

Implementing an Adverse Reaction Reporting Project Utilizing FHIR



Orion Health White Paper
Dr. David Hay
Product Strategist
102018



Introduction

HL7® FHIR® or Fast Healthcare Interoperability Resources is the next generation HL7 standard in healthcare data integration and is focused on decreasing interoperability costs and unlocking innovation in healthcare. FHIR represents a major upgrade in healthcare standards and is boosting access to health information and support for the health sector. From the beginning, FHIR has been developed as a standard that is implementer focused and is more straightforward to deploy compared to other standards that are available (or, indeed, some bespoke solutions). However, while implementer experience suggests that this is true, healthcare remains an extremely complex domain.

Because of this complexity, it is very important that the users of the data being shared – clinicians and patients – are involved in the design aspects of any project. Users need to be sure that what is delivered will meet the requirements that the project is intended to address. How can they be involved in such a technically challenging domain? In this white paper we will look at how this might be achieved, using an example scenario of capturing Adverse Reaction data. Our target audiences are the clinicians, business analysts and patients who want to play an active role in projects of this nature.

Adverse Reactions are defined in pharmacology as any unexpected or dangerous reaction to a drug, an unwanted effect caused by the administration of a drug. These unexpected effects of taking medications can cause harm to patients and in unfortunate circumstances even death. It is important when clinicians are prescribing medications for patients, that they are aware of any previous adverse reactions, so they can minimise any further harm. This is also important at a population health level to help identify medications with significant Adverse Reactions. However, the reporting of this information outside of the individual clinician's Practice Management System is limited, and so the Adverse Reactions project is intended to define a format for collecting this data, a way to record and report on the findings, and to improve the ease of generation of the report by pre-populating from existing data sources.

Fundamentals of FHIR

Before we go into the details of this project, we should review some of the basics of FHIR. These are the high-level concepts that you need to understand about FHIR to be effective when participating in FHIR related projects. FHIR covers the format of information and how the data is exchanged, so it is both a Model and an API. FHIR is built around the concept of “resources”, the basic unit of interoperability. The smallest amount of information that can be exchanged at one time, a resource represents a ‘thing’ of interest in healthcare. It could be clinical such as a problem, a medication or an allergy, or administrative such as an encounter, an appointment, or a Medical Practitioner. By itself, a single resource has only limited value, to exchange more meaningful information, you join resources together, a process called resource references. The information represented in this way, is able to be understood by a recipient.

Key components of FHIR

Resources

The individual building blocks used to tell a clinical or administrative story, made up of a number of elements. The specification defines what elements are in a resource (like a cookie cutter), an actual instance of a resource is like a cookie made by that cutter.

Examples of resources we might need in this project include:

- Patient
- Practitioner
- AllergyIntolerance
- Condition (Problems)
- Observation

View a complete list here:

<http://hl7.org/fhir/STU3/resourcelist.html>

DataTypes

The fundamental parts of a resource (If a resource is like a cookie, then these are the ingredients). Each element in a resource instance has a particular datatype that contains the value of that element (and a single ‘value’ may have sub values – like a person’s name).

References between resources; these are how we connect individual resource instances together. For example, an Observation instance representing blood pressure has a reference to the Patient who it is about, the Practitioner who took it and so forth. The ‘web’ of interconnected resources is also called a ‘Graph’ of resource instances.

Terminology

Terminologies defines the codes that resources can use. For example, a Condition resource might represent asthma – the code for asthma might come from the SNOMED terminology. Coded data is important for ‘semantic’ interoperability (being sure that the correct meaning is being shared) as well as for secondary use of data – analytics, quality measures and decision support. Each coded element in a resource is ‘bound’ to a set of possible values – using a special resource called a ValueSet to do so.

Profiling

The base resource types are, by design, very generic. Profiling is the act of ‘adapting’ a resource to a specific use. This can be very complex, but common examples of profiling include:

- Adding new elements (Extensions).
- Changing the binding of elements to a different set of values (by specifying a different ValueSet).
- Removing elements that are not needed or fixing their values (e.g. specifying that a patient identifier must come from particular registry).

