Paper 42-28

StARScope: A Web-based SAS® Prototype for Clinical Data Visualization

Fang Dong, Pfizer Global Research and Development, Ann Arbor Laboratories
Subra Pilli, Pfizer Global Research and Development, Ann Arbor Laboratories
Scott Pivirotto, Pfizer Global Research and Development, Ann Arbor Laboratories
Jeff Van Domelen, Venturi Technology Partners

Abstract

StARScope is a dynamic, clinical trial data visualization prototype. It was developed with SAS® version 8.2, including SAS/IntrNet, SAS Graphics and SAS ODS, and is a web-based application. The web server is on a Unix machine and the user interface is Internet Explorer (5.0 or above) running on any desktop machine. In developing the prototype, we took advantage of the SAS/IntrNet and SAS Output Delivery System to publish graphic reports with hyperlinks to relevant data. We also utilized the newly developed Pfizer corporate data standard--Global Reporting and Data Exchange Standards (GRADES), which follows the proposed pharmaceutical industry standards defined by the Clinical Data Interchange Standards Consortium (CDISC) group. This way, the prototype can be used by any clinical trial studies that comply with the GRADES. StARScope provides interactive access to clinical trial data to non-programmer clients, including clinicians, statisticians, and medical writers. StARScope heavily utilizes graphic presentations to help clients quickly capture salient information from the database in a pictorial fashion. Programmers can also use StARScope for data checking on the research report tables.
Skill Level: Intermediate SAS, Basic HTML including Forms, Java Script

Background

Research oriented pharmaceutical companies, such as Pfizer, Inc., conduct clinical trials to evaluate the efficacy and safety of new drugs. Clinical trials are conducted on thousands of patients before a drug can gain regulatory approval. During the process an enormous amount of data is generated. Traditionally, listing and summary tables are generated for the clinical scientists to review. The volume of the output can be huge, amounting to thousands of pages of data displays. Sifting through the mountain of paperwork quickly and effectively is a daunting task for clinicians, statisticians, and medical writers. Clearly, there is an unmet need for data visualization tools in drug development. Many have tried to fulfill this need, yet successful and user-friendly tools do not currently exist. Some of the difficulties in creating data visualization tools are due to the sheer complexity of clinical trials. Another concern is the lack of data standards, e.g., the same information of a patient's age can have different variable names. Efforts to develop user-friendly tools are ongoing, and "Big Al's World"[1] within Pfizer is one example. "Big Al's World" was well received by the clinicians inside the company and was recognized by the SAS users community in the pharmaceutical industy[2]. StARScope is a next generation prototype designed to fulfill the future needs of data visualization. StARScope utilizes corporate data standards and is a data-driven, dynamic, web-based system powered by SAS software solutions such as SAS/IntrNet, SAS Graphics and SAS ODS. Our goal is to help define the requirements and specifications for a corporate-wide data visualization tool. In the next few sections, some of the features of StARScope are discussed.

Creating A Standard Data Structure

At Pfizer, Inc., there has been a global initiative to develop a corporate data standard. When tasked to develop a data visualization prototype with GRADES, we decided to use an internally defined Reporting Data Set (RDS) structure as the input data structure for StARScope[3]. This has several advantages. First, the systems to support the GRADES standard are still in development. By utilizing the existing data structures defined by the GRADES system, we can focus on the data visualization functions of StARScope. Second, if we anchor StARScope to this reporting data set structure, all clinical trials that comply with GRADES can use StARScope to view data interactively.

GRADES define standards within data classes, which are sets of logically grouped data. The data class conceptualizes how users consider their clinical data. Data Class Definitions (DCDs) include specifications for data derivation algorithms, reporting (analysis) dataset structures, table shells, and reporting algorithms. The system creates reporting datasets using data derivation algorithms. The reporting dataset contains all the information and derivations necessary for the production of the DCD defined data presentations.

