Business Intelligence for Financial Risk Management

Jordan Prevé

Sponsor: BNP Paribas

Advisors: Prof. Arthur Gerstenfeld

Prof. Donald R. Brown

Prof. Jon P. Abraham

AG WS09

Abstract

This project evaluated the implementation of Business Intelligence (BI) platforms and alternative visualization techniques for risk management in fixed income trading at the sponsor, BNP Paribas. The project examined literature related to fixed income products and risk, the implementation of common BI solutions, analytical tools, and modeling techniques, as well as interviews with key stakeholders at the firm. Using this research, a proposed solution for capture, storage, analysis, and presentation was designed, and a prototype of the solution was implemented as a proof-of-concept. Testing with key users involved in portfolio risk management indicated the prototype was a marked improvement in usability, data access, and intuitive display of the information needed from the system.

Table of Contents

1. Introduction

In matters of finance, numbers mean everything from one trader's exposure to an entire company's risk if the market shifts a fraction of a point. Both empirical data from models and real-world market numbers must be taken into account to fully understand the positions held by traders and the impact of those positions to clients and the firm itself as markets move. Daily snapshots of model-based numbers are used to reconcile expected results with market and portfolio movements by management, while live risk data is used by traders to more accurately handle market movements.

Risk numbers for the fixed income trading group of BNP Paribas in New York are provided as snapshots by web pages collectively called the Risk Viewer, and in live form through a Microsoft Excel add-in called Westminster, a proprietary tool used within BNP Paribas that allows access to the firm's risk and pricing models. The Risk Viewer tool is used by the heads of trading desks and risk managers in the fixed income group to anticipate and reconcile portfolio results over time.

One of the most significant concerns raised at the firm with regards to portfolio risk data was of data quality and assurance. There are no certain indicators as to the quality or integrity of data displayed in a Risk Viewer snapshot, as both of these may vary with respect to the time the source risk data was generated from the models, as well as any errors generated by the underlying systems. Other concerns focused on the inability to display trends or historical tendencies in the data presented in Risk Viewer without first importing large sets of data into a spreadsheet application like Excel, or on the navigation difficulties of Risk Viewer.

The firm has endeavored to provide these services in past efforts through data warehousing and Excel extensions for data retrieval. However, the solution was more complicated to use than the system it augmented, and required information technology resources to maintain it. Requirements such as data integrity and reporting were not directly addressed, and the project subsequently suffered from limited adoption and resource limitations.

With the aforementioned business concerns driving the project, opportunities were examined to streamline and improve the different stages of risk aggregation and display – loading, storing, and displaying the data in an accessible and meaningful manner to users. This was accomplished through user and stakeholder interviews, analysis of similar development efforts in the past, literature on business intelligence (BI) architectures, and feedback on prototypes and design mock-ups built for the needs of the firm. The result of this research, planning, and design was a prototype system which loaded data from legacy records, stored it in an efficient manner, and was then able to flexibly recall it. The resulting system demonstrated the capability of business intelligence to meet the needs of BNP Paribas in assessing portfolio risk in fixed income more easily than before.

2. Background

The specialized nature of finance, technology, and their intersection necessitate the presentation of background material on both subjects. Such material is comprised of the basics of financial instruments and risk, principles of data storage, and means by which to analyze data. What follows is a brief review of the key subjects and literature involved in the planning, design, and implementation of a suitable system for portfolio risk capture and display in fixed income trading.

2.1 Financial Derivatives

Derivatives are financial instruments whose value is based upon other, simpler, instruments or market variables. These instruments or variables usually include prices of commodities, stocks, or bonds, or any other type of asset which is normally traded. Derivatives, however, can also be dependent on other variables, including weather or market indices – anything quantifiable (Hull 1). In fixed-income trading, derivatives such as options and forward contracts are often used to hedge risk in bond position (Hull 10).

