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.

6. Certain cases, source system doesn’t even have to be present.
Contains all application roles and business logic.

Controls access to all data.

Example:

Framework:
- Contains all data access and manipulation code as Stored Procedures.
- Contains all system data stored as relational tables.

SQL Server:
- Contains additional code used in database access.
- Contains the low-level implementation of auditing functionality.

Core Framework:
- Contains all base classes upon which all business objects are built (CLSAs).

Core:
- Contains any application-defined custom controls.
- Contains the Entity and Reporting subsystems.
- Contains all base forms, reusable UI components, and code.
- Contains all common business objects.

System Modules:
- Compose to a consistent interface for loading and navigation.
- Contains module-specific code that cannot be shared outside of the module.
- Contains module-specific configuration and navigation instructions.
- Compose module-specific UI screens.

Simple container application used to insert system modules into the Main App.

Example:
- Module 1
- Module 2
- etc.
This project also contains the application configuration file, which is then globally available to all projects contained within it.

The project simply acts as a container for all of the system modules. Display the base application form, and initiate the loading process. Its only other function is to handle the login process of inserted into or removed from. Aside from this project, almost any other function is contained within the system modules.
System modules will contain all of the UI objects specific to the Core.

UI objects in the Core:

System modules have full access to the Business objects and another system module is retrieved via the Core.

Individual system modules:

The module itself, if necessary due to security or performance, the module inherits from Core objects or will be contained specifically in the Programmed system. These objects will be based on items from the Core. This design allows for the inheritance and removal of objects, but does not directly reference any other system modules. These objects all reference the underlying core and framework.
Core contains most of the resource objects used in the application, such as graphics and icons.

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

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

Reporting lives here as well, as it is a central part of the system and report from anywhere in the system. Abstracted out to reduce the amount of code that needs to execute a specific object rather than the other system itself, which may not even be present in the application. If the user does not have rights.

Object core rather than the other system itself, which may not even be present in other systems, and so can retrieve what they need from the common object core, objects can be shared across all modules, and this project is where all of the business objects reside. By utilizing a
All project modules will reference this project, and so is its functionality will be globally available.

Data Binding
Serialization,
Object cloning,
Field level audit capability,
Basic error checking,
Edit levels,
Object functionality including:

The business class base objects encapsulate all common

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

STO Framework
create them.

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

underlying tables.

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

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

Requirements, an IsActive flag will be maintained per row to indicate the
Since deletion is discouraged due to historical recordkeeping
used to ensure data integrity during concurrent updates.

Tables will also include a Timestamp field on each row which can be
Tables and in associated queries.
distinction. This field will also be used to relate rows of data between
Tables must contain an automatically incrementing ID field for

All tables will subscribe to the following standards:

Database Conventions:

Else