Overview of the Technology Used

One major design consideration concerned which tools to use. Since StARScope is intended to be a web-based application, we chose SAS/IntrNet, SAS Graphics and SAS ODS because of our knowledge and experience as SAS programmers. Some basic HTML was necessary to create the user-interface and we found that incorporating JavaScript into the Graphical User Interface (GUI) increased the dynamics of the application. We found SAS/IntrNet to be easy to use for SAS programmers who wish to develop web based applications.

The three main web technologies used by StARScope are SAS IntrNet, JavaScript and Perl. First, we chose SAS/IntrNet as the CGI component because of its ability to connect to the underlying SAS data without requiring additional programming effort. Also, by choosing SAS/IntrNet existing SAS macros and programs could be easily converted to work with StARScope.

Second, we chose JavaScript for its ability to add dynamic content to static HTML pages. JavaScript can be easily incorporated into an HTML page produced with SAS code. JavaScript has the added benefit of being able to provide HTML form validation at the browser level without having to interact with the server. Finally, we chose Perl for its ability to easily access UNIX directory structures and to provide password encrypting and verification.

Details of the Application:

StARScope is a dynamic, data-driven web application. The first screen of the application is the Login screen where the user is asked to enter his or her userid and password. After successful login to the application, the user selects a study and a data version, which is of the reporting dataset structures from the

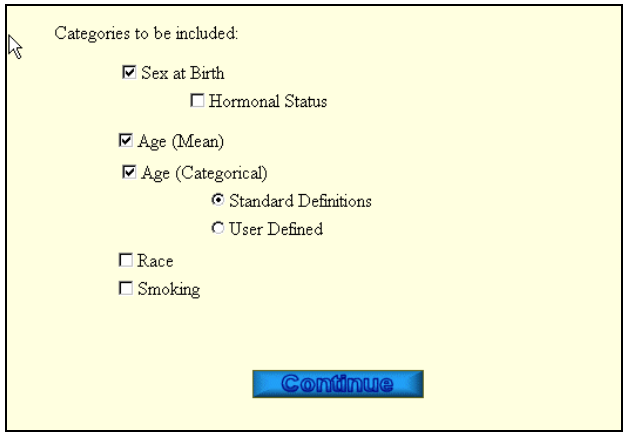
GRADES system. The user then selects one of the modules, e.g., Demo, AE, or Lab. There are several options available for each module, including selecting variables and selecting a table or graph output. After selecting the module and its related options, the user can then submit the job that will trigger execution of SAS code and show the output in the form of tables or graphs on the screen. StARScope will provide an option to download the output to a file or send it to the printer.

The flowchart for this process is shown in Figure 1.

Passing variables from the web Form to SAS

When developing StARScope, we use SAS/IntrNet as the Common Gateway Interface (CGI). A CGI is the method of transferring data from the browser to the server programs that run the application. SAS IntrNet uses a program called Broker to read the data submitted with the HTML form and create the SAS macro variables that are passed to the back end SAS program. It is important to note that only selected fields will be passed from the broker to the SAS program. As a result, macro variables will not be created for non-selected fields on the form. Figure 2 illustrates the selection of information that will be used for demographics data. Figure 3 lists the HTML code that created the form in Figure 2.

Figure 2



When the user presses continue, the broker application takes the fields submitted in the HTML form and passes them to the SAS program. The HTML form data are passed as macro variables into the SAS program. System macro variables are also passed and identified with a" _".

