Kndefjgawj principles

Klas, Nick, Deidre, Eric, Flavio, Jay, Gary, Arofan, Wendy, Jared

1. **Capture the meaning of the data at the source** (close to the domain expert who understands the domain). *This can take the form of a Domain Analysis Model (DAM). In a DAM there are entities and activities. The entities are the terms of the model. And the activities are the stories in which the terms are engaged. The DAM has been described as follows:*   
     
   *A representation of the static and/or dynamic semantics of a subject-area-of-interest (i.e. a “domain”) in a manner that enables harmonization of the various perspectives of the stakeholders in the domain while also providing the foundations required to build logical and implementable representations of the domain.”*
2. **Put it in understandable model for humans in an integrated sense.** *The model is NOT formal. In the Agile approach, there is a distinction between user stories and use cases. User stories come in a paragraph or two that recounts who is doing what to whom for the most part. Edge cases don’t need to be recounted. We are not yet talking UML.*
3. **Put it into form that is machine-processable** (so you can interrogate it/query it). *Story telling comes in sentences. Subjects in one sentence may become the objects in another and/or vice versa. There are any number of “systems” we can use to express these stories. One characteristic of each of these “systems” is that they are queryable. System examples include RDF, specific dialects of* [*Test Driven Development*](http://www.methodsandtools.com/archive/archive.php?id=72p7) *(TDD), various flavors of JSON and JSON-LD as well as domain-specific “systems” like FHIR.*
4. **Be able to test the semantics of the implementation against the logical model.** *Stories are instances that we tell in terms of schemas. In the process, we don’t always begin with a DAM. That would be top down. Instead the construction of a DAM may come after the fact: it is informed by storytelling. This would be bottom up.*
5. **The principle is when producing models is you create tooling so you can test if they fail.** *Whether we go top down, bottom up or most likely some mixture of the two doesn’t matter. What matters is that we complete a round trip. In a round trip conformance is bidirectional. Logical and domain specific models (DAMs) conform to stories and their schemas and vice versa.*
6. **In addition to be testable, models should be implementable**. *Typically, this is not the case. LOL. Not without a secret sauce…*
7. **You should make the rules for using something explicit.** *This is the segue. Now comes the secret sauce:*
8. **You should support communities of use and testability by providing a profile mechanism.** *Profiles come into play maybe at many points during a round trip.* [*UML Profile*](https://en.wikipedia.org/wiki/Profile_(UML)) *is an extension mechanism. Alternatively, the mechanism may be less formal. It can come during the binding phase in which we generate specific schemas to host our observations. When mechanisms are introduced thus we refer to them as the* [*deus ex machina*](https://en.wikipedia.org/wiki/Deus_ex_machina)
9. **Your metadata and data should be identifiable and manageable independent of its encoding.** *We can assure this by going bottom up, top down or some combination of the two so as to create and trace three models: a conceptual or domain specific model understandable to domain experts, a more formal platform independent model (UML perhaps) and a platform specific model that hosts and entertains instances (“languages” like JSON, XML and RDF).*
10. **No gratuitous remodeling. Maximize reuse; new modeling only when necessary**. *Avoid corner cases like the plague. Know that instances may not be supportable without “fudge”, i.e. the secret sauce. Profile until a use case becomes compelling.*