CHARTING A COURSE FOR THE FUTURE

by Kirby Urner
4D Solutions
February 14th, 2009

COLLECTING CLINICAL DATA: LEGAL VERSUS RESEARCH RECORDS

Let us agree that sharing data is a difficult problem, and also a basis for civilization. The challenge is not simply one of keeping information from degrading and getting lost, but of protecting privacy where necessary.

Because of HIPAA and for other reasons, we have an interest in withholding what we call PHI from those with no need to know. Medical research does require the concept of patient identity over time, in order to connect all the histories and complications, procedures and subprocedures, to the corresponding individuals in an unscrambled manner.

It doesn't work to obscure identity by scrambling the clinical attributes, as the research requires keeping patient identity intact. It does work to substitute pseudonyms to mask identity and to otherwise obscure PHI. The clinical attributes remain unscrambled and yet privacy is protected.

The above discussion does not address physicians' needs to access individual patient records, nor does it discuss physician identity in contrast to patient identity, in terms of protecting privacy etc. I feel it's important to continue distinguishing between what I will call the legal medical record (LMR) and the clinical research record (CRR). I make this distinction from long experience with the challenge of gathering CRRs at the point of care.

Because the end user of CRRs is envisioned to be a clinical statistician or data analyst, using aggregate data for retrospective overviews, there's a “take what we can get” attitude, meaning beggers can't be choosers at the point of care. The physician's primary responsibility is to the patient and keeping the LMR is a painstaking charting process that physicians learn in medical school.

The compilation of CRRs, on the other hand, is more like a scavanging operation. We hope the health care personnel will assist us, and we look to reward them for doing so, but we do not wish to interfere with upkeep of LMRs in any way.

Over the long haul, outcomes research is a source of critical information, so it's not that CRRs are unimportant or irrelevant, just that the harvesting of this kind of information takes different routes through a hospital system.

Outcomes researchers are permitted to siphon data from medical devices, circulate scannable forms, take information from PDAs etc.. They may also harvest information from the medical records themselves and indeed the astute reader is already asking why this dichotomy, why not simply use the electronic LMRs as a source of accurate and veridical data and simplify the entire picture?

I would call this a holy grail solution and am encouraged by all the talk of some electronic medical record standards. However, when push comes to shove, we're not there yet, at least not at Providence, so the people actually doing this work need to think in terms of the reality on the ground.

In point of fact, a doctor-dictated narrative, following a heart procedure, transcribed from audiotape for adding to the LMR, is not a friendly format when it comes to statistical analysis. Human language is difficult to parse by machine and no one has solved this problem satisfactorily.

The better option is to use a fill-in-the-blanks approach, where clinicians, physicians themselves if they have time, populate pre-existing tables with a lot of pre-coded questions. The answers are far from open ended, because when it comes to aggregate data, you want a lot of standardized responses. Multiple choice check boxes, numeric fields, pick lists of medical devices, characterize the ideal interface for gathering CRRs.

The astute reader will now see there's a lot of overlap between the LMR and the CRR for a given patient in that some of the LMR data fields are quite amenable to these canned "choose from these options" treatments. Blood type, patient weight, blood pressure, why not make these write to some electronic LMR in the background, with physician notes and dictations appearing alongside? Harvesting CRRs is then just a matter of choosing those fields amenable to statistical analysis, leaving aside a lot of the only-human-readable writings. As hospital systems increasingly computerize the LMR, the job of the outcomes researcher will become progressively easier.

Back to the challenge at hand, at Providence in 2009, the LMR and CRR data flows are still somewhat distinct, as is their purpose. LMRs are consulted by physicians in order to devise a best course of treatment for an individual patient. CRRs are consulted by physicians and research analysts for more statistical purposes, and it might be that the data is somewhat spotty, if the collections process is leaky. Again, the process of CRR harvesting is subordinate to the process of maintaining LMRs.

