CiviCore WidgetBuilder

CiviCore Team
Auren Daniel Pierce • David Wang • Scott Wiedemann
Client
Chic Naumer

COLORADO SCHOOL OF MINES

Faculty Advisors
Dr. Cyndi Rader • Dr. Roman Tankelevich

MACS 2009 Field Session
May 11 - June 19

TABLE OF CONTENTS

ABSTRACT

The goal of the CiviCore team during field session is to create a user friendly web interface called WidgetBuilder between a sample database and a set number of flash visualizations. It can be difficult to interpret and obtain statistics from large sets of raw data. Visualizing data is a common solution to this problem. Visuals for all sorts of data are found everywhere in your daily life, including the web. To transition from raw data to a visual representation can be tedious, time-consuming, and expensive. WidgetBuilder is the solution to these issues. The goal of WidgetBuilder is to empower users by providing them with a web interface to easily marry data in a database with a collection of various flash visualizations including charts, graphs and gauges. WidgetBuilder allows users to fully customize any visual of their choice to effectively astound their audience.

BACKGROUND & INTRODUCTION

CiviCore, LLC is a web-based development company that primarily creates web applications for the public sector and non-profit organizations. CiviCore provides and maintains many databases and several varieties of applications that help analyze data for their clients.

The goal of the CiviCore team during field session is to create a web interface between a sample database that contains information regarding Colorado public school information and a set number of flash visualizations that CiviCore has purchased called FusionCharts and FusionWidgits. Our team has created WidgetBuilder as this web interface.

Obtaining meaningful information from your data can be very valuable. The manner in which you represent and view data allows you to analyze and draw conclusions about important characteristics of a data set you may not have otherwise noticed. Visualizations like graphs, charts, and gauges can also allow others to easily recognize important aspects of the data you are trying to capture, not to mention that they are much more aesthetically pleasing than endless amounts of plain data in a database. However, it is often difficult to quickly visualize large sets of data clearly and accurately in a representation of your choice. WidgetBuilder resolves these issues by providing a simple to use web interface between a data source and visualizations. The goal of WidgetBuilder is to provide anyone with the power and tools to easily visualize their data with a collection of various visualizations.

REQUIREMENTS & TECHNOLOGY INFORMATION

A. Functional Requirements

This section lists and describes the functional requirements requested by CiviCore.

- User accounts for WidgetBuilder are created by a developer at CiviCore.
- WidgetBuilder requires the user's database type be Microsoft SQL. WidgetBuilder is able to connect to a msSQL database and provide a user with all the necessary information to use WidgetBuilder, which includes the table data and meta information for the tables and their fields.

- WidgetBuilder allows a user to easily view their data as plain text data.
- WidgetBuilder provides a simple to use web interface for the user to dynamically interact with. The web interface is flexible and allows for the customization of the different visualization schemes.
- WidgetBuilder supplies users with wizard guidance information. There are sample views of different visualizations for users to see. Sample visualizations and their corresponding data are provided in help sections in order to assist users on how to decide which choices are best for them.

B. Non-Functional Requirements

This section lists and describes the non-functional requirements requested by CiviCore.

- Microsoft SQL Server is the database server that WidgetBuilder interfaces with.
- FusionCharts and FusionWidgets are the flash visualizations that WidgetBuilder uses for the flash graphs, charts and gauges. CiviCore purchased and provided these tools.
- The programming work on the web server is primarily done in ColdFusion.
- FushionWidgets and FusionCharts take XML data as input to construct the visuals.
- JavaScript and AJAX techniques make the web interface faster and more efficient, intuitive for the user, and gives the interface a clean and professional look.
- JQuery is used to implement many of the functional designs and graphics in the web interface.

C. Technology Information

JQuery is a lightweight JavaScript library that emphasizes interaction between JavaScript and HTML. We use it for a few important visual effects and ease of javascript operations in the web interface such as the accordion style panels for the chart wizard. JQuery is under the General Public License and MIT (Massachusetts Institute of Technology) License both of which are free software licenses.

AJAX is used in WidgetBuilder to retrieve data asynchronously from the server in the background of the web interface without interfering with the user's display. The web interface is AJAX intense, which allows it to dynamically update and display information on the fly.

