**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**

1 INTRODUCTION 5

2 USER INTERFACE 7

3 ARCHITECTURE 9

3.1 Mondrian Wrapper 9

*3.1.1* *BlazeDS (Java)* 9

*3.1.2* *OLAP4J (Library)* 9

3.2 BlazeDS Wrapper 10

*3.2.1* *BlazeDS (ActionScript)* 10

*3.2.2* *Server Status* 10

3.3 Model 10

*3.3.1* *View Observer* 10

*3.3.2* *BlazeDS Observer* 10

*3.3.3* *State Observer* 10

*3.3.4* *Fields* 10

3.4 Controller 10

*3.4.1* *Model Controller* 10

*3.4.2* *View Controller* 11

3.5 View 11

*3.5.1* *Graph* 11

*3.5.2* *Graphing Libraries* 11

*3.5.3* *User Interface* 11

3.6 State Module 12

*3.6.1* *Saved States* 12

*3.6.2* *State Cache* 12

4 Initialization 13

4.1 Touch Initialization 13

4.2 UI Initialization 13

4.3 Database Selection 13

4.4 Module Initialization 13

5 EXTERNAL SAVE FILE 14

5.1 Database 14

5.2 Saved Sets 14

5.3 Trash 14

6 SUMMARY 15

7 REFERENCES 16

**TABLE OF FIGURES**

Figure 1. Datacus Conceptual Overview 5

Figure 2. Datacus GUI Window 6

Figure 3. Datacus GUI Window With Expanded Picker 7

Figure 4. Conceptual Overview of Datacus Architecture 8

# 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. Datacus Conceptual Overview. 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.

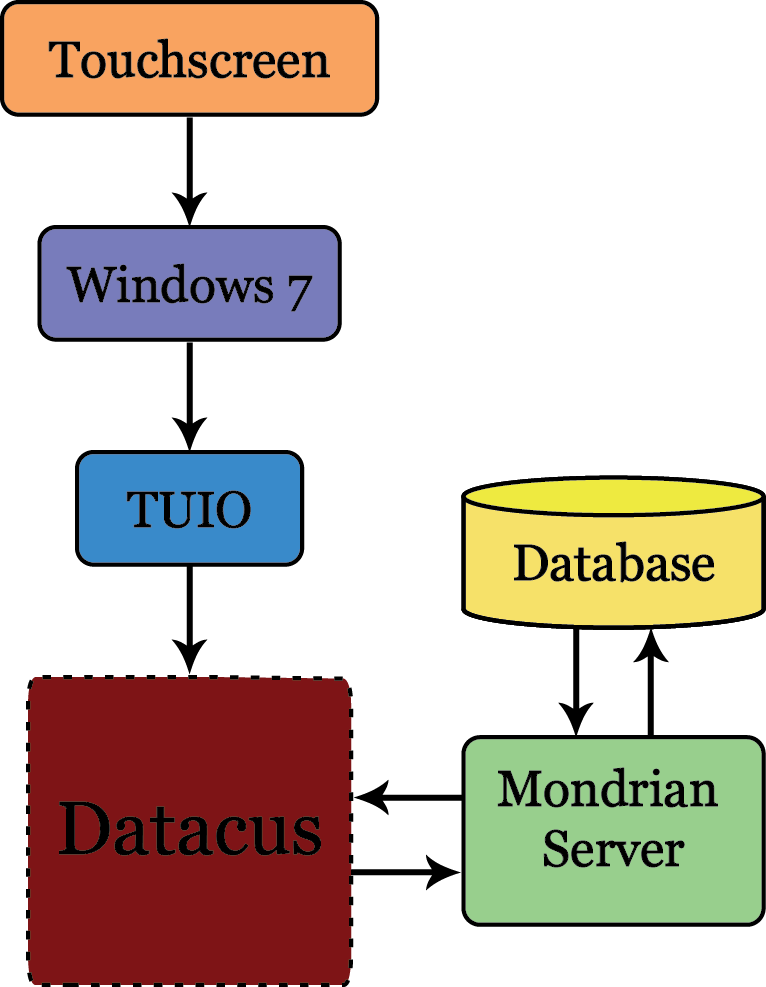


Figure 1. Datacus Conceptual Overview

This document defines the current Datacus design, which is intended to fulfill the requirements found in Datacus Requirements. This design will be updated as it evolves over the course of the project.

There are four primary aspects to the design of Datacus:

* User interface design
* Software architecture
* Initialization process
* External save files

Each of these design aspects is described below followed by a list of references.

