SAS® Macros in an Era of Apps

Sy Truong, Meta-Xceed, Inc. (MXI), Fremont, CA

ABSTRACT

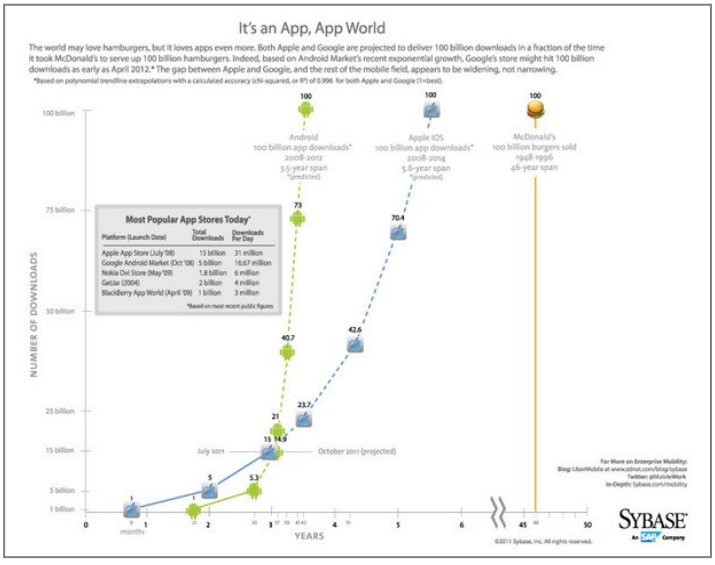
Software development is a dynamic environment shifting from large enterprise systems to small utilities or "apps". The success of marketplaces such as Apple's App Stores has proven that specialized small apps used for specific purposes are essential for large groups of users with diverse requirements. This paper takes a new view of SAS macros in the framework of mobile apps. SAS Macros can thus be showcased to a community of users such as an App Store and then automate the process of downloading and installing, analogous to mobile apps. Mobile App Stores has been successful in delivering millions of apps since it has made it easy for users to find their app through user reviews, documentation and screenshots. Some even allow users to test drive the apps. The Macro App Store allows users to submit the macro with sample data as a test drive. Once an app is found, it can be instantly installed and used upon a single click. The Macro App Store applies the same methodologies to many free SAS macros that were once left in obscurity.

100 BILLION DOWNLOAD

SAS macros have been a powerful tool to automate common tasks, but it has not garnered the popularity and usage outside the geeky programmer community. One of the challenges is having a centralized location where it is easy to find and download. This is in contrast to the explosive popularity and growth of mobile applications or "Apps". Just as a comparison, it took McDonald's about 48 years from 1948 to 1996 to have sold 100 billion burgers. As I was growing up, this always intrigued me with the large golden arches sign displaying the large numbers stating how many burgers are served. In comparison, Apple's App Store is estimated to reach the 100 billion download mark within 2014. That is a staggering amount of downloads in a relative short time. Android, the mobile operating system from Google, is estimated to reach the 100 billion download by 2012. This is even faster than the growth of Apple according to Infographics.

Some of the reasons contributing to this can also be applied to how we access SAS macros.

1. **Centralized App Store** – A one stop shop where users can go to find a variety of SAS macros
2. **Finding Your App** – A search capability analogous to Google where a short key phrase will result in relevant macros
3. **Easy Download** – A user friendly way of downloading free or purchasing paid SAS macros
4. **User Reviews** – Using the wisdom of crowds and crowd sourcing to identify the best macro based upon peer review
5. **Eco System for Uploads** – A community of users which allows programmers to upload their own



macros to be shared within the community

This paper presents a SAS Macro App Store providing some of the features that is useful in an App Store, but applied specifically to SAS macros.

