Implementing SAS/IntrNet® for Clinical Information Management

Sy Truong, Meta-Xceed, Inc., Fremont, CA

Abstract

Clinical information management is a challenging and complex task in an environment of changing regulatory requirements. Rather than adding extra layers of abstraction to bewilder users, this paper will demonstrate a new system called TRIALEX™ which simplifies the process of managing a clinical trial. The TRIALEX™ System represents the next generation in information delivery since it goes beyond the client/server model and is deployed completely within the thin client model. This system was developed entirely in SAS/IntrNet® to deliver an intuitive Web interface for accessibility and ease of use. This paper will also address some of the unique challenges faced when developing within a thin client architecture.

Introduction

There are many tasks involved during the implementation of a clinical data warehouse that can be automated. This can take place at the beginning design stage and carried through to the end product, the electronic submission. Early on in the design stage, it is important to use tools that not only automate, but also standardize objects within the clinical information system. This structure will set the foundation for the rest of the process. Among some of the design and implementation tasks include:

- Attribute Set up and Standardization
- Clinical Object Management
- Dependency Management
- Refreshing and Scheduling
- Information Delivery
- Review and Validation
- Documentation

These items are established during the design stage and are consistently carried out during the operations and conduct of a study. It is therefore crucial for the tools to have an intuitive interface for getting started quickly, while also having an iterative production mode to efficiently manage large amounts of information.

Attribute Set up and Standardization

An effective way to standardize objects within a clinical information system is to set up a hierarchy representing the organizational structure and work flow. This will ease the transition from managing information within the computing environment to the real world. An example of such a hierarchy is:

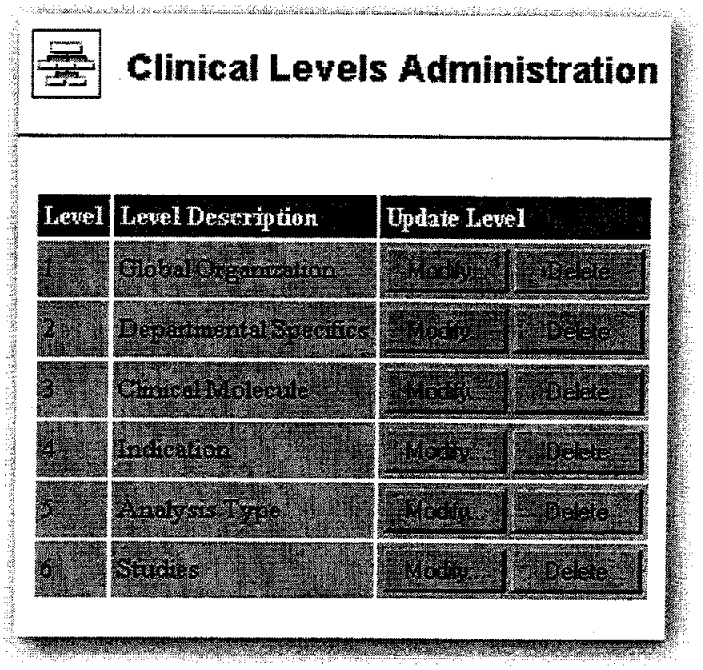

figure 1

Standards sometimes come from the day to day operations as analysts see re-occurring patterns. This usually happens lower in the hierarchy but once accepted as a standard, they are enforced from top down. For example, if a company name needs to be part of all reports in the header section, this can be defined at the global level.

figure 2

This header attribute of the report object is set to the object template that is inherited when new objects are created lower in the hierarchy.

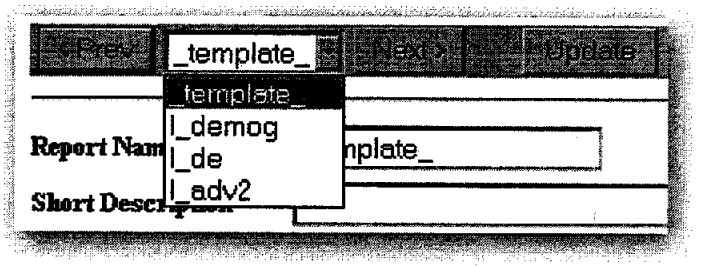

figure 3

There may be cases that contain exceptions so the object

has the flexibility to override standardized inherited attributes. However, standards are applied by default. A report object is just one example of how an object model can be used for standardization. This methodology can also be applied to other objects such as data, programs, and other user-defined objects.