USER SCENARIOS & USE CASES

A. User Scenarios

i. New Client
Lisa Harp is a young activist and a political writer who is writing an article about the poverty in Colorado. The article will be featured in the Rocky Mountain News and will be available on the web as well. Lisa has a source of information that describes where the worst poverty in Colorado occurs. She would love to have a graph at rockymountainnews.com that shows the change in poverty level by area code for the state. She wants to allow users to select a range of years and have the graph show the change in poverty levels. She hears about our interface and

FusionWidgets and works with a web administrator at the rocky mountain news and a developer at CiviCore to connect to the database and set up the graph.

ii. Existing Client
Bob Smith is the system administration at Under Privileged Children. Under Privileged Children has a database with CiviCore that tracks mentoring, donations, and grants. The database also provides statistics and a summary every month. Bob wants to add graphs to the nonprofit's webpage to display this data for each month. Bob contacts Steve Washington, a developer at CiviCore for Under Privileged Children. Steve will setup the database with our tool and Bob will be able to make his graphs.

B. Use Cases

i. Log in
Primary Flow
Pre-condition: User's account and database with data of interest has been set up
Post-condition: User logs in

1. User goes to WidgetBuilder login page.
2. User enters his/her credentials (username, password) assigned by CiviCore.
3. User logs in.
4. Goto case ii.

Alternative Flow
Pre-condition: User's account and database with data of interest has been set up
Post-condition: User fails to log in

1. User goes to WidgetBuilder login page.
2. User enters his/her credentials (username, password) assigned by CiviCore.
3. User entered wrong credentials.
4. Error is displayed.
5. Goto case i.

ii. Workspace Management
Primary Flow
Pre-condition: User has completed case i
Post-condition: User creates a new workspace

1. The user chooses to create a new workspace and clicks a button 'New Workspace'.
2. User enters the name of the workspace and.
3. Goto case iii.

Alternative Flow
Pre-condition: User has completed case i
Post-condition: User edits an existing workspace

1. The user chooses to edit an existing workspace and clicks an edit button by the existing workspace.
2. Goto case v.

Alternative Flow
Pre-condition: User has completed case i
Post-condition: User deletes a workspace

1. The user chooses to delete an unneeded workspace and clicks the delete button by the workspace.
2. Goto case ii.

iii. Data Management
Primary Flow
Pre-condition: User has completed case ii
Post-condition: User selects data of interests and applies filters/conditions

1. User selects a table where the data of interest exists. (The CiviCore Developer has already associated her account with databases she has requested to have access to.)
2. The user selects the fields (tool will query the tables and list the data it finds)
    2a. User applies conditions to the field.
3. User selects data to graph in the fields.
    3a. Conditions can also be applied to this data as well.
4. Goto case iv.

iv. Chart/Graph Management
Primary Flow
Pre-condition: User has completed case iii
Post-condition: The user chooses a graph to represent the data

1. User can preview various basic graphs and view each graph in detail with the data that has been selected.
2. The user chooses the type graph, chart or gauge to represent the data.
3. The user chooses the specific visualization they want to use.
4. Goto case v.

v. iframe Management
Primary Flow
Pre-condition: User has completed case iv
Post-condition: The user is satisfied with the iframe.

1. At this point, the user can see the iframe and the graphs on it.
2. Goto case vi.

Alternative Flow
Pre-condition: User has completed case iv
Post-condition: The user chooses how the graph or graphs are placed in the iframe

1. The user can drag the graphs in the iframe and scale them how they please.
2. Goto case v.

Alternative Flow
Pre-condition: User has completed case iv
Post-condition: The user chooses adds a graph to the iframe

1. The user can press a button to add a graph to the iframe.

2. Goto case iii.

Alternative Flow
Pre-condition: User has completed case iv
Post-condition: The user chooses removes a graph from the iframe

1. The user can press a button to delete a particular graph from the iframe.
2. Goto case v.

vi. iframe Retrieval
Primary Flow
Pre-condition: User has completed case v
Post-condition: The user embeds the iframe on a web page

1. The tool generates HTML code. The link to the html is be contained within an iframe. The hyperlink for the iframe is given as plain text on this page.
2. The user copies the iframe link.
3. The user copies the hyperlink to the iframe that contains the HTML code generated by our tool.
4. The user can now embed the link anywhere he/she pleases.
5. Goto case vii.

vii. Logging Out
Primary Flow
Pre-condition: User has completed case i
Post-condition: The user logs out

1. User clicks on log out link and logs out.

DESIGN

A. Overview

Communication is an important part of WidgetBuilder even though the user may be unaware of it. WidgetBuilder provides users with a web interface that allows them to view their data in database and represent it with various visualizations. Someone using WidgetBuilder must be able to connect to a database, retrieve the data of interest, and represent the data in a manner that suits them. A user's work must also be stored for later use. All of the low level communication must be seamless, clean, reliable and unnoticed by the user.