2.2 Risk of Financial Derivatives

Risk in financial derivatives is measured in dimensions denoted by Greek letters – the most common being delta (Δ), gamma (Γ), and vega (ν). Vega is not technically a Greek letter, but it is commonly used with the symbol for *nu*. There exist other Greek risk characteristics, however these are outside of the scope of this project and will not be discussed here.

Delta (Δ) risk describes the way an option's price will change as the underlying asset or instrument's price changes, providing a characteristic of the relationship between price of the asset and its derivative. Mathematically, delta for a stock's call option can be defined as:

Equation 2.2.1 $$\Delta = \frac{\partial c}{\partial S}$$

Where *c* is the price of a call option on the asset and *S* is the stock price (Hull 360). Since delta is defined as a first-order mathematical derivative, it is considered a "first order" measurement of risk.

Gamma (Γ) risk describes the rate of change of an portfolio's delta as the underlying asset's price changes. For a given asset, it is given by:

Equation 2.2.2 $$\Gamma = \frac{\partial^2 \Pi}{\partial S^2}$$

Where Π is the portfolio value, and *S* is the price of the underlying asset (Hull 369).

Vega is a measure of portfolio volatility – that is, the rate at which portfolio value will change given the rate of change in volatility of the underlying assets. This rate is defined mathematically as:

Equation 2.2.3 $$\upsilon = \frac{\partial V}{\partial \sigma}$$

Where *V* is the value of the portfolio and σ is the volatility of the asset.

2.3 Data Visualization

Visualization is the process of presenting data in a useful way to interested parties. When considered in a methodical way, visualization efforts take place in seven steps, according to Ben Fry. These are acquiring data, parsing and providing structure, filtering to narrow the scope of the data, mining in order to find patterns or hidden information, representation in the form of a chart or graph, refinement of the representation method, and enabling the user to interact with the data (Fry 1.1.7). Acquisition and parsing of data will be addressed in other parts of this literature review, however filtering, mining, and visual representation are all key elements of a visualization strategy when applied to the types of data involved in this project.

Fry further demonstrates that the seven steps of data visualization are not necessarily sequential, but instead a suggestion, dependent on the needs of the project. A proper visualization project uses the steps as a framework and then reorganizes the working steps to make the most sense for users and the data involved. Emphasis is placed on evaluating the needs of the entire project instead of basing decision making in a piecemeal form, developing each component one at a time (Fry 1.3). Particularly important in considering the project is the matter of when functions are actually needed - the temporal nature of data analysis.

2.4 Neural Networks

One field of data analysis and visualization which has applications in financial forecasting, prediction, and to some degree automated decisionmaking, is that of neural networks. Neural networks are adaptive systems which learn by example using sample data and results to develop predictive or classification models for use in data analysis or forecasting. Whereas a conventional model is part “art” and part scientific reasoning, based upon human understanding of the problem and possible ways to solve it, neural networks and their optimizations rely on human understanding of the data to make it more useful in training the system and to interpret more complex models’ outcomes, and use computer learning to form the model itself, given inputs and expected outputs (in the case of unsupervised learning, only inputs are provided).

Some of the caveats of neural networks, depending on the structure and style used in their implementation, may include divergent solutions or divergent responses to over-training the model, the stochastic nature of some neural network models, and the inherent complexity in optimizing a neural network model for the problem. While optimizations can be automated, current optimization algorithms are computationally intensive and take significant time to develop optimal models, requiring a certain amount of "art" be practiced by humans in establishing these models.

2.5 Operational Databases

Providing structure to data can be performed through the use of a database containing the information acquired through earlier steps of Fry's visualization model (Section 2.3). The subject of databases as they relate to this project can be divided into two sections – relational database implementation, and multidimensional, or OLAP, theory and implementation. These divisions take into account the planned stages of implementation within this project, as well as a progressing level of complexity. These two categories divide databases into operational and analytical roles according to their primary uses.