Clinical Object Management

Standardization is a good example of the benefits gained from organizing clinical information in an object-oriented model. The most common objects found in a clinical information system include:

Program *Report* *Data*

New user-defined objects can be created in addition to these common objects. For example, if an organization has developed a set of transformation macros, statistical models or algorithms that may be useful in multiple projects, new objects can be created to manage this and then integrated into the TRIALEX™ System.

There are three distinct types of object attributes. They include the following: common, object specific and user-defined attributes. All objects have standard common attributes such as object name, description and date/time of modification. There are also attributes that are specific to each object type. For example, a report object contains footers and page size attributes that are found only in a report.

FOOTER1	FOOTER2	LINESIZE	PAGESIZE
c:\Programs\X_XXXX.sas and this is some text that is long	Generated: [datetime] By: [userid] Data: XXXXXX.sd2 Status: Interim (Page XXX of YYY)	134	51
c:\Programs\l_demog.sas	Generated: [datetime] By: [userid] Data: a_ptinfo.sd2 Status: Interim (Page XXX of YYY)	134	51

A program object, on the other hand, contains attributes which are specific to programs, such as a header section to store program comments.

A third kind of object attributes, which is different from the common and object specific attributes, is the user-defined extended attributes. For example, a data set can contain common attributes such as data set names and data set labels. It also contains object specific attributes such as variable names and variable types. An example of a user-defined attribute for a data set could be the DEFINE statement used in a PROC REPORT. This attribute is customizable and will contain meaning specific to the organization where it is implemented.

These three types of object attributes allow for extendibility and flexibility in the management of clinical objects, while retaining structure for implementing standards.

Dependency Management

Once objects in a clinical information system have been set up in a standardized manner, understanding the relationship among the objects is the key to managing them. The relationship describes the flow of information between objects to form dependencies.

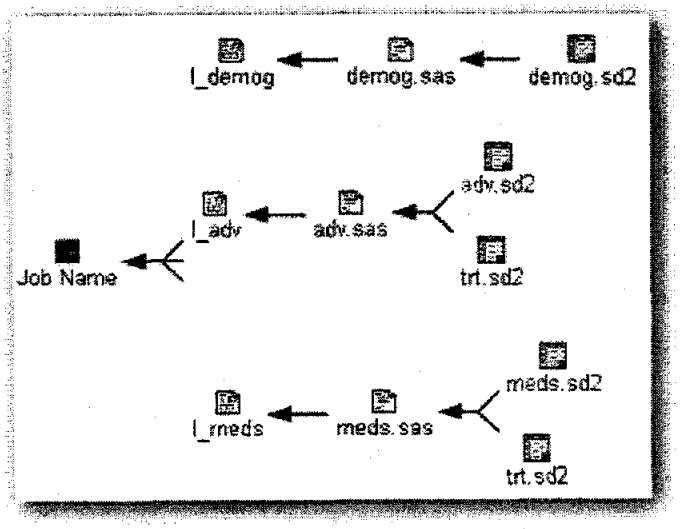

figure 4

Managing these dependencies becomes increasingly challenging as more objects are added to a project. The nature of clinical research also creates an environment of constant change, which adds to the complexity. The two main characteristics used in managing the dependencies include:

- The hierarchical order among the relationships between objects
- The time points at which the relationships were established and last changed

The TRIALEX™ System manages this by storing the information in a SAS® data set. This allows for quick updates and access. A partial view of this data from the above diagram would be:

OBJNAME	OBJTYPE	PARENT	OBJID
Job Name	Root		12473857818
l_demog	Report	12473857818	12490469155
l_adv	Report	12473857818	12490469314
l_meds	Report	12473857818	12490469480
Testing	Report	12473974979	12473178633
adv.sas	Program	12490469314	12490472146
Meds.sas	Program	12490469480	12490472254
Demog.sas	Program	12490469155	12490472412
Demog.sd2	Data	12490472412	12490472580

Once these characteristics are captured, it is then possible to manage the objects by having the ability to refresh the data flow in the right order and apply this to only those objects that have been changed.

Refreshing and Scheduling

The development process of managing a clinical warehouse requires multiple iterations. This is driven primarily by the changes in the data during the conduct of a trial. The changes in statistical models and reporting requirements also add to the need for frequent updates. It is important to understand the order of dependencies within the clinical warehouse, but knowing this by itself is not sufficient to optimally refresh a complex clinical information system. If the order is the only information acquired, the entire warehouse has to be refreshed in order to ensure that the information delivered is up to date. This requires large amounts of resources, especially when this process is repeated. By understanding when each object has been last updated, it is possible to refresh only those objects which are affected by the latest changes. This commonly occurs with the data but it also happens when there are changes to a program algorithm or reporting requirement. With this combined knowledge, the system can strategically apply updates only to objects which have been changed when compared to the previous update.

