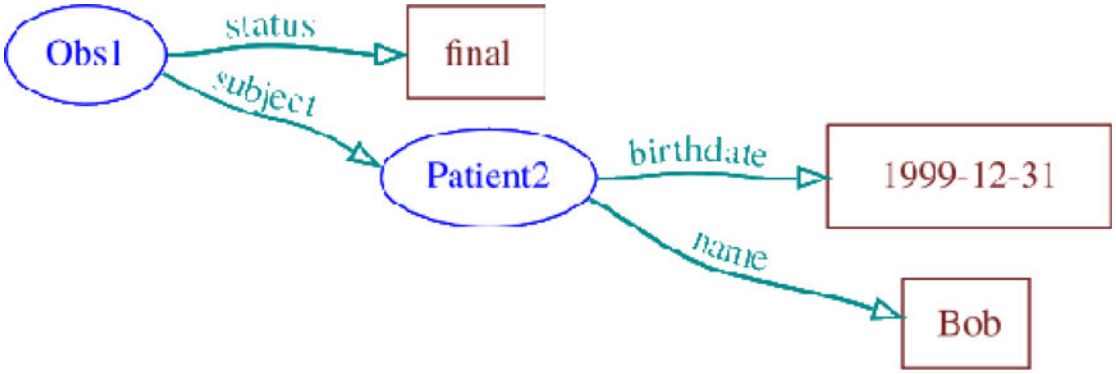
Name	Flags	Card.	Type	Description & Constraints
Patient			DomainResource	Information about an individual or animal receiving health care services
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	Addresses for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient Marital Status Codes (Required)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend) for the patient *SHALL at least contain a contact's details or a reference to an organization*
relationship		0..*	CodeableConcept	The kind of relationship PatientContactRelationship (Extensible)
name		0..1	HumanName	A name associated with the contact person
telecom		0..*	ContactPoint	A contact detail for the person
address		0..1	Address	Address for the contact person
gender		0..1	code	male | female | other | unknown AdministrativeGender (Required)
organization	I	0..1	Reference(Organization)	Organization that is associated with the contact
period		0..1	Period	The period during which this contact person or organization is valid to be contacted relating to this patient
animal	?! Σ	0..1	BackboneElement	This patient is known to be an animal (non-human)

Figure 1.

FHIR model of Patient Resource

```
"telecom": [
  {
    "system": "phone",
    "value": "(03) 5555 6473",
    "use": "work"
  },
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/us-core-direct",
        "valueBoolean": true
      }
    ],
    "system": "email",
    "value": "person@direct.example.org",
    "use": "work"
  }
],
```

Figure 2.
FHIR Extension Representation in JSON




```
PREFIX : <http://example.com/clinic/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

<Obs1>
  :status    "final" ;
  :subject   <Patient2> .

<Patient2>
  :name      "Bob" ;
  :birthdate "1999-12-31"^^xsd:date .
```

Figure 3.
(A) A diagram illustrating an example of an RDF graph recording an observation; (B) The same RDF graph in the RDF Turtle language.

```
PREFIX : <http://example.com/clinic/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

start = <ObservationShape>

<ObservationShape> {                # An Observation has:
  :status    ["prelim" "final"]; #   status in this value set
  :subject   @<PatientShape>     #   a subject matching <PatientShape>.
}

<PatientShape> {                    # A Patient has:
 :name       xsd:string*;           #   one or more names
 :birthdate  xsd:date?              #   an optional birthdate.
}
```

Figure 4.
Our example schema expressed in ShEx.

```
PREFIX : <http://example.com/clinic/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

<Obs1>
  :status    "finally" ;
  :subject   <Patient2> .

<Patient2>
  :name      "Bob" ;
  :birthdate "1999-12-31T01:23:45"^^xsd:dateTime .
```

Figure 5.

An RDF graph with two violations of the example schema.