API

Application Program Interfaces (APIs) define how different applications can communicate with each other. An essential pre-requisite to being able to choose different implementations for the same 'type' of functionality thus enabling an 'ecosystem' of applications.

Implementation Guide

Defines all the artifacts and documentation needed for a particular purpose. Our example project (Adverse Reactions) will be represented by an Implementation Guide.

Community

The last and the most important! There is a huge, global community that will assist implementers. You can get in contact at the FHIR chat: <https://chat.fhir.org/>

The Adverse Reaction reporting project

Now that the fundamentals of FHIR have been covered, we can start to think about how to approach the Adverse Reactions project. Below is a list with the overall approach that we will take:

1. Define the initial requirements
2. Describe the target architecture of the solution
3. Create one or more information models to define the exchanges
4. Validate the models (connectathon)
5. Create the FHIR artifacts
6. Deploy development resources
7. Validate artifacts (connectathon)
8. Review results and publish when ready

Although this looks like a 'waterfall' approach, in reality, each step is 'validating' the step before – and as we get more detail, and involve others in the project, we can and will revisit and refine previous steps. Let's go through these steps in more detail, describing what each is intended to achieve and giving an example from our Adverse Reactions project. We will discuss the tooling that is available to help us.

There are 2 'types' of tooling that we can use throughout this process:

1. Educational and design time tooling like clinFHIR and conMan – which have been developed to assist clinicians and designers who are not FHIR experts. These are mostly used in steps 1 to 4.
2. The official FHIR tooling – the Implementation Guide builder and Forge (the profiler) that are used to build the concrete FHIR artifacts. These are used from step 5 onwards.

Step 1: Requirements

In the first step we outline the problem we are wanting to solve and specify typical Use Cases (or Scenarios) that describe in more detail what a solution might look like. The more comprehensive the scenarios can be the better, and we should certainly try to describe the 'edge conditions' as well as the more common cases – we will be using these to validate the models and artifacts that we create in the later steps.

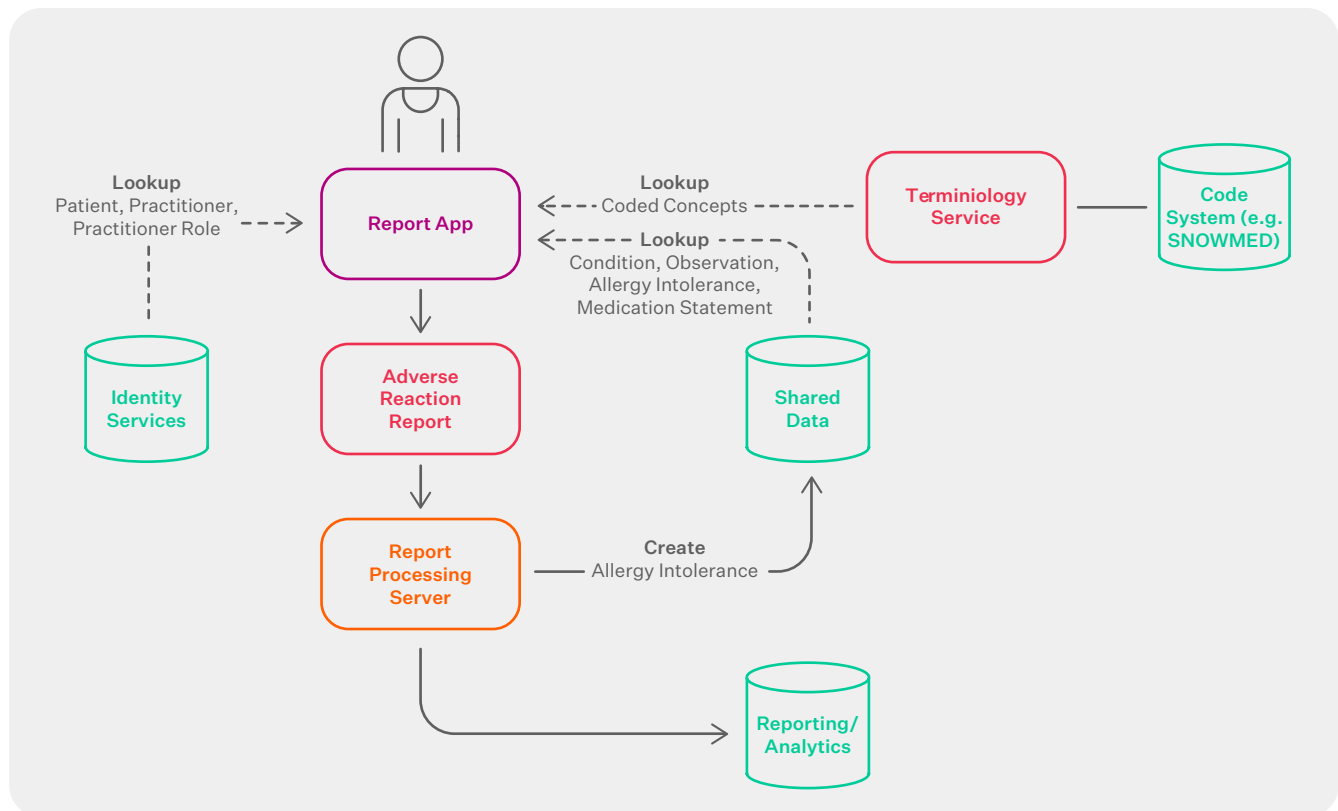
So, for our Adverse Reactions example, we want to have a simple way to capture information about drug related adverse events that occur in the community or in a hospital. This data can be stored in a central repository where the information can be made available for the delivery of clinical