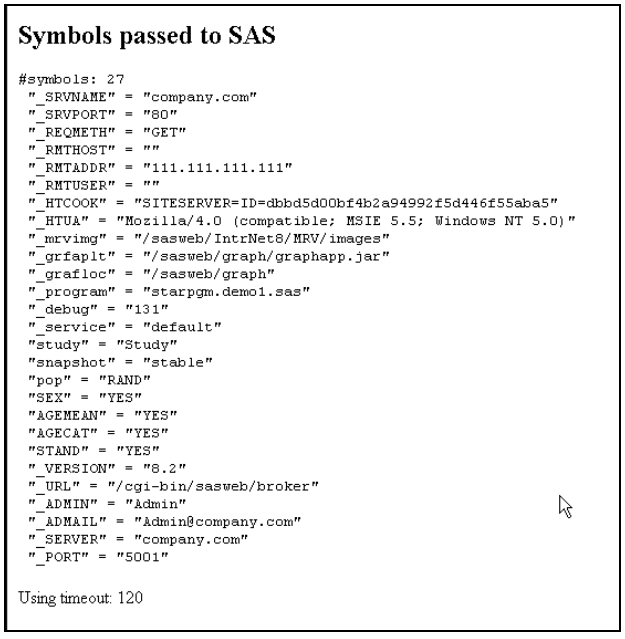
Figure 4 illustrates the screen after the user has clicked on the continue button. Here the de-bug feature of SAS/IntrNet is turned on.

A user can select multiple values on the check boxes on the web form show in figure 5. It is common to have a specific field referenced more then once in the HTML form. Figure 5 illustrates that check boxes for the laboratory tests are all named "lab" in the form, but take different values if checked.

Figure 3

```
<table border="0" width="50%" height="112" al>
  <tr>
    <td width="100%" height="19" colspan="2">Categories to be included:</td>
  </tr>
  <tr>
    <td width="10%" height="66"></td>
    <td width="90%" height="66">
      <table border="0" width="100%">
        <tr>
          <td width="100%" colspan="2"><input type="checkbox" checked  name="SEX" value="YES">Sex
            at Birth</td>
        </tr>
        <tr>
          <td width="13%"></td>
          <td width="87%"><input type="checkbox" name="HORMONAL" value="YES">Hormonal
            Status</td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td width="10%" height="19"></td>
    <td width="90%" height="19"><input type="checkbox" checked  name="AGEMEAN" value="YES">Age
      (Mean)</td>
  </tr>
  <tr>
    <td width="10%" height="19"></td>
  </font>
  <td width="90%" height="19">
    <table border="0" width="100%">
      <tr>
        <td width="100%" colspan="2"><input type="checkbox" checked  name="AGECAT" value="YES">Age
          (Categorical) </td>
```

Figure 4

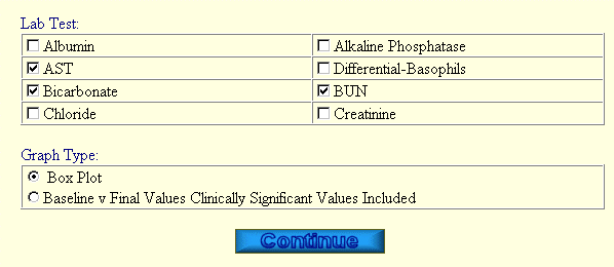


Symbols passed to SAS

```
#symbols: 27
 "_SRVNAME" = "company.com"
 "_SRVPORT" = "80"
 "_REQMETH" = "GET"
 "_RMTHOST" = ""
 "_RMTADDR" = "111.111.111.111"
 "_RMTUSER" = ""
 "_HTCOOK" = "SITESERVER=ID=dbbd5d00bf4b2a94992f5d446f55aba5"
 "_HTUA" = "Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
 "_mrvimg" = "/sasweb/IntrNet8/MRV/images"
 "_grfaplt" = "/sasweb/graph/graphapp.jar"
 "_grafloc" = "/sasweb/graph"
 "_program" = "starpgm.demo1.sas"
 "_debug" = "131"
 "_service" = "default"
 "study" = "Study"
 "snapshot" = "stable"
 "pop" = "RAND"
 "SEX" = "YES"
 "AGEMEAN" = "YES"
 "AGECAT" = "YES"
 "STAND" = "YES"
 "_VERSION" = "8.2"
 "_URL" = "/cgi-bin/sasweb/broker"
 "_ADMIN" = "Admin"
 "_ADMAIL" = "Admin@company.com"
 "_SERVER" = "company.com"
 "_PORT" = "5001"

Using timeout: 120
```

Figure 5



When multiple selections are made for the same field, the broker passes them in an array (Figure 6).The form field is parsed into sequential macro variables and a new variable is created,'*var*0', which provides the size of the array. The macro variable &lab0 will contain the number of variables that were created and &lab1 through &labN will contain the values.