# 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 2. Datacus GUI Window represents the user interface viewing a graph and interacting with selected data. Figure 3. Datacus GUI Window With Expanded Picker represents the Datacus UI selecting parameters for the graph.

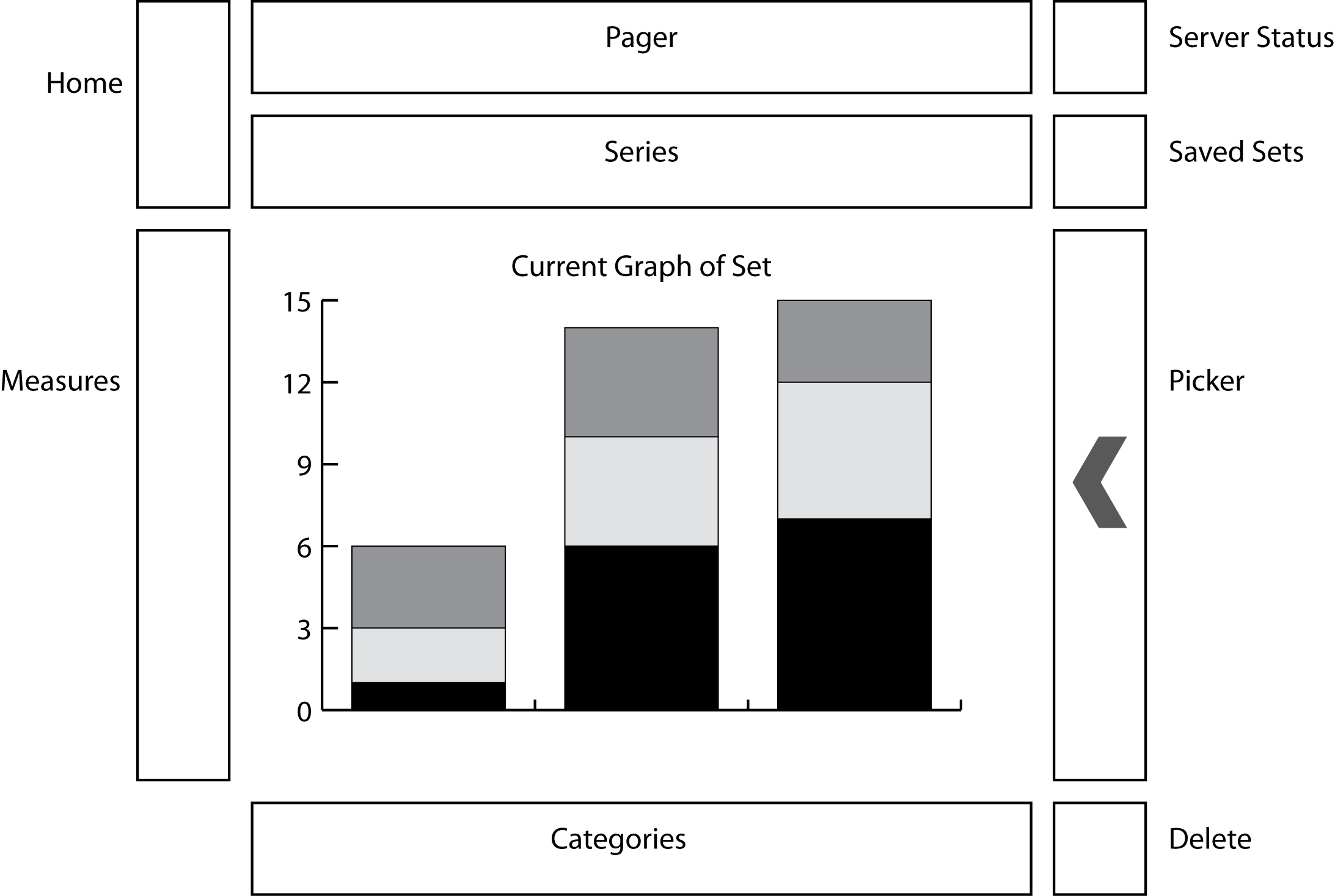