care. Examples of this are, the presentation of data to a clinician at the point of care and assisting Clinical Decision Support, research, population-based analytics and reporting. We want to allow anyone to submit a report – but tracking provenance so someone viewing the data knows where it came from (as would a computer system). Finally, we need to be cognisant of security and privacy – though this is likely to be part of an overall strategy rather than just in our project – there will be privacy-based requirements that we will need to support (such as Consent, Access Control and Audit).

From a tooling perspective, most of this will be textual documentation – though there are specific tools available. We do want to represent all the information in HTML format, this is an important part of our implementation guide. And we will add the scenarios into conMan when we start validating models in step 4).

Step 2: Target architecture

It will be helpful to understand the environment in which our project will function. This may consist of new and existing systems and the level of detail needed will vary enormously. We need enough detail to identify the ‘points of exchange’ where data moves between systems, as each of these exchanges will need to be modelled. This should be straightforward for this Adverse Reaction reporting project. Each report of a drug-related adverse event will be a structured and coded document that is authored by either a clinician or a patient and this is then submitted to a central repository. A lot of the data we want to collect will already be in clinical information systems, for example, a GP Practice Management System so we will endeavour to support pre-population from those systems to relieve the burden on the report submitters.



Data quality is going to be paramount. We require properly coded data to support analytics and eventual decision support, so a common terminology server should ideally be available. And the Identity – of the patient and the submitter – is also going to be important. We could let each submitter work this out, but it would be preferable to have a common service we can use. Here is how it worked for our project.

Step 3: Create information model/s

The third step is to define the information that we wish to exchange. In our project this is straightforward as there is only a single submission of a report and no workflow of any significance. In other projects you may need multiple models, and also more complex workflow to support them. If you want to learn more about this see: <http://hl7.org/fhir/workflow-module.html>.

Note that the scope of this project is simply to acquire the data, however there may be more complex workflow in the subsequent processing.

Like all the steps, there are different ways that we could do this, but given that we are using FHIR for the project, we are going to use Logical

Models and clinFHIR to create them. A logical model is a rather overloaded term, what we mean is we will use the FHIR infrastructure (particularly the datatypes and the terminology services) to create structured, coded representations of the information we want to exchange. We won't be restricted to the actual resource types (as described above), but by using the FHIR infrastructure we will make it easier to transition to 'real' FHIR resources in later steps. We will be able to delegate this later work to FHIR experts rather than domain experts, as we have been explicit about the data we want to share, and ideally the terminology bindings as well. (And now you can see why knowledge of data types and terminology is an essential skill for working with FHIR). Another term for the model is an 'Information Model'.

