HL7® FHIR® DevDays 2019

HL7 FHIR

Flicker or Bonfire – How design choices affect FHIR's power

Lloyd McKenzie

Redmond, June 10 – 12 | @HL7 @FirelyTeam | #fhirdevdays | www.devdays.com/us

Who am I

- Name: Lloyd McKenzie
- Company: Gevity
- Background:
  - One of FHIR's 3 initial editors
  - Co-chair FMG & FHIR Infrastructure
  - Heavily involved in HL7 and healthcare exchange for last 19 years
    - v2, v3, CDA, etc.
  - Unofficial FHIR community manager
  - Have helped numerous projects in the process to FHIR implementation
  - lmckenzie@gevityinc.com

Problem Statement

- FHIR is advertised as:
  - Fast
  - Easy
  - Powerful
  - Interoperable
- But implementations don't always meet those expectations
  - Why?
  - How can we do better?

Objective

- Considerations you might want to keep in mind as part of FHIR design
  - Some (perhaps all) will be obvious to some of you
- Provide ammunition to help challenge design decisions in other systems
  - Or better yet, influence decisions before they're made…
- These considerations have nothing to do with FHIR conformance
  - You can conform to FHIR while still exposing interfaces that aren't terribly fast/easy/powerful/interoperable

1. Use REST whenever you can

Why REST?

- FHIR supports REST, documents, messaging and other exchange paradigms
  - Use the one appropriate for your problem

- HOWEVER:
  - Just because you were doing documents with CDA or messaging with v2 or v3 (or something else)
  - Doesn't mean you should do the same with FHIR

Why REST?

- With REST, data is granular – no need to worry about what bits are grouped together or how they're presented
- Documents & messaging add an additional cost: implementations need to be consistent about what resources are present and how they're organized
  - For documents also consistency around 'sections'
- Messaging adds additional costs of standardizing event codes and allowed responses
- Biggest difference: Documents & messages are more purpose-specific
  - REST (not operations) is purpose-agnostic

But what if I need documents?

- IF
  - You need tight control over how data is presented to humans,
  - You need formal attestation of exactly how that data was displayed and/or
  - You MUST pass a collection of resources together over email or some other non-messaging/non-HTTP transport
- THEN
  - Use FHIR documents
  - (and try to use standard document profiles – e.g. C-CDA on FHIR)
- OTHERWISE
  - Seriously consider using REST

But what if I need messaging?

- You only need messaging if:
  - You need to support asynchronous routed delivery (i.e. where the message won't get to its end recipient within the HTTP time-out period) and there's no intermediary that can act as a queriable repository
- Otherwise
  - Contained resources and transactions with conditional references can handle situations where 'identity' isn't known/exposed
    - Though refactoring implementations to expose identity is even better
  - Asynchronous requests can be handled via Task
  - Complex behaviors can be handled via custom FHIR operations
    - Though custom operations won't save you much in the way of costs…

Impacts of transitioning to REST

- Will often need to expose data for query
- Permissions need to be context-independent
  - Based on role and action, not "why" you're doing the action

- **NOTE:**
  - If data isn't available via REST, you can't use SMART, CDS Hooks, Bulk Data, or all the other layers implementers are building on FHIR

2. Create Permissive interfaces

Instincts when creating a FHIR API are often **wrong**

- **Poor design**: If you don't support an element, mark it as max=0
  - **Better design**: Mark the elements you DO support as "mustSupport=true" – and ignore everything else
    - Don't make others spend money to exclude data you can ignore for free
    - Don't tell others it's an error for you to send data you might want to send later
- **Poor design**: Patient.name is 1..1
  - **Better design**: Patient.name is 1..*, with a 1..1 slice that describes the specific name you need (e.g. 'official', no expiry date, extension marking as "for use in Canada")
    - Expect clients to have only one FHIR interface, so don't prevent them from sending what others might need

Instincts when creating a FHIR API are often wrong

- **Poor design**: "The first identifier must be the XYZ hospital medical record number"
  - **Better design**: "At least one identifier must be their XYZ hospital medical record number (identified using Identifier.system=…)"
    - Don't enforce order unless FHIR has assigned meaning to that order
- **Poor design**: "When you query MedicationRequest, you'll only get active orders"
  - **Better design**: "At present, our system will only expose active orders. However, in the future, it may expose other orders, so always filter your query"
    - Assume that you'll eventually expose all data – and prepare (and test) that clients can deal with that

3. Don’t lock into “the way we’ve always done it”

Examples

- "Our specifications have always been PDFs before"
  - Implementers are really going to want computable specs (and HTML-navigable specs)
- "Our system doesn't need Provenance information"
  - Your system probably hasn't supported write access from external systems before – it's going to be important to track where data came from
- "We have no use-case to expose [Audit/Lab/CarePlan/…]"
  - Feel free to prioritize, however FHIR enables new business cases

4. Think about the community

The community is a core part of FHIR

- Costs are divided amongst the whole community
  - Try to minimize the overall costs, not necessarily your costs
- Align with the community where you can
  - Comply with relevant (and implementation-oriented) implementation guides
  - Before you define a new extension/implement a 'creative' solution, check with the community
- http://chat.fhir.org is your friend
  - The spec can't cover every nuance. Coordinating with your peers is what really produces interoperability

Summary

1. Use REST whenever you can
2. Create permissive interfaces
3. Don’t lock into “the way we’ve always done it”
4. Think about the community

In short: “Try not to be a ‘special snowflake’”

Challenges with driving better behavior?

- Sometimes 'faster/easier' is a trade-off
  - It might be slower and harder to create profiles/interpret profiles that are more re-useable for implementations
  - Try to think about the long-term/overall cost
- Economics often don't reward considering future costs/costs for others in the community
- Time pressures can reward "fast" over "right"

- Encourage your partners to look at the bigger picture – and reward those who do

Questions / Discussion

- lmckenzie@gevityinc.com

- Or, better yet, include the community and ask/discuss on chat.fhir.org