CREATING APPSTORE

The development of an App Store for macros requires an intimate understanding of how SAS Macros are developed and used. SAS programmers have the skills needed to create an "App Store", but one of the main challenges is to obtain skills that are outside the normal skill sets of a SAS programmer. Some of the technologies required to create a social network platform and an e-commerce site are not typically performed during statistical analysis and reporting. There are several different approaches, but the technologies I used for this project include: HTML, JavaScript, AJAX, PHP, mySQL in addition to SAS. I took a unique approach to the problem by viewing entire website and eco-system from a SAS programmer's perspective to solve problems. This initially started with the organization of the SAS macro code itself and continuing into the process of developing the backend server that stores the macros.

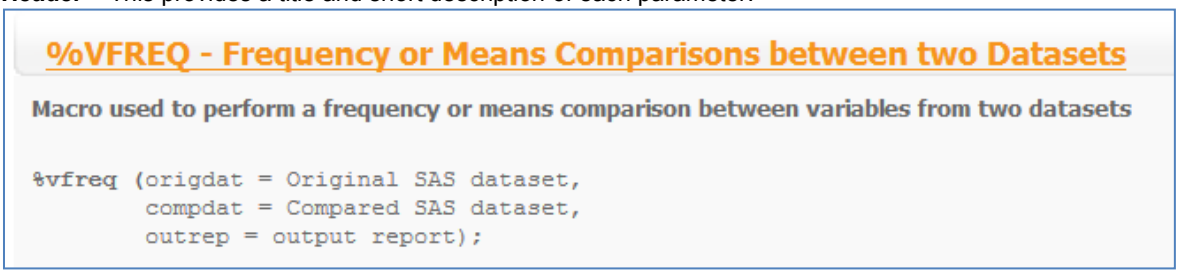
The following describes some main steps required in the development of an App Store for macros. Although some steps require programming languages or techniques not familiar to SAS programmers, I draw analogies to SAS for each component, describing how they relate to SAS techniques and software development approaches. This is an effective technique for this project since the objective is to have a solution where the audience is familiar with, or are hardcore SAS programmers.

STEP 1 – Documenting SAS Macro

The first step is to organize and document SAS macros in preparation for it to be used more broadly. A macro that is used by the single author requires very little in terms of documentation compared to one used for a whole organization, or across companies. Since the Macro App Store will deliver macros to multiple users from different organizations, it requires more detailed documentation.

The reference documentation for a macro details what the macro does and what is required for each parameter. For example, a macro named %VFREQ is used to verify data using PROC FREQ is shown here.

1. **Header –** This provides a title and short description of each parameter.



2. **Parameter Detail –** This details each parameter and documents the expected values.

Where	Is Type...	And represents...
origdat	C (required)	A two level dataset specification of the original dataset to be compared. include: • analib.c_adae • rawdata.conmed
compdat	C (required)	A two level dataset specification of the new dataset used for comparison include: • work.c_t_demog • analib.c_adae

3. **Functional Detail –** This explains in more details the functionality of the macro to provide context so the user knows how to use it.

Details

The VFREQ macro is used to perform the verification between two datasets that are similar. For all variables with the same name, it will perform a PROC FREQ or PROC MEANS depending on weather the variable is considered categorical or continuous. The VFREQ will consider a variable to be continuous if it is of numeric type and has more than 20 distinct values. Otherwise, it is considered a categorical variable. For those frequency counts where it finds more distinct values in one data compared to the next; it will identify these differences and summarize them in the output report.

