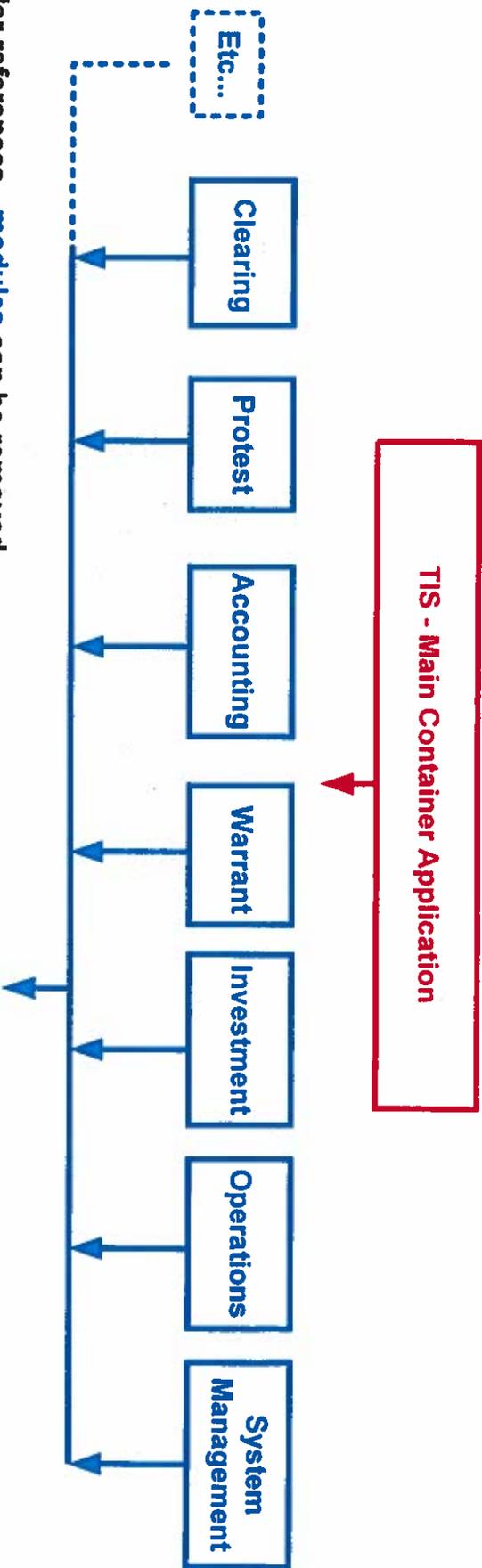
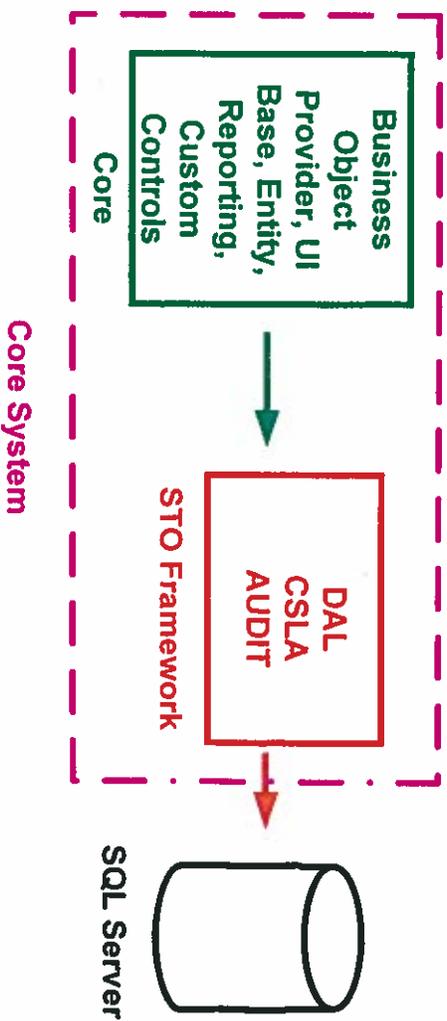


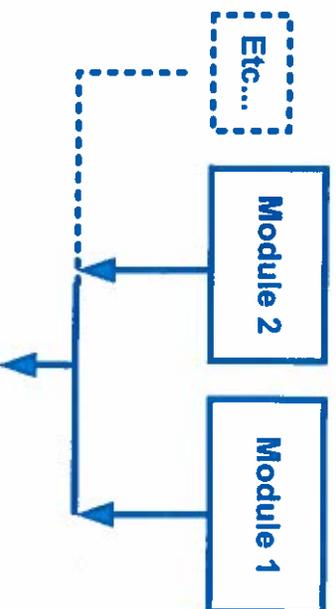
Treasurer's Information System - Architecture Overview



1. No circular references - **modules** can be removed without breaking the system.
2. Business Object Provider can exist on local machine, or on app server for remote access (web).
3. Works with our CSLA based framework.
4. Business object code written once. Consistent and easier to maintain. Available to every module through the provider.
5. Inter-module communication handled by provider. In certain cases, source system doesn't even have to be present.