An operational database is designed to handle records of individual transactions both written and read from it at a rapid rate. Hence, these types of databases are sometimes referred to as transactional databases, and the technology encompassing them as OnLine Transactional Processing (OLTP). These systems tend to require the ability to both read and write to the database quickly, and to ensure data integrity while writing – often from multiple accesses to the same database (Hernandez 4).

In order for transactional databases to meet the requirements of speed and integrity over both read and writes, these databases usually need to be highly normalized. Normalization is a process by which tables in a database are built to reduce or eliminate redundant data through the use of table relationships, hence the term "relational database" also applying to these systems. There are varying levels of normalization, applied as algorithms known as Normal Forms.

The application of Normal Forms aims to make the database more robust against what are known as modification anomalies, which can affect the integrity of the data. There are three sorts of significant anomalies encountered in denormalized relational databases – insert, delete, and update, which roughly correspond to the most common actions performed in a transactional database. Insert anomalies create constraints in the data set that do not accurately reflect constraints in the business case – for instance in a table containing zoo animals and their paddocks (assuming no null-valued columns), there cannot be a new paddock without having an animal. Delete anomalies operate in the opposite manner with the same constraints as insert anomalies – if you were removing the last animal in a paddock you would delete the paddock as well, or if deleting a paddock you would have to delete all the animals within it. Update anomalies take place when records belonging to the same entity may have some modifications made and others left alone.

There are two significant Normal Forms which may be considered in data warehousing design within this project – First Normal Form (1NF) and Third Normal Form (3NF). Additionally, some emphasis will be placed on Domain-Key Normal Form as well (DKNF), which emphasizes sets of acceptable values in determining data to normalize. First Normal Form establishes that for every entry in a table, you cannot have multiple values for the same attribute – that is, having fields Value_1, Value_2, etc. all of which draw from the same set of values, for a given record. Third Normal Form enforces not only the record independence and subject pertinence required of 2NF, but goes on to require that additional fields in a table record be attributes of the primary key of the record. That is, if you have four fields in a record, A through D, and a combination of A and B forms a primary key for the record, C and D must both be attributes of that primary key (they must describe it somehow) (Hernandez BC-10).

Database normalization works by using a principle known as relations, and as mentioned earlier relations are the namesake of the relational database on which OLTP systems are based. In relational databases, multiple tables are given logical “relations” at the query and retrieval level of the database by associating records on fields which contain identifying attributes (Hernandez 12).

ANIMAL_ID	**ANIMAL_NAME**	**WEIGHT_LB**	**TYPE_ID**
1	bob	310	tortoise
2	jane	480	tiger
3	jonas	440	tiger

TYPE_ID	**DIET**
Peacock	birdfood
Tiger	tourists
Tortoise	vegetables

Table 2.5.1 – Example of Database Relationships in a Zoo

In the case of Table 2.5.1, the relationship between the two tables is on the `TYPE_ID` field, such that finding the diet of any animal is a logical join between the two tables on `TYPE_ID`. So querying for `ANIMAL_ID=3` in the results of a join between the two tables would result in “tourists”. As a note, the example of Table 2.5.1 uses `ANIMAL_ID` as the primary key. It is also possible in the example to use `TYPE_ID` and `ANIMAL_NAME` together to form a primary key, though there is the risk of two tigers named Jonas, for instance, if it was a large zoo.

2.6 Analytical Databases

Compared to operational databases that specialize in transactional processes, analytical databases are geared towards multidimensional evaluation of complex data sets. While not all implementations of analytical databases support multidimensionality at the lowest layer of the system, all analytical databases share the same intention of improving accessibility and speed for analytical operations across large sets of data. The technology of analytical databases is called Online Analytical Processing (OLAP, as opposed to Transactional Processing) (Melomed et al 16).