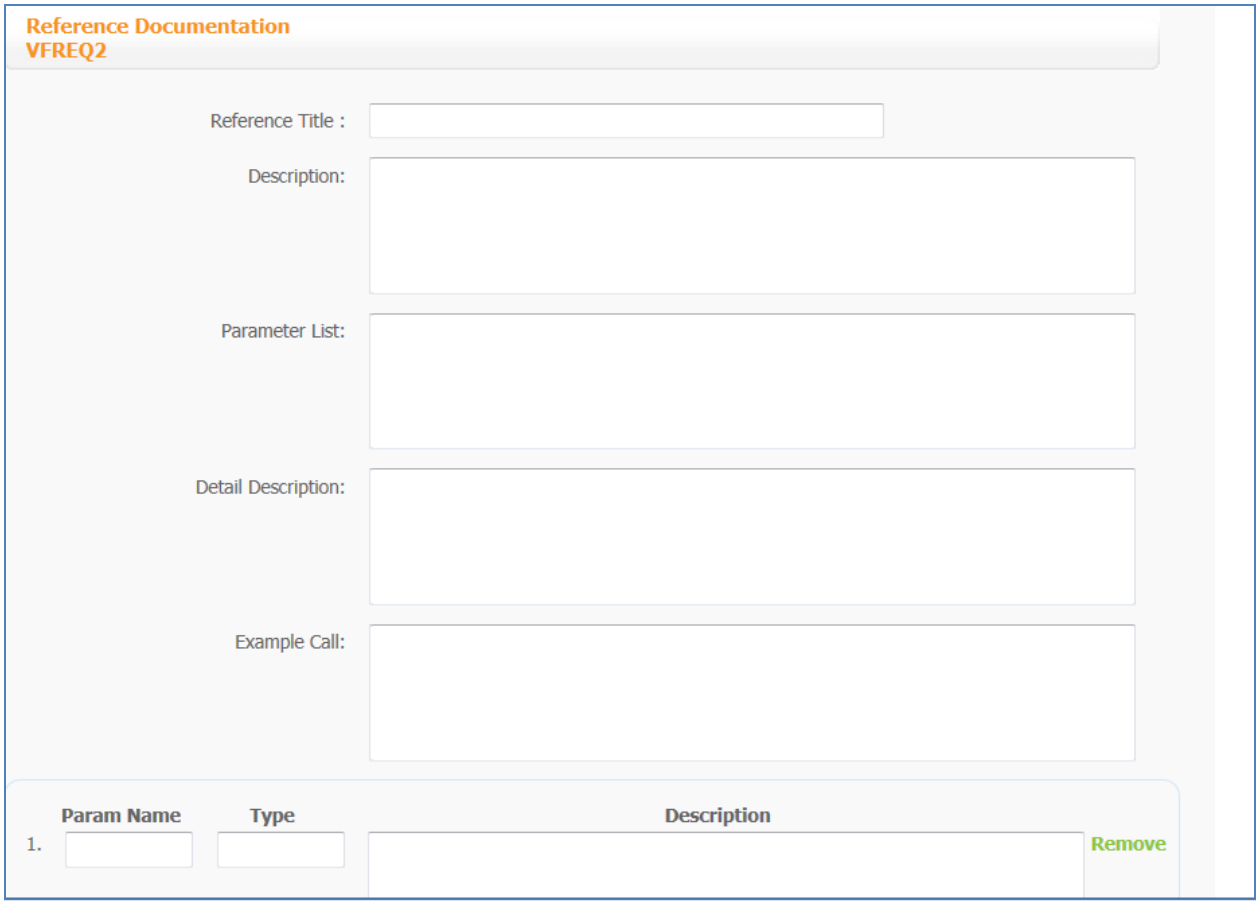
4. **Example Calls –** This provides example macro calls with sample parameters including syntax usage.

Examples

```
%vfreq (origdat = rawdata.conmed,
        compdat = analib.conmed);

%vfreq (origdat = rawdata.conmed,
        compdat = analib.conmed,
        outrep = ..\output\verification_check.html);
```

This documentation is essential for users since it functions like a user manual. I also find it useful to develop this documentation as a way of providing more perspective for a user. Because of this, I often write this documentation before starting to develop the macro since it allows a holistic view of the macro, before coding the first PROC.

The final documentation is delivered to the user in HTML, but the App Store will provide a form to fill out, so the user would enter this as text as shown below:



This will be accompanied with descriptive information such as what key words are to be used to search for the macro. All this information is used, making it easy for users to review and download the SAS macro.

