Lessons Learned from the Implementation of Clinical Messaging Systems

Mike Barnes, MD
Regenstrief Institute, Indianapolis, IN

Abstract

The Regenstrief Institute has designed and implemented two clinical messaging systems over the past six years, both called DOCS4DOCS®. These systems receive HL7 messages from data sources and deliver results to clinicians via the web, fax, or as HL7 directed to an EMR. This paper focuses on some of the lessons we have learned, both good and bad. We discuss important issues in clinical messaging including provider mapping, document delivery and duplicate prevention, creating uniform HL7 outbound feeds, user authentication, the problems of allowing Active-X controls, why automatic printing of documents is not important although a frequently requested feature, and assorted other pearls of wisdom we have acquired.

Introduction

Clinical messaging systems take results that a data source such as a hospital would normally deliver via an internal or external mail system and delivers them electronically. Electronic delivery may improve the incomplete and delayed delivery of care that researchers have documented.1-3 Clinical Messaging has been found to be a good initial project for Regional Health Information Organizations (RHIOs) because it has value for both the clinician (quicker delivery) and the hospital (lower delivery cost)4. Clinical Messaging systems can be easier and quicker to implement than full Electronic Medical Record (EMR) systems because they do not need to have a master patient index or map lab codes (although they can if they wish).

Over the past six years, Regenstrief has implemented two clinical messaging systems, both called Docs4Docs. The initial Docs4Docs implementation was for a single hospital system consisting of four hospitals. The second implementation of Docs4Docs was a complete rewrite based on our experiences up to that point. The primary design difference was that the new version could handle multiple data sources. In Indianapolis, the five major hospital systems formed a consortium and selected Doc4Docs as its clinical messaging system. A new corporation, the Indiana Health Information Exchange (IHIE), was responsible for business operations and administrative activities, such as implementing practices, managing users, tracking missing documents, marketing, etc.

Four major hospital systems are currently operational, with a fifth in final testing. Over 1800 practices receive documents for 4100 clinicians. More than 900,000 documents are delivered monthly via the web, and over 96,000 by fax. For hospital systems, the major types of documents delivered includes transcription, lab and microbiology, radiology, pathology, EKG, and admission and discharge notices.

Brief System Description

Much of the remainder of this paper will detail particular operational and implementation details where we have learned important lessons. However, a brief overview of the system architecture will facilitate later discussion.

Docs4Docs is implemented as a Java servlet on a dual processor AMD Opteron system running a Linux variant. Postgresql is used as the database. There is a two terabyte disk array connected to the processor via a storage array network (SAN). A separate Windows-based CPU runs the fax software, and our current hardware configuration allows for eight simultaneous outbound fax lines.

Docs4Docs obtains its HL7 messages via a central Regenstrief repository. Each HL7 message stream is categorized into one or more Docs4Docs services. A service in Docs4Docs parlance is a specific type of document, such as ‘Hospital A Transcription,’ or ‘Hospital B EKG.’ Service specific code processes the raw HL7 into XML, which is then evaluated by a set of delivery rules. These rules deliver documents to a practice via a web-based in-box or fax. Additionally, documents may be delivered as an outbound HL7 feed directly to an EMR.

Users authenticate themselves to their hospital system or IHIE’s web-portal and from there connect to Docs4Docs (described in detail later). They can then view their in-box and print results. The vast majority of users are practice front-office staff or medical assistants. Most log in and print results once or twice daily. Documents are kept and can be searched for at least a two year period.

The remainder of this paper will focus on individual issues of importance in clinical messaging.
Matching Providers

Delivering a message to a clinician requires extracting the clinician identifiers from the HL7 message, matching the identifier to a Docs4Docs clinician, and then delivering the message to the clinician’s practice. For the first step, we require all source systems provide copies of their provider identifier files containing the provider codes used in the HL7 messages. Since we did not think we could get each hospital system to send us provider files in a format we specified, we accept whatever they can give us. This is complicated by the fact that many hospitals have multiple provider files. Of the five hospital systems, two have something equivalent to a master provider file, while the rest have either provider files that handle only a single service (such as transcription), or a provider file that handles multiple services, with the remaining services handled by individual provider files. In an extreme case, one of our hospital systems has thirteen provider files, including seven for ADT and three for lab.

