

White Paper on AdHoc Web Reporting for Legacy System

1. Introduction

Legacy applications pose a key challenge to the way in which an organization is able to leverage them to ensure delivery of its existing services. These services are rendered to the organization's customers, suppliers, employees and other business stakeholders. In addition, these applications need to support and improve an organization's ability to maintain and/or even enhance competitiveness, without unduly compromising the protection of its investments in them. TWS is also one of the organizations, which has been catering to the agriculture industry with the help of legacy applications.

2. Background

TWS is a software solutions and services company that specializes in management information systems for agri-business. TWS has been developing systems for the agri-business for the past 25 years. The industry solutions that they provide include:

- Commodity Handling & Merchandising
- Fertilizer & Crop Input Retailing
- Feed Manufacturing & Retailing
- Seed Manufacturing & Distribution
- Agribusiness Retailing
- Fuel Distribution & Retailing
- Co-operative Management

3. The Problem

TWS existing software's were running in a client-server environment. The architecture of the existing software was not scalable. Some of the problems seem within the systems were:

- Legacy systems like D3 were used for data storages.
- The client applications were very similar to telnet client. The client applications were used as dumb terminals.
- The customer has to use dos based screens for order entries.
- Reports could not be viewed and printed properly
- Lack of user-friendliness, while using the system.
- Not flexible to changing business, market and end user needs
- Supporting the existing clients needs knowledge of D3 databases, internal structures, functions and procedures.
- Good working knowledge of D3 scripts and query building were required if new reports were to be added or existing reports were to be enhanced

4. Client Requirements

TWS wanted to create an enterprise level-reporting tool, which their customers could use to generate dynamic and static reports with the existing D3 systems as their databases. They wanted the new system to have strong reporting capabilities to improve control and management of the business. The tool would allow the system to adapt from a single user up to hundreds of users. They wanted to leverage the existing D3 customer's with the help of the new system. Some of the features to be incorporated into the system were:

- Pivot Report
- Custom Header and Footer Color
- Custom Group Headers and Footers
- Totalling and Subtotalling
- Adding Run time Columns
- Complex Formula supporting Run time Calculation fields
- Expanding & Collapsing
- Report Grouping and sub grouping
- Group level Totalling and Subtotalling with Repetitive Columns
- Multiple level Groups with totalling, Sub totalling
- Multiple Groups with totalling, Sub totalling and repetitive column
- Group By Year, Quarter and Month
- Group By Year, Quarter and Month with options for totalling and sub totalling
- Grouping by financial quarters (Q1/Q2/Q3/Q4)
- Suppress Group Column and Row
- Hiding Groups and Columns
- Adding Runtime columns with grouping functionality
- Custom heading in group summaries
- Pagination
- Null field handling
- D3 Custom Data Handling
- Supports for different date formats
- Exporting report's to Excel / Word / pdf formats

5. Challenges

The proposed architecture was a 3-tier model. The system would still be working with the existing d3 as its database. The challenge was to build a web based query-building tool that would connect to the d3 database and would show all the tables and columns of the database. The tool would also allow the user to build his custom filters on the table columns. Based on the table column type, the filter conditions had to change. The user would save the query as a template that would be later used to fetch data from the database. On the other hand a web based reporting tool needs to be created that would suffice the above requirement. The other challenges within the project were

- Configuring d3 database.
- Restoring and taking client data backups
- D3 internal structures, proc, functions and data types where very different from a conventional DBMS or RDBMS
- Working with multi value data types
- Working with d3 scripts and query manipulations
- Working with d3 date formats
- D3 incapability to distinguish between common data types such int, long, float, strings etc
- Managed .NET providers for d3 database
- The reporting tools capability to handle large unformatted d3 data
- The reporting tools capability to convert and perform calculations on the fly
- The reporting tools capability to do grouping and sub grouping. Grouping could extend to any level
- Groups having case sensitive data. They might not fall in the same group etc.
- Performing Group level totals and report level summaries
- Date wise report to have options of showing Financial Year / Quarter / Month etc.
- Options to export the report data to Excel, Word, PDF, and Html
- Identify repetitive data's and ignore them while performing calculations like group total and overall report summaries
- The reporting tools capability to generate OLAP / PIVOT reports