STEP 2 – Displaying Navigation Macro Information

Before a user can start to find a macro, it is required to have a website where there is a user friendly method for navigation. The website developed for this project was based upon Amazon.com, since Amazon has many years of perfecting its e-commerce website. The Macro App Store thus opens with the "Bestseller" displaying the most popular macros. The other sections of the website include areas such as a page to manage user account or macros downloaded. Each section is available through the navigational menu at the top. The following is an example:

The development of the menus requires a combination of HTML and JavaScript. Similar to how a SAS program can include a macro from another location by using the %INCLUDE statement or adding this to a SASAUTOS path, the HTML code uses the following statement to "include" the JavaScript routine used for managing menus:

```
<script type="text/javascript" src="http://meta-x.com/appstore/menu.js"></script>
```

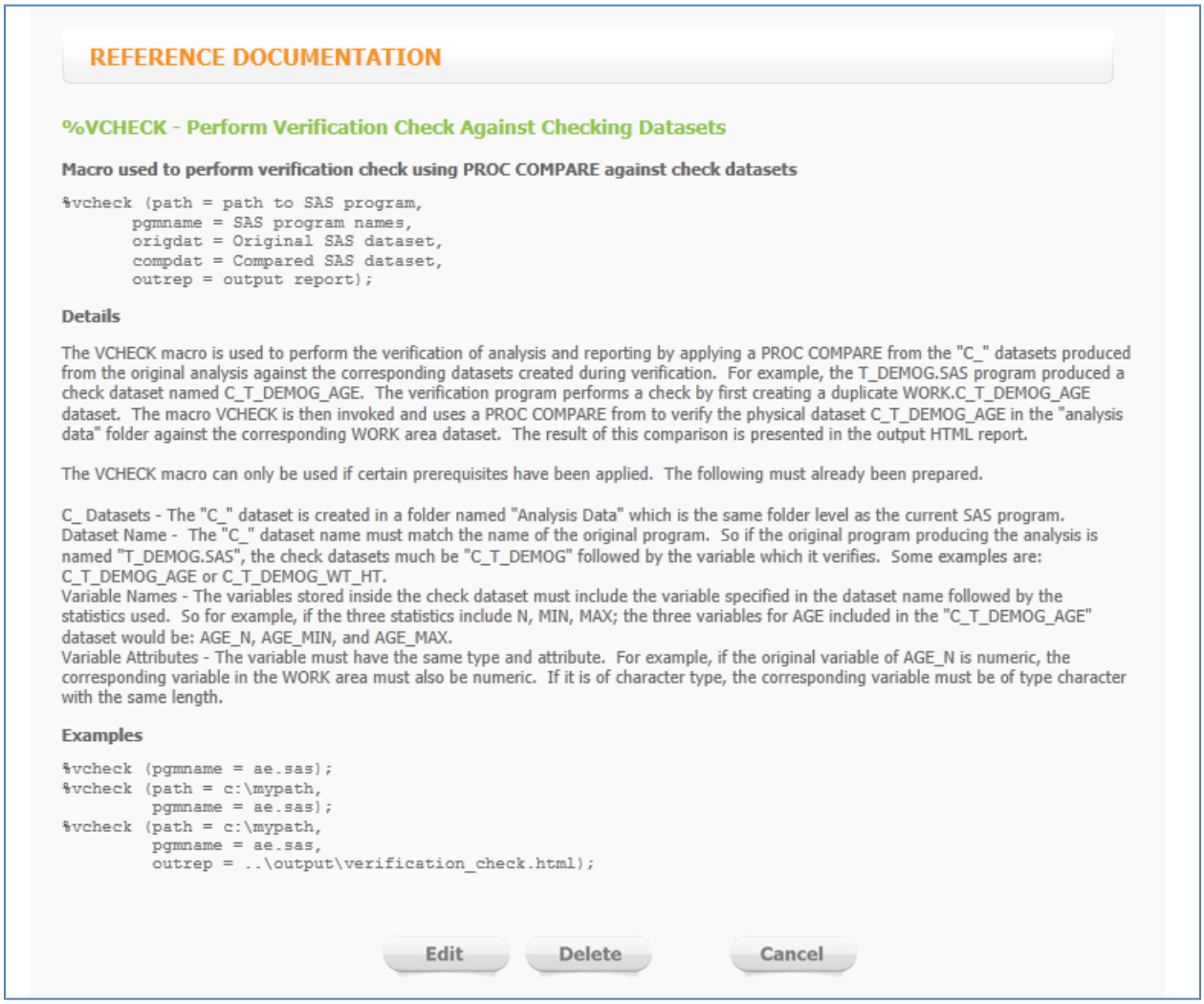
The logic of the MENU.JS script can be applied to the current HTML page so if the user places the mouse over a menu item and it will highlight in orange color and know where to link to. The example JavaScript code that performs this task is shown here:

```
// Menu JavaScript
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
x.src=a[i+2];}
}
```