Initially, we were excited to see all the different fields that accompanied the provider files we were receiving. Many had fields for physician license number, UPIN, fax numbers, addresses, etc. We had visions of using these fields to cross-link clinicians automatically. Our exuberance soon faded as we discovered that the provider files were filled with inaccuracies, and in some cases, the data would contain arbitrary values that would change with each update. We abandoned hope of doing any automatic processing on these files. The usual case was that data would be accurate at the when it was first entered, but would not be maintained with time. Our experience has been that hospitals with a single provider master seem to do a better job with updates than those with a single provider file per service.

Currently, we deal with approximately thirty different provider files from all the data sources we serve. All of these files have to be acquired and require periodic updates. We had hoped to have a unified method of provider file delivery, but this would require each hospital system implement our choice of protocol. Even though they contained no patient data, most hospital systems preferred sending us the data encrypted using a method of their choice, which was different for each system (following their local security conventions). For some we fetch an encrypted zip file, for others we get an encrypted PGP file, and some send us unencrypted files via secured FTP. One of our hospitals is capable of sending HL7 master clinician update records, but we have not implemented this functionality yet. Since we only deal with five or six different sources of provider files, the current acquisition methodology is feasible, but far from ideal. It certainly will not scale.

Once the provider files are uploaded from the source system, they are compared with the current provider data, and the changes are stored in a queue. IHIE staff members can then examine and approve the changes. This process usually goes quite smoothly, but there are occasional cataclysmic events. In one case a hospital system sent us the provider file for the northern portion of the state, as opposed to the central portion. The file format was identical, so Docs4Docs happily processed it. Another time, a system whose provider numbers are mostly numbers, but not always numbers, they were treated as new provider identifiers. In both cases, having updates placed in a queue for approval saved us from cleaning up a giant mess.

The final step of associating clinician numbers in the HL7 with a Docs4Docs clinician is linking all of the appropriate provider identifiers to the master Docs4Docs clinician’s record. As practices sign up for Docs4Docs, IHIE staff enters information about each clinician. They then select a screen where they type in some portion of a clinician’s name, and see all the possible matches from the source system provider files. They then link these matches to the provider. As stated earlier, the demographic information from the provider files is often out of date and IHIE staff coordinates with practice managers or nurses at each practice make sure the correct aliases are selected.

HL7 to XML Processing

A message to be processed and delivered to clinicians arrives at a central Regenstrief Institute gateway where it is stored and fetched by Docs4Docs. This gateway handles all the VPN and data-in-motion security issues. Once acquired, it is then processed by Docs4Docs into a common XML structure representing all the data Docs4Docs requires for document display and delivery.

One of the key design concepts of Docs4Docs is to place all program code specific for a particular HL7 source in one place. Currently, we implement a Java subclass that takes the message and sets a number of parameters. These parameters are then turned into XML. There is no institution specific code anywhere else in the system. The advantage to such a system is it isolates hospital system specific code to a single location.
Document Delivery and Subscriptions

Once the HL7 has been converted to the intermediate XML format, it is delivered to clinicians. Each HL7 message has a number of attributes associated with it that are useful for delivery. The most common two are the type of physician (ordering, referring, primary care, etc) and patient class (inpatient, outpatient, emergency, etc). These attributes are used to create a set of rules by which clinicians can be subscribed (via IHIE practice setup personnel) to receive just the documents they wish (for example, only outpatient documents where the provider is the ordering clinician).

It is not just the clinician who has a say in what documents will be delivered. Each Docs4Docs service may require that certain types of documents be delivered. For example, a lab may insist that reports always be sent to ordering clinicians.

After much experience, we have refined the subscription mechanism so that it is robust and easy to administrate. At the service level (i.e. ‘Hospital A Lab’) there are two sets of subscription rules: one set for the default most clinicians will want, and the other set to force delivery. These are defined at the time a service is created. Practices can have IHIE set subscription rules for each individual clinician at the practice level. For practices with more than one clinician, a single clinician’s rules can be used as a template for the entire practice.

If a service has forced delivery rules, then any document that cannot be delivered goes into a dead letter in-box (if the service requests it). Each service usually has an account within Docs4Docs which allows the service to manage its undeliverable messages.

In practical terms, almost all clinician are happy with the default service rules, so it is usually not necessary to define rules for individual clinicians at the practice level.

Preventing Duplicate/Excessive Report Delivery

Once the reports are turned into XML, they are stored in the database and delivered based on the various rule sets. However, reports require special handling to prevent the same report being delivered multiple times.

