Architecture Design

For MVC Reporting Website Framework

(MVC RWF)

Version 1.0

Submitted in partial fulfillment of the Masters of Software Engineering Degree.

Thaddeus Thomas Tuck

CIS 895 – MSE Project

Department of Computing and Information Sciences

Kansas State University

Committee Members

Dr. Daniel Andresen

Dr. Mitchell Neilsen

Dr. William Hsu

Change Log

Version #	Changed By	Release Date	Change Description
1.0	Thaddeus Tuck	5/12/2018	Initial Release
1.1	Thaddeus Tuck	6/11/2018	Updates of Overall Class Diagram and Formal Specification

List of Figures

Table of Contents

1. Introduction

The purpose of this document is to provide the architectural design of the MVC Reporting Framework project. This documentation presents the class, sequence diagrams, and the formal specification of the MVC Reporting Framework.

1.1 Background

The purpose of this framework is developed to fulfill the need for a development team of varying web-development experience to be able to implement, deploy, and maintain an Enterprise level web project while respecting best-practices and standardized implementations for features.

1.2 References

[1] Tuck, T. Vision Plan 2.0 Retrieved 05/12/2018, from Thaddeus Tuck's Project Page Web Site: http://thaddeustuckmastersproject.azurewebsites.net

[2] Naga Sowjanya Karumuri, Vision Plan 2.1 Retrieved 05/12/2018, from Sowjanya's Project Page Web Site: http://people.cis.ksu.edu/~sowji/100jiMSE/

[3] Hill, K. System Architecture Design 1.0 Retrieved 05/12/2018, from Kyle Hill's Project Page Web Site: http://mse.cs.ksu.edu/hill/

2. MVC Reporting Website Framework

To estimate the effort, cost, and schedule for this project the best method I have found so far during the creation of this documentation is the COCOMO II model.

2.1 Class Diagram for Base Framework Classes

Based on the Vision Document the framework uses a View, Controller, Service, and Repository architectural pattern. The architecture is an extensible pattern that builds standardized view components from standard extensible models. The class diagram below shows a partial view for the base class interaction of the framework excluding the models. However, it should be noted that since most of benefit of using this framework is the automated view element generation the model and controller setups are the base requirement.

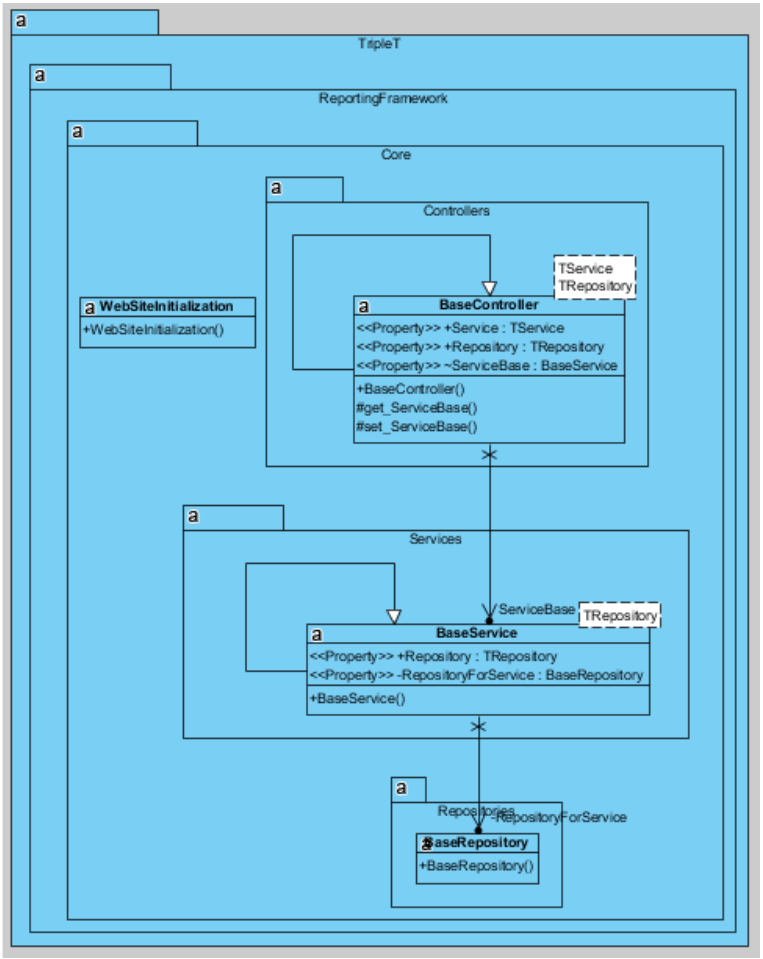

Figure 1: Base Interacting Classes for the Framework

2.2 Overall Project Class Diagram

The architecture is an extensible pattern that builds standardized view components from standard extensible models. The class diagram provides the overall view of the framework. The central component of the framework for table-based reporting is the model architecture shown in the center of the following diagram. A separate image will be available on the project website for easier viewing.