The concept of n-dimensionality (beyond n=3), while difficult to visualize in one's mind, is a relatively simple way to describe the sort of data sets usually under complex analysis. A simple way of understanding 4-dimensional sets, however, is to visualize 3-dimensional cross-sections of the data set along the time axis, such that each point in time is a normally represented cube. A common business requirement is to be able to analyze sales or some similar data over time periods, offering historical trending and the opportunity for forecast models, which takes [usually] two-dimensional data and extends it into three-dimensional space with a time axis. In structuring an n-dimensional cube (also known as an "n-cube" or "hypercube") for analytical databases, fields in the source data are divided into dimensions and measures (or facts). Put simply, measures are what a user would intend to analyze, while dimensions are what define the scope of the measures (Melomed et al 69).

For example, an analysis cube for the zoo previously discussed (see Table 2.5.1 if this means nothing to you) might contain information on animals, locations in the zoo, and animals' vital statistics gathered by the staff veterinarians. So the dimensions would be criteria like species, paddock, the source of the animal (other zoo, captive born, wild capture, etc.), and dietary category. The measures, or facts, could be the animal's name, and the vital statistics of the animal (medical information). One last dimension is needed – time helps in this case, since it could answer some questions that information from a single point in time would not be able to. While very few of the measures mentioned above are aggregatable, if any, an analysis model of the zoo's animals would be able to provide information to answer questions like "How are we at maintaining the health of our vegetarian animals sourced from Africa over the last 5 years?" or "We changed our lettuce supplier this year, and want to make sure our tortoises are still well-fed with the new vegetables – are they?". Because an analysis cube brings together all of this information in an immediately intuitive and accessible form, even with five dimensions a user with limited experience could answer these questions with access to the data set. There also exists greater potential for use of predictive models, such as simple statistical models or as advanced as neural networks, to evaluate the entire data set and find trends or patterns.

One of the significant differences, however, between conventional relational databases and either those with multidimensional layers built on top of them, or ground-up multidimensional databases, is that the data is very often denormalized in the underlying tables. While in some cases cubes can form relational links to one another, in larger cubes with denormalized tables, the various dimensions take the place of a Normal Form table structure, separating the data out and limiting redundancy in the final form presented to the user.

2.7 Kaizen and Lean

Kaizen, Japanese for continuous improvement, encompasses a field of efficiency planning and manufacturing engineering which focuses on processes within an organization, ideally with employee participation. Usually Kaizen philosophies are encouraged as part of a Lean implementation, where a firm tries to eliminate wastes in common manufacturing processes in the form of both literal wastes (rework and packaging, for instance) and figurative yet still critical ones (time sinks and inappropriate skill allocations, for example). Recently, Lean and Kaizen continuous improvement have been applied to business processes, software development, and virtually any value-added element of a firm. The idea is the same – to reduce rework, eliminate wastes, and to improve quality and productivity.

Information technology (IT) tends to assume a supporting role to the profit centers of a firm, where they might be charged with writing proprietary pricing software for traders at a bank, or architecting an electronic records solution for staff at a hospital. They create the tools which employees need to more efficiently generate revenue for the company. In this way, the function of technology workers in the modern financial firm is very similar to that of mechanics or manufacturing engineers at a manufacturer. While there is no clearly defined production line, the same wastes, costs, and impact can be seen as when process analyses are performed on manufactured goods.

3. Methodology

The goal of the project, through careful analysis and evaluation of business needs and available technology solutions, was to develop a prototype solution for capture and display of portfolio risk data. The prototype and resulting recommendations took into consideration the business needs of the firm throughout the stages of extracting and loading risk data, storing it, and presenting it to users at different levels of the organization. Also included in the research leading to a solution and recommendations was the analysis of user impact, total cost of ownership of available solutions to the business, consideration of maintainability and extendibility, and continuous feedback throughout development from key stakeholders in the business.

3.1 User Interviews

Critical to the design of an effective solution to the firm's risk display and management needs was to understand the specific requirements of the existing system's users. It was important to not only to gain an idea of how one person might use the system, but what they use it for, and also how users at different levels of the organization use the system. For instance, the difference between solutions geared towards individual fixed income traders, and the needs of the organization's regional Chief Risk Officer.

