DATACUS:
Multi-Touch Data Explorer

Design
November 8, 2010

Ryan Daubert
Buck Heroux
Adam Jackson
Ian Smith
Matt Smith

CSCI 4308-4318. Software Engineering Project 1 & 2
Department of Computer Science
University of Colorado at Boulder
2010-2011

Sponsored By:
Nicholas Goodman

Dynamo Business Intelligence Corporation
Seattle, WA

PROJECT PROPOSAL

The goal of this project is to build a system using the latest hot technologies! Use multi-touch input devices and advanced analytic graphics and build the most advanced, most enjoyable data exploration application ever. Multi-touch, made popular by the iPhone and iPad, allow for the most natural interactions with applications ever conceived. Data exploration, as a natural task, explore, visualize and repeat is a natural fit and one that fits brilliantly with multi-touch.

Think of this project as building the cool interface from the movie "Minority Report" for exploring business data.

The project team will build:

- An interactive data explorer that will explore multi-dimensional data.
- Users will be able to graph, chart, and explore data with different multi-touch gestures.
- Target audience is business analysis users.
- Client will be written in Silverlight *or* Flash.
- Windows 7 multi-touch devices are a requirement, but other mouse/TUIO devices (large screens) are a design target.
- Data, including hierarchical data navigation, results, and querying available via web services (XMLA). In other words, this is a UI/query app, not a data crunching application.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

DynamoBI is start-up with eight employees spread between Seattle, San Francisco, and India. Their core product is an open source business intelligence (BI) database built for analytics, charting, and dashboards. DynamoBI's goal is to create purpose built tools that make business intelligence easy and enjoyable. They are interested in applying new UI technology to discover ground breaking improvements in the experience of data exploration through multi-touch display interfaces. DynamoBI intends to open-source the Datacus project and use it as a basis for potential future products. DynamoBi's core product and speciality is an open source BI database. This database's purpose is 100% about analytics, charts and dashboards. Placing the Datacus package in front of their core product makes for a perfect demonstration environment for the databases they provide.

A conceptual diagram provided by DynamoBI of the proposed Datacus system can be seen in Figure 1. The diagram shows Datacus accepting touch input from a touchscreen, which is interpreted by Windows 7 and the TUIO protocol. Datacus also sends and receives data to a Mondrian server, which then retrieves information from an OLAP database selected by the user. The touch events produced by the user manipulate the content viewed and the visual representation of the data in the user's database. By doing so Datacus visually presents the selected information and allows the user to explore data through the interface.

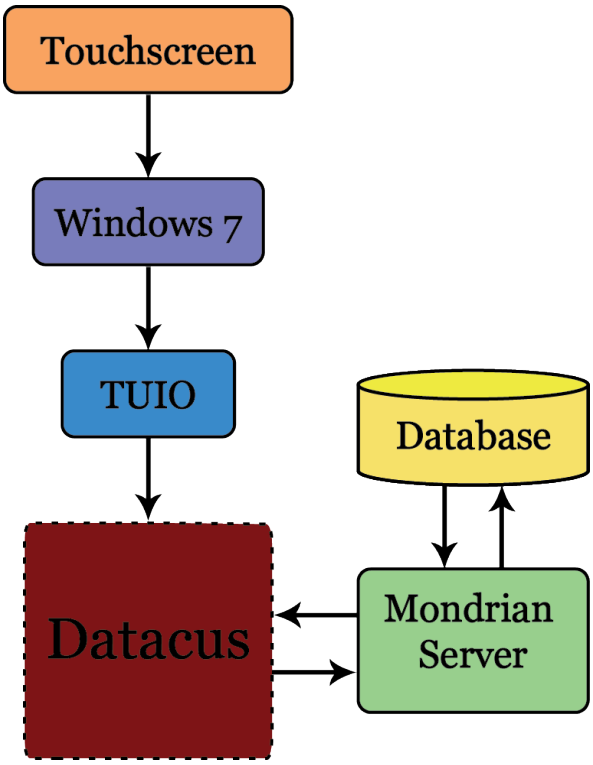

Figure 1: Datacus Conceptual Overview

2 USER INTERFACE

Datacus provides users with one graphical user interface. The user interacts with this interface by using touch gestures. The touch gestures can change the visual representation of the graph or the query data that the graph is generated from. Figure *1. Datacus GUI Window* represents the user interface viewing a graph and interacting with selected data. Figure *2. Datacus GUI Window With Expanded Picker* represents the Datacus UI selecting parameters for the graph.