TMS RFP Project - TIS Architecture
TIS - Main Container Application



Main App:

Simple container application used to insert system modules into

System Modules:

Contain module specific UI screens.

Contain module specific configuration and navigation instructions.

Contain module specific code that cannot be shared outside of the module.

Conform to a consistent interface for loading and navigation.

Core:

Contains all common business objects.

Contains all base forms, reusable UI components and code.

Contains the Entity and Reporting subsystems.

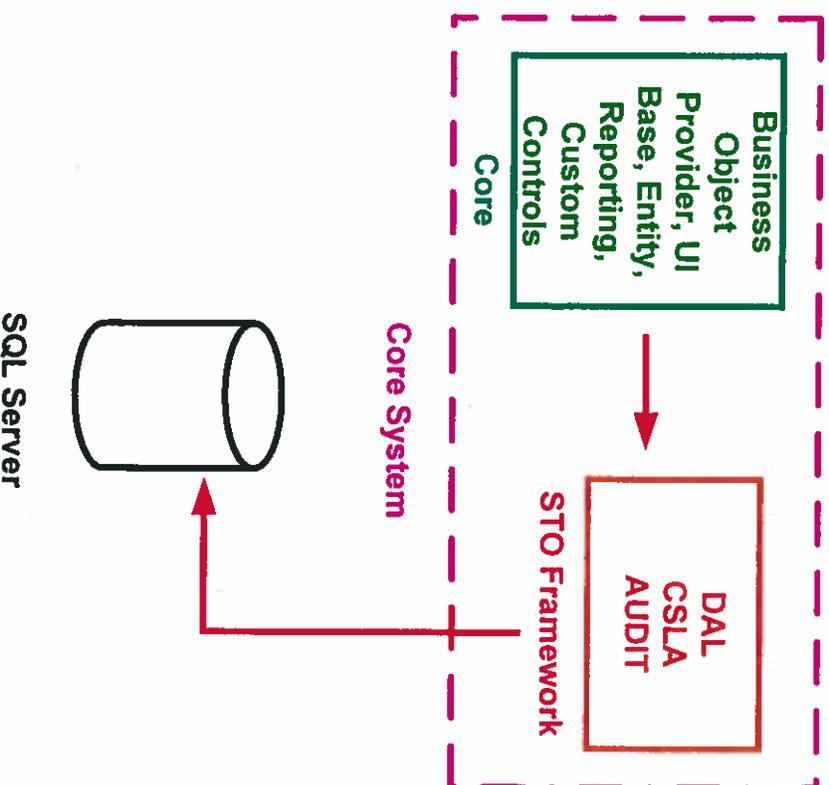
Contains any application defined custom controls.

STO Framework:

Contains the base classes upon which all business objects are built. (CSLA)

Contains the low-level implementation of Auditing functionality.

Contains additional code used in database access.