The JavaScript logic loads the graphic images of the menu items which are of PNG format. This allows the user to experience the interactive menu as it swaps out menu images with different orange colors upon a mouse over event. This is accomplished by storing the images in an array statement in the same way that SAS uses array statements. The looping mechanism to traverse the array uses the syntax of "for i=" rather than SAS "do i=". Besides the minor difference in syntax, the main constructs and function of the program have many similarities between HTML, JavaScript and SAS.

STEP 3 – Displaying Macro Information

The macro information is displayed in HTML as described in step 1. In addition to standard HTML, CSS or Cascading Style Sheets is used to set the font and style such as in this example:



REFERENCE DOCUMENTATION

%VCHECK - Perform Verification Check Against Checking Datasets

Macro used to perform verification check using PROC COMPARE against check datasets

```
%vcheck (path = path to SAS program,
       pgmname = SAS program names,
       origdat = Original SAS dataset,
       compdat = Compared SAS dataset,
       outrep = output report);
```

Details

The VCHECK macro is used to perform the verification of analysis and reporting by applying a PROC COMPARE from the "C_" datasets produced from the original analysis against the corresponding datasets created during verification. For example, the T_DEMOG.SAS program produced a check dataset named C_T_DEMOG_AGE. The verification program performs a check by first creating a duplicate WORK.C_T_DEMOG_AGE dataset. The macro VCHECK is then invoked and uses a PROC COMPARE from to verify the physical dataset C_T_DEMOG_AGE in the "analysis data" folder against the corresponding WORK area dataset. The result of this comparison is presented in the output HTML report.

The VCHECK macro can only be used if certain prerequisites have been applied. The following must already been prepared.

C_ Datasets - The "C_" dataset is created in a folder named "Analysis Data" which is the same folder level as the current SAS program.
Dataset Name - The "C_" dataset name must match the name of the original program. So if the original program producing the analysis is named "T_DEMOG.SAS", the check datasets much be "C_T_DEMOG" followed by the variable which it verifies. Some examples are: C_T_DEMOG_AGE or C_T_DEMOG_WT_HT.
Variable Names - The variables stored inside the check dataset must include the variable specified in the dataset name followed by the statistics used. So for example, if the three statistics include N, MIN, MAX; the three variables for AGE included in the "C_T_DEMOG_AGE" dataset would be: AGE_N, AGE_MIN, and AGE_MAX.
Variable Attributes - The variable must have the same type and attribute. For example, if the original variable of AGE_N is numeric, the corresponding variable in the WORK area must also be numeric. If it is of character type, the corresponding variable must be of type character with the same length.