WidgetBuilder is communicating with two Microsoft SQL databases. The first is the user's database. To establish a connection to a database all the user needs to do is log in to their account. Accounts for WidgetBuilder are created by a developer at CiviCore. The other database holds our proprietary interface information as well as user's workspaces. A workspace can be thought of as a user's session of previous work created in WidgetBuilder. The user can edit, delete, or create a new workspace.

Other forms of communication besides those needed by the databases also exist. The FushionWidgits and FusionCharts visualizations read XML data. WidgetBuilder constructs

precise XML based on the chart type and options the user may specify and feeds it directly into the flash files. The visualizations themselves are wrapped in an iframe to be embedded in different types of web pages.

Figure 1 shows a user's database on the bottom and the FusionWidgits and FushionCharts on the top. The components that are surrounded by black borders are part of WidgetBuilder. These components include the main web interface the user interacts with and a database with our proprietary interface information. The components outlined in red on Figure 1 are outside components. These are the FusionWidgits and FusionCharts and the database where the user's data exists.

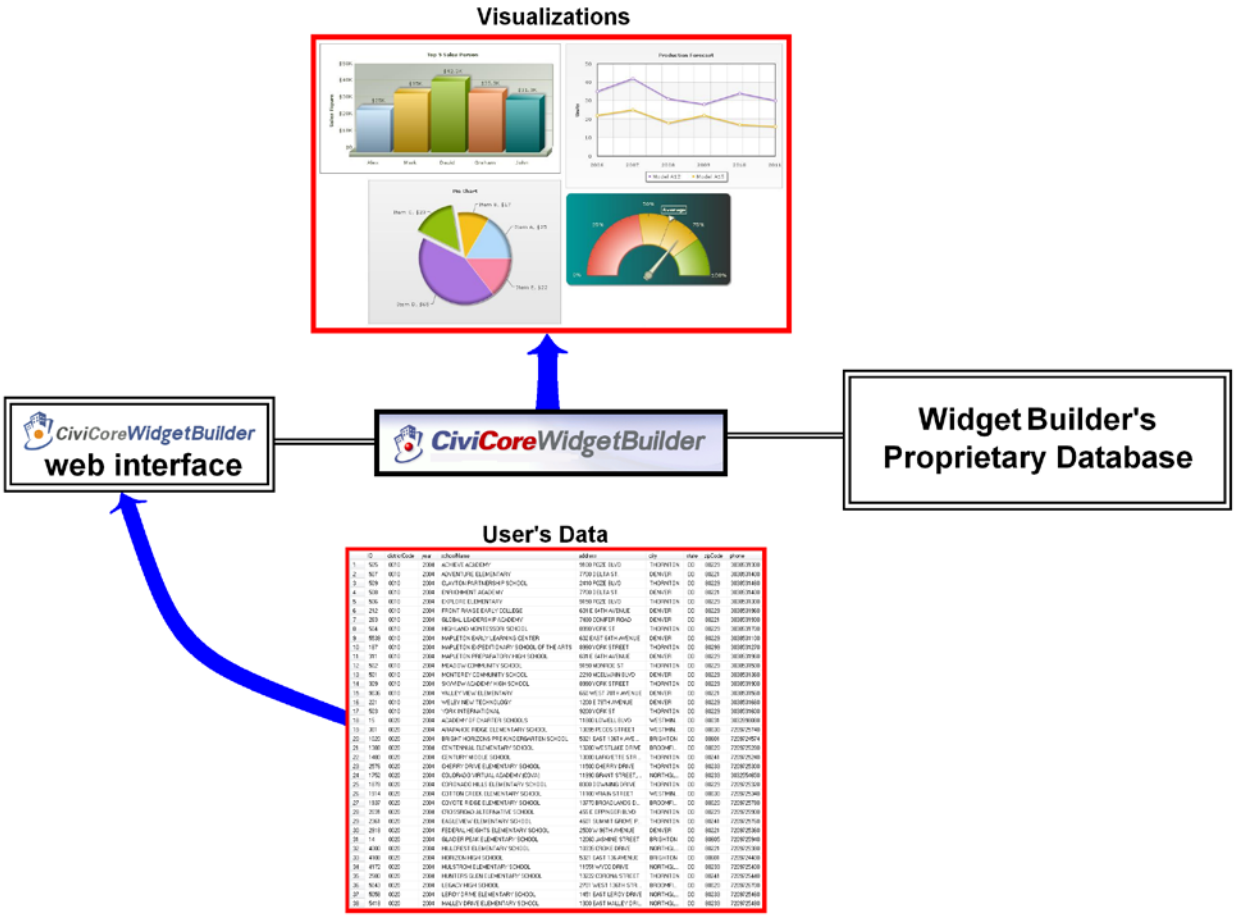

Figure 1: System Integration Overview

B. Detailed Description

i. Main Internal Components

This section describes the main components in our tool, the components shown in black borders in Figure 1. The main internal components are the web interface and the WidgetBuilder database.

Users interact with the front end of our database through the web interface. The web interface is the key component that manipulates and connects all the other components in Figure 1. A user logs into the interface with credentials provided by CiviCore. In their account they can create a workspace, edit