In the Adverse Reactions project, we used the clinFHIR Logical modeler, to learn more about this you can read <https://orionhealth.com/global/knowledge-hub/blogs/clinfhir-a-tool-for-clinicians-business-analysts/>. We displayed the live model in meetings, so we were able to build it in real-time. Because the Logical Modeler supports terminology services, we could get examples of coded data where needed to help define the model.

Below is a screen shot of part of the model.

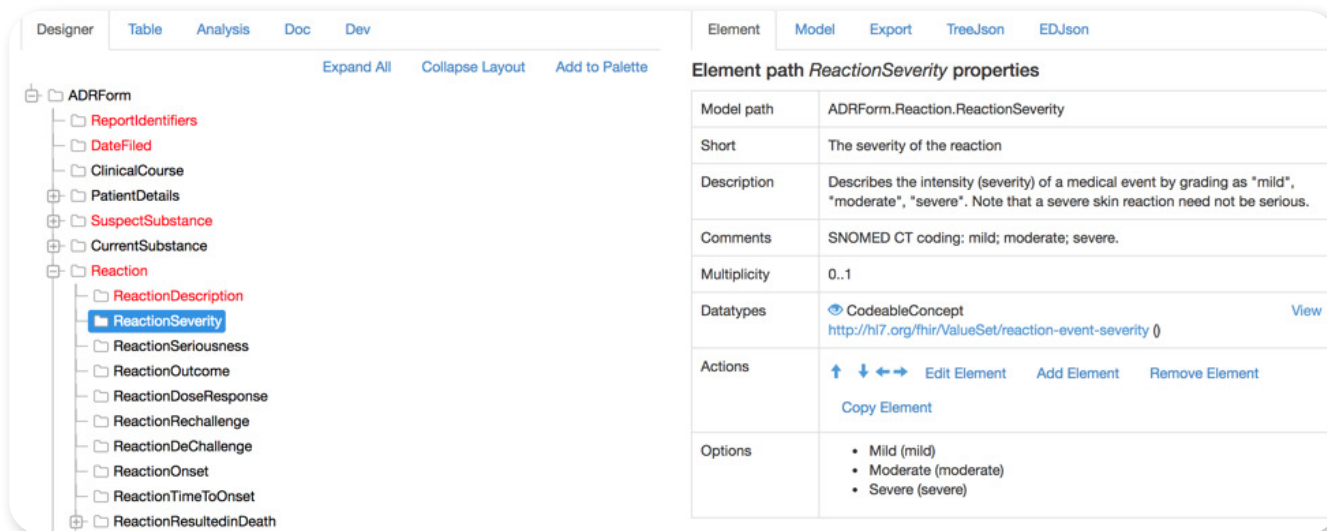


FIGURE 2: Screenshot of section of information model.

Note that there is no relationship to any core resource type, this is a clinician’s view of the data. Also, we have been able to list the possible values for reaction severity (bottom right) due to the link to the Terminology server (Ontoserver in this case: <http://ontoserver.csiro.au/>). We could have used any FHIR terminology server that supports the required API.

Another task for this step is to identify the ValueSets that we are going to need. They are essential to building the FHIR artifacts in step 5. In some cases, there will be existing ValueSets that can be used, otherwise they will need to be created.

Step 4: Validate the models

Once the data items are correctly identified, there is great value in being able to validate these, by taking the scenarios that we identified in step 1 and ‘populating’ the information model with sample data. This will allow us to identify missing data, or irrelevant elements. Based on this validation, we can go back and update the model if needed.

This is what conMan was designed to do, one of its functions is to take the Logical Model and generate a form based on that model with places to enter sample data. This can be done by any user, and there are also places for users to record comments about the model, and to review comments made by others.

At the time of writing this paper, we have done the first round of this and we are planning a more extensive round of testing at a HL7 Connectathon at the Health Informatics New Zealand conference.