CLINICAL REGISTRIES

CRRs (clinical research records) are what populate Registries, and these reflect the key interests of researchers. Many are devised by experienced professionals and shared from a central standards-setting source. Submitting data to these registries is important on several levels. The ability of a hospital to keep up such registries is a sign of health in the sense that if the CRRs are this good, then the LMRs must be all that much better.

Understaffed, under-computerized hospitals will not have much time for clinical research. High volume hospitals may also fall behind, despite major investments, because these are difficult problems, as mentioned in the opening sentence. It's not just about budget, but about making smart investments in complementary technologies. Some medical devices are more forthcoming than others with their data. Some interfaces are easier to use. Service providers compete on many levels to solve both the LMR and CRR challenges.

Given the overarching requirement that hospitals deliver the best medical care possible, the medical profession has a strong incentive to share about "what works" across hospitals. Software engineers have this same incentive, even in a competitive environment, because bad design has a way of getting in the way, no matter where it occurs. There's a relentless pressure to find workable solutions and to share them.

DATABASE TOOLS

In terms of storing and retrieving tabular data, the kind storable in rows and columns, a lot of experience and consensus has developed around using Structured Query Language (SQL). An "SQL engine" or "SQL database" is an environment that supports a specialized language involving the verbs INSERT, SELECT, UPDATE, DELETE, and also CREATE TABLE and GRANT as in "grant permission". The SQL environment includes the concept of access rights in other words, which implies the concept of users.

However, control over records may not be sufficiently expressed in SQL and indeed there's strong case for "wrapping" SQL's functionality with a layer of code that works at a higher level. In this higher level, we bring everything about a patient together, instead of scattered through tables. We also have a concept of the user and access privileges.

The ORM or Object Relational Mapper, is what gets between an end user, seeking information from a SQL engine, and the SQL engine itself. The ORM will obey a "fetch" command, which will translate to SQL under the hood, and return a list of objects ready for templating and displaying in the end user's web browser.

The results of a Google or Amazon search may be described in terms of this picture. The key words generate a "fetch" operation and the returned objects gather together URLs (web page addresses) complete with blurbs (web page excerpts), or, in the case of Amazon, book covers, reviews, pricing, publishing information. In the actual tables, these data are scattered, just as procedural details, subprocedural details (e.g. lesions bypassed), and PHI (name, address, SSN) may be scattered across multiple tables in a clinical reseach database.

At the object level, after the fetch has been run, the tabular information has become integrated. It's a relatively simple matter to hand off a "dictionary" to some boilerplate (template) and have the information display out to the user in a familiar, organized format.