Each document coming into Docs4Docs receives a unique identifier called the service unique ID (SUID). This identifier is computed in different ways depending upon the service. Any new document for a service having the same SUID as another document for that service is considered to be a replacement.

Our primary mechanism of duplicate detection involves message digests. A message digest is a means of hashing a document into a string of bits (in our case 160 bits) with the property that any two message digests from different documents have an extremely low rate of being identical. We perform a message digest on the incoming HL7, and if it has the same message digest as another message with the same SUID, then it is considered a duplicate. For the purpose of creating the message digest, we do not consider the MSH segment, since a system sending the same message again may have changes in this segment. We have experienced episodes of source systems resending, for no apparent reason, tens of thousands of previously sent documents. We also have a lab system that routinely sends 20 to 40 percent of its lab results as duplicates.

We recently augmented the message digest detection scheme with a second one that detects changes to documents where the HL7 has been changed, but not in a way that is important to the report. For example, we have a lab service that routinely updates patient billing information and resends the HL7 of about 900 messages daily. This information is superfluous to Docs4Docs. To prevent this type of duplication, we create a message digest of the XML report and compare that against a similar message digest for the same SUID. It would be possible to only use the XML message digest to prevent duplication, but we have found it useful to use both types of message digests (for the HL7 message and the XML), because they detect different types of problems.

Occasionally, hospital systems generate repeated messages at the time of message creation. One example of this is a lab system that might send partial battery results as each new test is received. Depending on the lab system, we may or may not be able to determine if the battery is complete. For those where we cannot, we have the ‘fermenter.’ The fermenter is a holding area for a particular Docs4Docs service where each test is held for a certain number of minutes before being released. Updated versions of documents in the fermenter just replace the previous one.

Fax lab reports require special attention to prevent excessive faxing. Normally, Docs4Docs delivers reports as they are received. For HTML delivered reports, lab reports with the same accession number are grouped together at the time they are printed, with the user determining how many times a day the reports are printed. For fax reports, there is no user determining the frequency of printing. If we faxed a lab report each time one came in, a provider would get one page per battery instead of grouping them
together by accession. To prevent this, we put lab reports to be delivered via fax in a special holding area where they are delivered at a frequency specified by the practice, with the default being twice a day. When the reports are sent, any reports with the same accession number are grouped together. Non-lab reports get faxed as soon as they are received.

Printing and Faxing

We use XSLT to take the XML formatted documents and transform them to HTML for display. For faxing, we use XSL-FO (XSL Formatted Objects) which performs similar functions to XSLT, but is oriented to printed documents. In our case, documents bound for the fax server are converted to PostScript.

One of the more successful things we did is to make all documents look as uniform as possible. Each document has branding information (the hospital icon and contact information) at the top and a uniform patient header. In particular, all lab reports have the same format no matter where they originated. We also map all abnormal flags to similar values so that providers don’t have to take lab origin into consideration when evaluating results.

Using HTML as an output display for printed reports does present a number of challenges. When a user prints a group of reports (as opposed to displaying single reports on the screen), we need to keep track of the number of lines on each page and perform our own pagination. We also have to produce and format our own headers and footers since browsers typically don’t allow HTML to control header and footer information directly. We have used an Active-X control to directly control the browser headers and footers, but the disadvantages outweigh any advantages (described later).

HL7 Outbound

As more and more practices use EMRs, demand increases to supply outbound HL7 feeds to these systems. Initially, Docs4Docs provided a simple pass-through message of the original HL7 message to an EMR, but this was not particularly useful since the receiving EMR was responsible for dealing with many different types of HL7 messages, each with their own syntax and semantics.

We took our existing XML document and applied XSLT to create an outbound HL7 message with similar syntax and semantics across all Docs4Docs services. This was yet another unexpected benefit of using XML as an intermediate format.

The EMRs receiving our HL7 outbound also have to deliver the messages to the appropriate clinicians, so we translate, as much as we can, provider numbers from the original HL7 to a Docs4Docs master provider number. All clinicians that are subscribed to get HL7 outbound delivery are guaranteed to have this number. This allows all EMRs to use a single provider file, generated by Docs4Docs, for all delivery mapping.