Below is an example of one of the forms. This is a compressed view showing only elements that have data entered against them, we have access to all the elements in the model if we need them.

Simple allergy
A patient visits their GP with a rash 2 days after starting penicillin for a sore throat. There are no breathing difficulties. The patient is diabetic, currently receiving metformin

Notes about model - eg missing elements

(ARForm-1)

Model Element	Card.	DataTypes	Value	Notes
> patient	1..1	--> Patient		
name	0..1	HumanName	John Doe	
> event	1..1	--> AdverseEvent		
manifestation	1..*	CodeableConcept	Rash	
✓ suspectMedication	0..*	--> MedicationStatement		
code	0..1	CodeableConcept	penicillin	
> medicationsList	0..1	--> List		
✓ medication	0..*	--> MedicationStatement		
code	0..1	CodeableConcept	metformin	

FIGURE 3: Screenshot of Form Example

Step 5: Create the FHIR artifacts

Now that the models are confirmed, it is time to create the actual FHIR resources that will be needed when we implement this. (Remember that the Logical/Information Model is not what is exchanged, it is simply a design artifact to get to that). Implementers are going to need 'real' FHIR artifacts, which are collected into an Implementation Guide. For more information see here: <http://hl7.org/fhir/implementationguide.html>.

This can be a complicated step. Some of the things that we need to consider include:

What is the exchange paradigm?

Are we interacting with a real-time interface in a 'conversation', or are we just submitting a report? If it's a report, then is it a Document <http://hl7.org/fhir/documents.html> or a Message <http://hl7.org/fhir/messaging.html>? In our case, we will treat it as a Document, as it is something we want to keep long-term and refer back to. (HL7 defines a number of characteristics of a document as part of the CDA (Clinical Document Architecture) standard – see http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7). We can still pull the coded data out for analytics as required. Being a document means we can also create a profile on the document itself <http://hl7.org/fhir/documents.html#profiles> which would assist with validation. This allows us to specify the sections for example.

Another option we could consider would be to use Questionnaire /QuestionnaireResponse resources to collect the report data, which would be processed by the server and create all the desired resources required. There is a project currently underway to standardise this approach, for more information see here: <http://www.healthintersections.com.au/?p=2835>.

What are the resource types that will be needed?

We are going to need a number of resource types to represent this information. For example, there is a Patient resource, an AllergyIntolerance if we believe this to be an allergy, AdverseEvent if not. Then there's Observations for things like a rash, MedicationStatement to describe drugs and more. It is important to use a FHIR expert for this step and this is why we had the Information Model for the clinicians.

In reality it will be more complicated than that, as it is likely we have specified information that is not in the core resources – so extensions will be needed. In other words, we need to create or preferably re-use resource profiles.

What are the ValueSets needed?

Ideally, we would be able to use some of the ValueSets defined by the FHIR specification, but there are bound to be some that are specific to this project. These should have been defined (or at least identified) as we built the information model, but there may well be work here to create or locate them.

How will the resources be organised?

The answer to this is substantially defined by the exchange paradigm. It is certainly a bundle (collection) of many resources, but if it is a Document or a Message then the specific relationship needs to be created. The conMan tool can help, as there is a 'Scenario Builder' function that builds 'graphs' of resources by selecting them from a palette and 'wiring them up'. Further, we can add sample information into the resources, as we did when validating the Information Model to help with this design.

Here's an image from partial work in the Adverse events project.