TERMINOLOGY

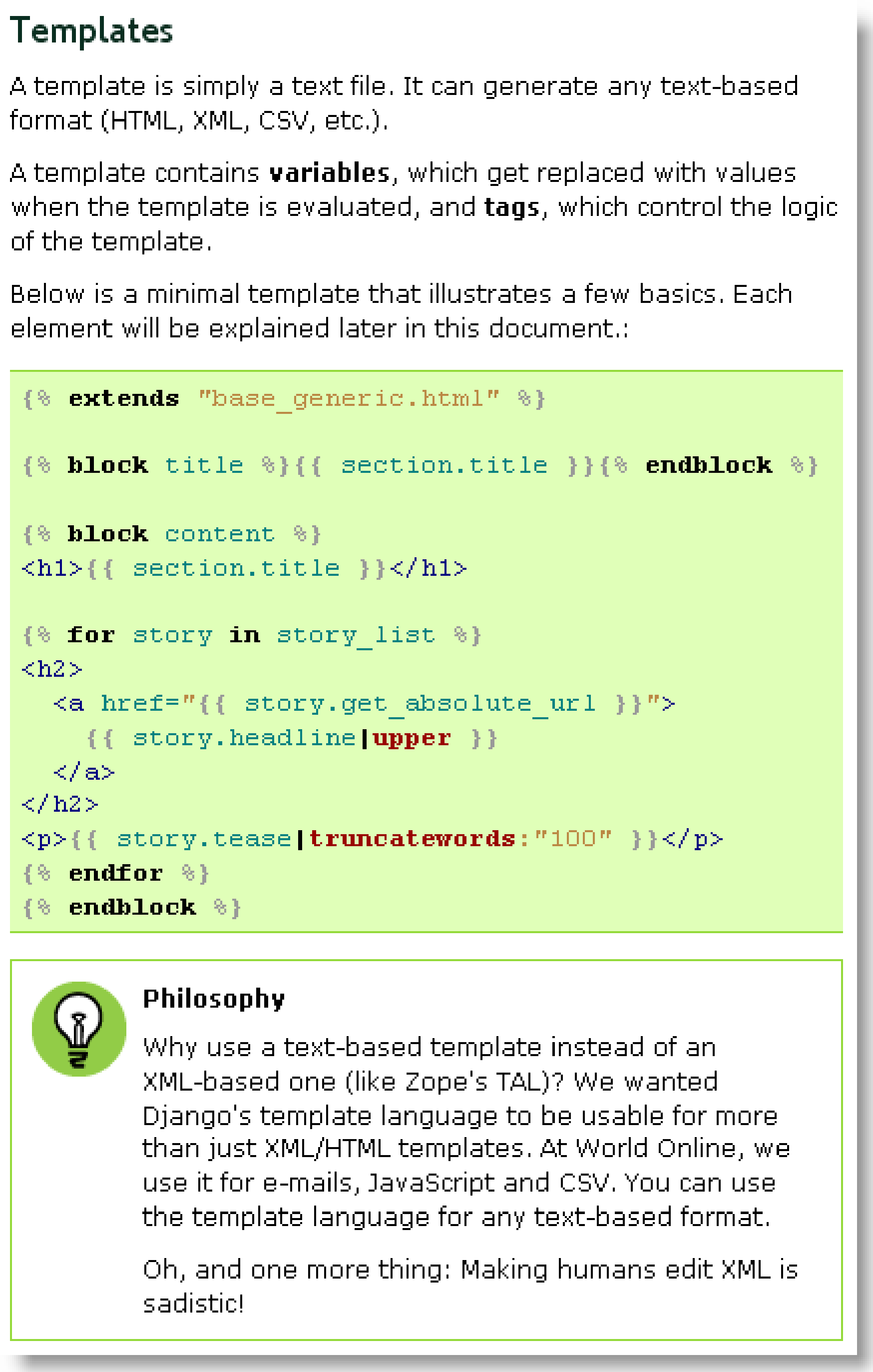
Reviewing the above section, we think in terms of Model, View, Controller (MVC). There's a huge literature spanning many computer languages regarding the MVC design.

I will consider these in the VMC order (View, Model, Controller), which I think corresponds to their ease of conceptual accessibility (the Controller is the hardest to understand).

View

A personnel database might include a worker's picture, home, pager and cell phone numbers. A clinicial research database might show a patient's PHI along with each procedure of a certain type. We've all seen web pages. Imagining these visualizations is not difficult if you have ever used a browser to select an item for purchase.

In the real world, it's common for disparate clinics, even within the same hospital, to be unconnected, such that heart procedures will not display in conjunction with treatments for urinary disorders or child births. These are not LMRs, remember, which do integrate all this history for a given doctor's view.



Templates

A template is simply a text file. It can generate any text-based format (HTML, XML, CSV, etc.).

A template contains **variables**, which get replaced with values when the template is evaluated, and **tags**, which control the logic of the template.

Below is a minimal template that illustrates a few basics. Each element will be explained later in this document.:

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Philosophy

Why use a text-based template instead of an XML-based one (like Zope's TAL)? We wanted Django's template language to be usable for more than just XML/HTML templates. At World Online, we use it for e-mails, JavaScript and CSV. You can use the template language for any text-based format.

Oh, and one more thing: Making humans edit XML is sadistic!