Figure 6

```
"snapshot" = "stable"
"LAB" = "AST"
  #symbols: 4
   "LAB0" = "3"
   "LAB1" = "AST"
   "LAB2" = "BICARBONATE"
   "LAB3" = "BUN"
```

When only one value is selected, the broker passes it as a variable, not as an array, (see Figures 7 and 8).

Figure 7

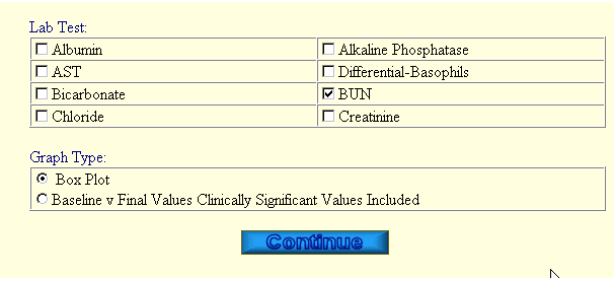


Figure 8

```
"snapshot" = "stable"
"LAB" = "BUN"
"GRAPH" = "BF"
"GRP" = "ABM"
```

If only one laboratory value is selected, the broker will not parse the field. This means that &lab0 and &lab1 will not be created. Only '&lab' will be available. This poses a challenge to the backend SAS processing since values will be passed as either an array or as a single variable. Therefore the code in Figure 9 is used in SAS to check whether only a single value was passed, and if so, render the single variable to a one-dimension array. If only one value is passed, the value of &lab0 will be null, and the value &lab will be move to &lab1 and &lab0 will be set to 1. Now the form data can be processed the same way as when multiple values are selected. The backend SAS program can then process consistently using an array regardless how many boxes the user checked.

Figure 9

```
%global lab1 lab0 ;

%if &lab0= %then %do;
  %let lab1 = &lab;
  %let lab0 = 1;
%end;
```

Passing the data presentations back to the Form from SAS

SAS ODS was used to send the SAS output to the browser window. Because StARScope is data driven, there are no static web pages stored. The HTML is sent back to the browser by assigning _webout as the output file. The ODS statement in Figure 10 is the standard ODS statement used with StARScope. The option 'no_top_matter' is used because the SAS output is being appended to HTML code and the 'no_bottom_matter' is used because custom HTML is being added to the bottom of the SAS output. StARScope uses gif files for graphical output. To avoid saving the files on the server, the gif files are stored in a temporary catalog and sent to the browser window. The path statement below tells SAS IntrNet that the gif files are in its temporary directory.

Figure 10

```
ods listing close;
ods html body=_webout ( no_top_matter
no_bottom_matter)
      path=&_tmpcat (url=&_replay)
      rs=none ;
```

The HTML code is written using SAS 'put' statements within a DATA *_NULL_* step to create forms and make sure the macro variables are properly quoted so it is resolved at the desired time. Since the & symbol is the delimiter within an URL, SAS must not resolve the & within the string.

Since URL string can contain many different fields, it is easier to create the string in multiple steps. Then the different strings representing different values within the data may be concatenated together to form the whole URL. Figure 11 shows an example of creating an URL in a data step and putting it out to the browser. Otherwise it is very confusing to write the long URL in one 'PUT' statement.

Figure 11

```
file _webout;
http='_program=stgm.dem_list.sas&_service=default&ref=load';
pop='&POP=' || "&pop";
where='&where=' || "&where"

put ' <a href="';
put "&httpbase" http +(-1) pop +(-1) wher +(-1)  ;
put "and trt="" +(-1) trt  """>';
```

Creating Hyperlinks

One of the main StARScope features is the ability to 'drill' down to the underlying data. This is accomplished by using hyperlinks within the graphs and reports. Users need the ability to easily access the source data that produced the graphs and reports. This is accomplished using SAS graphs with the *html* and *html_legend* options. These options allow the procedure to create hyperlinks to the underlying data.

