Team Seven Architecture Document

Eric Sundquist, Jeremy Szyba, Shane Taylor, Charlie Thompson, Matt Yale, Jasmin Zehic

October 28, 2015

Revision History

Date	Name	Description
10/27/2015	Eric Sundquist	First release

Introduction

This document describes the architecture and design for the Productivity Analysis System (PAS) application being developed for Commerce Bank. The PAS is an internal tool which allows commerce bank professionals to track and follow key measurements and goals of the company. They system uses a web application connected to a shared database to manage input of data and display reports and charts on the state of various measurements and goals. Our team is specifically designing a dashboard to track measurements and goals over time across multiple departments.

The purpose of this document is to describe the architecture and design of the PAS in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Users at Commerce Bank. Users want assurances that the architecture will provide for system functionality and be reliable, accurate, and highly usable.
- Developers at UMKC. Developers want an architecture that will minimize complexity and development effort, be easy to understand, easy to expand on, and facilitate the task.
- Project Manager. The project manager is responsible to assign tasks and coordinate development work. He wants an architecture that divides the system into roughly equal components by size and. The architecture should support development in parallel with minimal dependencies.
- Developers at Commerce Bank. These developers will be responsible to maintain the code after this project is submitted. They want an architecture which will be extensible and maintainable into the future.

This architecture document will present the 4+1 model of architecture. It will represent the architecture from these perspectives:

1. Logical View. This view will show the major components, their attributes, their operations, and the relations between them. It will use object oriented design with class diagrams and sequence diagrams.
2. Process View. This view will show the threads of control and processes used to execute the operations identified in the logical view.
3. Development View. This view will show how system modules map to the development organization.
4. Physical View. This view is concerned with the physical parts of hardware the software will run on.
5. Use Case View. This view is used to motivate and validate design activity. Use cases will walk through use case scenarios following the interaction between high-level components which demonstrate the overall cohesiveness of the product architecture and design.

Design Goals

The design priorities for the PAS are listed below:

- The design should minimize complexity and development effort.
- The design should take into consideration the nature of the team doing development. Specifically, the developers rarely work in the same place at the same time and can only commit 6 - 9 hours per person per week to the project. The design should therefore be made to consist of several small modules and pieces which have low inter-dependencies.
- The design should emphasize maintainability as the development phase is limited to one semester and the team at Commerce Bank is responsible to understand and reuse the code for the rest of the product's lifetime.
- The design should emphasize completing the basic requirements first, with the ability to add extra features without integration issues as the semester progresses.

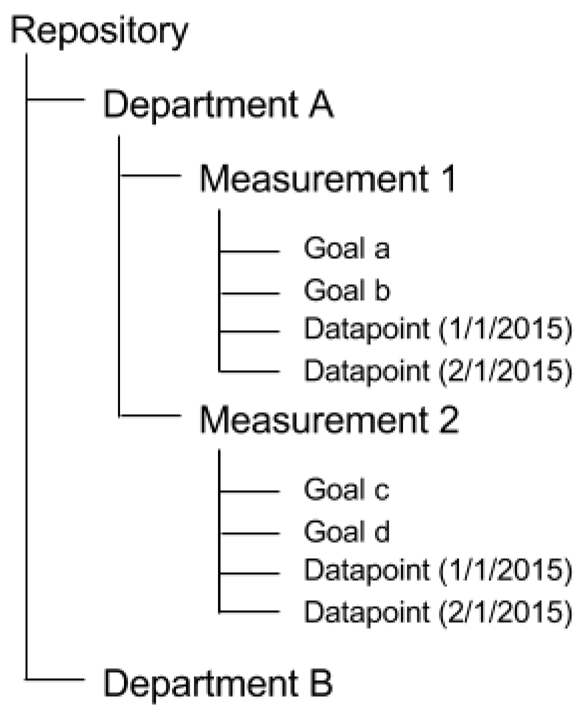
System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expected system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.

The system should organize data related to measurements. A measurement has monthly datapoints which may be manually entered or automatically calculated based on other measurements. Measurements may have one more goals. A goal may be cumulative (from the start of the year) or based on the most recent datapoint. Goals may have different levels of success, including meets, meets plus, and exceeds. Measurements are grouped together to form departments.