3.2 Total Cost of Ownership (TCO) Analysis

In considering the project's prototype deliverable as an enterprise software solution, it was critical to evaluate the options available to the firm based not only on how well the solutions meet the users' needs, but on the cost incurred by the enterprise in deploying and supporting the application(s). Total Cost of Ownership, or TCO, is a metric which combined the purchase cost, support costs, relative hardware cost, and relative costs to deploy the solution in terms of personhours (Solution Matrix). A comparison of TCO allowed costs to be compared among both Free or Open-Source Software (FOSS) and commercial software.

3.3 Lean Evaluation

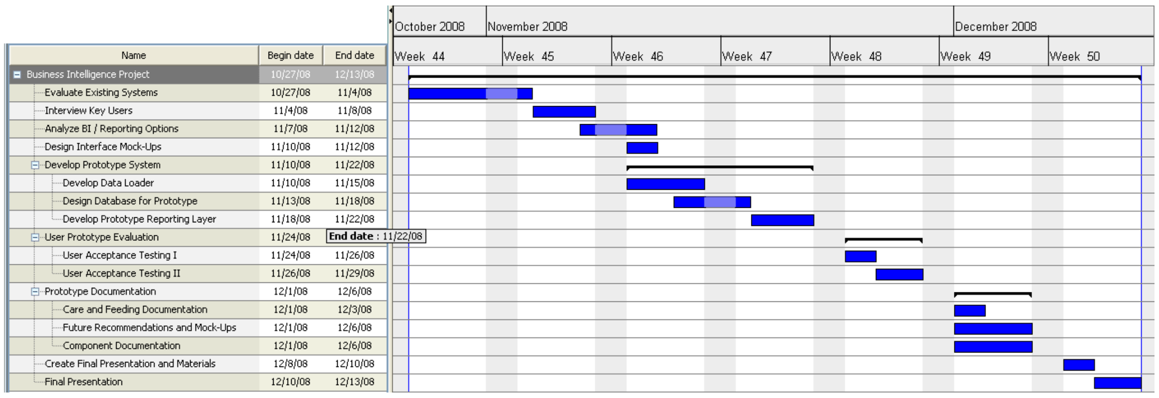
Using Lean and 5S as a foundation for business process evaluation, the project considered the way in which users, administrators, and developers work with the existing risk snapshot platform in fixed income. These manufacturing process improvement methodologies were adapted to use in a financial information technology (IT) context by translating many parts in Lean to their applicable IT equivalents, as discussed in the Literature Review. Recommendations were made based upon the principles of these process improvement schemes, and such principles were also taken into account in the prototyping of recommended solutions.

3.3 Mock-up and Prototyping

Designs and mock-ups of analysis and reporting environments were created in order to gauge user response to development ideas. These mock-ups allowed quick determination of whether key users would find a feature or layout useful in their activities. This saved time in development of the prototype by keeping significant design changes in the early stages of prototyping.

3.4 Timeline

The first week of the project was spent getting acquainted with the systems for risk management and display currently in place at the firm, along with necessary on-boarding tasks. The second and third week focused on interviews with users of the current systems and determining the best methods to source the data needed for effective risk presentation. Between the third and fourth weeks, time was spent first analyzing available options that met the needs identified with users, and then in the remaining time developing a working prototype. Additionally, during the fifth and sixth weeks, the prototype and design mock-ups were reviewed by users and improved based upon their feedback, and transition documentation was written up for effective transfer of the prototype and analysis documentation at BNP Paribas.



