Form driver notes  
  
The difference between intake form, repeat form and follow up form may have nothing to do with the form driver rather with how the database is set up.  The form driver could likely be the same for all three cases, with the only variations being whether or not the previous value of a field is offered as default.  
  
In the database, forms that are part of a repeating set, such as visits, as opposed to a non-repeating set such as registration, would have an additional index or subscript indicating their place in the series... Visit #1, visit #16, etc. The formdriver would deliver its output to the thispatient.Visit(16) structure just as it would deliver a registration form output to a thispatient.Registration (no subscript) structure.  
  
But the form driver would inherently always capture date and time of entry and person entering for every update.  And the audit trail feature would allow updating any field regardless of its place in a data series; I would be able to update a patient's address (and the audit trail would show the older entries) and I would be able to correct the fourth blood pressure entry in a series of six (and the audit trail would show that there are earlier changed entries for the fourth member of that series). So, conceptually, if I enter a new diagnosis for visit #16 for a patient at the time of the visit, then  
 thispatient.visit(16).current.diagnosis = pneumonia  
 thispatient.visit(16).audit-trail(1).metadata.datetime = (current date and time)  
 thispatient.visit(16).audit-trail(1).metadata.enterer = Jonathan Teich  
  
 Later, if Akhil changed the diagnosis for visit #16 after the fact, then  
 thispatient.visit(16).current.diagnosis = bronchitis  
 thispatient.visit(16).audit-trail(1).diagnosis = pneumonia  
 thispatient.visit(16).audit-trail(2).metadata.datetime = (date and time of the change)  
 thispatient.visit(16).audit-trail(2).metadata.enterer = Akhil  
  
  
Comments:  
Comment is a field of its own, unless we believe that comments are a required attribute of the basic field structure (I don't think that's a good idea, it leads to way too many comment fields and you'd have comments all over the place that should be together). You might have one comment field per form, e.g., one place to put comments about this instance of capturing vital signs, rather than separate comments on the systolic blood pressure, on the diastolic blood pressure, etc., etc.  
  
(Advanced) Should allow smaller forms to be embedded (linked) in larger forms -- enforces consistency, e.g., for vital signs section that could be used in many forms.  If done, we need to track what forms are using each modular sub-form, so we can review those if/when it changes.  
A simpler, distinct variant is to allow a sub-form to be copied into a larger form under construction.  
  
Selection content is irrelevant to the basic design of the form driver; it just needs to know there is a content-sensitive field like a drop-down.  At run time, need to be able to invoke the form with different available selections in the same field, of course, for different patients (eg selecting from the problem list), different providers (selecting different schedules), etc.  So the definition of the field in the formdriver specification contains a pointer to its content set.  That pointer can of course be set up at runtime by the preceding code.  Similarly, the validation guardrail and available ranges for a numeric field may be set up at run time.  
  
 Also, we toyed with the idea of allowing unit conversion factors to be included in some way as well, so the same database element can be accessed in its intrinsic units (eg, Celsius) even if the form is using different units (Fahrenheit) -- better than requiring two different fields in the database because all of the patient's temperatures would be together. There would be a field in the form record for "data as entered" and "normalized data as stored".  I'm not quite sure about this one yet, although it sounds good.  
  
Bear in mind that the form data may not be ready to save until the user presses a Save button or similar (i'd expect that to be the most common case); thus, in these cases one needs to store the form data in its own temporary structure and write it to the actual data structure when the form is confirmed.  
  
 What we may want to do is define a few major visual form types -- tables, one-column and two-column questionnaires, etc.