The system accepts user input to create new measurements and goals, as well as to enter monthly datapoints. The system will output user-customized reports based on the data. The system will also provide the user with a dashboard of customized data visualizations representing the data. (The dashboard is the specific goal of this part of the overall project.) The dashboard will organize widgets which the user can customize to obtain various perspectives on the data.

The data is modeled as shown below.



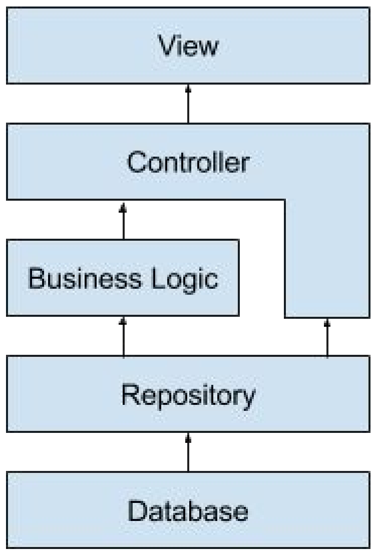
Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction. In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

High-Level Design (Architecture)

The high level design consists of five major components:

- Database
- Repository (Data Access Layer)
- Business Logic
- Controller
- View



The database is the central repository for data on measurements, datapoints, goals, etc.

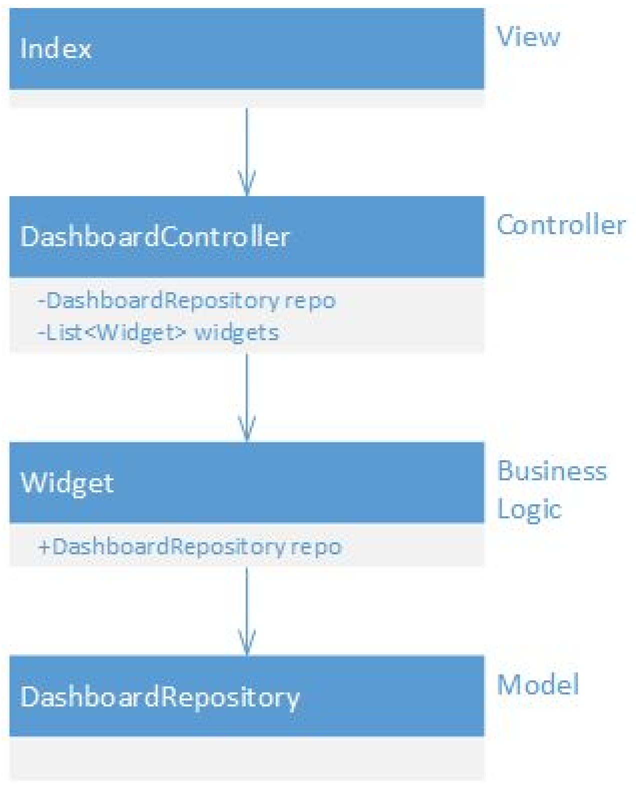
The repository is a data access layer, allowing for programmatic retrieval of information inside the database.

The business logic is responsible to organize the various activities the program performs, such as creating reports and creating dynamic widgets to display on a dashboard.