Text 1: talking back to end users, preparing visualizations with templates

CRRs gather piecemeal as clinicians have time to gather and record them. Patients streaming through a busy CVL or CVOR will generate CRRs in GE's Hemo or in the S2 registry in PATS respectively.

In practice, a web page is typically written in a “templating language”, a fill-in-the-blanks affair. The more sophisticated templating languages allow for “forms within forms” meaning an outer shell, like a page of a newspaper, might have preallocated column and box regions which in turn have their own templated contents.

A header, where PHI is displayed, may be designed separately from the enclosing frame, which may feature the hospital name and logo.

In the more contemporary MVC design, the templating is handled by a modified XHTML and CSS within the context of a web framework. Statistical displays such as graphs and charts, insert into this flexible regime, characterized by “floating” as well as “fixed” elements.

Model

A script of CREATE TABLE and GRANT commands defines a set of tables, imposes some low level restrictions. These tables fill with data. The CREATE TABLE command specifies the names of the columns, but also what types of data those columns expect.

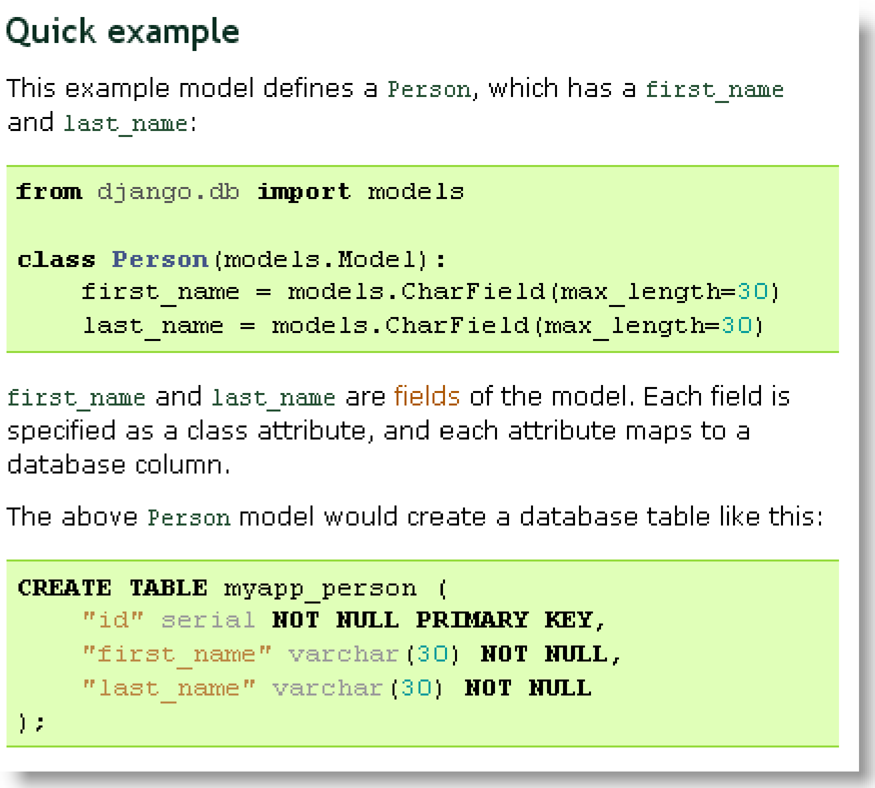
PRIMARY and FOREIGN KEY modifiers describe how the tables relate to each other, e.g. “this medical record number column is unique to each row” (as in a demographics table, e.g. PATS Demog) would be a PRIMARY KEY directive. A table so protected would disallow any two rows from sharing the same medical record number.

If we're talking about multiple hospitals, where the same numbers might indeed point to different patients, then the hospital itself must be identified, to prevent name collisions (mixups in identity).