During the conduct of a study, the source data is changed periodically. This affects the analysis and reporting of clinical reports. It is useful during these situations to have the refresh occur on a regular schedule. For example, if the data base is to be updated every night, a schedule can be set up to repeat the refresh on a nightly basis after the data has changed. This will allow the analysts and statisticians to get the latest information automatically in an optimal manner.

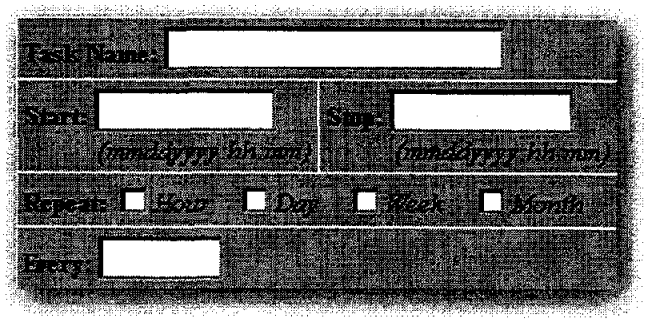

figure 5

There are many options possible for setting up a schedule for updates. The interface allows for precise control over these options while remaining relatively easy to set up.

Information Delivery

The scheduler is a useful tool for automating updates to the clinical data warehouse, but it is also useful when implemented in conjunction with information delivery. The scheduler by default generates a log of the job executed and sends this via email to the person who initiated the schedule, after its completion.

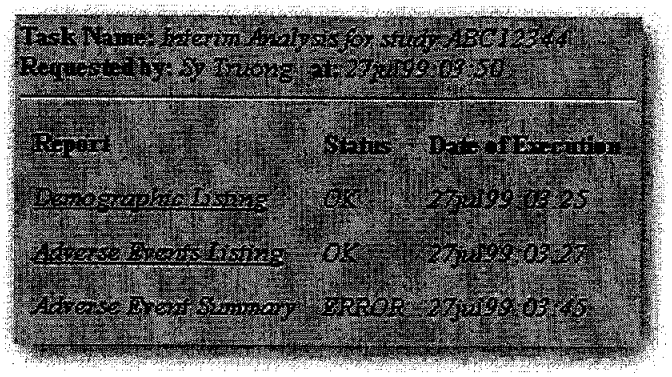

figure 6

This contains hyperlinks in a form of a table of contents (TOC) which goes directly to the web server where the reports have been published. In addition, an option can be set for the scheduler to automatically email the resulting TOC to specified users within the organization.

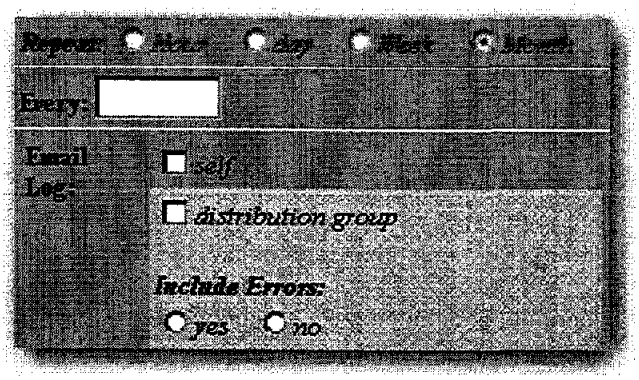

figure 7

This automates the delivery of the information with the use of email and an Intranet. There are two tools used within the scheduled job which enable this information delivery. They include:

- txt2html – tool to convert report text to html
- TOC – table of contents generator

By default, SAS programs produce reports in text format. The txt2html tool automates the generation of the html and publishes this directly to the web server in the proper location.

The TRIALEX™ System organizes the directory structure on the web server in the same manner in which the project directories are developed. The process is therefore transparent to the user when he or she generates reports to a study directory, since it is mirrored on the web server. This allows the analyst to develop SAS programs which produce ASCII text reports while at the same time updating the equivalent html version on the web server.