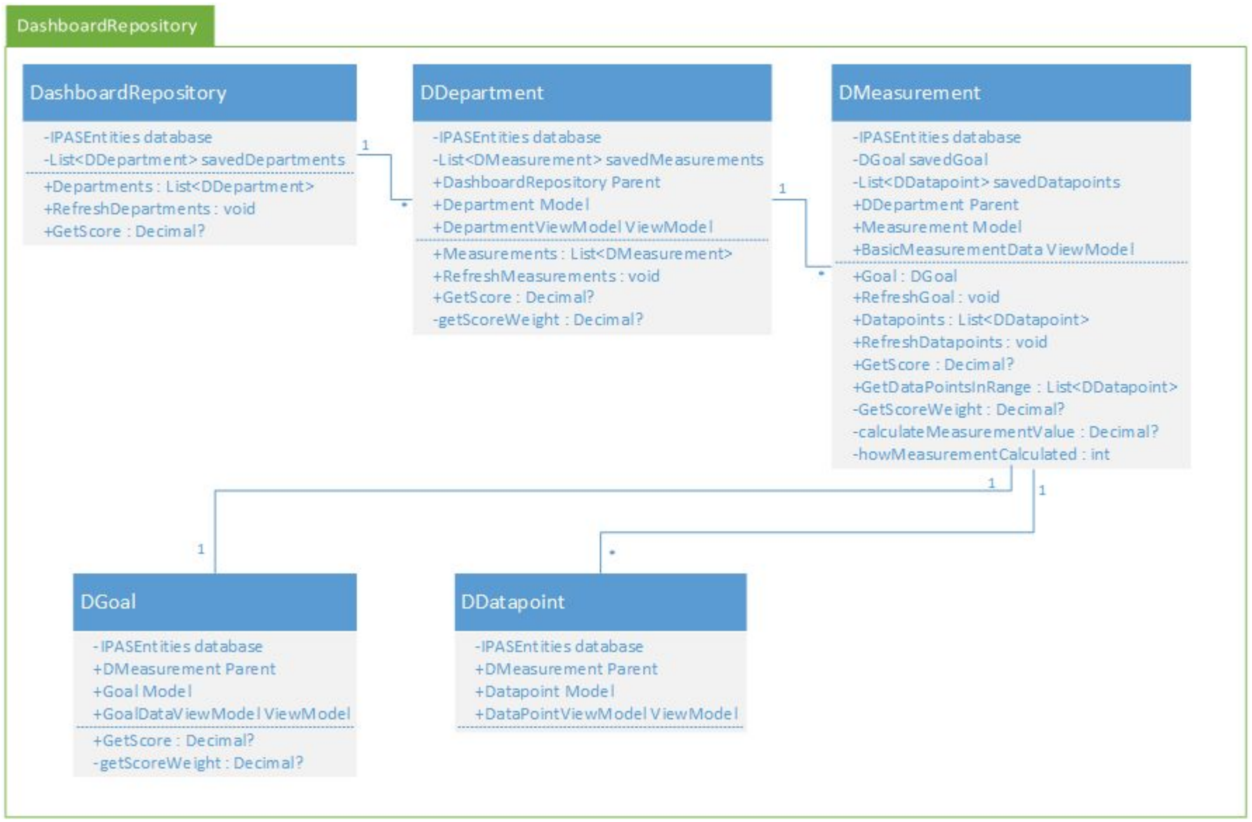
The controller is the main driver of the application. It presents information to the user and reacts to user inputs.

The view is the results the user sees. It displays reports and widgets on a dashboard and allows the user to input data.