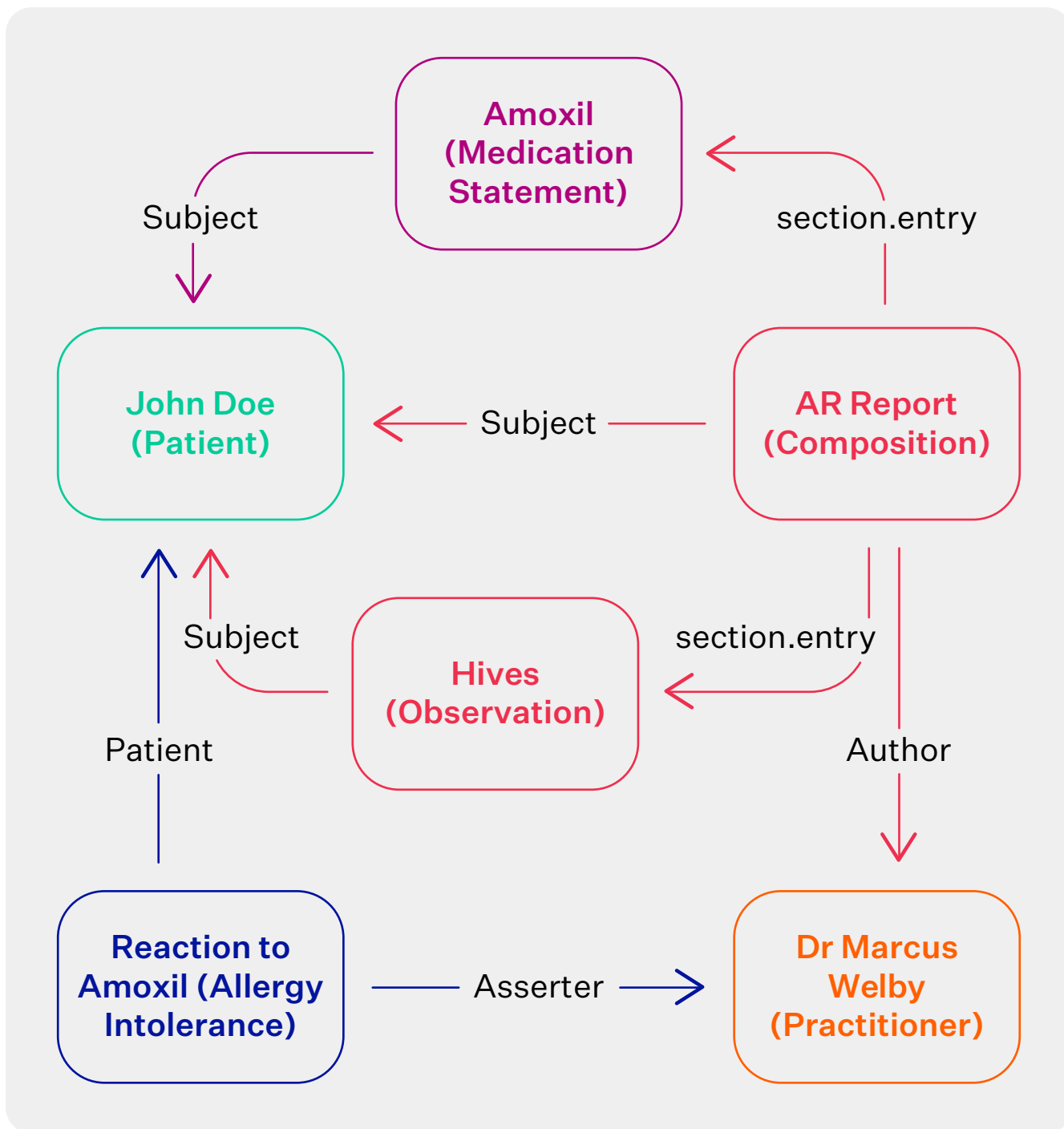


FIGURE 4: Partial Work in the Adverse Events Project

What other artifacts are needed?

There may be other artifacts required, for example ConceptMaps for converting from one terminology to another, CodeSystems to define a separate code system or CapabilityStatement resources to describe required API interfaces.

From a tooling perspective, we might use conMan to help us in identifying the resource types as described above, but we'll use the official tooling to generate the Profiles and Implementation Guide.

Step 6: Deploy development resources (optional)

We are now at the stage in our process where we have something that implementers can take a look at. The Implementation Guide describes what we are trying to do and has references to all the artifacts needed. It is possible to stop here, publish a draft Implementation Guide and get feedback from the community.