Publishing reports directly to a web server is an elegant way of distributing information because the information is instantly available within an Intranet. This overcomes the inefficiencies of delivering printed reports. Another advantage to generating reports directly to a web server is that it is one step away from creating the final PDF document for electronic submission. The TRIALEX™ System is integrated with Adobe Acrobat® 4.0 to convert all html reports for a specified project into one PDF document while retaining all the hyperlinks and proper page breaks. Similar to html, the PDF document can be delivered to internal members of the group who may have varying desktops (i.e. Macintosh, PC, and UNIX). However, the PDF document does have some characteristics which are different from html. These include:

- It is self contained without the need for a web server.
- It can be printed with more consistent results.
- Password protection can be applied without the need of a server.
- It is locked from having changes or edits applied.

These characteristics make PDF files more convenient to distribute to members outside the organization such as collaboration partners or CROs (clinical research organizations). The primary external partner, however, is the FDA. PDF format is very useful in delivering information to the FDA since it is the standard format for electronic submission.

A key to locating the information of interest is having a table of contents. TOC is another tool which is used in information delivery and in other ways. In the example of the scheduler, it generates a section of the email which links directly to the reports on the web server. HTML is one type of output which it generates but it can also be generated in RTF format. This becomes useful for distribution during the authoring of a statistical report.

This is used in conjunction with a tool similar to the txt2html, named txt2rtf. All reports can be generated in RTF along with an accompanying table of contents which contains hyperlinks.

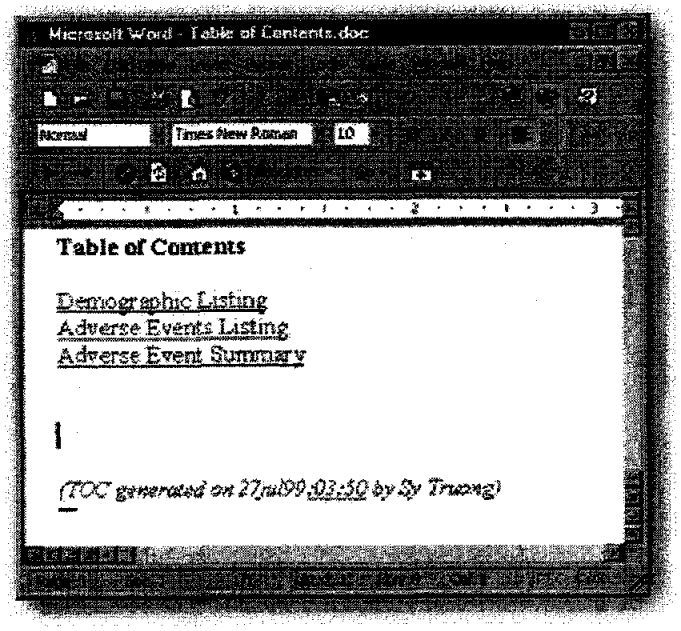

figure 8

This is very useful for the statistician or the author of the final statistical report, since he or she can cut and paste references to appended listings and tables from the TOC directly into the report. Not only does this contain the title text of the appendix, but it also retains the hyperlink to the actual appendix. The final statistical report authored in a MS Word document will then contain hyperlinks directly to the appended tables and listings without the author having to hard code the hyperlink paths manually.

Review and Validation

Delivering information within an organization is the first step towards facilitating a review process. The review process usually consists of a team with different members playing different roles. The two primary roles consist of:

- The author of the object
- The reviewer of the object

The author initiates the creation of a clinical object. The reviewer can be an active collaborator by supplying needed feedback and corrections.

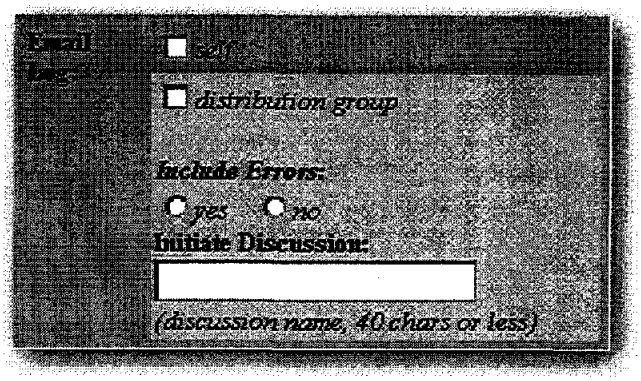

figure 9

Once the review discussion option is set, an email is sent to all the members of the group requesting them to participate in the discussion. If the situation requires that the object be updated with additional information or corrections, the reviewer can respond with comments detailing the needed tasks.