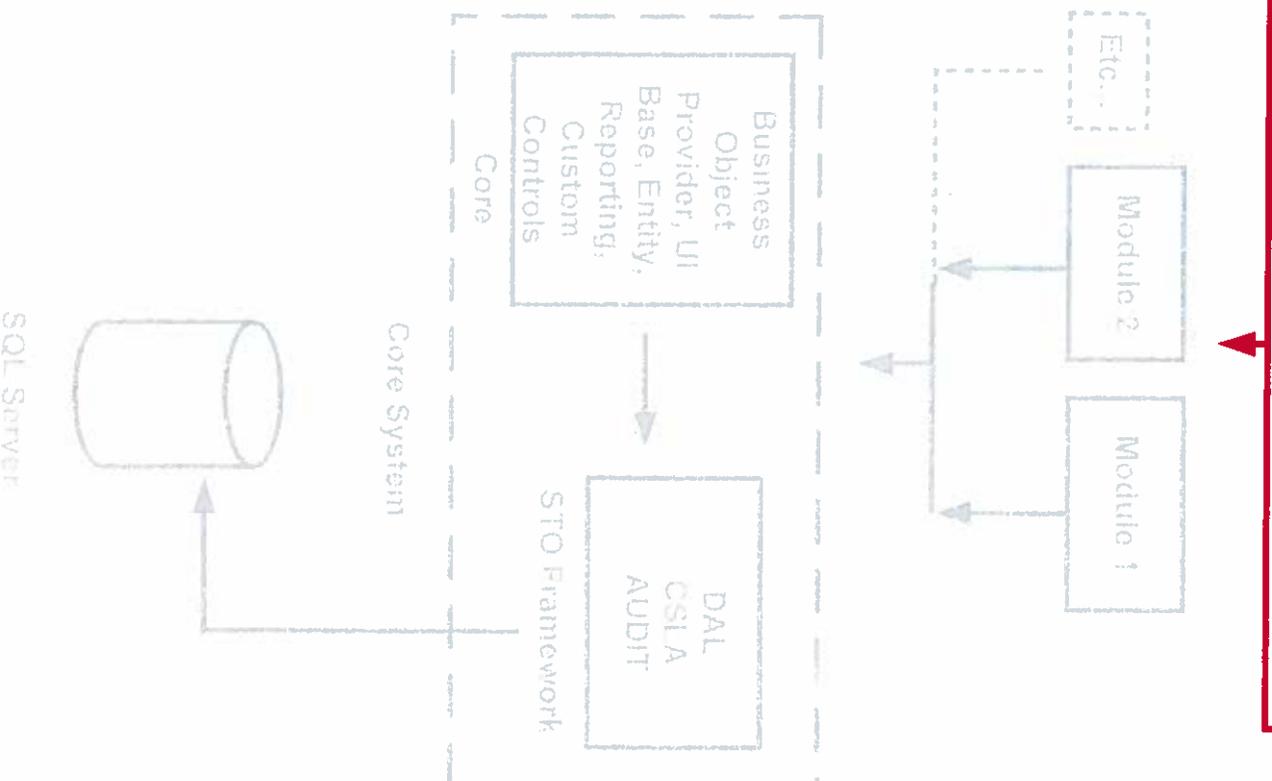
SQL Server:

Contains all system data stored as relational tables.

Contains all data access and manipulation code as Stored Procedures.

Contains an application role - TISRole - that controls access to all data.