But an alternative is feasible if there is the resource available and that is to stand up a 'development' environment against which developers (and advanced clinicians) can use to further exercise the guide, through a connectathon. The infrastructure required is going to depend on the project, but for our project, we'd be looking at:

A server that can store the report. Most of the freely available reference servers (like HAPI and Vonk) can store documents, but other development work might be needed if we wanted to extract and store the individual resources from the Document bundle. (Or, especially for testing purposes, we could use the Batch operation to do this, see here: <http://hl7.org/fhir/http.html#transaction>).

- A source of data for pre-population of reports. This can be an 'ordinary' FHIR server with some sample data pre-loaded, or a co-operating PMS (Practice Management System) vendor.
- A Terminology server loaded with the ValueSets and CodeSystems is required.
- An Identity server which could be an ordinary FHIR server (maybe the same as the data server) with some Patient resources and Practitioner or PractitionerRole resources on it.

If you required a cost-effective option, you could use a cloud server. If you require a cost-free option an instance of the HAPI CLI server http://hapifhir.io/doc_cli.html or Ontoserver terminology server <https://ontoserver.csiro.au/stu3-latest/> are available.

We have not discussed security and privacy in this paper but depending on your circumstances you may wish to set up infrastructure like OAuth or SMART, insist on SSL, and implement a privacy layer.

Step 7: Validate artifacts

In the same way, as we validated the information model, it is good to validate the FHIR artifacts (represented as an Implementation Guide) as well. There are a few ways we could do that.

- Have the Implementation Guide reviewed, this is what happens in a HL7 ballot.
- Use the scenario builder in conMan to build instance graphs that are compliant with the Implementation Guide (possibly using a Document profile to test this).
- Hold a connectathon (virtual or physical) and have implementers test against the development resources. This is the best approach, albeit time-consuming or even include all of these.

Step 8: Review and publish

This is the final step in the process and occurs after you are sure that the Implementation Guide is fit for purpose. As described in the beginning, there are a number of validation points and places where iteration may occur, so the actual path to completion might vary.

Conclusion

The Adverse Reactions project is intended to define a format for collecting this important data, a way to record and report on the findings, and to improve the generation of the report by pre-populating from existing data sources. The FHIR standard has been used as a foundation to define a format for implementing the Adverse Reaction reporting project utilising this innovative FHIR standard.

Dr. David Hay - Biography

David is a Product Strategist for Orion Health. He is also active in the international standards community as the chair emeritus of HL7 New Zealand and is a co-chair of the FHIR Management Group, charged with guidance and development of the latest HL7 standard. David graduated from medical school, then moved into the Health IT sector. He started a company developing Practice Management Software to GP's (GPDat). This was the first such program in New Zealand to receive electronic laboratory results.

Leaving the vendor space, David worked at EDS as a Solutions Architect, before returning to health IT working as an Architect (Solutions and Enterprise) for the Auckland based services management organisation 'healthAlliance' producing a number of innovative solutions: an internal eReferrals application, a community-based Case Management solution, plus an application to track and report on clinical tasks within the hospital. All of these applications remain in current use today. While there,

he also participated in a number of national programmes, including the medical records transfer project GP2GP, ePrescribing, and was one of the authors of the Interoperability Reference Architecture that governs information flow within the health sector.

He currently serves on the Health Information Standards Organisation (HISO) committee, which provides technical advice on standards to the New Zealand Digital Advisory Board. He has provided outstanding leadership towards helping to create and evangelise the innovative FHIR standard, and frequently writes about FHIR on his blog <https://fhirblog.com/> and has developed a range of open source tools widely used to educate and assist the developers of FHIR <http://clin.fhir.com/>

David is a true champion of the New Zealand IT healthcare sector, he has spent many years both professionally and personally designing, educating, and advising on health informatics. In recognition of this David has been awarded the Excellence in Health Informatics award 2016 by ITx New Zealand.

His ongoing commitment to FHIR has helped to provide the momentum needed for international adoption. FHIR represents a major upgrade to standards that will boost access to health information, which will improve the access to a patient's health information globally.