This exact situation is what pertains at Providence, where PSTV and PPMC each assign medical record numbers (MRNs) independently. The PATS Demog file was devised such that the all-important “account number” would prefix the MRN with a single letter, V or P. Other first letters mapped to other hospitals.

All of this tabular “metaphysics” is considered the Model. The so-called Relational Database Management System (RDBMS) gets stuffed into this category.

In the case of MDRC, the Model that Cecil wrote for us in Sybase, a welcome professional contribution from IS, was all defined in terms of SQL tables. There was no corresponding visualization or view however, no direct user interface to the end user other than SQL itself.


Quick example

This example model defines a `Person`, which has a `first_name` and `last_name`:

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

`first_name` and `last_name` are fields of the model. Each field is specified as a class attribute, and each attribute maps to a database column.

The above `Person` model would create a database table like this:

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Text 2: The same model works with a variety of SQL engines...

Controller

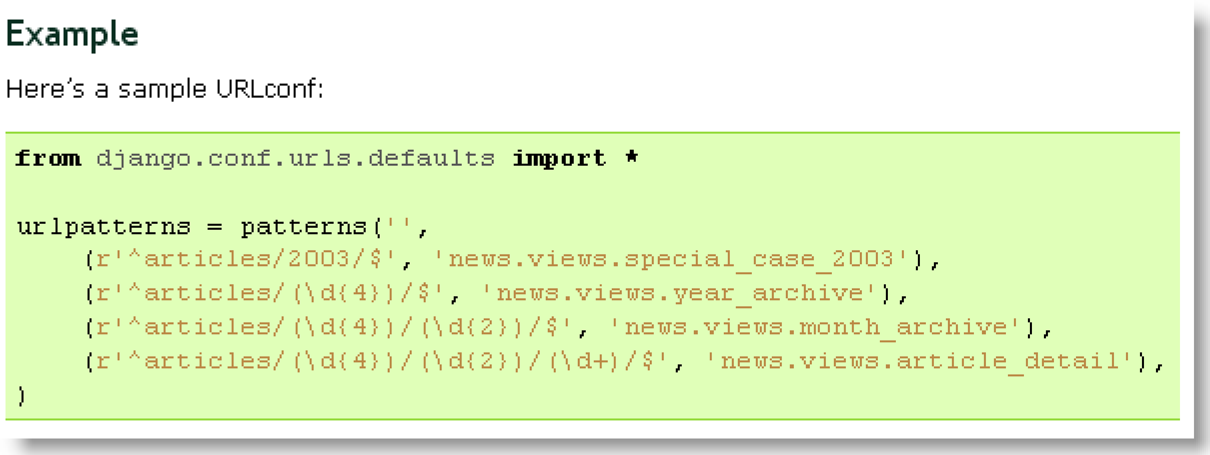
In today's MVC designs, the end user's communication with the SQL engine is mediated by an HttpRequest, that which goes from a web browser to a server, containing whatever the server needs to know about the user and the request (some kind of search, or perhaps a post of new data). The server is responsible for returning an HttpResponse.

The Http refers to the “hypertext protocol” and is familiar to all web browser users thanks to the http:// that occurs in front of almost every address (or URL) fed to a browser.

The controller sits in the server and receives the HttpRequest object, which it unpacks and deals with.

There's a “switchboard” aspect to this job, as URLs must be dispatched to various functions. For example, a URL with the pattern /providence.mdrc.cvl/mrn=”V12347890” would be parsed and dispatched to the function capable of fetching a patient by medical record number.

Recall there is no static web page anywhere in this picture. What the user finally sees on the screen is created dynamically within fractions of a second, in response to data queries and fill-in-the-blanks templates, all stuffed back into an HttpResponse object and returned by the web server (e.g. Apache, see below).


