DevDays 10th Edition

HL7® FHIR® DevDays

NOV | 2020

The Use and Limits of Validation

Grahame Grieve, Health Intersections / HL7

HL7 FHIR DevDays 2020, Virtual Edition, November 17–20, 2020 | @FirelyTeam | #fhirdevdays | www.devdays.com/november-2020

ORGANIZED BY firely

Who am I?

- Grahame Grieve
- FHIR Product Director
- FHIR Community Lead
- Author of:
  - FHIR Schemas (XML & JSON)
  - FHIR Validator

Learning Objectives

- Learn what validation achieves
- Know how to check that resources and systems are correct
- Learn the limits of validation – what it does not achieve
- Choose the right validation approach for your needs and constraints

Validating a resource

- **Structure**: All the content as described, nothing extra
- **Cardinality**: Check min & max of all properties
- **Value Domains**: Check all specified type values
- **Vocab bindings**: Check codes & displays are correct
- **Invariants**: Check that co-occurrence rules, etc are correct
- **Profiles**: Check that profiles are correct
- **Definition vs Instance**: e.g. Check a QuestionnaireResponse against matching Questionnaire
- **Business Rules**: Checking duplicates, references, authority etc

Validation Approaches

- Format Specific Schema:
  - XML Schema - generated with the specification
    - +Schematron – also generated
  - JSON schema – generated with the specification
  - Shex (RDF schema) – under development
- FHIR Specific Validator – Custom code written to validate resources
- FHIR Servers – Server validates a resource, for a specific operation/context

Validation Approaches

Method	XML	JSON	RDF	Structure	Cardinality	Values	Bindings	Invariants	Profiles	Definitions	Business Rules
XML Schema	✓			✓	✓	✓					
XML Schema + Schematron	✓			✓	✓	✓		✓	✓[1]		
JSON Schema		✓		✓	✓	✓			✓[2]		
ShEx			✓	✓	✓	✓	✓[3]				
Validator	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Validation Operation[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

XML Schema

- 2 x 2 sets of schema provided:
- Structure:
  - All in one large single schema (fhir-single.xsd)
  - Modular schema (fhir-all.xsd) (all depend on each other)
- Function:
  - Validation Schemas – as much validation as possible in XML schema
  - Code Generation Schemas – less validation (no xs:choice)
- Validate XML syntax, little value validation
- No schemas for profiles (schema is name dependent)

Schematron

- Generated from invariants in the specification
- Not all invariants have xpath equivalents
- Strictly an add-on to the main XML schema (no duplication)
- Not used much

- Schematron are generated for profiles, but limited
  - No support for slicing
  - no terminology validation

JSON Schema

- Provisional – due to issues with Json Schema tooling
- One single large schema (issues with json schema modularity)
- Schema depends on resourceType – error messages may be very difficult to interpret
- Splitting primitives into field and _field makes schema weaker
- Validate Json syntax, little value validation

- Could generate JSON schema for some parts of profiles, but not done yet

Software Validator

- Official FHIR Validator:
  https://github.com/hapifhir/org.hl7.fhir.core/releases
  https://confluence.hl7.org/display/FHIR/Using+the+FHIR+Validator

- A jar file you use to validate resources:

```
java -jar validator_cli.jar c:\temp\patient.xml -version 3.0
 -ig hl7.fhir.us.core#1.0.1 -profile http://hl7.org/fhir/us/core
```

- Part of the HAPI FHIR core, used for the core specification

Validator Output

```
FHIR Validation tool Version (5.1.22)
  Java:   11.0.8 from C:\Program Files\Java\jdk-11.0.8 on amd64 (64bit). 16344MB available
  Paths:  Current = c:\temp, Package Cache = C:\Users\graha\.fhir\packages
  Params: C:\temp\validator\poc.json -version 4.0
Loading
  Load FHIR v4.0 from hl7.fhir.r4.core#4.0.1 - 4575 resources (00:03.0615)
  Load hl7.terminology#2.0.0 - 3749 resources (00:01.0124)
  Terminology server http://tx.fhir.org - Version 1.0.362 (00:02.0225)
  Get set...  go (00:00.0040)
Validating
  Validate C:\temp\validator\poc.json 00:00.0143
Done. Times: Loading: 00:07.0163, validation: 00:00.0144

*FAILURE*: 3 errors, 1 warnings, 2 notes
 Error @ DetectedIssue.identifier[0].system (line 4, col49) : URL value 'http://xx/app/checkin' does not resolve
 Error @ DetectedIssue.patient (line 8, col29) : This property must be an Object, not a primitive property
 Error @ DetectedIssue.code.coding[0].system (line 12, col49) : URL value 'http://xx/app/alert' does not resolve
 Information @ DetectedIssue.code (line 11, col16) : None of the codes provided are in the value set
    http://hl7.org/fhir/ValueSet/detectedissue-category (http://hl7.org/fhir/ValueSet/detectedissue-category),
    and a code is recommended to come from this value set) (codes = http://covidcare.au/app/alert#trendNegative)
 Information @ DetectedIssue.code.coding[0] (line 11, col18) : Code System URI 'http://covidcare.au/app/alert'
   is unknown so the code cannot be validated
 Warning @ DetectedIssue.identified.ofType(dateTime) (line 9, col53) : The value '3020-10-27T13:33:15+11:00' is
   outside the range of reasonable years - check for data entry error
```