The application creates the graph of lab values at baseline verses the value at the final visit (Figure 12). The legend serves as a link to the graphs where only the particular treatment group is displayed (Figure 13). The URL is assigned by the variable in the html_legend and is created the same way a URL would be created in a put statement. The html statement assigns the URL for each point on the graph and is the link to a table of lab values (Figure 14).

Figure12

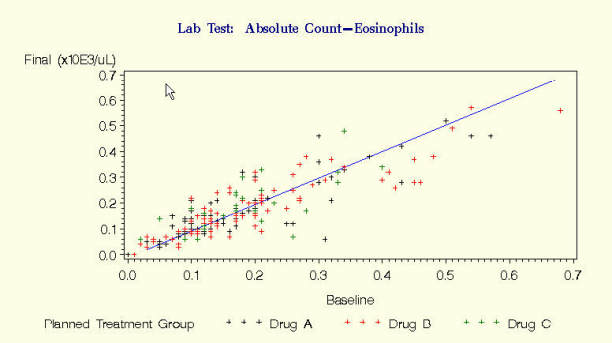


Figure 13

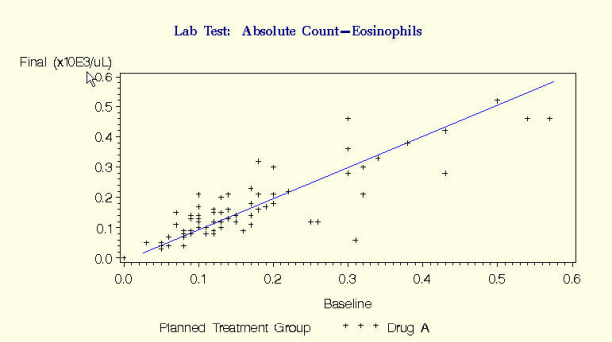


Figure 14

Subject Listing of Clinical Laboratory Values
Report Date: 11/06/02

Subject: 0008-0024-005
Age: 74 Sex: Male Race: White
Treatment: Drug A

Laboratory Test Name: Absolute Count-Eosinophils

			Original Units				Standard Units			
Study Day	Normal Range Indicator	Potentially Clinically Significant	Unit	Value	Low	High	Unit	Value	Low	High
Day -11	N	N	x10E9/L	0.31	0.05	0.5	x10E3/uL	0.31	0.05	0.5
Day 8	N	N	x10E9/L	0.35	0.05	0.5	x10E3/uL	0.35	0.05	0.5
Day 22	N	N	x10E9/L	0.26	0.05	0.5	x10E3/uL	0.26	0.05	0.5
Day 29	N	N	x10E9/L	0.06	0.05	0.5	x10E3/uL	0.06	0.05	0.5

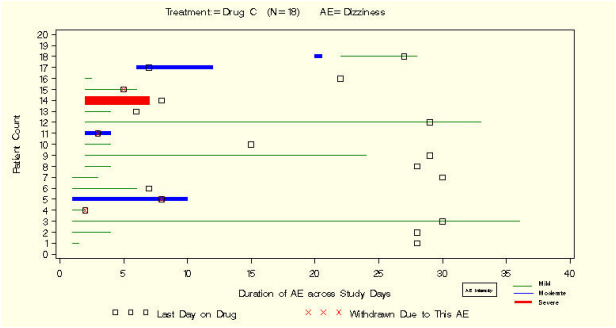
The SAS code for the graphs is shown in Figure 15. The variable "labdrill" is the full URL path for creating the listing of lab values and "trtdrill" is the URL for running the graph again with the particular treatment group.