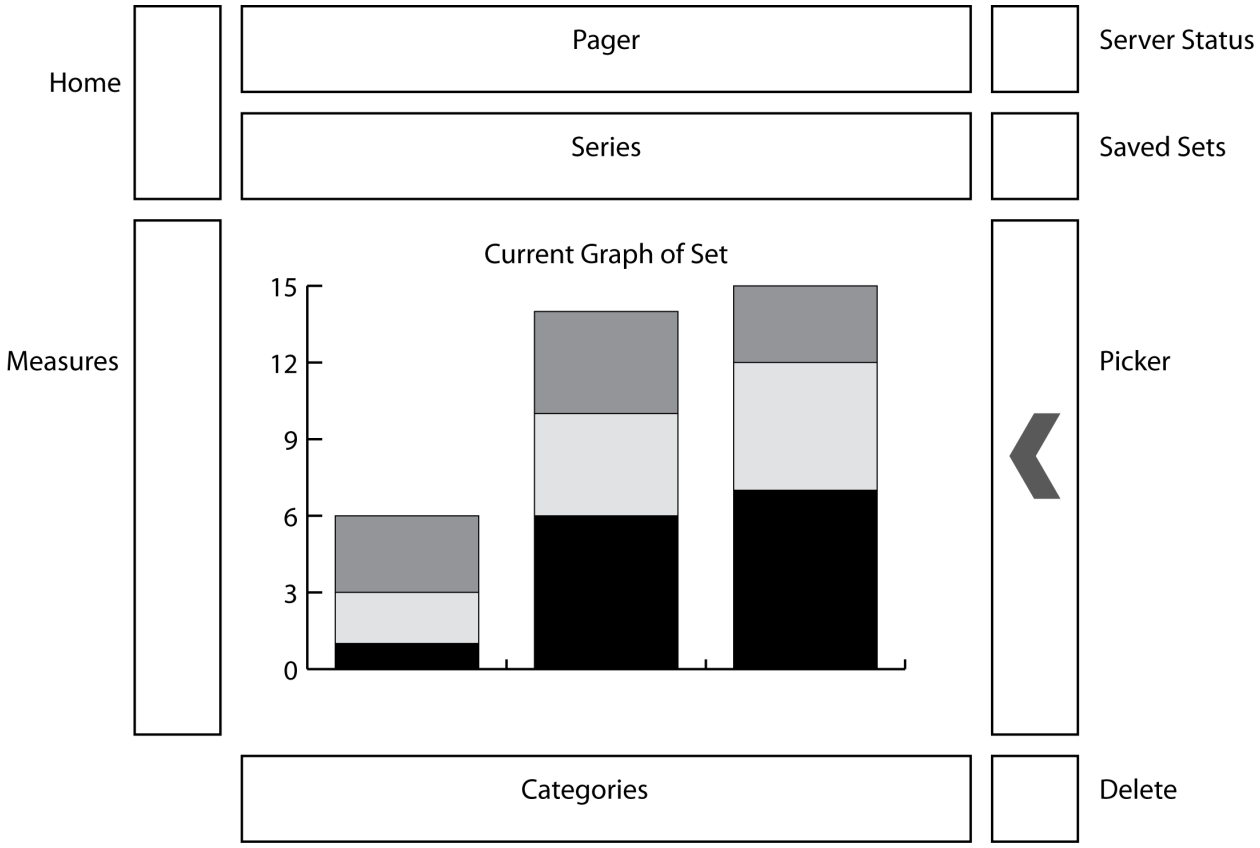

Figure 1. Datacus GUI Window

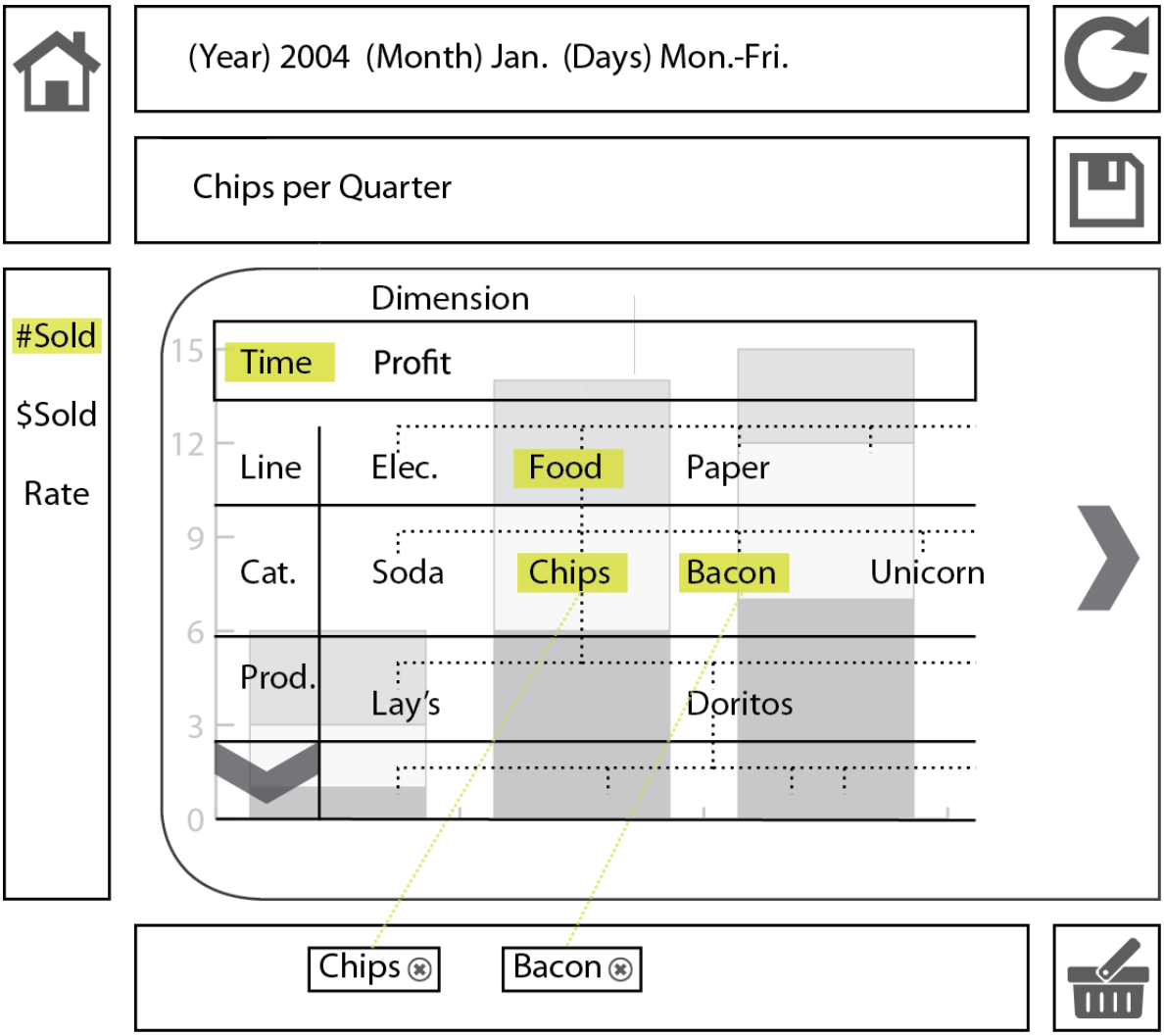

Figure 2: Datacus GUI Window With Expanded Toolbar

3 ARCHITECTURE

The high-level decomposition of Datacus consists of the *Mondrian Wrapper*, the *BlazeDS Wrapper,* the *Model*, the *Controller*, the *View*, and the *State Module*.