Name	Begin date	End date
Business Intelligence Project	10/27/08	12/13/08
Evaluate Existing Systems	10/27/08	11/4/08
Interview Key Users	11/4/08	11/8/08
Analyze BI / Reporting Options	11/7/08	11/12/08
Design Interface Mock-Ups	11/10/08	11/12/08
Develop Prototype System	11/10/08	11/22/08
Develop Data Loader	11/10/08	11/15/08
Design Database for Prototype	11/13/08	11/18/08
Develop Prototype Reporting Layer	11/18/08	11/22/08
User Prototype Evaluation	11/24/08	**End date** : 11/22/08
User Acceptance Testing I	11/24/08	11/26/08
User Acceptance Testing II	11/26/08	11/29/08
Prototype Documentation	12/1/08	12/6/08
Care and Feeding Documentation	12/1/08	12/3/08
Future Recommendations and Mock-Ups	12/1/08	12/6/08
Component Documentation	12/1/08	12/6/08
Create Final Presentation and Materials	12/8/08	12/10/08
Final Presentation	12/10/08	12/13/08

Figure 3.4.1 – Gantt Chart for Proposed Timeline

4. Findings and Analysis

In the initial weeks of research at BNP Paribas, the current system for displaying risk snapshot data was evaluated both as a business process and as a component of the firm's information technology infrastructure. As a business process, opportunities for streamlining the user experience were identified for improvement along with any bottlenecks to productivity. As an information system, compatibility with existing technology processes was examined in conjunction with normal enterprise considerations such as maintainability.

4.1 Risk Viewer

The current system in place within the fixed income group for risk capture and display is a suite called Risk Viewer, made up of a number of Microsoft Excel spreadsheets, batch scripts, and proprietary back-office functionality implemented in C and Visual Basic (VB).

The risk snapshot-generating parts of Risk Viewer for fixed income portfolios currently run once a day. Batch files control execution of the Excel workbooks, which generate both the HTML navigation structure and risk data through add-ins and custom scripts. Process visibility is provided by log files and email notifications generated by both the batch files and VB scripts within the workbooks, indicating any error triggers encountered during the generation process. A log of success or failure, along with any error codes, is provided in the form of HTML pages within the navigation structure of Risk Viewer.

Visibility at multiple levels of the portfolio hierarchy and of different shifts of measures or tenors are provided by Risk Viewer generating new HTML pages reflecting these various permutations and aggregations thereof. This process is resource-intensive and cannot be run efficiently on commodity hardware, and also requires a large amount of space for daily data – each day requires approximately 11,000 HTML files and one gigabyte of storage space to keep a complete snapshot.

4.2 User and Administrator Feedback

The Risk Viewer suite, while providing the data required for day to day risk management operations of the fixed income group, has grown over time to be more complex than originally designed. This is from a combination of changing business structure and varying user needs or requests, and has resulted in increasing difficulty of maintenance and to adapt to further changes in user or company needs.

A constant observation from users of the current system was that while Risk Viewer presents its data as tables in HTML files, they tend to need the data in Excel for calculations or out of convenience.

4.3 Lean Analysis

Investigating the way users and administrators of Risk Viewer interact with both the "application" and the data therein, a few things become clear. One is that the majority of users, as is the case with many financial users, work with their numerical data almost exclusively in Excel. While Risk Viewer provides a formatted, visible answer to questions about single points of risk data, it does not provide the data in a format easily moved to the user's spreadsheet in the exact format they need for their normal approach to number crunching.

This additional layer of complexity required for the average user of the Risk Viewer system to look at data in an environment familiar to them is a good example of waste in a common process. While the system meets the requirement of displaying snapshot data for a given point in time, it does not address user requirements of being able to directly access this information through their usual toolset. This extra step or series of steps can hinder adoption and satisfaction among enterprise software users.

While moving data that is directly available in Risk Viewer to Excel or other applications takes the extra steps detailed above, finding data over time requires even more steps and a user’s time. To get the value of a portfolio’s delta(s) over time, one must iterate through a page of data for each date, taking additional time and further compounding the aforementioned issue of moving data into the user’s preferred analysis toolkit (Excel, etc.).