Figure 2. Datacus GUI Window

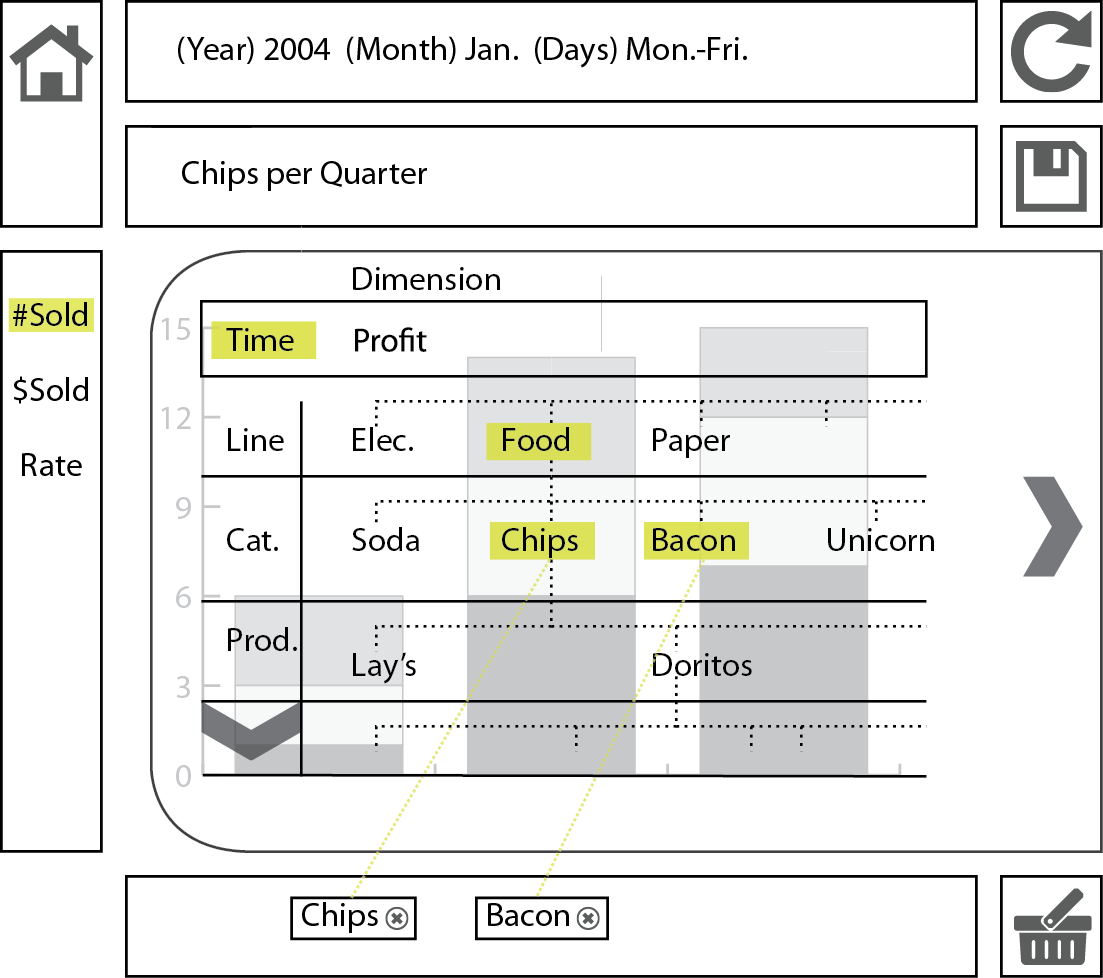


Figure 3. Datacus GUI Window With Expanded Picker

# 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* as shown in Figure 4. Conceptual Overview of Datacus Architecture.