One problem we have encountered is that a number of HL7 sources have purposely turned off the patient social security number (SSN) for privacy reasons. Along with other identification information, the SSN is a key component for an EMR matching a patient from the HL7 to one in its database. We have had to work with data sources to educate them of the importance of providing the SSN.

User Authentication

In our original implementation of Docs4Docs for a single hospital system, we used a traditional username and password scheme. As we moved to multiple hospital systems it became clear that each of the different hospitals security committees had differing opinions on what would constitute adequate security.

To address this, Docs4Docs got out of assigning passwords altogether (with one exception described below). Instead, we allow each hospital system’s portal to authenticate the user. Each portal provides a button to gain access to Docs4Docs. This button causes a special URL to be sent to Docs4Docs, containing the name of the portal, the name of the user as known to the portal, the current time, and a digital signature of the above items. When Docs4Docs receives this URL, it first verifies that the URL was correctly signed by the hospital portal and that the timestamps agree to within five minutes (this is to prevent a ‘replay attack’ where a malicious user could simply resend an old URL). Docs4Docs maintains an internal user name, as well as any number of portal aliases to the internal user name. When Docs4Docs gets a valid authentication request, it attempts to match the portal user name to the internal Docs4Docs user name. If successful, the user is logged in.

The advantage of this system is that each hospital portal can provide the type of authentication that fits its needs best. Docs4Docs administrators do not have to worry about granting authentication privileges directly to users, or dealing with password resets. There is still some administration overhead required to communicate user information from each hospital system to Docs4Docs, but this is usually streamlined to a single point of contact for each hospital.

One disadvantage of this system is that it requires the hospital to have a portal, and for the portal to
implement the Docs4Docs protocol. Fortunately, each of our initial hospital systems had available programmers who could implement our protocol, and after it was implemented in a variety of programming languages (Java and .Net being the most common), this code was shared with smaller portals having less programming resources. IHIE also maintains a portal for hospitals who do not have one.

We had thought we were totally out of the password business when we discovered that a large number of practices associated with one of our hospital systems had group logins through their portal. Instead of logging in under their actual name, all the users logged in under a single name, rendering audit trails essentially useless. To solve this, we created a special set of aliases so that the users would log into the portal under the practice name, but Docs4Docs would then generate a username and password screen allowing the user to personally identify him or herself.

**Autoprint and Active-X**

One of the features we were mandated to provide was autoprint—the automatic printing of documents as they arrived. This required that some form of software be installed on the client. In our case we choose a commercial Active-X object which allowed Javascript to control printing of documents. We also used it to turn headers and footers off of printed reports making them look more like reports and less like web pages.

Autoprint was a disaster. Many of the hospital systems had turned off the ability to install Active-X controls. This was compounded by the fact that the install would fail, but would look like it succeeded. So in order to install the control, a disk had to handed carried to the practice, or the Active-X control e-mailed to the client for installation. Since many users had little computer expertise, talking them through the process was time consuming. Additionally frustrating was encountering computers that were locked down and required an IT person to do the procedure manually.

We discovered that users didn’t mind their reports looking like web pages. For all new practices, we do not require installation of the control, and hope to phase it out altogether. Currently, of over 1800 practices, only two or three insist on using autoprint.

**Other Lessons**

One problem we anticipated, but not at the level it has occurred, is the frequency that hospitals make changes to their HL7. Hospitals have changed vendors, upgraded software, added new code sets, and countless other surprises. Sometimes they even tell us in advance. These changes have delayed us enhancing the system as much as we would like because of constantly switching to ‘emergency mode’ to handle the crisis de jour.

Document tracking, being able to track each document as it enters and flows through the system until it is delivered, is extremely important. IHIE receives many requests from practices reporting documents that they think should have been delivered but weren’t. We now have a great many functions to allow administrators to track document flow.

While Docs4Docs audits all patient accesses and user activity, we discovered how important it is to audit even the most mundane of administrative activity. Many problems reported to us as bugs, turned out to be administrators making mistakes. Adding extensive auditing to administrative commands has mitigated this problem greatly.

**Acknowledgement**

This work was performed at the Regenstrief Institute, Indianapolis, IN and was supported in part by grant number 1D1BTM00095-01 from the Office for the Advancement of Telehealth, Health Resources and Services Administration, DHHS to the Foundation for eHealth Initiative. The contents are solely the responsibility of the author and does not necessarily represent the official view of HRSA/OAT.

**References**