The author can apply the corrections and reply to the reviewer's comments with additional comments and updates to the object. This discussion is captured in a database which is organized by the discussion thread. This model for managing discussions is used commonly in electronic bulletin boards. The advantage that this has over meetings or emails is that it stores the discussion centrally in a "project memory" database which is searchable. This creates an audit trail, which becomes useful when reviewing how a decision was derived and can also be useful for validation.

The review tool is a great way for tracking decisions within a group. There are situations however when an audit trail of one person is required for validation purposes. Usually the one person is the author of the clinical object. An example of this can be applied to the program object. This object contains common attributes such as the time it was last modified. This helps in the management of the audit trail but it also contains object specific attributes designed for source control management.

PRGNAME	PRGDESC	Chktime	Chkstat	chkby
demog.sas	Demographic Listing	31Jul1999: 13:13:30	out	syman
trtfile.sas	Treatment File	30Jul1999: 07:57:57	in	
advsum.sas	Adverse Event Summary	30Jul1999: 07:57:57	in	

A separate source control data set interacts with the clinical object to track the history of the program along with associated comments explaining each programming session. This information along with the header comment section of the program object creates a good audit trail of the program object used for both validation and documentation.

Documentation

The same technologies used for managing dependencies, information delivery, review and validation are also used in automating the documentation process. The purpose of documentation is for team members to keep track of all the information being managed and to understand the data flow at all points of the development cycle. This helps team members stay in touch with the constant changes, and it also helps new team members transition onto the project by giving them a road map showing the flow of information.

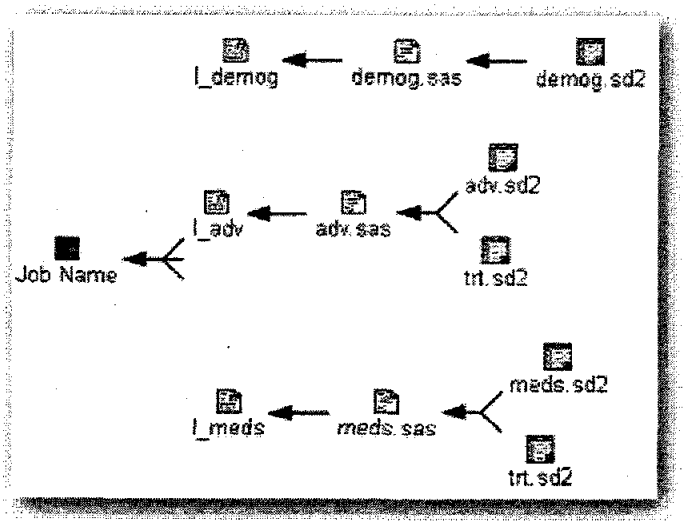

figure 10

This diagrams looks identical to the ones used during the dependency management. The difference is in the navigation. Rather than having the ability to edit each object, the drill down will present a view of the object's attributes. This view may be several layers deep as in the example of the data object. In the first layer of the data object, there are attributes similar to a PROC CONTENTS. The formats also have a drill down to more details, showing the values of the coded values. This gives a thorough documentation of the object organized in a manner that is easy to navigate. Since this is a useful tool for all team members to view at any point in the development cycle, the documentation can be generated at any time.

Implementation

There are too many modules in the TRIALEX™ system to allow for a comprehensive elaboration. Instead, this

section will expand on a few components and techniques. These components represent different aspects of the system so that these examples will be representative of the various technologies implemented. The three components this section will explore include:

- Txt2hml, an API (application programmer's interface) to the TRIALEX™ system used for generating html documents from text reports
- HTML wizards, implementation of CGI and Java Script to accomplish an effective user interface
- Object Relationship Diagram, implementation of html tables and DHTML to accomplish dependency diagrams

txt2html

This tool is classified as an API (application programmer's interface) because it is invoked from a SAS program. Although it is developed as a SCL program, it is invoked with a macro call. This interface was chosen because macro is a familiar construct for SAS programmers. The syntax for this tool is:

```
%txt2html(repname=report name,
          path=new html path [optional],
          titleby=html title variable
                  [optional]
          );
```

The only required parameter is the report object name. By default, the system can figure out the path location on the web server to generate the new html file. The path parameter is therefore optional in the event where the user wants to hard code the location. The titleby option specifies the column from the report object that the html title is derived from.