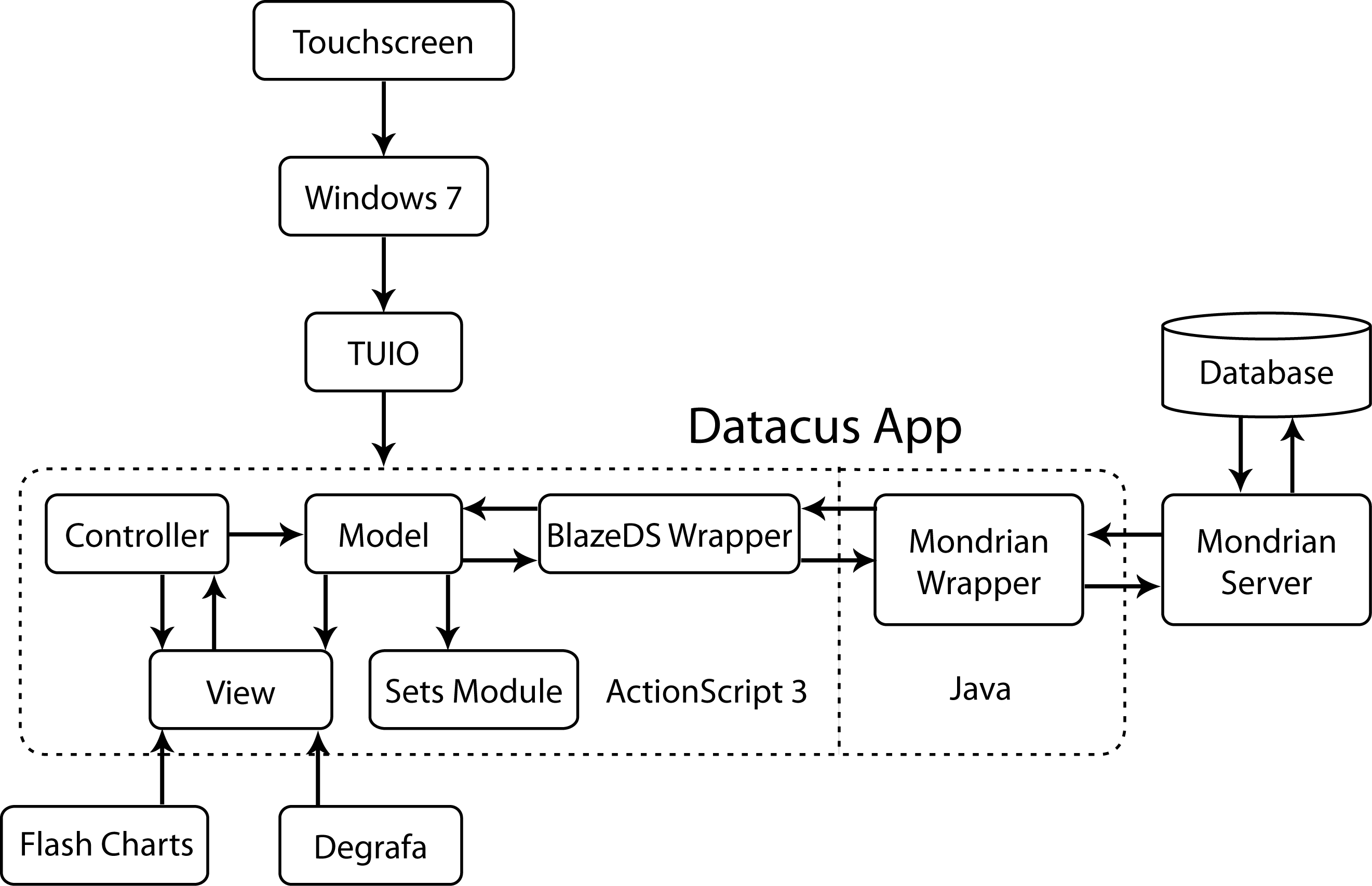


Figure 4. Conceptual Overview of Datacus Architecture

## 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*.

### *BlazeDS (Java)*

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

### *OLAP4J (Library)*

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

## BlazeDS Wrapper

Relays queries from the *Model* 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.

### *BlazeDS (ActionScript)*

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

### *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*.

## Model

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

### *View Observer*

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

### *BlazeDS Observer*

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

### *State Observer*

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

### *Fields*

The Model 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.

## Controller

The main purpose of the *Controller* is to handle the user’s touch gestures. It receives input and initiates a response by making calls on *Model* objects.

### *Model Controller*

Based on the input gesture, the *Controller* will tell the *Model* whether to update or not.

### *View Controller*

The *Controller* will instruct the *View* to change depending on the user’s touch gesture.

## View

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

### *Graph*

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

### *Graphing Libraries*

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

### *User Interface*

The User Interface Elements are contained within the *View*. 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.

## State Module

The *State Module* interacts directly with the *Model* 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).

### *Saved States*

The current state of the *Model* 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.

### *State Cache*

The State Cache implementation involves saving each state of the *Model* 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*. 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.

# Initialization

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

## Touch Initialization

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

## UI Initialization

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

## 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.

## Module Initialization

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

# 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.

## Database

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

## Saved Sets

This value is an array of *Model* states.

## Trash

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

# SUMMARY

This document describes the design plan for the Datacus software package. This includes design related to the User Interface, the Architecture, the Initialization process, and the External Save File to be used for saving states and holding previously deleted information in which the user can undo commands from. This document is expected to evolve as the project progresses and the Design needs to reflect the state of the project.

# 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/>