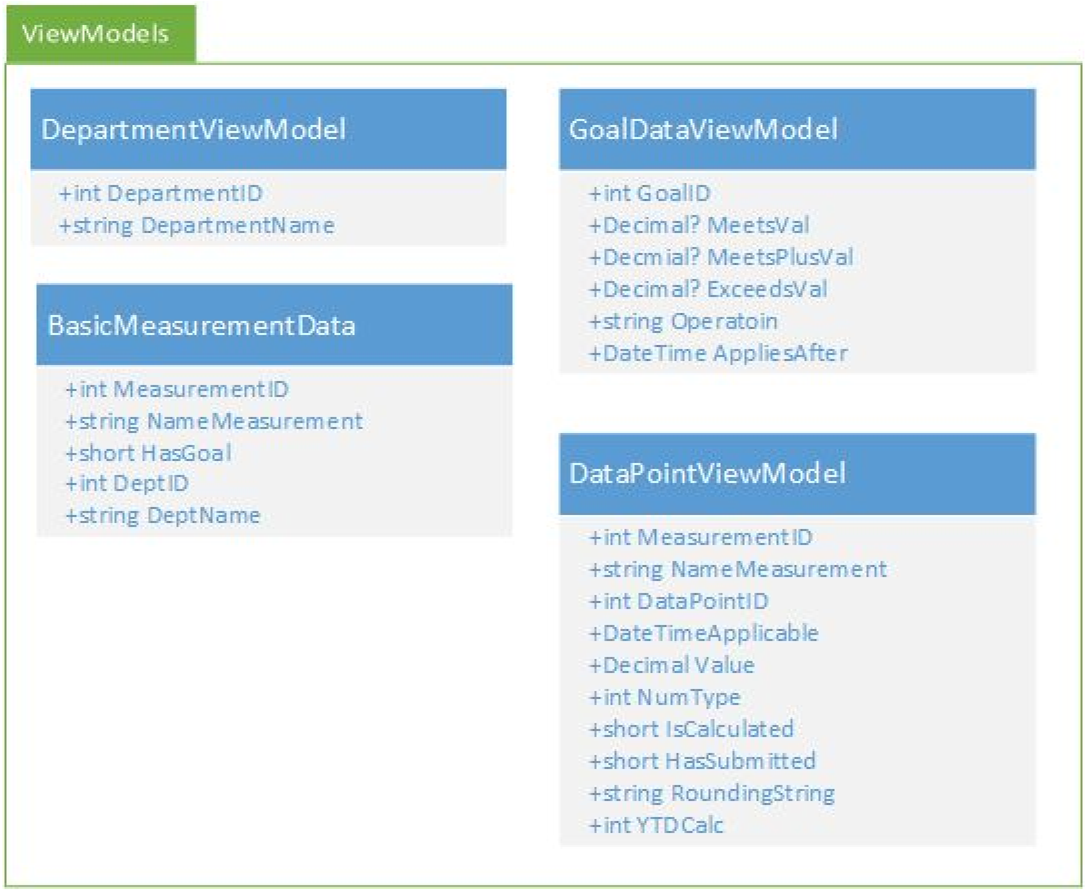
Mid-Level Design



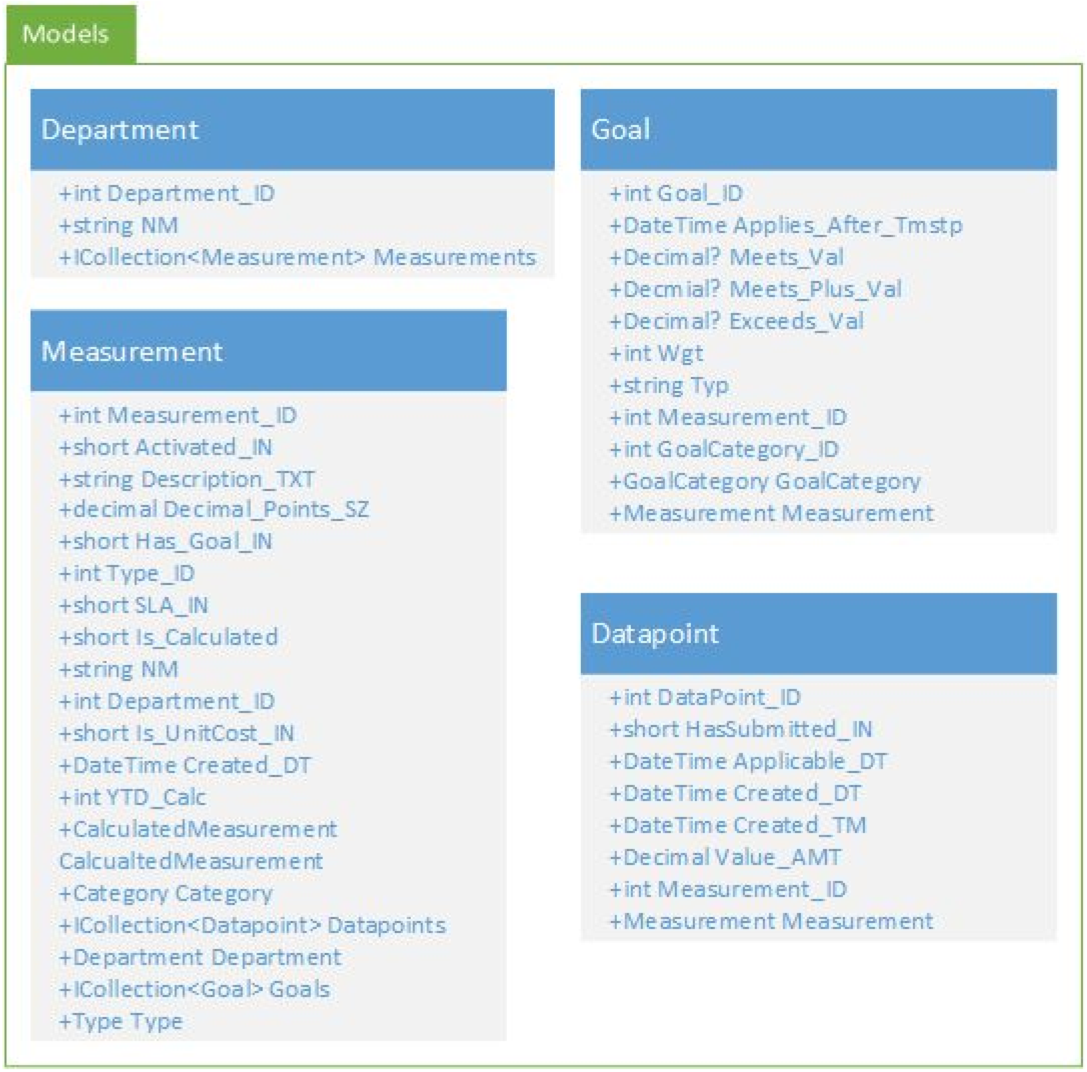
Detailed Class Design


DashboardRepository
DashboardRepository
-IPASEntities database
-List<DDepartment> savedDepartments
+Departments : List<DDepartment>
+RefreshDepartments : void
+GetScore : Decimal?
DDepartment
-IPASEntities database
-List<DMeasurement> savedMeasurements
+DashboardRepository Parent
+Department Model
+DepartmentViewModel ViewModel
+Measurements : List<DMeasurement>
+RefreshMeasurements : void
+GetScore : Decimal?
-getScoreWeight : Decimal?
DMeasurement
-IPASEntities database
-DGoal savedGoal
-List<DDatapoint> savedDatapoints
+DDepartment Parent
+Measurement Model
+BasicMeasurementData ViewModel
+Goal : DGoal
+RefreshGoal : void
+Datapoints : List<DDatapoint>
+RefreshDatapoints : void
+GetScore : Decimal?
+GetDataPointsInRange : List<DDatapoint>
-GetScoreWeight : Decimal?
-calculateMeasurementValue : Decimal?
-howMeasurementCalculated : int
DGoal
-IPASEntities database
+DMeasurement Parent
+Goal Model
+GoalDataViewModel ViewModel
+GetScore : Decimal?
-getScoreWeight : Decimal?
DDatapoint
-IPASEntities database
+DMeasurement Parent
+Datapoint Model
+DataPointViewModel ViewModel
1
*
1
*
1
1
1
*