6. Solution

An engineering team was setup to evaluate different enterprise reporting products that existed in the market. The primary goal of the product was to suffice all client requirements. The tool would also be generic in nature, so that it works with different database like SQL, Oracle etc in addition to the d3 database, in case the client ports his data to one of the RDBMS in future. As the reporting was to be deployed on .NET, a .NET based managed providers were needed to work with d3 databases. PICK.NET and MV.NET were identified as the managed providers to work with d3 database. The engineering team also worked on an in house tool as a proof of concept and designed some prototypes. The team identified the pros and cons of the major reporting tools and made a comprehensive comparison between the best products available in the market and the tool that was designed in house.

COMPARISON BETWEEN REPORTING TOOLS

REPORTING TOOLS				
Features	WebGrid	Excel	Crystal Report	In-House Grid

Performance	Average	Good	Good	Good
Support	Poor	Good	Good	

Run Time				
Grouping	√	√	X	√
Totalling and Subtotalling	√	√	X	√
Adding Run time Columns	√	X	√	√
Complex Formula supporting in the Run time Calculation				
Exporting	√	X	√	√
SubReport	X	X	√	X
Expanding & Collasping	√	√	√	√
Pivot Report	X	X	X	X
Totalling and Subtotalling with Repetitive Columns Consideration	X	X	X	√
Grouping with Hiding Group Columns	X	X	X	√
Multiple Groups with totalling, Sub totalling	X	X	X	√
Multiple Groups with totalling, Sub totalling and repetitive columns	X	X	X	√
Group By Year, Quarter and Month	X	X	X	√
Group By Year, Quarter and Month with totalling and sub totalling	X	X	X	√
Group By Year, Quarter and Month with totalling and sub totalling with repetitive columns	X	X	X	√
Custom Header Color	X	X	X	√
Custom Footer Color	X	X	X	√
Custom Group Footers	X	X	X	√
Custom Group Headers	X	X	X	√
Suppress Group Column and Row	X	X	X	√
Suppress Group Column and Row with Custom Header and Footer	X	X	X	√
Pagination	X	X	X	√
Null Field handling	X	X	X	√
Null Field, String handling with respect to D3 data	X	X	X	√
D3 Custom Data Handling	X	X	X	√
Different Date format supporting	X	X	X	√

The comparison clearly showed that the in house tool would be a better choice than the products existing in the market. Based on the proof of concepts,