Examples

```
%vcheck (pgmname = ae.sas);
%vcheck (path = c:\mypath,
         pgmname = ae.sas);
%vcheck (path = c:\mypath,
         pgmname = ae.sas,
         outrep = ..\output\verification_check.html);
```

Edit Delete Cancel

In a static web page, you can link to this page with another HTML page such as:

```
<a href="http://vstatus.html">%VSTATUS Macro</a>
```

This hard codes the HTML location and the label of “%VSTATUS Macro”. In a more dynamic fashion such as in our “Bestsellers” App Store page, this same function can be accomplished using PHP. The link is modified to the following:

```
<a href=" index.php/yourApps/detail/43">%VCHECK Macro</a>
```

The hardcoded HTML page location is replaced with a PHP program named INDEX.PHP. The parameters to this program specify the detail information for the app with the ID number of 43. The PHP code to parse this information is shown here:

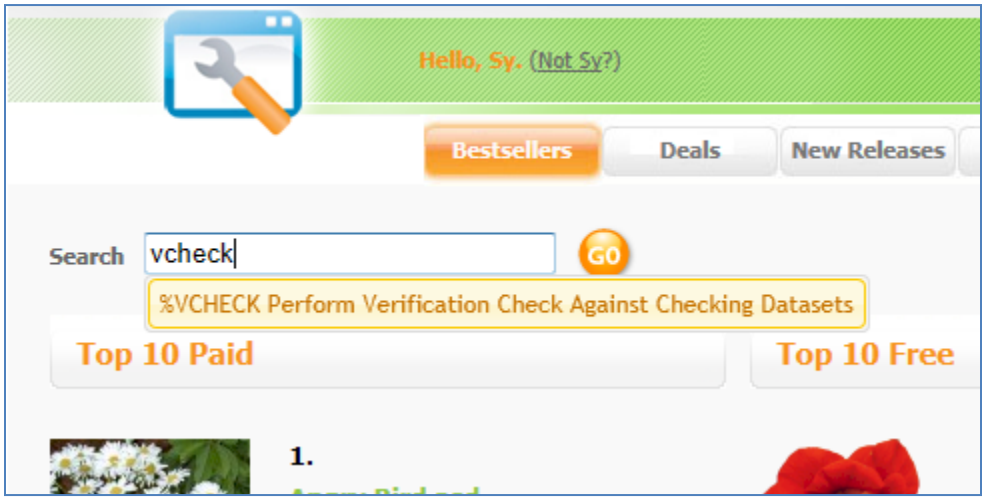
```
<?php
$this->load->view('header');
//-------------------------------------------
switch ($flagView) {
  case 'yourApps':
       $this->load->view('yourApps/yourApps');
        break;
```

```
                        case 'newApp':
            $this->load->view('yourApps/newApp');
             break;
       default:

}
//-----------------------------------------------
$this->load->view('footer');
?>
```

This logic is equivalent to the CASE statement in PROC SQL, or the use of a series of IF THEN ELSE. It allows for a more dynamic selection of macro within the App Store since new macros are added and deleted and thus requiring dynamic PHP code to handle this, rather than static HTML.

STEP 4 – Searching for Macro

A Google like search can be applied to locate a specific macro. As soon as you start to type a portion of the macro name on the search criteria entry, the "instant search" pops up a list of matching values.



When the macros are uploaded, keywords such as the name of the macro are recorded. The user is presented with the choice of clicking on the "GO" button to apply the search, or select the suggested pop-up to go directly to the macro. If the user were to click on the GO button, several parameters are passed to the search engine.

```
<form action="index.php/search" method="post" id="frm_search" name="frm_search">
   <div id="search_title">Search</div>
   <div id="search_box"><input name="Keyword" type="text" id="txtSearch"
value="vcheck"/></div>
   <div id="search_button"><input name="bnt_go" id="bnt_go" type="image" src="http://meta-
x.com/appstore/images/go2.png" /></div>
   <input name="action" type="hidden" value="search">
</form>
```