Validator Capabilities

- Validate against any milestone version
- Validate against any published profile, any invariants
- Supports most publicly available terminologies (e.g. not CPT-4)
  - We add support for others by request but capacity is limited
  - Can use a different FHIR terminology server
- Multi-lingual (in theory)
- Special Security Mode
- Also supports:
  - Questionnaire, MessageDefinition, Measure, ObservationDefinition

Validator Capabilities

- Validate against core specification, stated profiles, and requested profiles
- Validate cardinalities, value domains, syntax specific features
- Validate all terminology if code system is known
- Check invariants & slices
- Validate against other rules stated in the specification
- Some specific best practice warnings

Hosting the Validator

- Run the jar directly
- Load in a HAPI based process
- Nearly here:
  - Run as a local web service
  - Run as a local desktop app
  - Use http://validator.fhir.org (demo)

Validator – Choosing Version

- You can specify the version
- Or you can let the validator try to guess
- Defaults to R5 (internal current version)

Validating References

- Validating References
  - `"subject" : { "reference" : "just-some-random-url" },`
  - Is this valid?

  - → setting "assumeValidRestReferences"
  - Validator won't check the reference directly – host it in Java (HAPI)

Validating Against Profiles

Can validate against profiles

- Specify multiple profiles when you run the validator
- Validate Profile claim inside the resource
  - Anything in Resource.meta.profile
- Validate Resource against an IG generally
  - Uses ImplementationGuide.global.profile (not much used in IGs right now)
- Todo: Allow to specify conditions for when to apply profiles

Other Validators

- There are other validators (e.g. DotNet – pretty capable)

- Official validators are those that pass the test cases
  - https://github.com/FHIR/fhir-test-cases/blob/master/validator/manifest.json
  - 374 tests that test all aspects of validation. ~1 new test / wk
  - Bug reports always get turned into test cases

Server Validation

```
POST [base]/Patient/23424/$validate?mode=update
Content-Type: application/fhir+json

{
  "resourceType" : "Patient",
  "id " : " 23425 ",
…
```

Server Validation

Typical Server strategy:

- Validate with one of the main validators first
- Then check business rules and commit logic

What/When should the Server Validate?

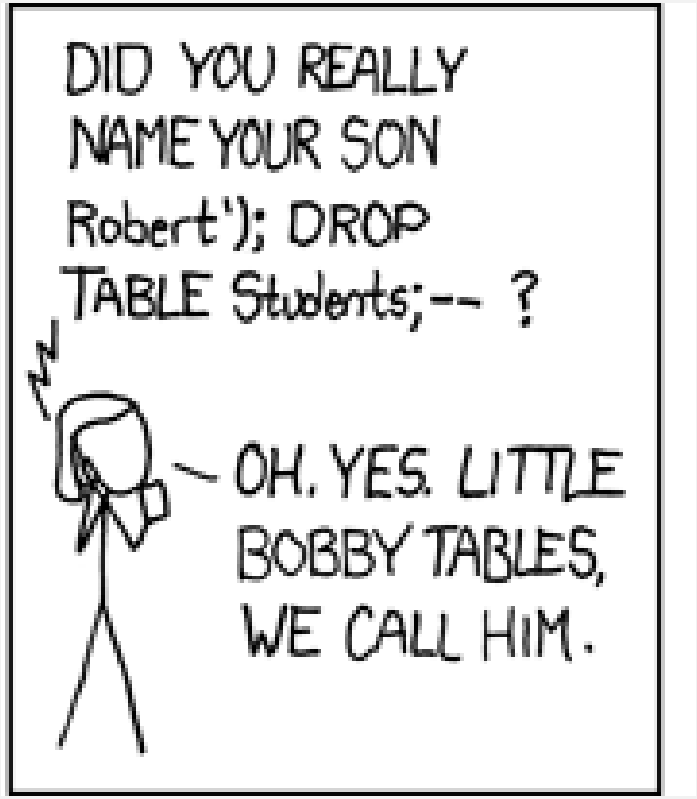
- Check data entry quality / external business rules
- Check security requirements
- Check internal business rule issues
- Check Development quality

How much is quality worth?

- Fully costed quality costs up to $millions
- Many institutions turn quality checks off

Validation and Security

- Some insecure content is not valid
  - E.g. Active HTML content
- Some potentially dangerous content is valid
  - E.g XHTML and SQL escape code hacking e.g.
  - Validator can be run in special security mode to make this content illegal
  - There is still conformant content that is a security risk



What is the right validation approach?

During development:

- Validate everything aggressively. Use an official validator
- Particularly anything you generate

In production

- That depends…
- My recommendation: validate only what you need

Contact

- During DevDays, you can find / reach me here:
  - Via Whova App – Speaker's Gallery
  - On chat.fhir.org
  - Email: grahameg@gmail.com
  - Twitter: grahamegrieve