TIS - Main Container Application

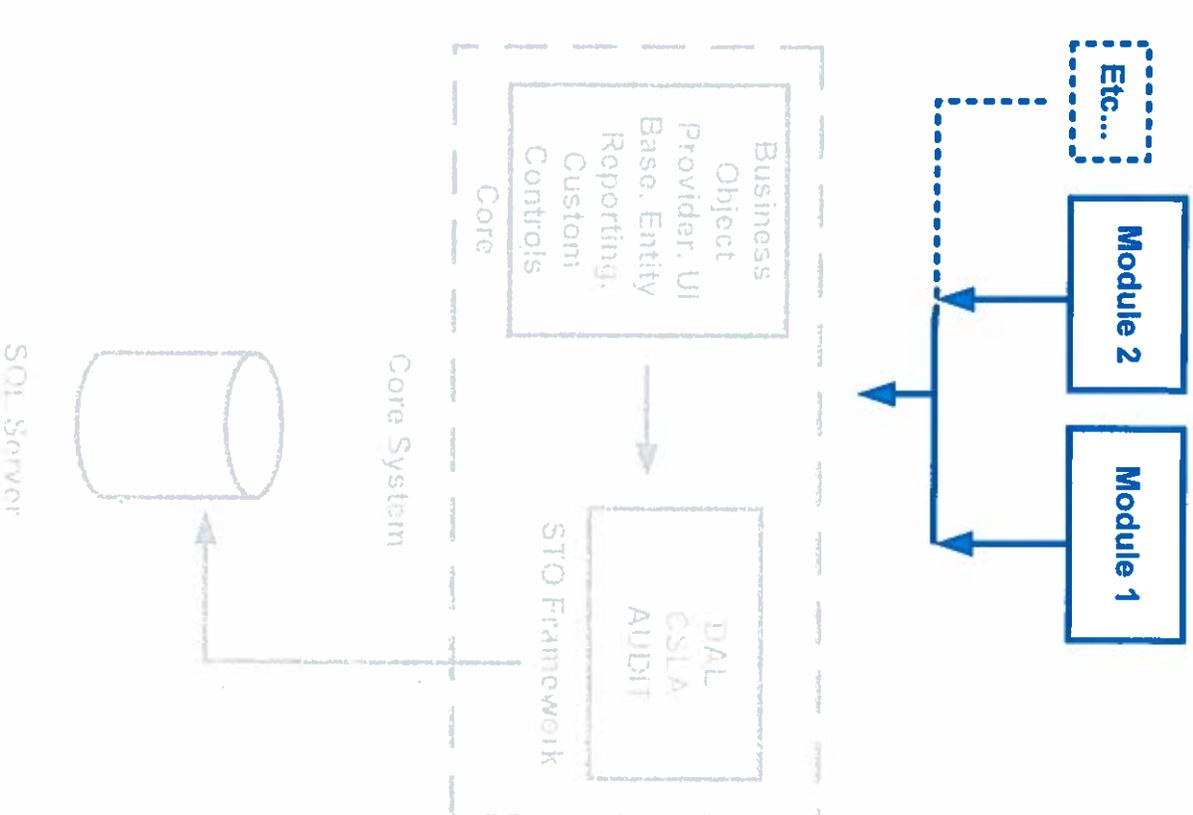


TIS - Main Application Project

This project simply acts as a container for all of the system modules to be inserted into or removed from. Aside from handling the logon process, its only other function is to display the base application form, and initiate the loading of the system modules.

This project also contains the application configuration file, which is then globally available to all projects contained within it.

TIS - Main Container Application



Individual System Modules

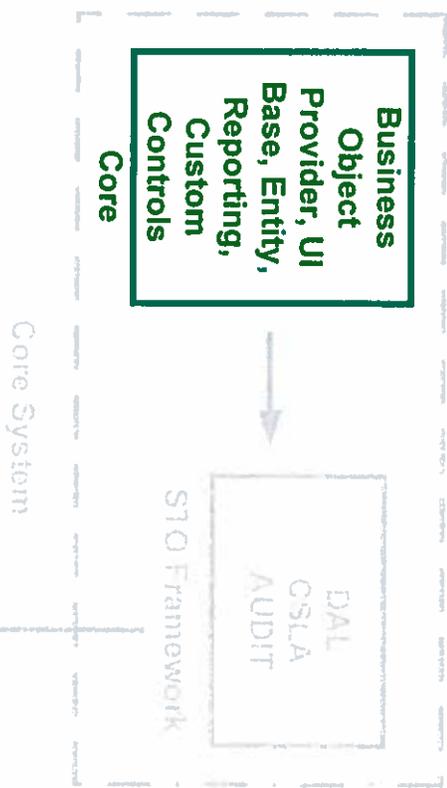
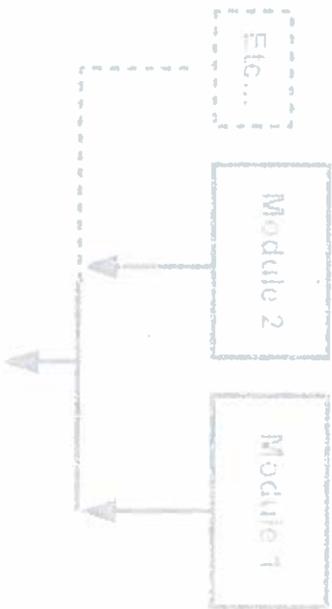
These projects all reference the underlying core and framework objects, but do not directly reference any other system modules. This design allows for the insertion and removal of systems from the main application without introducing any breaking changes. Any data a system module needs from another system module is retrieved via the Core.

System Modules have full access to the Business objects and UI objects in the Core.

System Modules will contain all of the UI objects specific to the programmed system. These objects will be based on items inherited from Core objects, or will be contained specifically in the module itself if necessary due to security or performance issues.

Each module will implement a consistent interface so that the main container application project can interface with it in a predetermined manner.