Author Manuscript
Author Manuscript
Author Manuscript

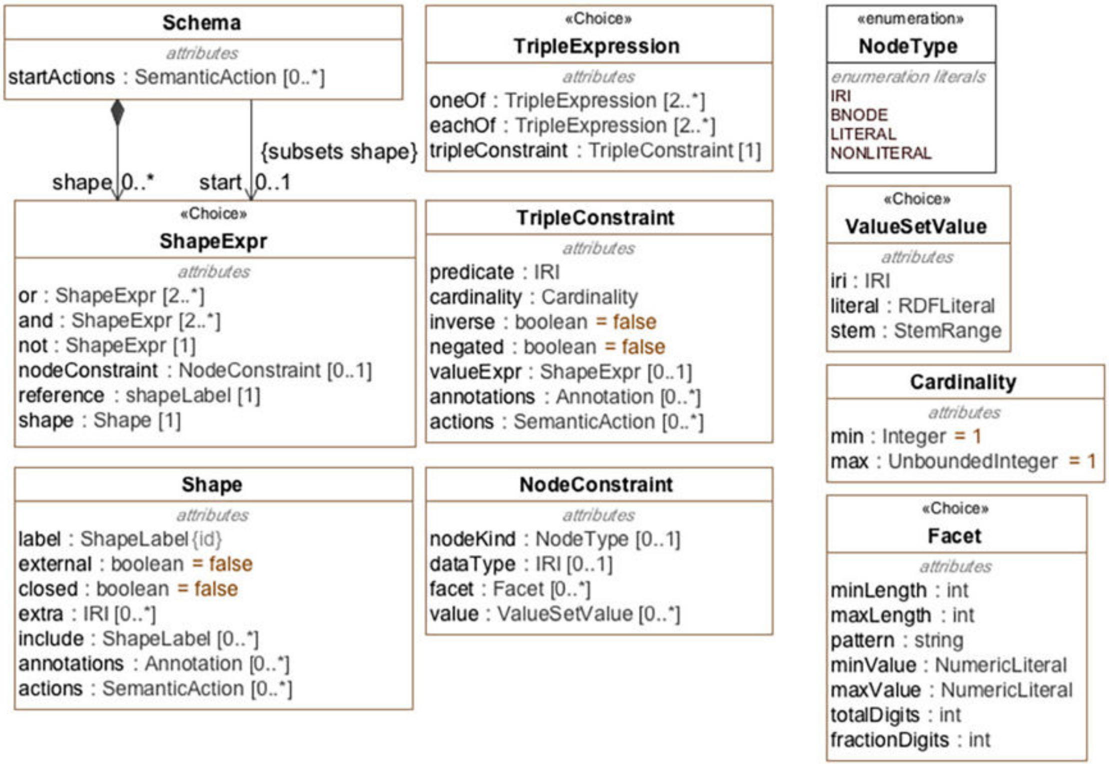


Figure 6.
The ShEx abstract syntax in UML

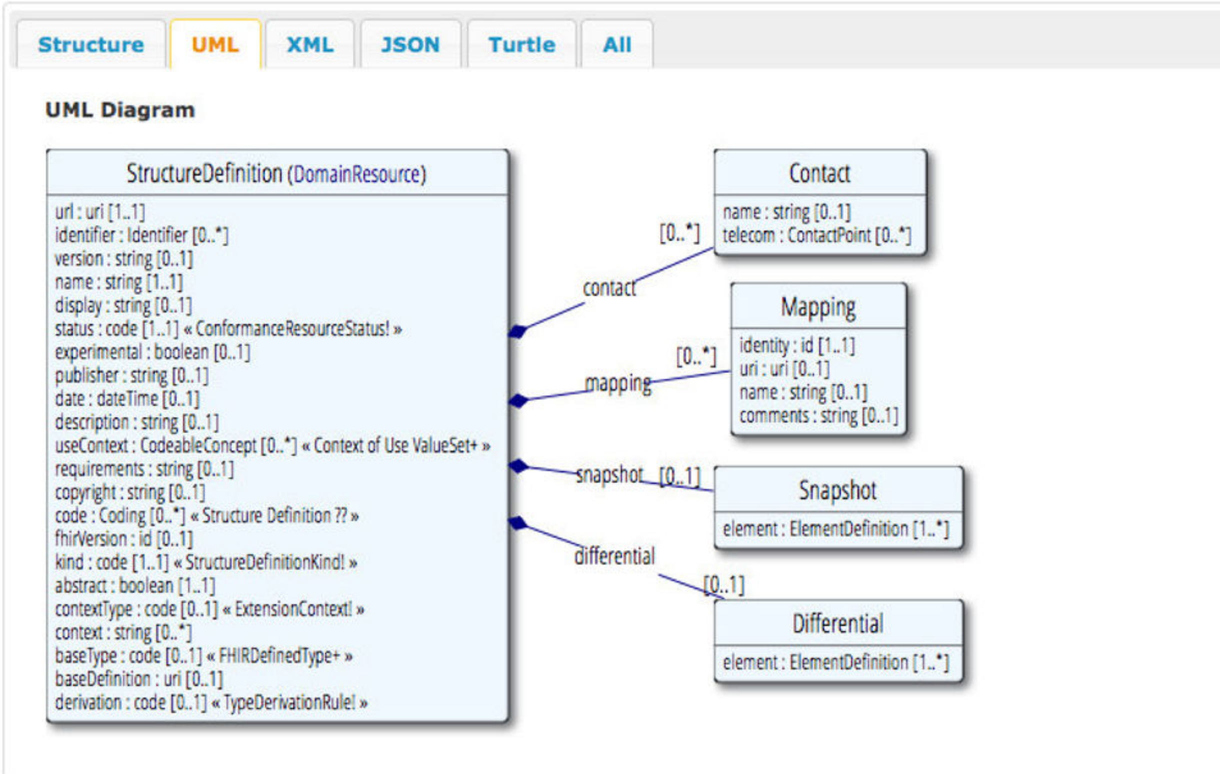


Figure 7.