Example

Here's a sample URLconf:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^articles/2003/$', 'news.views.special_case_2003'),
    (r'^articles/(\d{4})/$', 'news.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'news.views.article_detail'),
)
```

Text 3: A switchboard for dispatching URLs, Django web framework

The controller will pass V1234567890 to the a fetch function, a set of SQL SELECT statements will fire in the background, drawing data from multiple tables. The returned “object” is much like a dictionary, and goes to the relevant template for displaying a single patient, perhaps with an associated list of procedures ordered by date in a procedures block.

All of the XHTML gets stuffed into the HttpReponse object (like a suitcase) and goes back to the browser, where it unpacks into the standard web page display for a single patient, complete with hospital logo and perhaps a Chinese spelling of the name in a native language field, if relevant (the industry standard is Unicode, meaning it's now quite practical to store data in multiple character sets).

Session management and user identity likewise enter the picture within the controller. Is the current user authorized to pull up a CRR for a patient by medical record number, including PHI? Because the user is logged in with a password, the HttpRequest is able to inform the server “who wants to know” and a request may be denied, or perhaps a pseudonym is automatically substituted in place of the real PHI, and clearly labeled as such.

The analyst or researcher who wants access to CRRs for research purposes is automatically protected from seeing actual PHI, and yet patient identities remain intact in the sense spelled out in the opening section.

The controller is also able to keep track of user activities, perhaps forcing a log off after a set interval of inactivity. These security features keep a system HIPAA compliant.

In many cases, a researcher is not asking to see a particular case history, but all procedures in a specific time period that meet various criteria. In the MVC design, the nature of the request is sent via an HttpRequest from the browser. The response might be to prepare a file, perhaps many megabytes in size, with a message back to the user that “your file is ready”, with a link to the file, in whatever format. In other words, using the browser for visualizations does not force aggregate data to show up on screen. An analyst might import the requested data file into SPSS or Mathematica or Excel for further analysis.

ECONOMIC CONSIDERATIONS

Not so long ago, a SQL engine was an expensive big ticket item, perhaps licensed from IBM or one of those, and tended by highly degreed individuals proud of their exclusive credentials. Not just any kid with a laptop could download a state of the art SQL engine, web server, web framework, and start messing around as a hobbiest, in some cases transforming into a major contributor. The PC revolution, launched by Apple, IBM and Microsoft, did not initially provide access to SQL engines, and the web had yet to be invented.

Once the Internet became a reality, engineers of considerable talent could find one another and the shared vision was to cut loose from expensive licensing, which made access to tools of the trade unaffordable, not just to small businesses, but to the very graduate students who had helped code the assets in the first place.

As it turned out, using the Internet to co-develop the basic tools was a highly scalable activity in the sense that clearly important projects would snowball and bugs would get fixed, enhancements applied, at a far greater rate than any one private enterprise could afford, at least for those few projects. This was the so-called “open source” revolution, about which much has been written, the upshot of which is we now have robust SQL engines, web servers, web browsers, computer languages, that are not only free, with their source code available, but in many cases are “best of breed” in the sense that proprietary tools of the same genre are not necessarily of higher caliber, or even if they are, the added bells and whistles are not sufficiently attractive to trump the free versions, which are in every way as reliable and up to the task at hand.

For example, the best web server and the most widely used, is Apache. SQL engines MySQL and PostgreSQL drive large ecommerce sites in private industry. Computer languages such as Python, Ruby, PHP, C++, Java, Perl etc. are freely downloadable and their inner workings are unencumbered by restrictive copyrights.

Copyrights still apply, but the restrictions are often permissive, i.e. “you may modify this software only if you pass that right on to others” (GPL) or “you may do anything you like with this software and you don't need to tell anyone” (BSD or MIT licenses).

Because of these developments, it is realistic to design small research databases of a somewhat “academic” flavor, using tools that not so long ago would have carried a prohibitive price tag. This doesn't reduce development costs to zero, because the tools are generic and need to be customized to the task at hand.