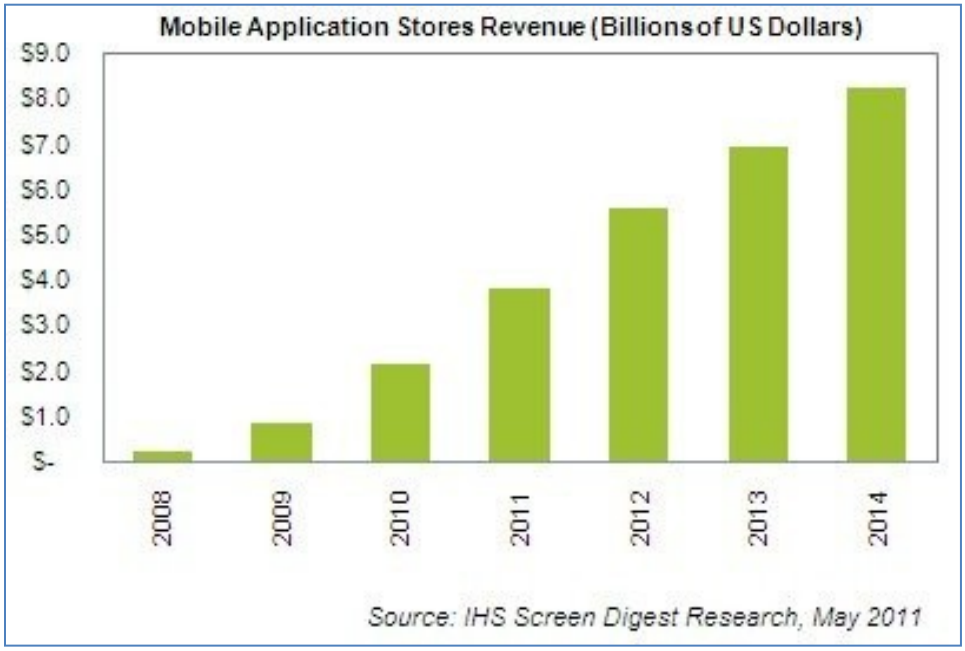
The value of the search criteria is passed to the same navigational INDEX.PHP program. However, the parameter TXTSEARCH delivers the values which will be used with the example keyword "vcheck" for the search algorithm. The following PHP logic is used in conjunction with SQL to apply this search.

```
//search for macro
function get_num_row_by_where_string($where_string)
{
   $sql = "SELECT AppIDNumber FROM apps ".$where_string;
   $query = $this->db->query($sql);
   return $query->num_rows();   }
```

The function call within PHP passes standard SQL to a mySQL database in this case. The SELECT statement is dynamically applied with the WHERE clause for `$where_string` which was captured from the `txtSearch` from the user. The SQL in this example is the same as PROC SQL with the only difference in that it is wrapped around a PHP function rather than a PROC SQL statement.

CONCLUSION

The growth of mobile App Store has gone through the roof in the past couple of years, and it is projected to continue on this meteoric rise. The success and popularity of the App Store is due to making it easy for users to find and download their Apps and have the ability to use it instantly. Many of the Apps are free so it draws users in. Once users see the value, they start to purchase paid apps thus resulting in tremendous revenue growth.



IHS shows how the growth was over 77% in 2011 to $3.8 billion and does not see the trend stopping. Users enjoy this instant gratification of finding, downloading and using Apps right away. However, in the world of SAS Macros, there is nothing instant about the process of obtaining a SAS Macro. This paper presents an approach of applying the methodologies of an App Store to how users will find and download SAS macros.

There are different types of software ranging from large sophisticated enterprise solutions on one end of the spectrum, and smaller “Apps” on the other. The Apps perform just a few things, but what it does, it accomplishes them well. This approach can also be applied to SAS macros in that they are small tools designed to perform specific tasks. In an era of App Stores, it is wise not to dismiss Macros or Apps because of their relative size or what they can do. In aggregate, the little Apps or Macros working together can deliver function that larger enterprise system will find difficult to match.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sy Truong
MetaXceed, Inc. (MXI)
42978 Osgood Rd
Fremont, CA 94539-5627
tel: 510.979.9333
fax: 510.440.8301
E-mail: sy.truong@meta-x.com
Web: www.meta-x.com

Zach Epstein “Major mobile app store revenue will grow 77.7% in 2011”, May 5th, 2011