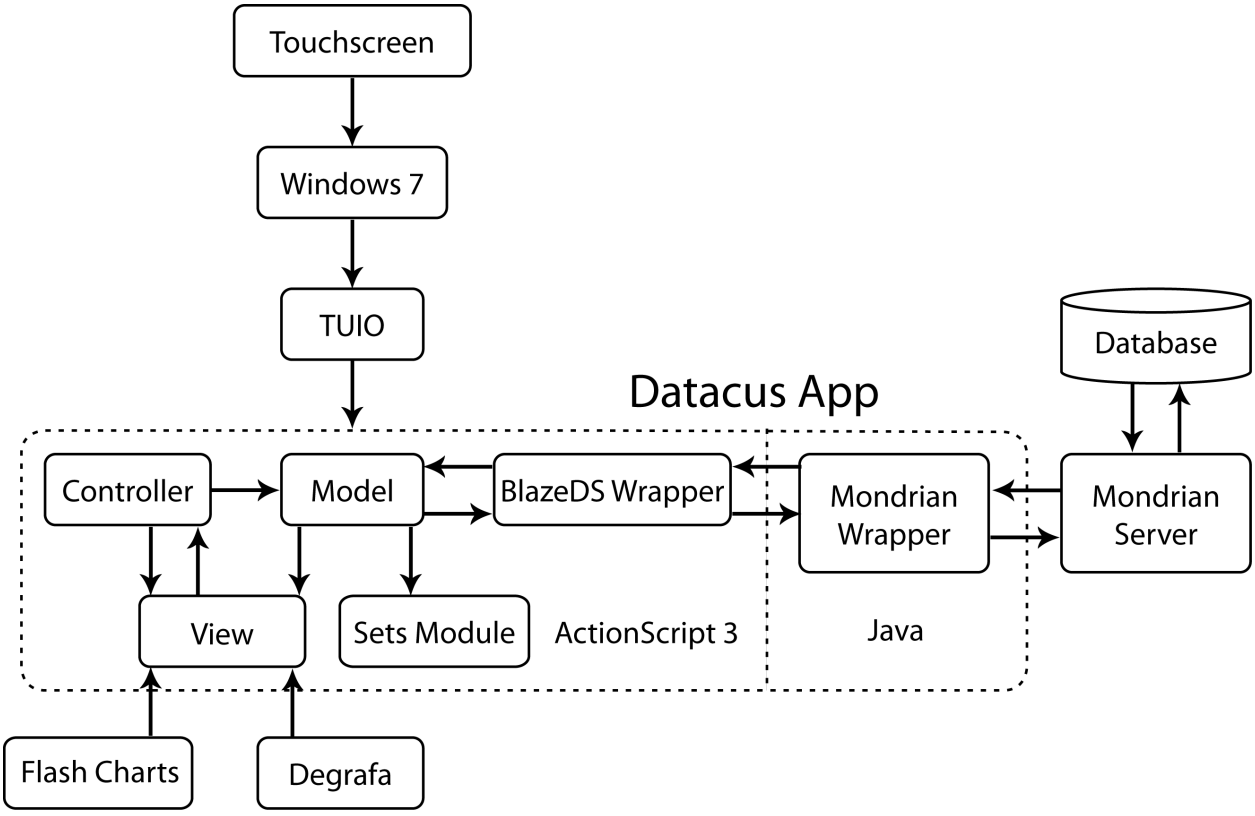

Figure 3. Conceptual Overview of Datacus Architecture

3.1 Mondrian Wrapper

The Mondrian Wrapper relays queries from the BlazeDS Wrapper to the Mondrian server and then translates response data from XMLA into Java objects. The Mondrian Wrapper contains the Java code for the BlazeDS processes and uses OLAP4J to relay data from the Mondrian Server to the BlazeDS Wrapper.

3.1.1 BlazeDS (Java)

Interacts with Java objects produced by OLAP4J and sends them to the BlazeDS Wrapper.

3.1.2 OLAP4J (Library)

Sends MDX queries to the Mondrian Server and saves the resulting OLAP responses as Java objects.

3.2 BlazeDS Wrapper

Relays queries from the Model Elements to the Mondrian Wrapper and translates response data from Java objects to ActionScript objects asynchronously. Contains the

ActionScript components of BlazeDS, which allows Datacus to request data from the Mondrian Server via an RPC call without having to pause for a response.

3.2.1 *BlazeDS (ActionScript)*

Transfers data from the Mondrian Wrapper by making remote calls to the Java component of BlazeDS.

3.2.2 *Server Status*

Keeps track of the interaction of the user's database and the Datacus software via the BlazeDS Wrapper. An icon in the User Interface visually represents the Server Status. The Server Status is a Flash animation that gets updated every time an interaction happens between the BlazeDS Wrapper and Mondrian Wrapper.

3.3 Model

The Model Element manages data and notifies observers when the data has changed. The Model Element contains the View Observer, the BlazeDS Observer, and the State Observer.

3.3.1 *View Observer*

The view observer changes the view any time the active data is changed.

3.3.2 *BlazeDS Observer*

The BlazeDS observer changes the view any time the active data is changed.

3.3.3 *State Observer*

The state observer notifies the State Module whenever the model is updated.

3.3.4 *Fields*

The Model Elements will have the following fields:

- *Data*
  - All of the current active data.
- *State*
  - The selected categories of active data.
- *Labels*
  - The selectable categories dynamically retrieved from the database.

3.4 Controller

The main purpose of the Controller is to handle the user's touch gestures. Based on the input gesture, the Controller will tell the Model whether to update or not. It will also instruct the View to change depending on the user's gesture.