The TRIALEX™ system is configured with the location of the web server root (i.e. c:\webroot) and the analysis root (i.e. c:\analysis). The report object contains the path to the ASCII version of the report. For example, the demographic listing has the path:

```
C:\analysis\project100\study100\listings
```

In this case, the location of the newly created html file is algorithmically derived to the location:

```
C:\webroot\project100\study100\listings
```

This is derived by swapping out the analysis root and replacing it with the web root. By design, the structure of the web server mirrors the analysis structure. It is therefore possible to dynamically derive the destination of newly created html files on the web sever.

There is a problem with creating html files from ASCII reports since text reports were originally intended to be printed on physical pages of paper. HTML, on the other hand, is intended to be viewed on a browser and does not by default come with the concept of a page. This becomes a problem when html reports are printed or converted to PDF since page breaks are lost. A work around to this problem is integrated into the txt2html tool. This is accomplished by encapsulating each page within the report into an HTML table cell with the width and height set to one hundred percent. This creates a concept of a page so page breaks will occur at the right places during printing and during PDF conversion. The example HTML code for this is:

```
<div align="center"><center>
<table width="100%" height="100%">
  <tr>
    <td><pre>
This is the content of the report
Page 1
    </pre>
    </td>
  </tr>
</table>
</center></div>
```

txt2html is one example of a tool within the API category, which allows programmers to access the TRIALEX™ system tools directly from their programs. Other APIs allow programmers to derive new clinical objects, retrieve their attributes and access TRIALEX™ system information and tools.

HTML Wizards

An effective user interface is one that is intuitive for the user and does not require a steep learning curve. The use of wizards fits this requirement when implemented properly. Wizards are a set of dialog boxes linked together in a linear fashion with buttons including previous, next, cancel, and finish. This is commonly implemented in Windows applications when the goal is to collect information in sequential order. If the information needed is not collected or organized in a linear fashion, a tabbed dialog box is more appropriate. For sequential interactions, wizards are the most intuitive interface.

There are some challenges involved when implementing this with an html front-end since by the nature of this environment, it is stateless. This means that the server does not know the state of the user once the html pages are delivered to the browser. The system has to create a mechanism to track the users so that it can present pages representing dialog boxes of a wizard, in the proper sequence. This is accomplished by the use of hidden fields which keep track of a user's location or state. For example, the html form contains the following hidden fields:

```
<form name="myForm"
 action="/cgi-bin/broker.exe">
<input type="hidden" name="_PROGRAM"
 value="TRIALEX.TRIALEX.wizard.scl">
<input type="hidden" name="_SERVICE"
 value="default">
<input type="hidden" name="form"
```

```
 value="wizard page 1">
<input type="hidden" name="usrname"
 value="syman">
<input type="hidden" name="session"
 value="-0.32336358235063">
...
```

This html page is assigned a unique session identifier number, which is also kept on the server. When the form request comes back to the server, it is matched up and identified. Other fields which help identify the form include the form field and the user name. This technique allows for the system to track each form and therefore enable the management of sequential dialog boxes of a wizard.

Another way of implementing a wizard is to string the dialog boxes into one long html form with spaces between the dialog boxes to create an appearance of separate screens. The navigation between the dialog boxes are handled by Java Script. An example of a simple wizard with only two dialog box is shown here shrunken down to show both dialog boxes in one long screen as seen in figure 11.

The Java Script code segment to handle the Next button is:

```
<input type="button" value="  Next &gt;"
name="next" onClick="linkHandler('#Page2')">

function linkHandler(loc){
   window.location.href = loc;
   if (loc=="#Page1")
   document.newstdy.stdyname.focus();
   if (loc=="#Page2")
   document.newstdy.finishb.focus();
}
```

Once the next button is clicked, the location of the browser navigates to the location of page 2, which is an anchor lower in the same page. Since the page is long, this jump appears to navigate from one dialog box of the wizard to the next. This same technique is applied to the back button. The advantage of this method is that it does not go back to the server to retrieve the next page so the response is very fast. The disadvantage is if there is information from the server that is required in between the wizard dialog boxes, then this method will not work. There are situations where a combination of the two methods can create a seamless series of wizard dialog boxes. It may appear to be fragmented when viewed from the development side but when implemented properly, this will appear very smooth and intuitive to the user.