TIS - Main Container Application



Core:

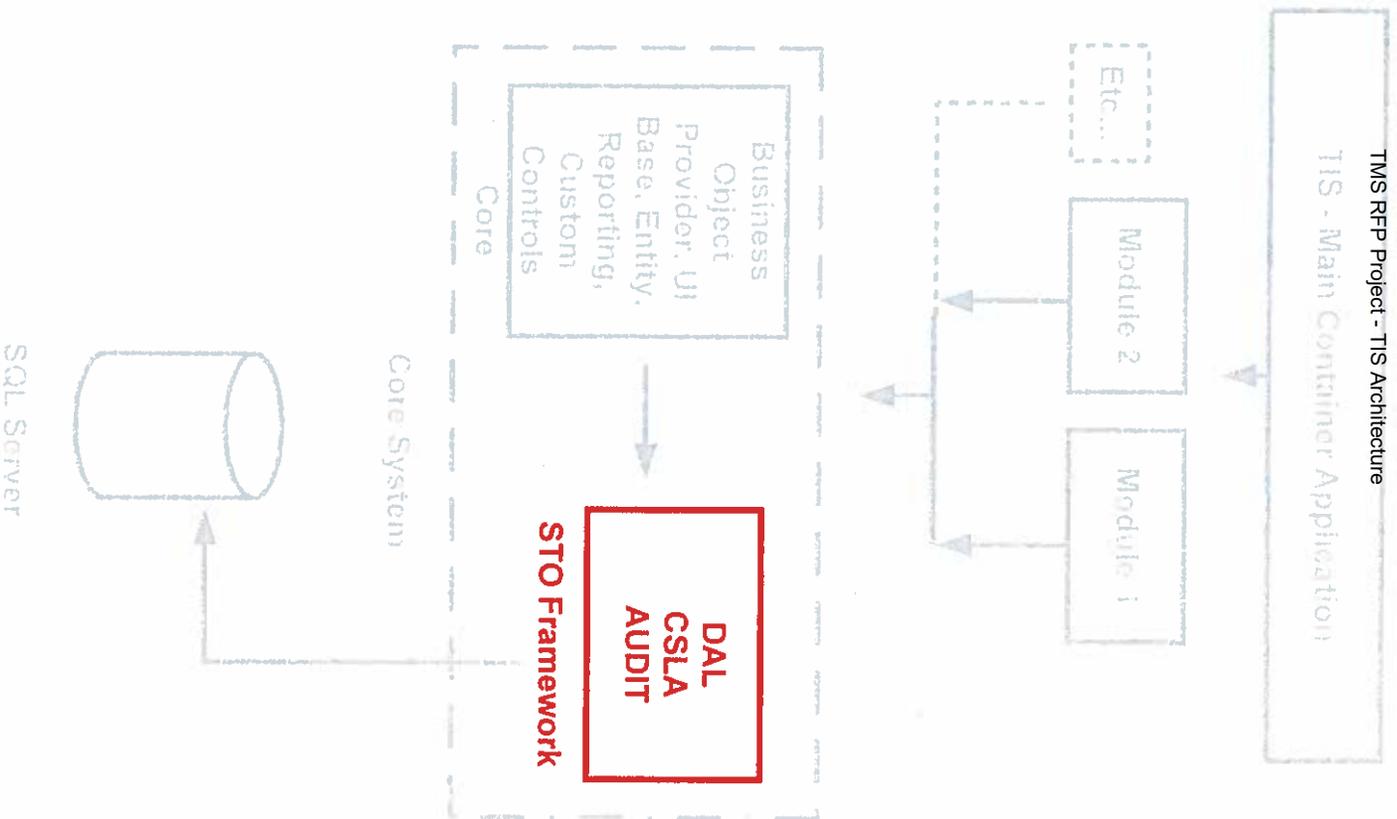
This project is where all of the business objects reside. By utilizing a common object core, objects can be shared across all modules, and the code needs written only once. Several systems depend on data from other systems, and so can retrieve what they need from the object core rather than the other system itself, which may not even be present in the application if the user does not have rights.

Reporting lives here as well, as it is a central part of the system and is available to all modules as needed. A common report controller object is available to handle reporting. Most functions have been abstracted out to reduce the amount of code needed to execute a report from anywhere in the system.

The Entity Management system is also placed into this project so that entities can be available across the system and in every module that needs them.

Custom controls either created or modified for inclusion in the application also live here so that they may be globally available to the UI developer. These objects include form building objects and controls as well as all of the base UI forms that are inherited by the individual module screens.

Core also contains most of the resource objects used in the application such as graphics and icons.