If all medical equipment and pharmaceuticals were free, one would still need trained physicians and technicians to practice their skills. Free medical supplies would not make physicians any less relevant, would only amplify what they might accomplish in terms of providing medical care. The analogy is apropos.

SUMMARY

As of 2009, Providence has a reputation as a center of excellence for outcomes research and has every intention of maintaining that reputation.

The infrastructure for gathering clinical information for statistical analysis has undergone rapid evolution as the technologies have developed. For example, angiogram CRRs used to be saved in DBF format (associated with xBase e.g. dBase and FoxPro) by the Quinton workstations. The Quintons were recently replaced by the Hemo system from GE, which wraps a SQL engine by Oracle.

An ongoing question facing Providence management is how to best collect and store CRRs, with an eye towards leveraging any connections with the LMR infrastructure. To the extent that legal medical records have an electronic component, statistical analysis systems should be able to tap in to that data, in compliance with privacy rules.

To the extent that CRRs must be gathered independently of the LMR infrastructure, the challenge is to (a) maximize the relevance of the collected data and (b) minimize the burden placed on physicians, nurses, technicians, anyone providing direct patient care. At CORE, we always thought in terms of carrots (rewards) e.g. perfusionists will enter all these values in the course of a heart operation, because of the studies they will be able to engage in, and because of the personal track record each will build, in the form of logged cases, important for certification.

When it comes to relevance, in-house researchers and canned registries, such as NCDR, provide much of the guidance. When it comes to minimizing the burden, I see a need for a discovery process, not a one size fits all approach.

The implementation architecture I most recommend is MVC for anything multi-user, with a SQL engine for the Model and a web browser for Viewing. If there are proved industry standard tools of a free and open source nature, then the case for using more expensive infrastructure should be made by the vendor, as keeping costs to a minimum is a part of keeping CRR data collection a low overhead activity.

Vendor lock in is a real danger and takes many forms:

- the CRRs may be hard to extract in mass quantities in any usable electronic format
- the design of the Model may lag the state of the art
- deficiencies in the product itself may not be addressed in a timely fashion
- those working in-house who develop talents and skills around MVC design have no opportunity to apply these capabilities

My current chief concern is the MUMPS based Patient Analysis and Tracking System, which has become the core repository for case histories stretching back several decades. PATS is not a SQL engine and has a closed architecture. Cracks have appeared and the Model does not scale. I do not have current information on how the ACC submission process is going, but when Axis Clinical removed our ability to import from scan forms, insisted we enter each record by hand through the user interface, we had to be increasing the burden somewhere.

I recommend developing an in-house culture that standardizes around more affordable, more scalable, and more reliable tools. This is mainly a dream at this point, however dreaming is a recognized step in the management process, thinking of Appreciative Inquiry in particular.

A key question at this juncture would be: do we want to explore my recommendations in more detail?

If the answer is yes, then I suggest we implement a discovery protocol based on interviews. Part of what we would like to discover is who are the stakeholders and what CRR harvesting efforts are currently envisioned or already underway outside of the ones that we already know about. I would also recommend a series of formal presentations in group settings i.e. with stakeholders present, where these tools and concepts are demonstrated in more detail. These could be small events of two or three interested parties, or larger events.

Just as physicians must keep up with their studies in order to remain well versed in their fields, so must information workers have opportunities to improve their skills. Research hospitals need to budget for a kind of in-house teaching function.

My model for this would be Google Code University, which puts material on the web. Given what I've experienced at Providence over a fifteen year period, a group such as CORE (Center for Outcomes Research and Education) and / or MDRC (Medical Data Research Corporation) could be active players in this effort. The idea would be to build up a knowledge base of resources, including people to contact for consultation. A schedule of presentations would include outsiders and visiting MVPs. Collaboration across hospitals would be encouraged.

One purpose of this emerging education intiative would be to continue work on integrating CRR and LMR infrastructures wherever possible. This would be a timely endeavour given all the attention being given to streamlining LMR systems as a way of lowering costs.

Hospitals doing focused development in this area will be making a contribution to an important area of research.

Providence is well positioned to provide welcome leadership in this area.