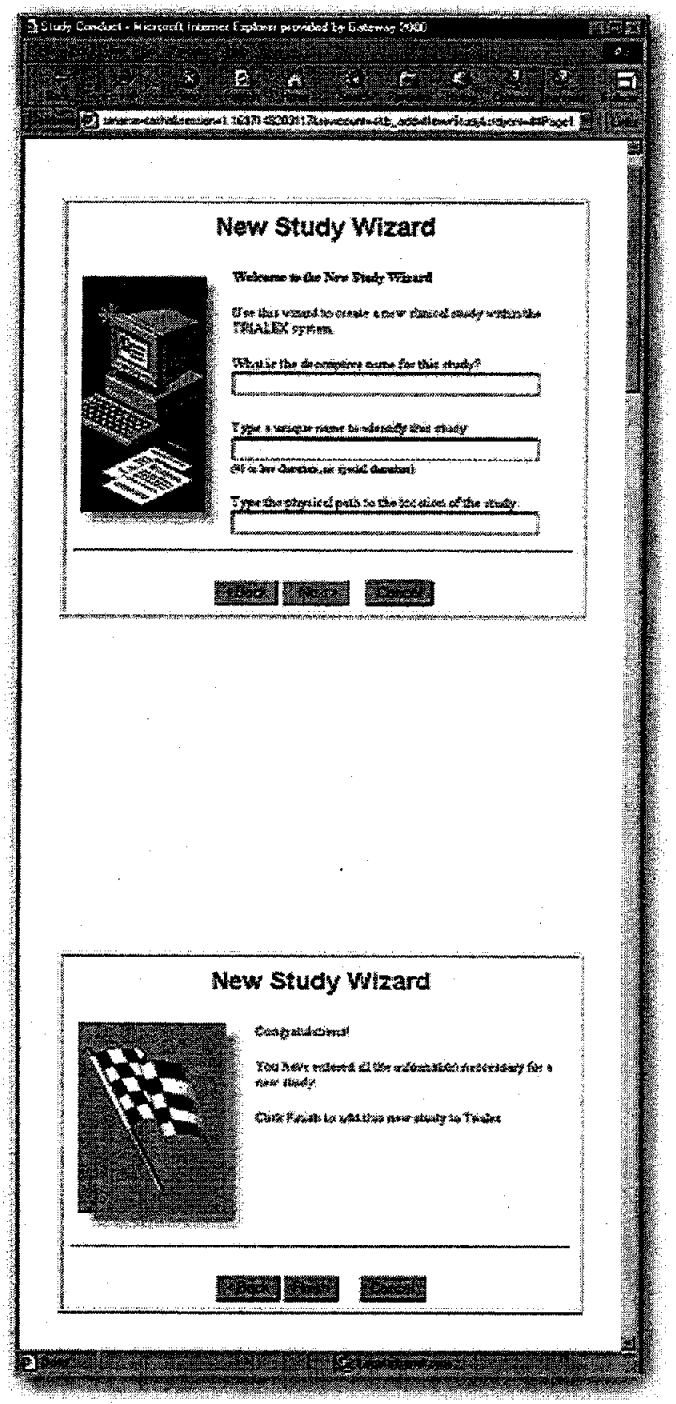

figure 11

Object Relationship Diagram

Diagrams are very effective at conveying data flow, especially for complex information systems. The object relationship diagrams (ORD) used in the TRIALEX™ system is used both in the design stages for creating the dependencies between objects and also for documentation. Rather than explaining all aspects of the implementation of the ORD, which includes adding and

deleting objects to the diagram, this section will focus on techniques used to display the ORD onto a web page.

Before generating the html to display the diagram, the information pertaining to the diagram needs to be captured and organized. A view of this data is:

OBJNAME	OBJTYPE	PARENT	OBJID
Job Name	Root		12473857818
l_demog	Report	12473857818	12490469155
l_adv	report	12473857818	12490469314
l_meds	report	12473857818	12490469480
Testing	report	12473974979	12473178633
Adv.sas	program	12490469314	12490472146
Meds.sas	program	12490469480	12490472254
Demog.sas	program	12490469155	12490472412
Demog.sd2	data	12490472412	12490472580
Adv.sd2	data	12490472146	12490472711
Meds.sd2	data	12490472254	12490472798
trt.sd2	data	12490472146	12490472903
trt.sd2	data	12490472254	12490473074

Note that each object has a unique identifier that is the object ID number. It also contains a parent column which identifies the object upon which it is dependent. This allows the system to figure out the order in which to display each object.