an existing one, or delete a workspace. The user will now reach the main page of the application. From here a user can edit data used in his or her visuals, create new visuals, delete visuals, apply filters to data, change visual options and reposition the graphs as they appear in the iframe, as well and numerous other operations. Figure 2 in the additional details section shows a basic flow of the web interface.

The proprietary database holds a wide array of interface information. This information includes account credentials, a user's existing workspaces which encompass what visualizations are being used, the database and data of interest, the iframe hyperlink, and any options such as filters or positions. More details on the database scheme are given in the Additional Design Details section.

ii. Main External Components

This section describes the external components that WidgetBuilder is built around, namely the user's database and the visualizations. These components are shown in red borders in Figure 1.

The user's database is referenced from our database in order to obtain the information. The web interface references the data of interest and builds the visualizations from this data. WidgetBuilder requires the user's database type be Microsoft SQL. WidgetBuilder will graph any type of data that can be stored in a msSQL database, therefore the user needs not to worry about the type of data they want to visualize.

The visualizations are shown in red as outside components of WidgetBuilder, but the flash files are stored on the CiviCore development server. The visualizations are an important part of WidgetBuilder but no direct development on the flash files was done by our team. FushionWidgits and FusionCharts take XML data as input to construct the visuals. The xml is what WidgetBuilder constructs to pass to the flash components.

C. Additional Details

i. Web Interface Overview
Figure 2 shows the layout and design of WidgetBuilder web interface.