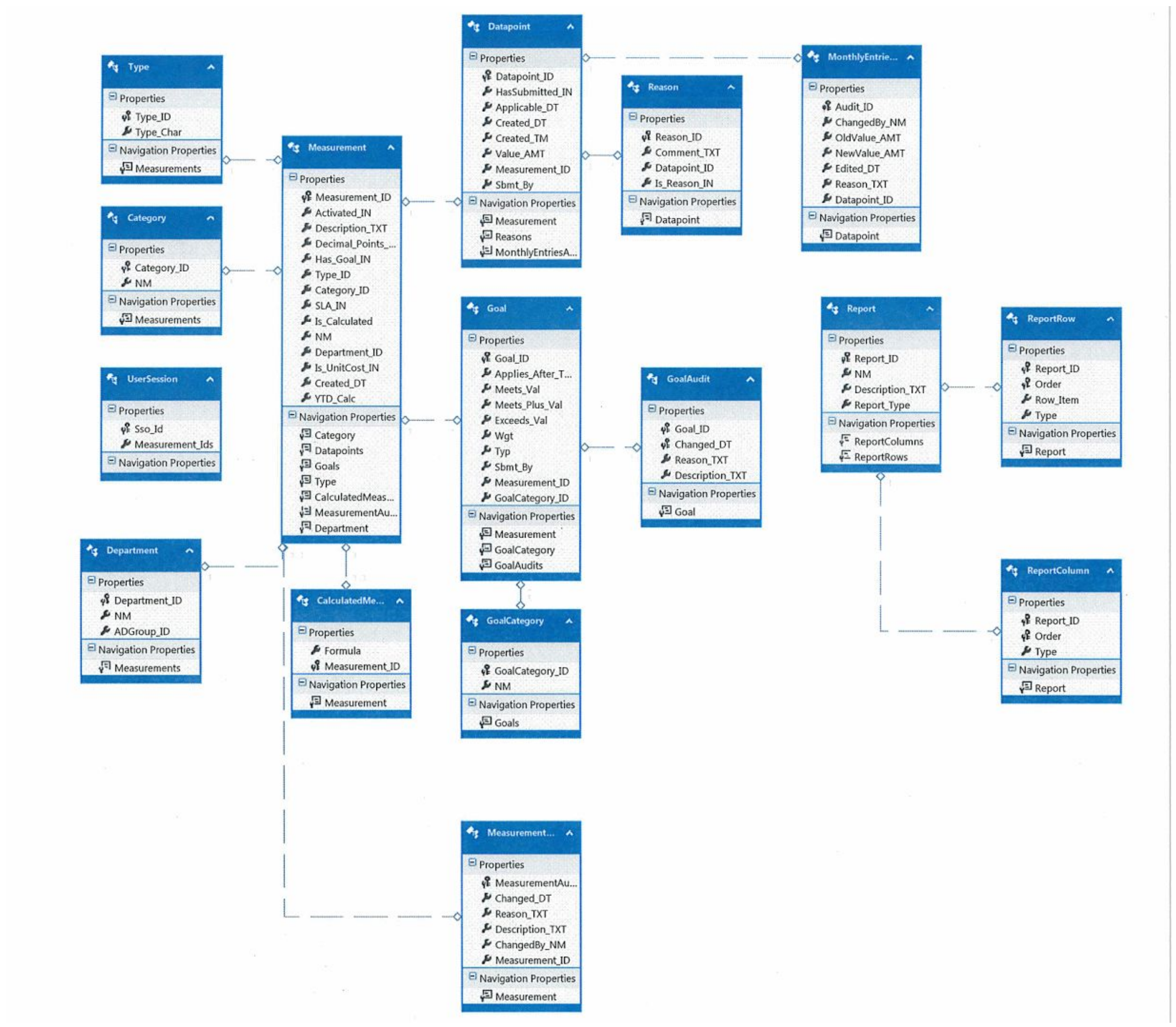
DashboardRepository classes


ViewModels
DepartmentViewModel
+int DepartmentID
+string DepartmentName
BasicMeasurementData
+int MeasurementID
+string NameMeasurement
+short HasGoal
+int DeptID
+string DeptName
GoalDataViewModel
+int GoalID
+Decimal? MeetsVal
+Decmial? MeetsPlusVal
+Decimal? ExceedsVal
+string Operatoin
+DateTime AppliesAfter
DataPointViewModel
+int MeasurementID
+string NameMeasurement
+int DataPointID
+DateTimeApplicable
+Decimal Value
+int NumType
+short IsCalculated
+short HasSubmitted
+string RoundingString
+int YTDCalc