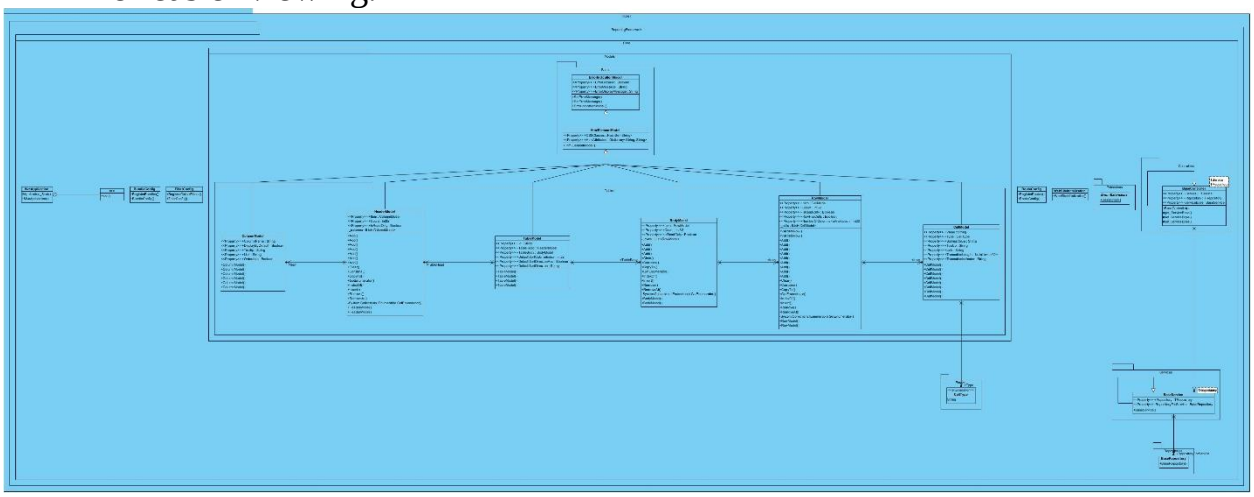

Figure 2:Overall Class Diagram for the Framework

3. Formal Specification

```
--------------------------------------------------------------------------------
-- MVC Reporting Website Framework Architecture
--
-- This file contains a formal specification of the table models and base structural classes.
--
-- File:    MVCReportingWebsiteFrameworkArchitecture.use
-- Author:  Thaddeus Tuck
-- Date:    June 10, 2018
--------------------------------------------------------------------------------

model MVCReportingWebsiteFrameworkArchitecture

--------------------------------------------------------------------------------
-- Base Model Classes
--------------------------------------------------------------------------------
class ErrorIndicationModel
attributes
        ErrorOccured : Boolean
        ErrorMessage : String
        ErrorDisplayMessage : String
operations
        SetErrorMessage(s : String, ds : String)
end

class HtmlAttributes
```

```
attributes
   key : String
   value : String
end

class HtmlElementModel < ErrorIndicationModel
attributes
        CSSClasses : Set(String)
        HtmlAttributes : HtmlAttributes
end

--------------------------------------------------------------------------
-- Table Classes
--------------------------------------------------------------------------

enum CellType {String, Date, Number}

class CellModel < HtmlElementModel
attributes
   Value : String
   cellType : Set(Sequence(CellType))
   SortingValue : String
   Tooltip : String
   Link : String
   TruncationLength : Integer
   TruncationIndicator : String
end
```

```
class RowModel < HtmlElementModel
attributes
  Count : Integer
  IsReadOnly : Boolean
  RowHasCells : Boolean
  NumberOfColumnsForValidation : Integer
  Cells : Set(CellModel)
operations
      ValidateRow(i : Integer) : Boolean
end

class ColumnModel < HtmlElementModel
attributes
  ColumnName : String
  DisplayByDefault : Boolean
  Tooltip : String
  Link : String
  Orderable : Boolean
end

class HeaderModel < HtmlElementModel
attributes
  Columns : Set(ColumnModel)
  Count : Integer
  IsReadOnly : Boolean
end
```

```
class BodyModel < HtmlElementModel
attributes
  Rows : Set(RowModel)
  Count : Integer
  IsReadOnly : Boolean
end

class TableModel < HtmlElementModel
attributes
  Id : String
  TableHead : HeaderModel
  TableBody : BodyModel
  DefaultSortColumnIndex : Integer
  DefaultSortDirectionAsc : Boolean
  DefaultSortDirection : String
end

--------------------------------------------------------------------------
-- Table Associations
--------------------------------------------------------------------------

--------------------------------------------------------------------------
-- Structural Classes
--------------------------------------------------------------------------

class BaseRepository
```

```
end

class BaseService
attributes
        Repository : BaseRepository
end

class BaseController
attributes
        Service : BaseService
        Repository : BaseRepository
end
```