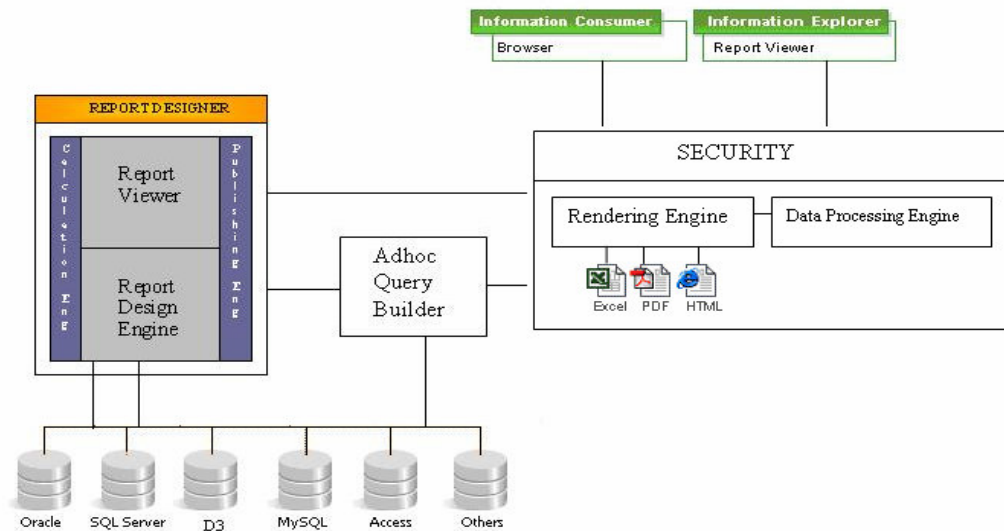
7. Application Development

7.1. Architecture

7.1.1. Architecture Principles

- Service based 3-tier architecture
- Separation of data and data management logic from application logic to hide physical database implementation from application logic
- Browser-based / thin client front-end
- Presentation layer patterned on Model View Controller
- Loosely coupled object-oriented components
- Improve maintainability, usability and scalability

7.1.2. Component Architecture



Adhoc Query Builder: It allows users to create and manager user defined queries by selecting any table and its columns. Users can build and save his filter conditions. The query builder intelligently identifies the type of data source and changes the filter types that can be applied to the table. Some of the inherent features are

- Building dynamic queries

- Dynamic filter types
- Compatibility with almost all known data sources
- There is no post back during query building
- Client side implementation architecture
- Allows saving the queries in a template format
- Templates are be used for generating reports OLAP Pivot, Crystal reports

Report Designer: Main features of the reports designer are:

- Data sorting, grouping, applying filters
- Unlimited hierarchical reports
- Unlimited multicolumn reports
- Reports with grouping
- Unique ability – containers
- Unique ability - segmented pages
- Generate cross tab reports
- Pass user parameters to the report
- Summary and Group calculations

Publishing engine: Will publish the generated reports in the web server, so that other users can view the reports.

Report Viewer: Provide design time capabilities to view the reports.

Data Processing Engine: OLAP cubes are used for data processing. Once the user has identified the data source, the cubes will process the data and dynamically create 3 dimensional view reports. These reports can be drilled down to the nth level. User can decide, which of the column data needs to shown at the row, column, data or the page level. Various report formats can be selected

Rendering Engine: Will generate reports in different formats like Html / Word / Excel / Pdf etc.

7.1.3. Analysis modeling

- Use cases were identified for the system
- The business objects, attributes and methods were abstracted from the use case views, object diagrams and sequence diagrams
- Classes, properties and attributes were modeled. Mapped classes to tables and defined keys. Defined operations and associated error messages. Define queries required.
- The dependencies and integration points were identified.

7.1.4. Design Modeling

- Refined the analysis model into the design model
- Identified new design level classes and attributes.
- Studied the detailed interaction design (sequence and collaboration) and state transitions.
- Identified persistent classes. For each persistent class, data access methods were identified. Microsoft application blocks were used for data access
- Mapped persistence classes and their attributes to tables and columns.
- Identified queries that were not generated by default. Modeled the Query objects and their types (Simple Query, Paging Query, Cursor Query, Multi-row Cursor); model input and output parameters
- Defined interfaces for each of the components and defined dependencies with each other

7.1.5. Deployment Platform

- Microsoft Windows 2003 server
- Microsoft IIS Web server 6.0 to facilitate sessions and Web Garden
- Microsoft Office 2003
- D3 Database & Microsoft Access 2003 database for ACL