The FHIR *StructureDefinition* Resource

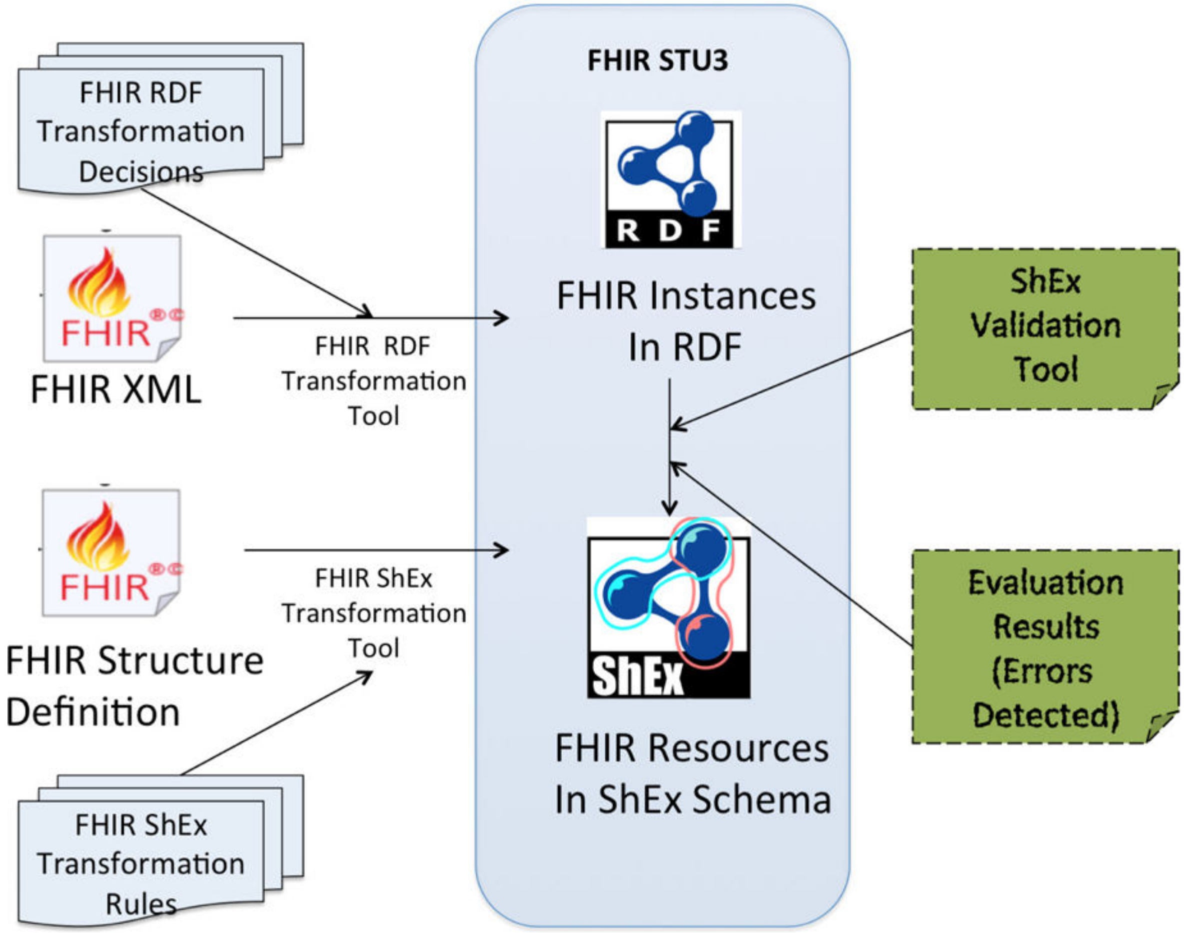


Figure 8.
The components of our FHIR ShEx transformation approach