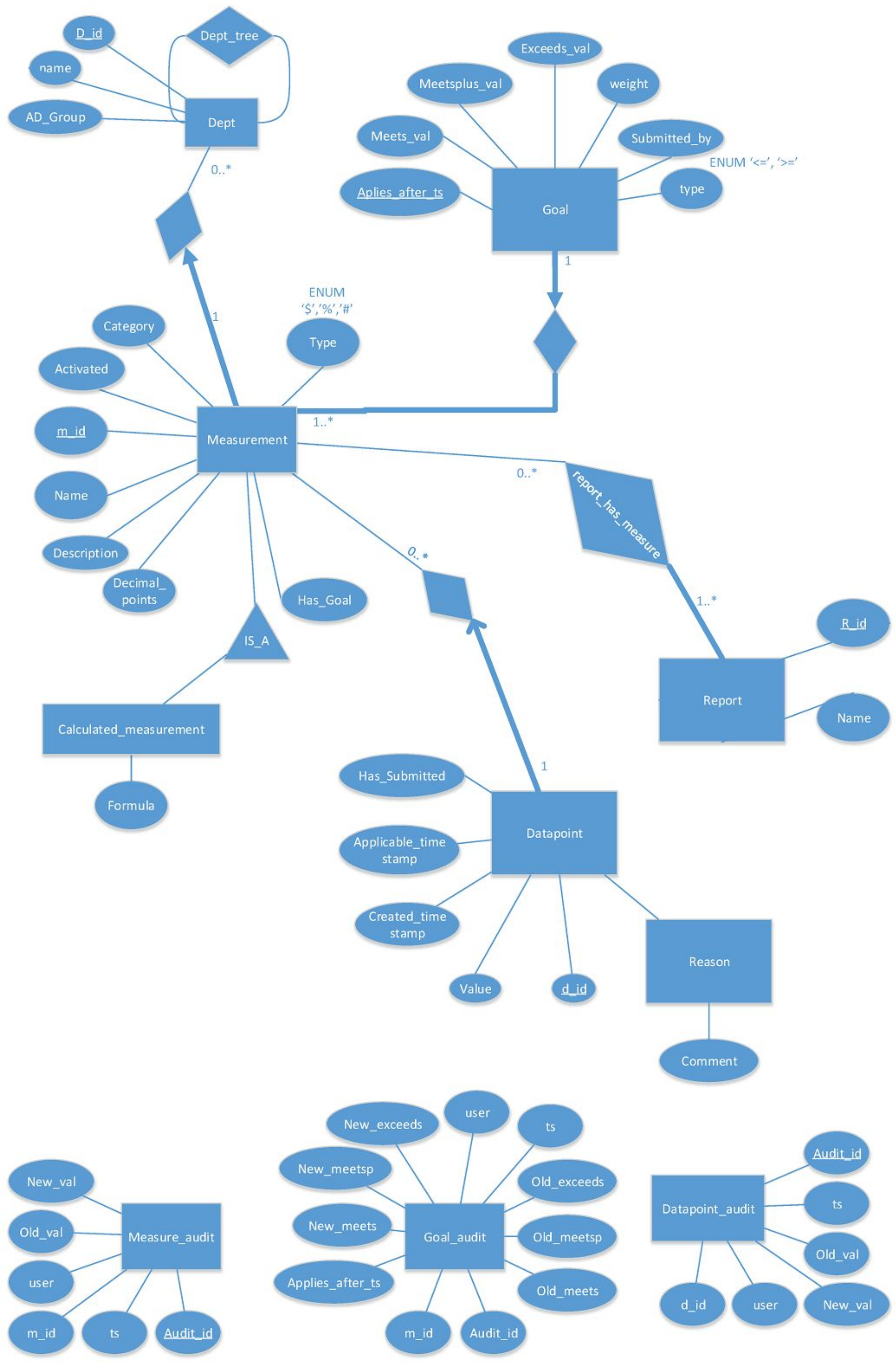
ViewModels


Models
Department
+int Department_ID
+string NM
+ICollection<Measurement> Measurements
Measurement
+int Measurement_ID
+short Activated_IN
+string Description_TXT
+decimal Decimal_Points_SZ
+short Has_Goal_IN
+int Type_ID
+short SLA_IN
+short Is_Calculated
+string NM
+int Department_ID
+short Is_UnitCost_IN
+DateTime Created_DT
+int YTD_Calc
+CalculatedMeasurement CalcualtedMeasurement
+Category Category
+ICollection<Datapoint> Datapoints
+Department Department
+ICollection<Goal> Goals
+Type Type
Goal
+int Goal_ID
+DateTime Applies_After_Tmstp
+Decimal? Meets_Val
+Decmial? Meets_Plus_Val
+Decimal? Exceeds_Val
+int Wgt
+string Typ
+int Measurement_ID
+int GoalCategory_ID
+GoalCategory GoalCategory
+Measurement Measurement
Datapoint
+int DataPoint_ID
+short HasSubmitted_IN
+DateTime Applicable_DT
+DateTime Created_DT
+DateTime Created_TM
+Decimal Value_AMT
+int Measurement_ID
+Measurement Measurement