The TRIALEX™ system generates each object in its own html table independently before combining them into the final ORD. For example, the first report object, l_demog has the html code generated as:

```
<table border="0" cellpadding="0">
  <tr>
    <td align="center">
    <a HREF="javascript:doMenu('l_demog',
       '12473857818','1','12490469155')">
    <img src="report.gif" width="14" height="14"
      border="0"></a><br>
    <small><font face="Arial">l_demog</font>
    </small></td>
    <td><img src="arrow1.gif" width="36"
          height="14"> </td>
    <td>
        <!--- #include(12490472412) //-->
    </td>
  </tr>
</table>
```

The object type column determines the proper GIF image to include. The algorithm also counts how many dependent "children" the object has to determine the proper arrow image. There is also a comment text describing any dependents that may need to be included following the current object. A Java Script menu is also invoked when the user clicks on the object. The menu would look like figure 12.

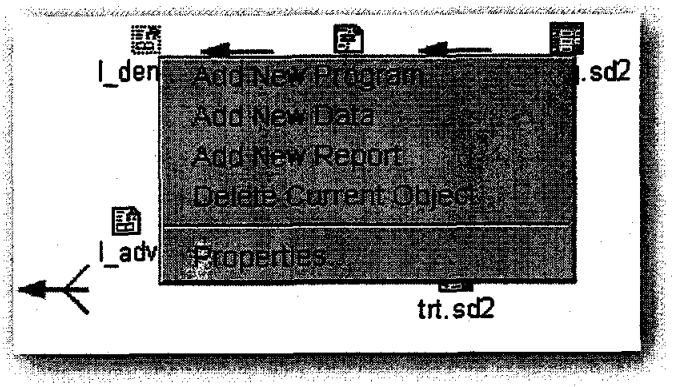

figure 12

After all objects have their own table generated, a second process parses through each item searching for the comment text with the "#include" statement. Each times it sees this, it pastes the html table for the specified object in place of the comment text. This algorithm recursively loops through all the items and draws one large html table with cells containing other html tables to form the entire object relation diagram. Since the html tables are set up without any width applied to the borders of each cell, the users see one smooth image containing the flow diagram. The borders are never shown but are just for demonstrative purposes; this is the same diagram with the border set to have a value of 1.

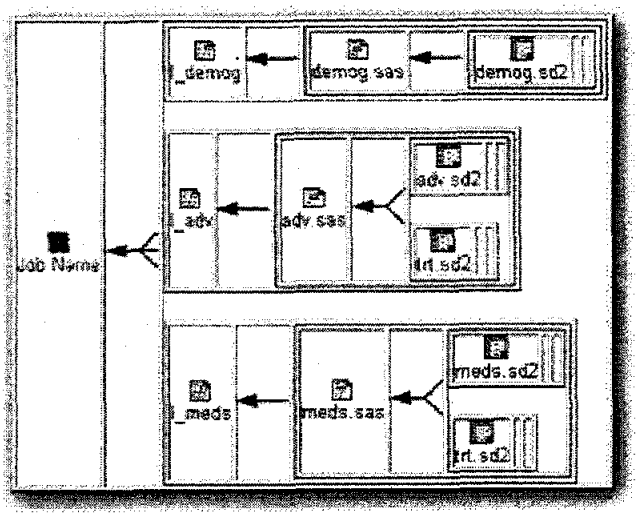

figure 13

The sizes of the html tables are also unspecified so if the diagram is large, it can resize to fit whatever the browser is stretched to. This diagram can therefore be viewed on many different resolution monitors since it is redrawn to match the size of the screen. This also applies to printed versions of the diagram since the size of the tables are redrawn for the specified printer on which it is printed.

Conclusion

Managing a clinical information system is a complex task in a dynamic and ever changing research environment with challenging regulatory requirements. Facing up to this challenge requires innovative solutions which empower the group to work collaboratively internally while also being able to deliver information to outside organizations including the FDA. This means pushing the edge of technologies such as SAS/IntrNet® and also integrating it with other web technologies such as email, Java Script and dynamic html. It is also useful to implement existing technology models and adapt them to clinical information as in object-oriented structure and electronic bulletin board discussions.

It is important to keep a clear vision of the final deliverable when working with large amounts of information. The goal is to deliver consistent and accurate information to the FDA in the format that is required by their guidelines. If this vision is clear from early on during the design stage of a clinical information management system, and then implemented through to the resulting electronic submission, chances for success are greatly increased.

ACKNOWLEGEMENTS

We wish to thank the "team" at Pharmacyclics, Inc. including David Eber and Kathy Boussina for their ideas and contributions to this project.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries.

® indicates USA registration.

Author:
Sy Truong
Meta-Xceed, Inc.
sy.truong@meta-x.com
http://www.meta-x.com