



The Challenge

Core objectives for the GeoStratSys Desktop application, in order of importance:

Reliability and Availability

One of the problems with existing applications is that databases can get overly cluttered and must be optimized regularly, which decreases performance. The new system's database should be easily maintained by the management team and through community efforts to ensure the data remains as reliable and accurate as possible.

Scalability

The system should be able to handle demands from a growing number of users and processes.

Maintainability

The new application must be designed so that the domain logic is centralized and never duplicated.

Rich Client Functionality

Users are familiar with rich controls and responsiveness. Any enhancements should continue in this style of user experience.

Offline Capability

The new application must be able to work even when the user is not connected to the internet. For those who will run the application with occasional or intermittent connectivity (example: field geologists) the ability to work while disconnected is very important.

Web Access

Some parts of the application need to be exposed to the web, such as reporting and graphical tools. Another improvement would be to extend the application to support an executive dashboard for viewing key performance indicators (KPIs) or similar information.

Intelligent Installation and Auto-Update Functionality

We need to design a deployment method similar to that of web applications, where GeoStratSys Desktop can be easily installed by clicking on a URL from a website. The application must be able to update while it is running, and the updates should guarantee the integrity of the application and its related files.

Modular Framework

The new application should be designed so that modules or plug-ins can easily be created by third-party developers to extend the application.

Cross-Platform Compatibility

The GeoStratSys Desktop should work on not only Windows operating systems, but also Macs and potentially Linux.

The Solution

GeoStratSys Desktop was designed using Microsoft Visual Studio 2008 (which includes the Microsoft .NET Framework 3.5) technologies for both the client-side and server-side partitions. The UI leverages XAML, which is an XML based vector technology that allows a clear separation of presentation and functionality. In addition, the local database is completely XML based to allow for easy extensibility and modification. Any remote data is accessed through secure Web Services.

Achieving Reliability, Availability, Scalability, Offline Capability, and Modular Framework:

We will be using both a SQL Service database on the server and an XML database on the client.

On the Server

In order to support the Reliability and Availability requirements, the database server will be a SQL Server instance. All of the data from our legacy application (PaleoStrat 2.0) will need to be migrated to the new SQL Server database. A SQL migration script or .NET program will be written to facilitate this data transfer. This will allow users to continue adding data to PaleoStrat while the new application is being built. The script or migration tool will make it easier to refresh data on a regular basis from the production database into the development, testing, and staging database environments.

On the Client

One of the requirements of the application is to support users who are not always connected to the internet, such as geologists who collect data in the field. The database on the client will be a set of XML files which can also be used on mobile devices such as PDAs and Tablet PCs.



GEOSTRATSYS DESKTOP TECHNICAL OVERVIEW

Another benefit of having an XML database is that it can take some of the load off the database server, thus addressing the Scalability requirement. The Synchronization Services API, which is modeled after ADO.NET data access APIs, is a much more intelligent, service-based way of synchronizing the data. It makes building applications for occasionally connected environments a logical extension of building applications which have a consistent network connection. Microsoft Outlook is an example of the online/offline functionality that the Synchronization Services API will enable.

Achieving Maintainability:

In order to avoid embedding business logic in the behavior of the UI elements, such as forms, controls, reports, or database queries, a layered architecture will be used. The layered architecture's main principle is that any element of a layer depends only on other elements in the same layer or on elements of the layers beneath it. Using a layered architecture will make the code for this application much easier to maintain. The layers that will be used in the GeoStratSys application will be:

UI (presentation layer): Easiest to recognize, this layer is responsible for showing information to the user and interpreting the user's commands. Sometimes, instead of a human, the user could be another system.

Application layer: This layer is meant to be very thin and is used for coordinating the actions of the domain model objects. It is not supposed to contain business rules or domain knowledge, or even maintain state - that is what the domain model is for. The application layer is very useful for coordinating tasks and delegating actions to the domain model. Although it is not to be used to maintain state of a business entity, it can maintain the state that tracks the current task being performed by the user or system. It is very important that the application layer does not get in the way of the domain model representing the important parts of the business model.

Domain layer: This is where the business logic and rules of an application live, and it is the heart of the software. The domain layer controls and uses the state of a particular business concept or situation, but how it is stored is delegated to the infrastructure layer. It is absolutely critical in Domain-Driven Design that the domain layer contains the business model, and that the domain logic is not scattered across any other layers.