In order to eliminate or reduce the wastes present in the process of retrieving snapshot risk data from Risk Viewer, the replacement implementation would have to provide both flexible destinations of data and flexible rendering of data to that source. The time of users, especially those at the higher levels targeted by snapshot risk solutions, is critically important to the firm. Providing these users with the ability to re-form data to answer most questions about traders’ risk reduces the amount of time needed to pull the data, compared to manually extracting it from Risk Viewer pages or working with complex proprietary tools to pull up historic data from legacy or back-office applications.

5. Results

This project set out to both investigate the current solution in place for snapshot risk data and to develop a proof of concept of an alternative which took into account the findings in a Lean-based analysis. The tangible result of the project was the implementation of a business intelligence (BI) stack prototype in three layers – storage, where the data was kept in a specialized analytical database; extraction, testing, and loading, where data was pulled from sources and set-up for insertion into the storage layer; and presentation, where information was taken from the storage layer and given to users in a useful and intuitive way.

5.1 Storage – Application of Analytics Databases

The storage of risk data was accomplished through the use of an analytics database. This decision was made based upon the time-related nature of the data being stored and the dimensionality of the data – the structure of portfolio-level risk lent itself to a multi-dimensional storage solution because of the shifts involved in higher-order risk measures. After careful consideration of multiple OLAP database providers, including free / open source options, and evaluation of the tools available within BNP Paribas to test, the Analysis Services platform on Microsoft SQL Server 2005 was selected. The advantages in the context of this project were rapid development time, ready availability of an environment in which to develop this proof-of-concept, and relatively low cost compared to other analytics databases capable of dealing with extremely large data sets such as Oracle solutions.

The cube itself was configured with two measures, or facts, and five dimensions. These dimensions were Time, Portfolio [Number], Tenor, Product (which described the Delta and Gamma measures) and [Gamma] Shift, which also described Gamma. The Portfolio dimension was given additional attributes to describe a given portfolio's position in the organizational hierarchy and its trading currency. To support a non-trivial sorting scheme, an integer key was added to the Tenor dimension on which to sort the stored tenors. This allowed business logic to define the sorting of tenors through external logic, as opposed to a simple scheme such as alphabetical or numerical sorting.

5.2 ETL – Automation of Data Extraction and Loading

The proof-of-concept BI system was designed to live on top of existing Risk Viewer processes. Thus, it would extract data from the final HTML product of Risk Viewer and prepare it for import to the analytics provider, Microsoft Analysis Services. To interpret the Excel HTML exports, a Python script using the BeautifulSoup parser was written to find the risk values and export them to comma-separated-value (CSV) files. These CSV files are a common means of transferring tabular data across systems, and tend to be the most reliable way to migrate data between sources.

The extraction, transformation, and loading (ETL) tool used within SQL Server 2005 was Integration Services. Using this built-in ETL facility, the CSV files created by the Python translation script are imported, denormalized, and stored in cache files for each measure in the cube. The cache files are then used to load the dimensions with their unique values, and subsequently load the cube itself with data points.

The Python component of the acquisition stage of the prototype was used to reduce time to develop, and would ideally be replaced with a more direct solution to the same source systems which Risk Viewer uses in future migration plans.

5.3 Presentation

The presentation layer of the prototyped business intelligence stack was comprised of a direct query component, rapid-development applications which provided connectivity to the database, and left the door open to server-side reporting. While they may seem like redundant solutions to a problem of how to pull data from a source system, different tiers of capability and interest exist among users, and each of these caters to one such tier.

5.3.1 Direct Query Interface

To more advanced users of Risk Viewer, the system presents more constraints to how they use it than it provides useful information. While the information is the same as they would get through other means, not having it the exact way they would like to see it proves a hindrance to their acceptance of the tool in any significant capacity. In most other applications in the firm, these users' needs are met by providing direct query access to a source system, often by means of SQL (structured query language, commonly used in relational databases).

The storage provider for the prototype, Analysis Services, supports a query interface called multidimensional expressions (MDX). Using components freely available in most Windows installations and through other means in alternative operating systems, developers can integrate MDX query handling into Excel or other solutions that need access to information in Analysis Services a user-customizable way.