Figure 15

```
proc gplot data=lab uniform ;
    plot  labf*labb=trtgrp /  vaxis=&min to &max by &scal
    haxis=&min to &max by &scal
    annotate=triangle  html=labdrill HTML_LEGEND=trtdrill;
    label labb='Baseline' labf = "Final (%cmpres(&unit))";
    format labb labf ;
run;
```

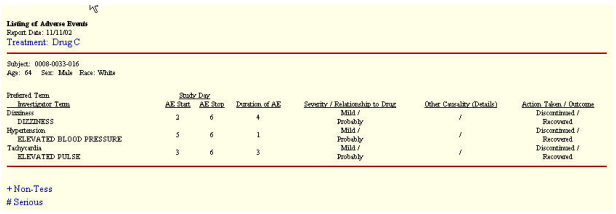
The next example of creating hyperlinks with SAS ODS is in the graph of Adverse Events (AE). One of the things that the clinical scientist looks for is the time to onset of an AE and the intensity of the AE. StARScope enables the scientist choose the AE that he or she is interested in and display a graph that shows the duration, intensity and onset of that AE (Figure 16).

Figure 16



The scientist can also drill down to a specific subject's data via a hyperlink to look at all the AE information for that subject in the graph (Figure 17).

Figure 17



Lessons Learned

Overall, we are very pleased with how well the prototype has been received by the pilot users from the client community. The prototype is very user friendly and easy to use. With the pilot study that was used in the prototype, there was a need to break the blind for one arm (Treatment group) mid-course during the study. StARScope was used to display safety data for that treatment group, which helped the clinicians quickly identify any critical issues. StARScope was also used for identifying data anomalies, such as discrepant dates or values, which would not be found in a timely manner with the normal data cleaning cycle.

The developers' intention was to create a system that was data driven and did not require file storage on the server. This proved to be a daunting requirement considering the stateless nature of

the web, where the state information is not kept from page to page.

Some of the pitfalls in the web applications are inherited from this stateless nature of the web. It is difficult to pass information from one web page to subsequent pages at the browser level and it is inefficient to constantly communicate with the server. Although we could have resolved this issue by using cookies, we decided early in development not to use cookies because their particular complexities, and discussing them is beyond the scope of this paper. We also chose not to write any intermediate data back to the server to increase efficiency and decrease storage on the server. Therefore we designed the application to be driven by the underlying data, to create output 'on-the-fly' (not stored) and not require intensive user interaction.

As a final note, although SAS/IntrNet only requires basic SAS knowledge, being well versed in SAS Macro Language is necessary to develop a comprehensive SAS/IntrNet based application. Knowledge of HTML is also required along with some experience with Java Script.

Conclusions

Overall, SAS/IntrNet is a useful tool to design a web based data browser. It allows one to take full advantage of the SAS analytical and reporting power to present data graphically and dynamically. A SAS program can easily be adapted to work with SAS/IntrNET and deliver the presentation on the web. The prototype was well received by our pilot clinical users. What the end users found the most useful is the dynamic and interactive data browsing for relevant data points.

REFERENCES:

1. Synowiec, R (2001)
Data Visualization of Phase 1 Clinical Studies-"Big Al's World". Proceedings of the Twenty-Sixth Annual SAS User Group International Conference, 26.
2. Synowiec, R (2001)
Data Visualization of Phase 1 Clinical Studies-"Big Al's World". Best Paper, PharmaSUG 2001
3.The name StARScope comes from the department where the developers work in. StAR stands for Statistical Analysis and Reporting. StAR plus data visualization becomes StARScope.