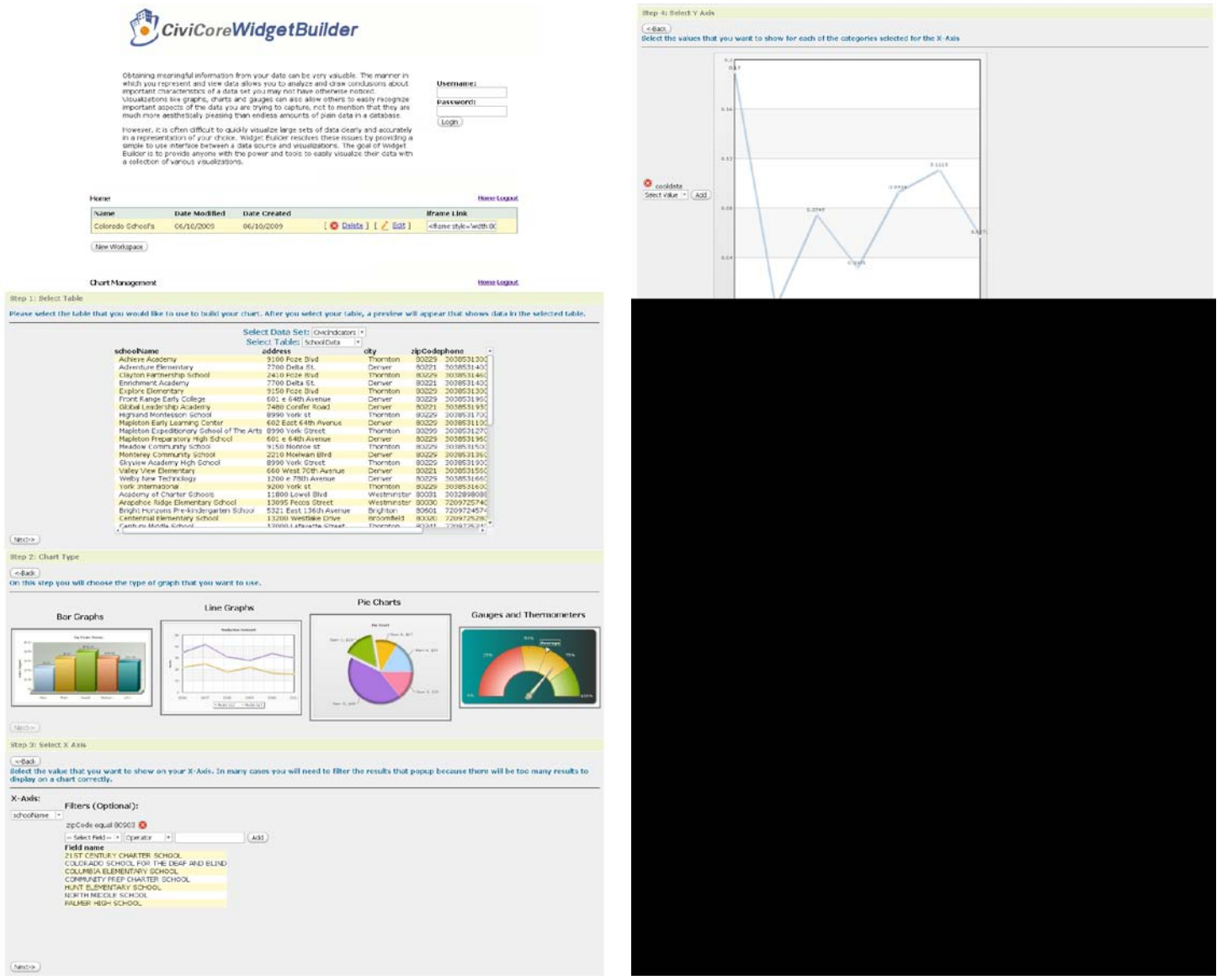

Figure 2: Web Interface Flow Design

ii. Database Schema

Figure 3 below shows an overview of the database schema and descriptions of the tables are given.