3.5 View

The View Element contains the Graph, the Graphing Libraries, and the User Interface. As the system receives input, the View Element notifies the Controller Element of the object selected by the user. As the View Element receives input from the Model and Controller Elements it updates the Graph accordingly.

3.5.1 Graph

The resulting graphical object from the selected input data when data is updated in the Model Element.

3.5.2 Graphing Libraries

Flash Charts is the primary graphing library used for producing charts and graphs within Datacus.

3.5.3 User Interface

The User Interface Elements are contained within the View Element. These elements include the Picker, all Selection Bars, and the Visualization of Server Interaction.

- *Picker*
  - Visual element that allows the user to select parameters to graph.
- *Selection Bars*
  - Visual elements that manipulate the model by user interaction.
- *Visualization of Server Interaction*
  - Indication of Datacus' interaction with the user's database.

3.6 State Module

The State Module interacts directly with the Model Element for the purpose of storing the current state. The State Module writes both Saved States and a State Cache to an External Save File (.xsf).

3.6.1 Saved States

The current state of the Model Element can be explicitly saved in order to retrieve it for later use. The current Series, Categories, and graph preferences are saved in each Saved State.

3.6.2 State Cache

The State Cache implementation involves saving each state of the Model Element as it is edited to an External Save File, but tagged separately from Saved States. This allows users to read from the file and Undo any previous actions preformed on the Model Element. There will be a Trash Can represented on the UI for users to "delete" items within the UI environment that will be saved to the file.

4 Initialization

The steps required for synchronous execution at every startup of Datacus consist of:

4.1 Touch Initialization

Datacus starts a TUIO Manager to monitor touch events. Touch events are handled similarly to mouse click events.

4.2 UI Initialization

The Degrafa graphics library provides visualizations used within the Datacus interface and initializes during the startup of Datacus.

4.3 Database Selection

A screen appears to select which database to connect to from the External Save File. If the application is being run for the first time, Datacus will ask the user to setup an initial connection to their OLAP database.

4.4 Module Initialization

Datacus initializes each module respective to the database selected from External Save File.

5 EXTERNAL SAVE FILE

The External Save File (.xsf) is a plain text file that represents all of the local work done in Datacus in the following JSON format. A leading # represents variable information.

```
{ ‘database’ : ‘#database_location’ ,
        ‘saved_sets’ :[ { ‘#state’ },{ ‘#state’ } ],
        ‘trash’ : [ { ‘#state’ },{ ‘#state’ }]
}
{ ‘database’ : ‘#database_location’ ,
        ‘saved_sets’ :[ { ‘#state’ },{ ‘#state’ } ],
        ‘trash’ : [ { ‘#state’ },{ ‘#state’ } ]
}
```

The External Save File will be opened at the initialization of Datacus and written to whenever a new database is connected to, a set is saved or when set is put into the trash. This should effectively replicate the environment of Datacus that was last used with respect to each database.

5.1 Database

The database value is the location of a database that has been previously connected to.

5.2 Saved Sets

This value is an array of model states.

5.3 Trash

This value is an array of previous model states that can be used to undo the current state.

6 SUMMARY

7 REFERENCES

There are a few documents related to this paper that are useful for further reading on certain topics.

[BlazeDS 10]

"BlazeDS." Wikipedia. June 7, 2010. Wikimedia Foundation, Inc. (October, 2010) <http://en.wikipedia.org/wiki/BlazeDS>

[Degrafa 08]

"Degrafa : Declarative Graphics Framework." 2008. Degrafa Team. (October, 2010) <http://www.degrafa.org/>

[Flash App 10]

"Flash Application Design: Understanding Flash applications." adobe.com. Adobe Systems, Inc., 2010. (October, 2010) <http://www.adobe.com/support/flash/applications/app_design/app_design04.html>

[Flash Server 10]

"Flash Application Design: Using an application server." adobe.com. Adobe Systems, Inc., 2010. (October, 2010) <http://www.adobe.com/support/flash/applications/app_design/app_design06.html>

[Flex 10]

"Source of DynamicChart." 2010. Flex Blog. (October, 2010) <http://www.flex-blog.com/samples/sample13/srcview/index.html>

[olap4j 10]

"olap4j: Open Java API for OLAP." 2007-2010. olap4j Team. (October, 2010) <http://www.olap4j.org/>

[Tuio 10]

ìTuio as3 library.î 2010. Tuio Flash Blog. (October, 2010) <http://bubblebird.at/tuioflash/tuio-as3-library/>