# Section 5 – UI Layer

UI – BO there are many views of how a user interface should be built to handle errors. One view is that the user should be prevented from entering any invalid data into the user interface. I.e. masked edits, preventing letters in a numeric field etc. The other extreme is that the user interface should allow invalid data and the validation logic (in the business object) must validate this data and present the appropriate error to the user explaining the error. Most applications exist somewhere between these two extremes but regardless of which extreme the onus is on the business object developer to protect the business object from misuse (i.e. no invalid data must be allowed into the business object).

SmartDate NNB allows empty date. Empty date can be either MaxDate or Min Date. This is configurable. For comparisons if empty is MaxDate then empty gr8ter than other date else less than.

E.g. A non compulsory e.g. Shipment date and a loop if ShipDate < Today should return valid even when there is no ship date.

Also should do money type.

Null values in the database: vs not entered values convert zero to null in VB6 what happened to this.

## Reporting

Reporting is a term used for a very wide set of artifacts. Traditionally anything that can be printed is viewed as a report. Business intelligence tools that allow for cubing slicing and dicing etc are also often viewed as reports even though they are seldom printed.

Reporting is one of the areas that is seldom addressed in frameworks. The reason for this is that it is extremely hard to generalize reporting requirements, most reporting tools are datacentric and therefore expect to build up reports directly from the database. Implementing any reasonable number of reports using a traditional reporting tool such as Crystal reports bound to the database is extremely problematic since it breaks all the hard work done in building a multilayered architecture by binding the Presentation layer directly to the datasource. Each report is typically bound to its own view of the data via a view, stored procedure or sql statement. Reports of this type are almost impossible to test in an automated fashion and any automated tests are extremely fragile. After the implementation of reporting it therefore becomes almost impossible to make any changes to the databases for fear of breaking all reports. In cases where users are provided with tools that allow the user to build reports that directly report of off the transactional database tables this problem becomes even worse.

In addition to the above issues reporting in this manner often frequently requires the duplication of logic that is already contained in the business object layer and logic is often duplicated between reports.

We have seen many a well written well architected system fall apart due to the inappropriate implementation of reporting strategies.

We therefore believe that an appropriate reporting philosophy and architecture is essential to any enterprise application. For these purposes we have divided reporting requirements into a number of categories and will then suggest guidelines, appropriate solutions and technologies.

* + - 1. Business artifacts: These are reports that are typically produces as part of the business process e.g. Invoices, Order confirmations. These should always be accessed and produced directly from the business application and should be produced from the business objects. Active reports provides a model for directly reporting from a business object a collection of business objects an XML extract from business objects or from a dataset extract from business objects. Crystal reports is somewhat more limited but can also be used to bind to DataSets extracted from business objects. The framework provides a number of mechanisms to extract the appropriate data from the objects in a format useable by all reporting engines (see ReadOnlyDataSetProvider)
      2. Small Business listings (predefined reports): These are relatively short listings of data that viewed and printed with some frequency (usually daily or weekly) e.g. All orders captured yesterday all orders of a particular type captured. These listing will often be primarily viewed on a screen and printed only if required. The number of items in these listing is usually not high. These listing should typically be included in the business application. The application will provide filtering/searching sorting viewing on screen with the possibility to print, email or export on demand. These reports may provide additional flexibility such as allowing the user to select which fields to see in the report and to reorganize the order of the fields. Typically these will be presented as a ReadOnlyGridControl and the resultant set of Business objects will be printed on demand. Once again you can extract the result using the ReadOnlyDataSetProvider and use any standard reporting engine. A short listing is considered to be a listing where the querying and rendering of the result to a user interface can be achieved in a realistic time period for the application. By reporting from the Business objects encapsulation of the datastore is maintained.
      3. Large Business reports (predefined reports): These are reports are similar to Business listings but occur where the user is provided with a set of standard reports that will typically run over many pages. The user is provided with functionality to print these reports without previewing these on a screen and with very limited searching/filtering or sorting capabilities. These were the mainstay of reporting in the mainframe days however the number of this type of report and the usefulness is becoming more and more limited. This type of reporting needs to be performed using a reporting engine that is installed as near to the underlying database as possible. This type of reporting should be developed to use stored procedures that at least provide some encapsulation of the underlying tables and can also provide calculations and business logic.
      4. End user (Ad hoc) reporting: This type of reporting is provided to the relatively sophisticated user who has the ability to develop there own reports based on ad hoc queries. To facilitate this type of reporting a separate reporting database should be developed this reporting database is always denormalised. The data from the transactional database should be transferred to the reporting database at regular intervals. There are many tools and technologies for achieving this. It is outside the scope of this book to provide more details of these technologies but suffice it to say that by providing a single place where all data in the transactional database is transformed to reporting database you achieve a reasonable degree of separation between the transactional system and the reporting system. (See …)
      5. Business Intelligence reporting: Business intelligence reporting is similar to end user reporting but is targeted at the less sophisticated user. Once again the reporting database should be separated from the transactional database and the reporting database is always denormalised to provide optimum reporting capabilities and performance. The denormalisation process and resulting database will usually be appropriate for End User reporting and Business Intelligence reporting and many BI frameworks and tools provide both capabilities.