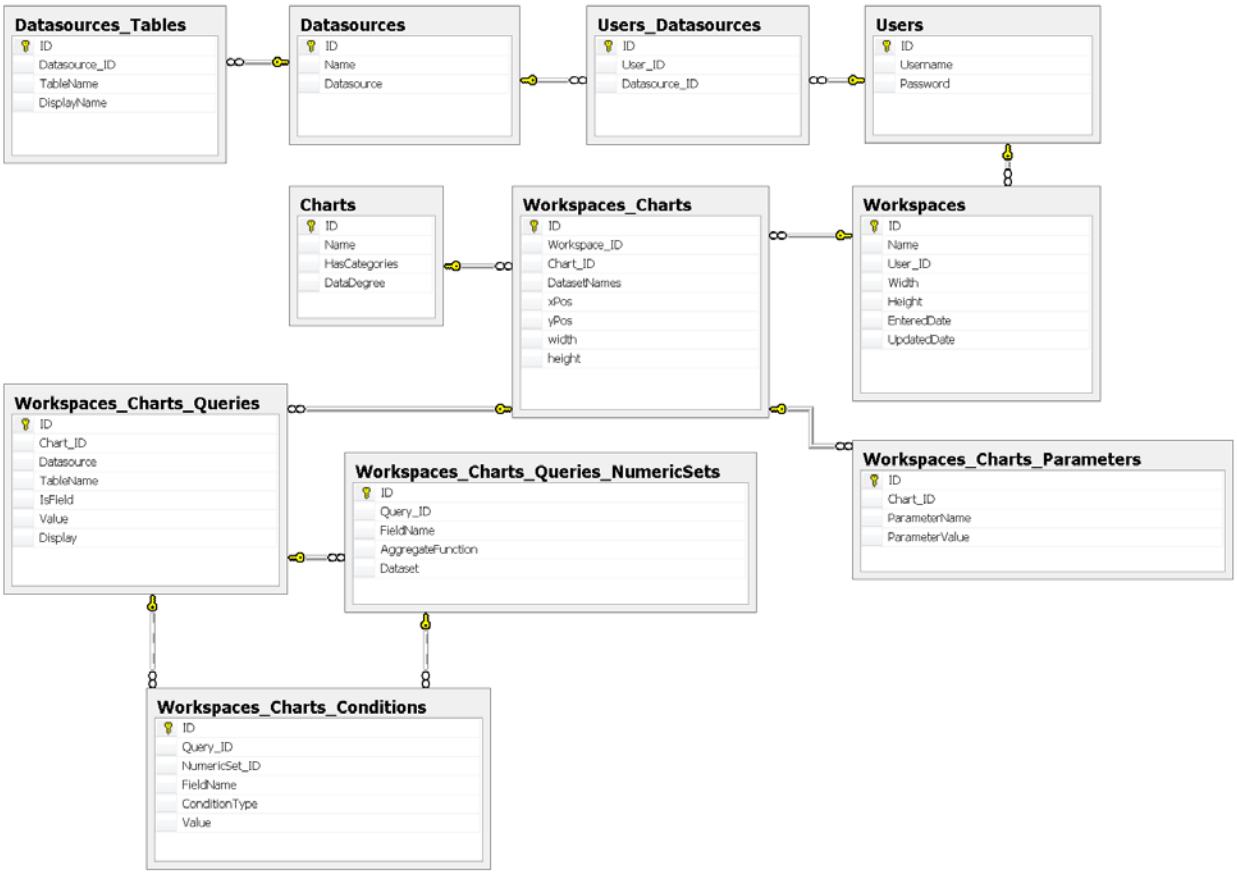

Figure 3: Database Schema

Table Descriptions:

- Datasources holds information about where a user's databases is located.
- Users contains all credential information for every user. (username, password)
- Users_Datasources is the join table for Users and Datasources.
- Workspaces contains information about a users previously done work.
- Charts is a static table that holds information about the chart types.
- Workspaces_Charts is the join table for Workspaces and Charts. A join table contains the ids of entries of two tables that provides communication between those tables. This join table holds information for the workspace related to the charts a user has defined.
- The Workspace_Charts_Parameters table holds custom options that the user has defined for their charts appearance.
- Workspaces_Charts_Queries and Workspaces_Charts_Queries_NumericSets hold the information that is passed to a SQL statement in order to retrieve the user's data of interest for a particular visual.
- Workspace_Charts_Conditions contains information about the filters a user has put on their data.

- Datasources_Tables is a table that holds view information for a user. Initial designs for WidgetBuilder included a settings file that would contain information for each user about how to display certain tables, columns and data in their database. This functionality has been implemented by using views. A view for a table would be set up cooperatively with the user and a developer at CiviCore. The view gives a cleaner look to the data in the database.

ISSUES & PROJECT PROGRESSION

Because CiviCore already had many of the necessary tools required to create WidgetBuilder on site, no issues that the team encountered delayed the project for more than 48 hours. The project depended heavily on CiviCore's servers because the sample user database, filesystem, FusionCharts and FusionWidgets were all stored there. The team only encountered a few issues dealing with server latency and downtime that interrupted work.

Below is a calendar that outlines the progression of the project. Forecasted work is shown in orange and the actual accomplished deadlines are shown in gray. All events, tasks and deliverables are shown as well.

