2015-07-20 Design Forum

Attendees

- Darius
- Burke
- Jonathan Teich

Agenda

- Encounter Statuses and Drafts

Notes

Burke: we need to explicitly have "status" on Obs, Order, Note; it's not enough to have this be inherited from Encounter.

JT: You might even have a "complete" encounter that still has pending Obs in it.

Burke: we have treated this as an encounter addendum

Sign off on individual components, or on the whole encounter?

JT describes an emergency encounter, includes some notes and orders, and he can sign these things individually

- E.g. can sign off on an order and disposition (to send the patient home), before having signed off on the attestation/note

- (In BICS you signed the disposition, in Epic you don't, it's implied.)

Burke: at RI this is treated as an addendum.

- First encounter '"ED Release Order" (has order and disposition)

- Later, you come back, find this encounter and say "I want to adend this" (which is actually another encounter under the hood)

Typical BICS and Epic statuses (stati?) for anything:

- 1. Draft - only I (the author) can see it

- 2. Preliminary ['shared' in Epic] - others can see it but it is marked preliminary. It could be changed, in which case they would see the latest version with a flag to see the earlier ones.

- 3. Final - it's no longer changeable

Notes could be draft/preliminary/final

Orders could be draft/final

Dispo is only final

The parent structure of the Encounter is start/end date/time, venue, provider, type (clinical, lab, procedure, etc.), status (usually to separate no-show, telephone, in-person encounters)

Burke: in this model, our OpenMRS encounter is things that clinically belong together, but the status is tracked on the encounter transaction (e.g. technical transaction), e.g. "at this time Jonathan finalized these X things"

Burke: What if there's a requirement like "an encounter of this type must have a note and a disposition"? Where is this checked?

JT: Each component should know about its dependencies, it shouldn't be checked at the whole encounter level

Burke: Encounter Type might define things that must be in an Encounter. Depending on this we can report or alert on this later, but there's no official point at which we could say "the encounter can be finalized because it has X and Y" (and this is good).

There are some discrete workflow steps that can't be done until one or two other things are done (e.g., need dispo entered, and need admitting transfer entered, before patient can be admitted); but there's no dependency to "finalize" any entire encounter.

When an encounter of a certain type is initialized, it does create various to-dos which must *eventually* be completed, such as (for a clinician encounter) a note and disposition.

[Consensus] for an inpatient admission visit, each cluster of orders signed together is an encounter; each daily note/record between clinician and patient is an encounter. Each new set of vital signs that I take at a different time (e.g., every four hours) is an encounter, perhaps unless there is some specific meaningful cluster such as a brief conscious-sedation procedure with vitals taken every two minutes.

Conclusions:

- We need to introduce EncounterTransaction before we can add statuses (because status will go on EncounterTransaction)

- We need statuses on Obs, Order, and Note* (when we add Note)

- For Obs, status should leverage same "HL7" status field used for exceptions, pending results, etc.

- Shouldn't have a "final" EncounterTransaction that includes non-final Obs

[ At this point we realize that Burke and Jonathan don't mean the same thing by EncounterTransaction. :-) ]

Burke: it means "all the data that came together in one transaction under the hood".

JT: Maybe don't use the word "encounter"? Maybe "Session" or "SessionTransaction" or "Transaction" or "TechnicalTransaction"

I'd suggest: a Session (or, if you prefer, Transaction; I like Session but I'm easy on this) is when you add/edit/delete/change status of one or more elements at the same time, signed or acknowledged together. (Element = one note, one set of vitals, one order, etc.)

You have a preliminary observation O1 that came in transaction T1 and is in encounter E1. What happens when the final observation comes in the next day?

=> the final observation O2 is in another transaction T2, in the original encounter E1. And we void O1 when O2 comes in.

Darius: can we model (do we need to?) the fact that we void an obs in a transaction?