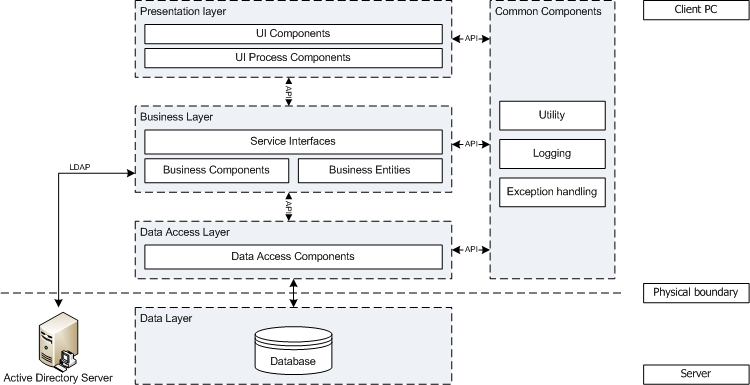
# LOGICAL VIEW



***CIE-PDB Logical view***

## Overview

The architecture of the CIE Pricing Database system is designed following the well-known n-tier, Microsoft .NET approach architecture. The diagram above shows main logical layers of the architecture and the vision of how they fit with each other.

The next sections will describe the role/purpose of each layer and the technologies that will be used.

* Client can use services provided by the CIE-DB application by sending requests to the database server. Presentation Layer, Business Layer, and Data Access Layer will be running on client.
* Data Layer will be running on Database server.

## Architecturally Significant Design Packages

### Presentation Layer

This layer controls the displaying to the end user. For the presentation layer of CIE-PDB application, window form based and .NET framework were chosen.

* Managing requests/responses from/to users
* Controlling the display of UI.
* Assembling a model that can be presented in a view.
* Performing UI validation.
* Providing a controller to delegate calls to Business Logic layer.
* Handling exceptions from other layers that throw exceptions.
* User Control will be used to improve the user facility and usability.

### Business Logic Layer

This layer manages the business processing rules and logic. It consists of a set of Business Objects to support business logic implementation of the system. The presence of this layer is to add flexibility between the presentation and the data access layer so they do not directly communicate with each other.

* Handling application business logic and business validation.
* Managing transactions.
* Providing interfaces for interaction between layers.
* Exposing business services provided by this layer to the presentation layer.
* Managing implementations from the business logic to the data access layer.

1. Handling exceptions thrown from data access layer.

### Data Access Layer

This layer manages access to persistent storage. The primary reason separating data access layer from the Business Logic Layer is to make sure the business implementation and the data are loosely coupled. With this separation, it will be easier to switch data sources and share Data Access Objects between applications later, increasing the reusability, maintainability, and flexibility of the application.

* This layer consists of Data Access Objects and manages reading, writing, updating, and deleting data of the application stored in Microsoft SQL Server 2005 database server (see Data Layer section).
* .NET namespace SqlClient provider will be used for accessing data.

### Common Components

This group of components provide some classes/packages that will be widely used by the other layers of the application.

* Utility package: contains common utility classes that provide common functionalities such as: string processing, date conversion, database access helper, ...etc.
* Logging package: This package contains classes for logging information to the log file for debugging/auditing purpose.
* The log4net component from Apache (<http://logging.apache.org/log4net/> ) will be used to implement this package because it is possible to enable logging at runtime without modifying the application binary. The log4net package is designed so that these statements can remain in shipped code without incurring a heavy performance cost.
* Editing a configuration file, without touching the application binary, can control logging behavior.
* There are four levels of log: ERROR, INFOR, DEBUG, WARNING. In Test environment, all level of log can be turned on (by editing the configuration file) to provide the full error message. In production environment, log Error level can be turned on to provide only the friendly error message.
* Exception handling package: contains classes supporting for handling exceptions.

### Data Layer

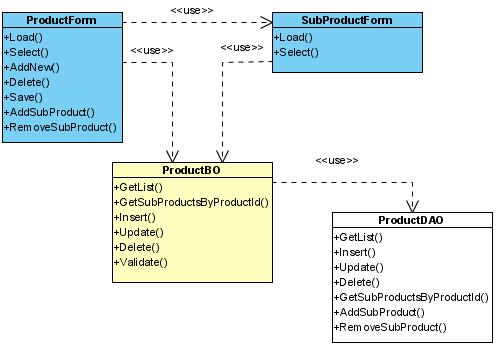
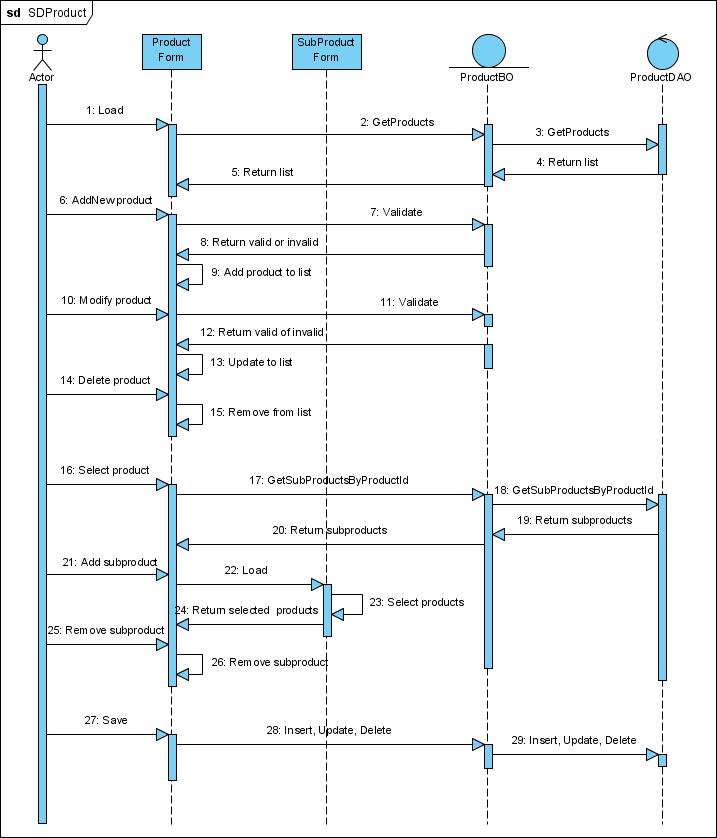
In CIE Pricing Database application, Microsoft SQL Server 2005 will be used to store business data. The Data Layer Access accesses data stored in this database by using SqlClient .

# Data View

* ERD: available
* Table details: trainees to defined (based on provided screen/form descriptions)

# Class View

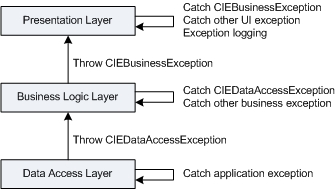
## Common design for a CRUD function

* + Class Diagram
  + Sequence Diagram
* 
* ***Maintain products class diagram***
* 
* ***Maintain products sequence diagram***

## Exception handling mechanism

CIE Pricing Database application defines two exception objects described as following:

* CIEDataAccessException is used as exception object thrown out from Data Access Layer
* CIEBusinessException is used as exception object thrown out from Business Logic Layer



***Exception handling mechanism***