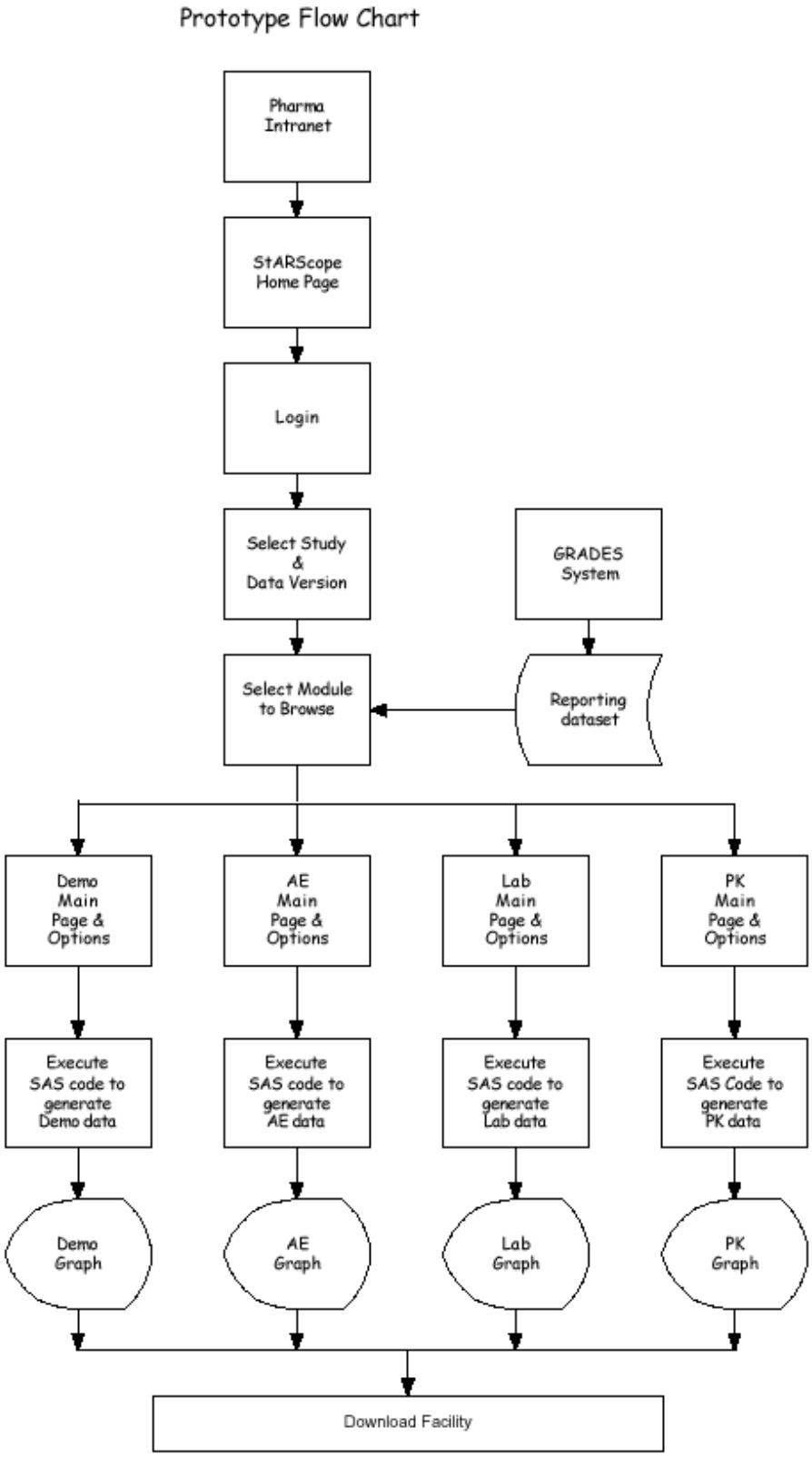
ACKNOWLEDGEMENTS:

The authors wish to thank John Arbuckle, Jim Sundberg, Steve Pohl, Rich Synowiec and many other colleagues for their advice and support in the StARScope development project.

CONTACT INFORMATION

Fang Dong, PhD
Ann Arbor Laboratories
Pfizer Global Research & Development
2800 Plymouth Road
Ann Arbor, MI 48105
Email: fang.dong@Pfizer.com

Figure 1.


Prototype Flow Chart
Pharma Intranet
StARScope Home Page
Login
Select Study & Data Version
GRADES System
Select Module to Browse
Reporting dataset
Demo Main Page & Options
AE Main Page & Options
Lab Main Page & Options
PK Main Page & Options
Execute SAS code to generate Demo data
Execute SAS code to generate AE data
Execute SAS code to generate Lab data
Execute SAS Code to generate PK data
Demo Graph
AE Graph
Lab Graph
PK Graph
Download Facility