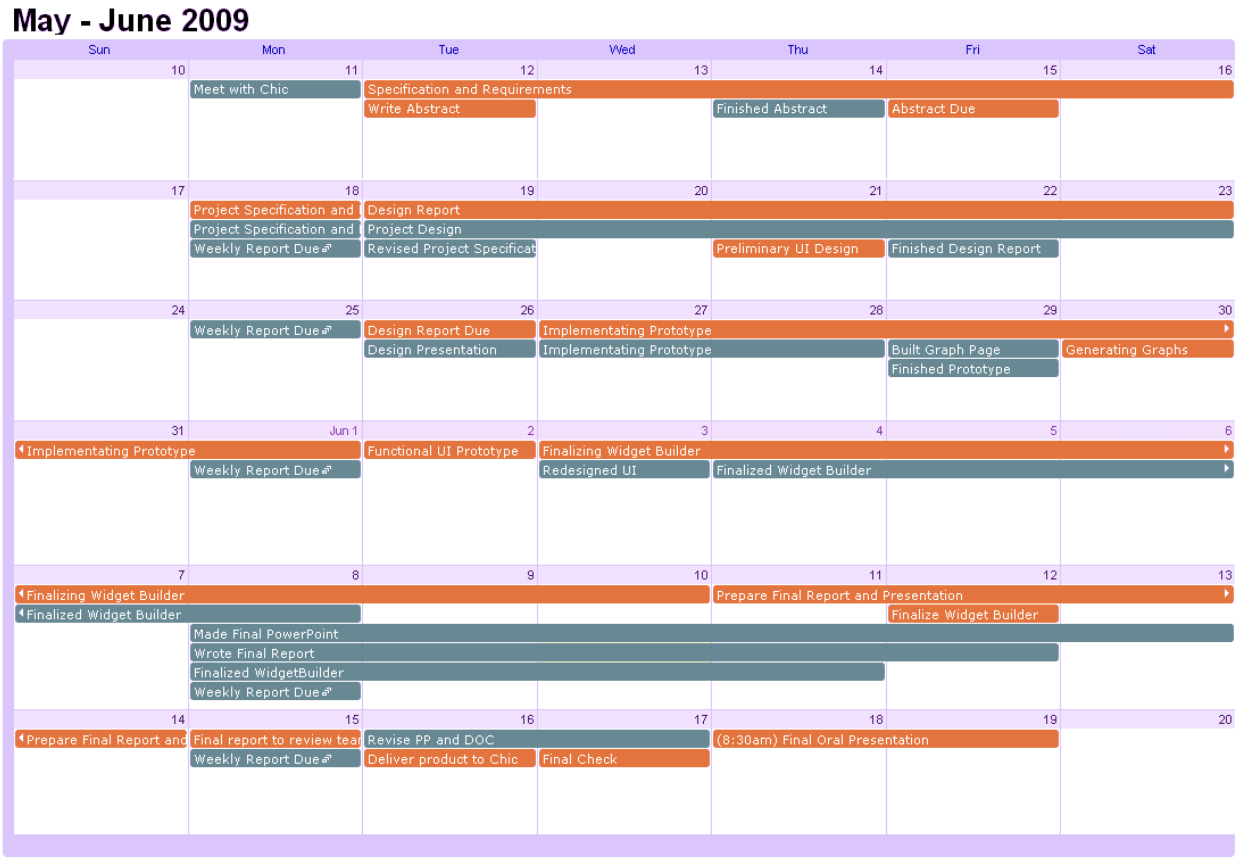

Figure 4: Calendar

FUTURE WORK

Further work on the application will allow it to connect to different formats of data, and not limit the format to a msSQL database. Integrating other database types and flat files such as Excel sheets or csv files would allow users more freedom in their data configuration.

Extended work would allow other developers outside of CiviCore access to a development version of WidgetBuilder called the WidgetBuilder Development Suite. Developers could modify and expand WidgetBuilder in order to deliver meaningful results to their own clients. This will allow other developers using the WidgetBuilder Development Suite to provide their own users with a more specific version of WidgetBuilder and fast way to view other data in numerous ways.

CONCLUSION

The CiviCore's team goal of creating a user friendly web interface between a sample database and a set number of flash visualizations was met in the six weeks allocated for the project. Many of the team's expectations for WidgetBuilder were exceeded, but some considered features for the project were not included. Overall our team is very pleased with WidgetBuilder.

GLOSSARY

AJAX (asynchronous JavaScript and XML), can retrieve data asynchronously from the server in the background without interfering with the display and behavior of the existing page.

ColdFusion is an application server and software language used for Internet application development such as for dynamically-generated web sites.

FusionWidgets is a collection of real-time flash gauges, self-updating charts and financial charts like Gantt charts, funnel & pyramid widgets, bullet graphs, sparklines etc. It is perfectly suited for use in dynamic web applications, executive dashboards and real-time monitors.

HTML, an acronym for HyperText Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document—by denoting certain text as links, headings, paragraphs, lists, etc.—and to supplement that text with interactive forms, embedded images, and other objects.

iframe is an inline frame places another HTML document in a frame inside a normal (rather than frameset) HTML document.

JavaScript is a scripting language used to enable programmatic access to objects within other applications. It is primarily used in the form of client-side JavaScript for the development of dynamic websites.

JQuery is a lightweight JavaScript library that emphasizes interaction between JavaScript and HTML.

Microsoft SQL Server is a relational model database server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL.

Views consist of a stored query accessible as a virtual table composed of the result set of a query.

XML (Extensible Markup Language) is a general-purpose specification for creating custom markup languages.