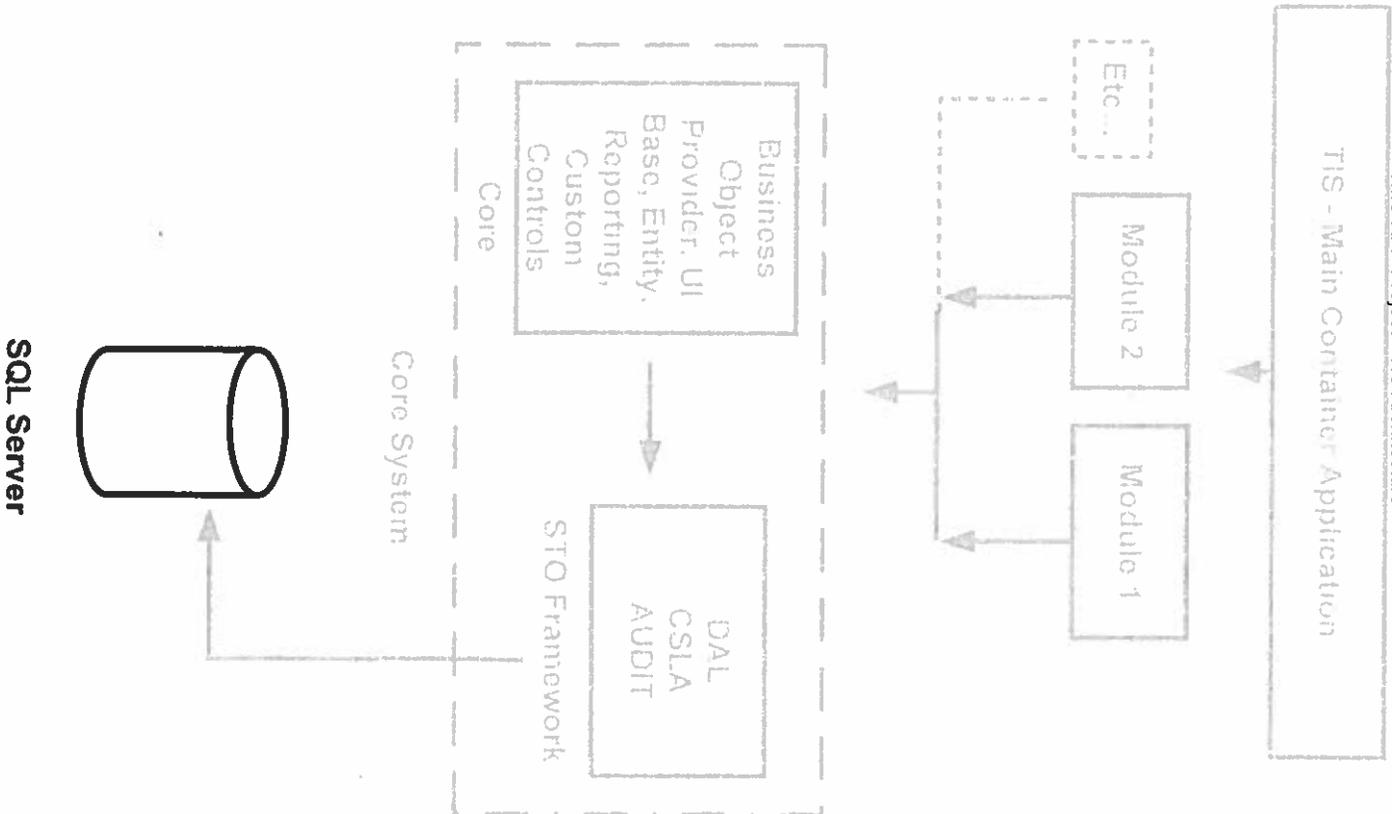


STO Framework

The code stored in this project rarely changes. It is primarily used as a base for the business classes to inherit from, and also for specialized data handling with regards to NULL values and Dates.

- The business class base objects encapsulate all common object functionality including:
- Edit levels,
 - Basic error checking,
 - Field level audit capability,
 - Object cloning,
 - Serialization,
 - Data Binding

All project modules will reference this project, and so its functionality will be globally available.



Database Conventions:

All tables will subscribe to the following standards:

Tables must contain an automatically incrementing ID field for distinction. This field will also be used to relate rows of data between tables and in associated queries.

Tables will also include a timestamp field on each row which can be used to enforce data integrity during concurrent updates.

Since deletion is discouraged due to historical recordkeeping requirements, an IsActive flag will be maintained per row to indicate the active state of the data.

Each row in a table will contain a LastUpdatedBy field to enable audit tracking of edits to the data.

The created application role: TISRole will have execute permission on all stored procedures. Users of the system will be added to the Application Role only. They will not have any direct access to the underlying tables.

Each Stored Procedure will be named in accordance with the standards set in defining the business objects. This will allow for easier maintenance, and help to facilitate the code generation process used to create them.