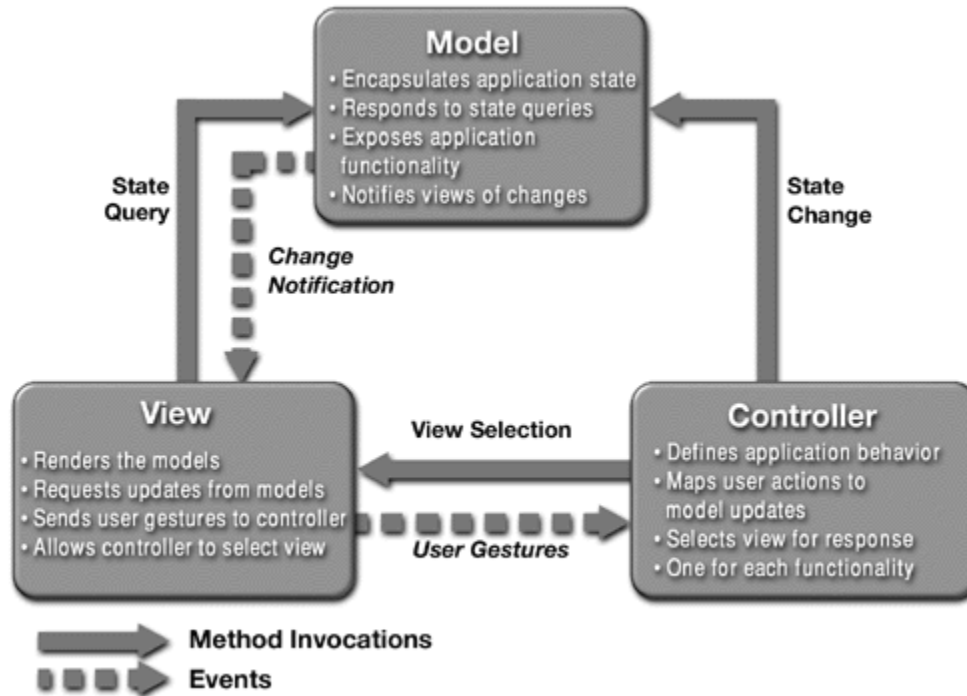
7.1.6. Development Platform

- Microsoft Windows XP
- Microsoft IIS Web server 5.1
- Microsoft Visual Studio 2005
- Microsoft Office 2003 and tools

7.2. Presentation Development

7.2.1. Architecture

This is based on the Model View Controller (MVC) pattern to facilitate loose coupling of input, output and processing logic.



7.2.2. . NET development Approach

- . NET Framework (ASP.NET, HTML and JavaScript)
- Model-View-Controller approach
- Model (Business Process) is developed using C# classes
- View is implemented using ASPX pages, which contain HTML and the client side scripts
- Controller is implemented as the Page class, which is the code behind file. This class manages the interaction between the business process and Web page.
- Visual Studio 2005 is used for building and compiling the ASPX pages
- Developed Unit Test Cases, Unit Test Plan, Unit testing and the integration testing strategy for the tool

7.3. Testing

7.3.1. Development Testing: Following were the processes used within the project by the development team

- Prototype reviews
- Use case reviews
- Model walkthroughs
- Object oriented code testing for methods, classes, class integration
- Traditional code testing – black box, white box, boundary value, coverage and path

7.3.2. QA Testing: Following were the processes used within the project by the internal quality assurance team

- Technical Design reviews
- Code Inspection
- User interface testing
- System testing – Functionality, stress, installation and operation
- Regression testing

7.3.3. Testing Tools

- For UI regression testing, WinRunner was used. It has the options to record and playback online test cases. The test cases could be modified using a scripting language. Tools like Ants-Load runner was used for Load testing. Virtual clients, measuring 50-100 were used for load testing
- Ant-Memory profiler was used to identify the size of objects. The tools helped in identifying how many objects were alive and how many objects had been destroyed.
- Ants-Performance profiler was used to identify bottlenecks in the users code. It helped in measuring the time taken for each of the individual function calls.

7.4. Performance and Application tuning

Performance, scalability and availability are critical requirements for any large application. It was the case with TWS. Performance was the key, as the application had large number of users. Performance was addressed by:

- Making prototypes of the Query Builder and measuring the query building time
- Measuring the time taken for the Stored proc / query to execute using d3

- Measuring time taken by the Pick and MV .Net providers to execute queries
- Measuring the Web Applications request, response and throughput time
- Measuring the web request hits/ seconds and application level errors
- Monitoring and measuring network bandwidth and network latency
- Measuring OLAP cube and Pivot report generation time
- Measuring the time taken to export to other formats like Excel / Word / Pdf
- Performance tuning was done at
 - Code level (e.g. Code reviews)
 - Web server level (e.g. implementing web garden and creating application pools)
 - Provider level (changing parameters)
 - Database level (e.g. Using Stored procedures, Generating Normalized data views, Data schemas imported into XML etc.)