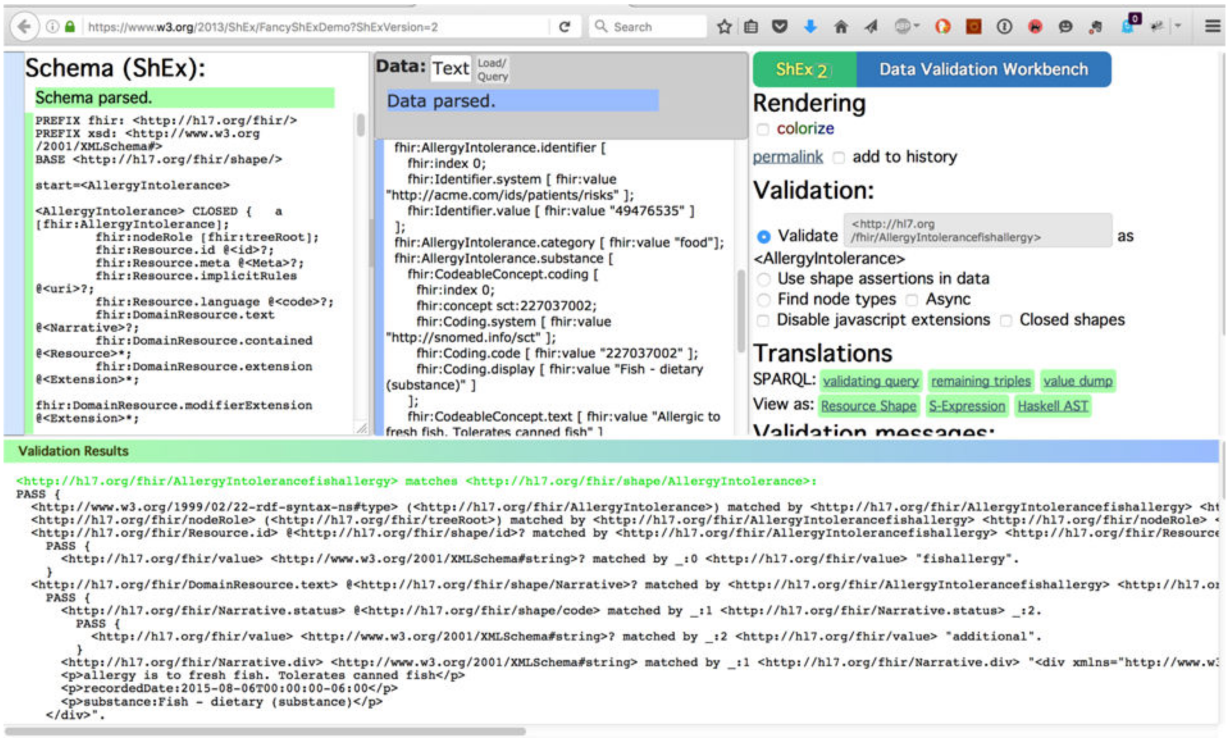


Figure 9.
Fancy ShEx Demo Screenshot

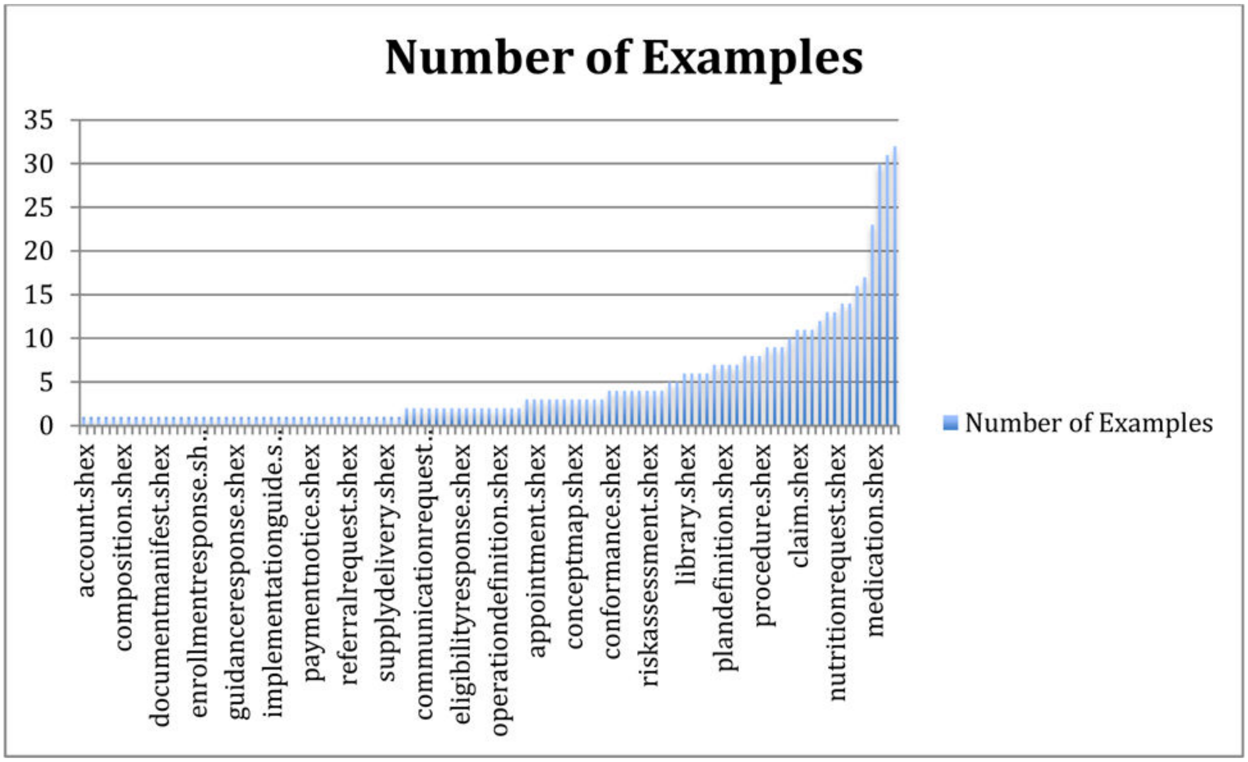


Figure 10.
FHIR Resource Example Distribution

Table 1

The components of FHIR Structure Definition and Element Definition and their corresponding transformation decisions in ShEx.