Infrastructure layer: This is where general, technical, plumbing-related code happens, such as persisting objects to a database, sending messages, logging, and other general cross-cutting concerns. It can also serve as a place for an architectural framework for the pattern of interactions between the four layers.

Achieving Rich Client Functionality:

Since most users are accustomed to Windows applications, the client application will also be Windows-based, but it will be much more than just a traditional Windows application. The GeoStratSys Desktop will be a smart client application implemented using the Windows Presentation Foundation (WPF).

Smart Client Definition

The term "smart client" means different things to different people. We define a smart client application using descriptions adapted from the MSDN Smart Client FAQ:

A smart client uses local resources and provides a rich user experience.

It is a connected application that can exchange data on the Internet or on an enterprise network.

Even though it is a connected application, can be used even when it is not currently connected.

It has an intelligent deployment and update story, maintaining relatively the same ease of deployment and management as web applications. Intelligent deployment means that the smart client application is deployed to an application server, and from there it is deployed onto the local client system. Intelligent update allows the application on the client system to receive updates that are deployed to the server. This also satisfies the Intelligent Installation and Auto-Update Functionality requirement.

Smart clients also combine the best features of both Windows and web applications.

Windows Application Benefits

The advantages of Windows applications are that they are able to provide a rich user experience, they are not too complex to develop, and they can use local resources. Using local resources allows Windows applications to be responsive, interact with connected devices, and do other things that web applications cannot easily do.



GEOSTRATSYS DESKTOP TECHNICAL OVERVIEW

Web Application Benefits

The positive aspects of a web application are that it is easy to deploy and manage, since you deploy it to a server not to the client computer, and it has a very broad reach: even PDAs and cell phones can access a web application!

Windows Presentation Foundation (WPF) and Silverlight

WPF and Silverlight are intended to be the next-generation graphics API for Windows applications on the desktop. Applications written in WPF are visually of a higher quality than Windows Forms applications. Some of the relevant highlights of WPF for the GeoStratSys application are:

Resolution independence: Because WPF uses vector graphics, unlike most Windows-based applications of today, graphics and text are scaled to the resolution of your screen. This means that a user can literally shrink or enlarge elements on the screen independently of the screen's resolution.

Declarative programming: Windows Forms do not have built-in support for declarative UI definitions. The .NET Framework as a whole has allowed developers to use declarative custom attributes classes, methods, and assemblies, as well as XML-based resource and configuration files. WPF takes this declarative-based model to a new level with Extensible Application Markup Language (XAML). Using XAML in WPF is very similar to using HTML to define a UI for a web page, but with even more benefits. Not only does XAML give a great range of expressiveness for the look and feel of a UI, but it also allows for parts of the behavior of the UI to be declarative.

Rich composition and customization: It is very easy to customize controls in WPF with little or no code. Almost any type of control can be composed with another control. Applications can also be very easily "skinned" to look differently without requiring any code.

Easy deployment: Using .NET Framework's ClickOnce technology will provide a way to install and run GeoStratSys on the client machines by simply clicking on a URL. ClickOnce ensures that installation will not affect other applications because all files required for the application are placed in an isolated area and it also prevents any custom registration.

Achieving Web Access:

GeoStratSys client application instance will use web services to synchronize its transactions with the server.

Achieving Intelligent Installation and Auto-Update Functionality:

The GeoStratSys application will take advantage of the ClickOnce deployment tools and .NET Framework technology to satisfy the Intelligent Installation requirement. Since the .NET Framework also has built-in support for automatic updates and for rolling back to previous versions, the Auto-Update requirement will also be satisfied. This will also make GeoStratSys very easy to deploy.

Achieving Cross-Platform Compatibility:

The GeoStratSys application will take emerging XML based technology of XAML and Silverlight, which will allow our application to run online and offline, on Windows-based systems, Macs, and even Linux.

Technical Breakdown of GeoStratSys Desktop:

User Interface:	WPF and Silverlight
Application:	Application Services
Domain:	Domain Classes .NET Framework 3.5
Infrastructure:	ADO.NET Entity Framework Synchronization Services for ADO.NET Windows Community Foundation SQL Server XML Database