Models

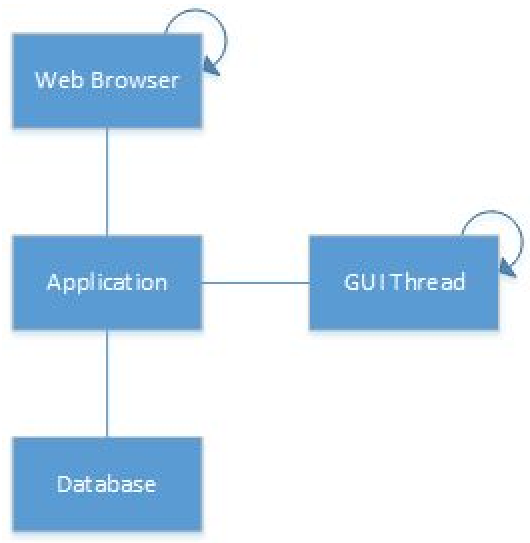


Database Components

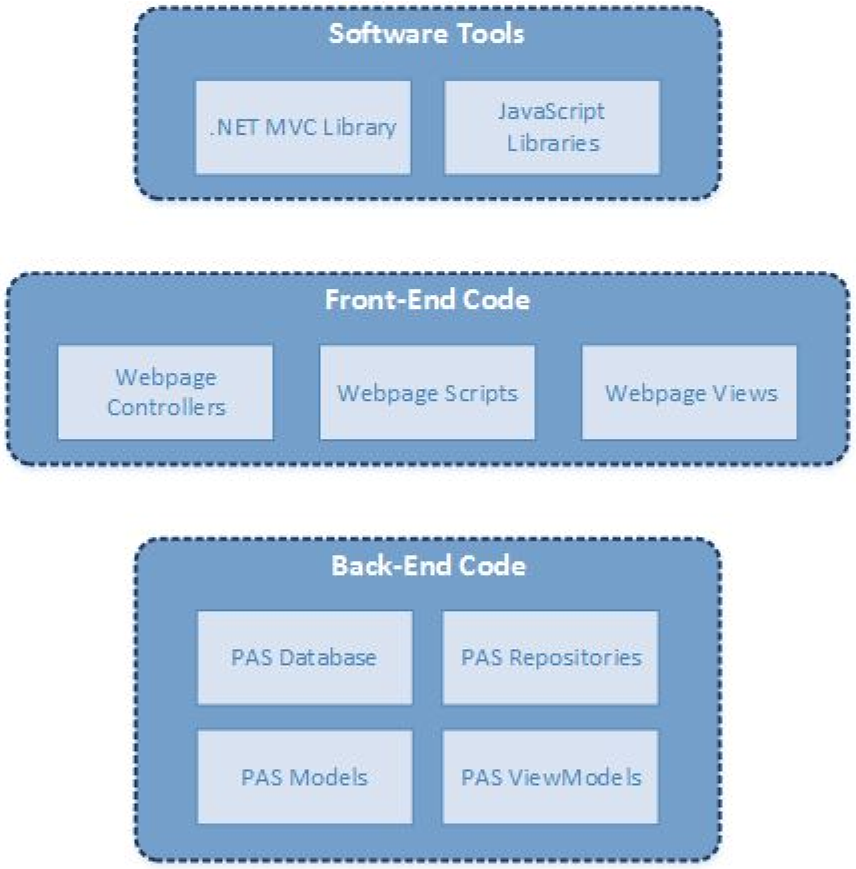


Database components

Process View



Development View



Physical View

The PAS will run on code accessed inside Commerce Bank computers. The database will be housed on site at Commerce Bank.

Use Case View

Title:	View measurement and goal information
Actors:	PAS user
Preconditions:	PAS has measurement and goal information already entered and stored
Basic Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard displays most recently viewed configuration 3. User quickly sees the relevant information and can make analysis	

Title:	Customize dashboard view
Actors:	PAS user
Preconditions:	PAS has measurement and goal information already entered and stored
Postconditions:	PAS saves state of dashboard for next use
Basic Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard displays most recently viewed configuration 3. User adds, removes, resizes, and rearranges widgets as desired 4. System remembers state for next use	
Alternate Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard is empty (first time use) 3. User adds widgets as desired 4. System remembers state for next use	

Title:	Customize a widget
Actors:	PAS user
Preconditions:	1. PAS has measurement and goal information already entered and stored 2. A widget
Postconditions:	PAS saves state of widget for next use
Basic Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard displays most recently viewed configuration 3. User chooses a widget to modify 4. System shows options for modifying widget 5. User chooses options as appropriate 6. System displays the modified widget and saves its state for next use	
Alternate Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard displays most recently viewed configuration or blank configuration 3. User adds a new widget 4. System show default widget and options for modifying widget 5. User chooses options as appropriate 6. System displays modified widget and saves its state for next use	

Title:	Interact with information
Actors:	PAS user
Preconditions:	PAS has measurement and goal information already entered and stored
Basic Flow: 1. User logs on to PAS and navigates to dashboard 2. Dashboard displays most recently viewed configuration 3. User identifies information in widget and chooses to see its details 4. Widget changes appearance to give more details about what user selected 5. User can continue to drill down 6. User can move to previous views or to default view when finished	