Model	Element	Purpose	ShEx	Impl?
StructureDefinition	url	Resource identifier	rdf:type [*url*];	Y
	name	Natural language name of resource	ShEx shape name	Y
	kind	Identify whether the definition is a resource, primitive-type, complex-type or logical model	fhir:nodeRole [fhir:treeRoot]; if top level and *kind*=resource	Y
	differential	Definition of the changes in the resource	(Defines changed triples)	NA
	snapshot	Definition of elements in the resource	(Defines the individual triples)	Y
ElementDefinition	path	The path of the element	See: Note 1	Y
	slicing	The element is sliced	See: Note 2	N
	min	Minimum Cardinality	ShEx cardinality	Y
	max	Maximum Cardinality	ShEx cardinality or omit entire property if "0"	Y
	contentReference	Reference to definition of content for the element	(must be de-referenced to get actual type)	Y
	type	Data type and Profile for the element	See: Note 3	Y
	type.code	Name of Data type or Resource	Target shape if no profile attribute	Y
	type.profile	Profile to apply (or IG)	See: Note 4	Y
	type.aggregation	Contained | referenced | bundled (Not used in FHIR's base resource definitions)	N/A	N
	fixed[x]	Value must be exactly this	(single member value set for simple types. TBD for structures)	Y
	pattern[x]	Value must have at least these property values	PATTERN facet	Y
	minValue[x]	Minimum Allowed Value	MININCLUSIVE facet	Y
	maxValue[x]	Maximum Allowed Value	MAXINCLUSIVE facet	Y
	maxLength	Max length for strings	MAXLENGTH facet	Y
	condition	Reference to invariant about present	See: Note 5	N
	constraint	Condition that must evaluate to true	See: Note 5	N
	binding	ValueSet details if coded	See: Note 6	Y
	binding.valueSet[x]	Source of value set	See: Note 6	–

Table 2

Different types of errors identified by validating the FHIR examples (n=511) against FHIR ShEx schemas (n=109).

Problem	Cause	Solution
@prefix xs: in header, *xsd:* in body	Error in FHIR to RDF converter	Fix header
FHIR elements with data type *xhtml* were not emitted.	Left as a "to do" in the FHIR to RDF converter.	Discussed in the HCLS task force. Emit *xhtml* as an *xsd:string*.
fhir:code data type represented as *xsd:token* in ShEx, but *xsd:string* in RDF	Subtle difference between the interpretation of types in XML and RDF. In XML, types are not explicitly declared – everything is a string. In RDF, the type of a Literal must be declared, and the string *"abc"* is not the same as the token *"abc"^^xsd:token*.	Created an RDF extension for the FHIR core data types to be used by the ShExGenerator and mapped *fhir:code* to *xsd:string* instead of *xsd:token*
fhir:anyURI data type represented as *xsd:anyURI* in ShEx but *xsd:string* in RDF	(Same issue as *xsd:token* above)	Corrected FHIR RDF generator to *fhir:anyURI* to *xsd:string* and add an additional match pattern.
Multiple *fhir:concept* triples were generated in the root of the CodeableConcept structure, an artifact of an earlier decision that had been reversed.	Different interpretations of the semantics of the *coding* element in CodeableConcept.	Brought up with the HCLS task force. Still under active discussion, but removed from the FHIR to RDF converter until resolved.
fhir:integer, fhir:unsignedInt and *fhir:positiveInt* all mapped to *xsd:int* in RDF	Error in FHIR to RDF converter	Map *fhir:integer* to *xsd:int, fhir:positiveInt* to *xsd:positiveInteger* and *fhir:unsignedInt* to *xsd:nonNegativeInteger*.
Bundle.entry.resource, *DomainResource.contained* and *Parameters.parameter.resource* reference untyped resources.	*Any* FHIR resource can appear in these slots.	Complete validation requires that all FHIR resource definitions appear in the ShEx Schema, as exemplified by *fhir.shex*. An alternative would be to declare Resource to be "open" in ShEx, meaning that it would still be considered valid in the presence of additional triples.
Unknown predicate-*Resource.contained*	Error in FHIR to RDF converter	Change to *DomainResource.contained*.
Reference type could not be determined from URL	RDF specification currently calls for an rdf:type arc. The rdf:type could not be determined for non-FHIR resource links	Recommendation to weaken final specification and create a FHIR profile for linked data compatible instances with stronger rules. Recommendation to FHIR community to add a *type* attribute to a resource reference so the type can be determined even when the link is not dereferencable
Multiple Coding's in a Required CodeableConcept	Error in ShEx validator – "EXTRA" directive not being processed correctly.	Will be fixed in subsequent release