5.3.2 Standalone Applications

For users in the middle tier – those who require greater control over the data but do not require the advanced features of an exposed query language – there are rapid-development toolkits available for Windows and rich Web platforms which allow developers to quickly roll out PivotChart-like access to OLAP data sources like Analysis Services. One such toolkit is DevExpress, which was used in this project to make a small Windows application which displayed the Analysis Services cube data as a PivotChart and an associated scatterplot. While not giving the absolute control over output like MDX querying would, it allows users to reformat and filter data if they are familiar enough with the data being used to make sense of the options, aggregations, and tools available.

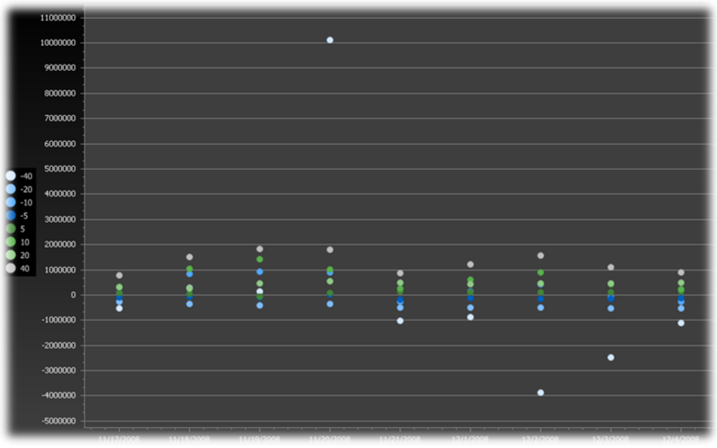


Figure 5.3.2.1 – Sample Scatterplot from Windows Application

5.3.3 Server-Side Reporting

For users who need the data but do not have the understanding of, or time for, more complex options, there is the third tier. This tier primarily provides access to data that has been designed to fit the needs of business users ahead of time, either by developers with domain knowledge or power users within the business.

Server-side reporting generally falls into two categories – one is developer-designed, where the information technology team for an office would meet with users and determine what they want for reports, and then design templates to be used in generating reports with the source data. The other method is called "ad-hoc reporting", where users can take advantage of (usually web-based) tools to build their own reports through an interface similar to a PivotTable.

The other advantage of server-side reporting is the ability to schedule a report for publication on the company intranet or distribution through email. This regular, up-to-date dissemination of information can help to eliminate the time wastes made by having a user request reports daily, especially those requiring complex aggregations or other operations and thus having long lead times in generation.

6. Conclusion

This project's goal was to evaluate and implement business intelligence (BI) for BNP Paribas to handle snapshot risk data for their portfolios. In the course of seven weeks, multiple ETL, storage, and presentation components were evaluated on their merits and availability to determine the best possible prototype platform for the firm's needs. A single BI stack, the tools for which were present on instances of Microsoft SQL Server 2005 already deployed in the office, was selected for its robustness, ease of development, and ready availability.

6.1 Improvement

When evaluated against the existing tools for displaying snapshot risk, the prototype system offers the same data visibility at the portfolio level, though it provides more intuitive access to different views of the data. Additionally, the new system can simultaneously display different points in time of these data views, and through creative methods of handling data, could even display full views over multiple time periods (like a 3-dimensional scatterplot, for instance).

From a technology perspective, improvements were in potential maintainability of a system built with components from the prototype. The technologies and their relative connections are shown in Figure 6.1.1 for the original system.

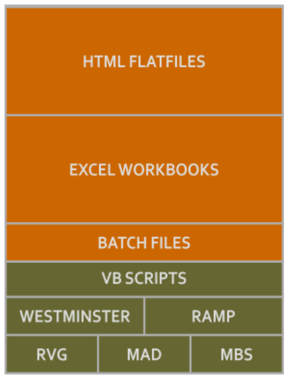


Figure 6.1.1 – Current System

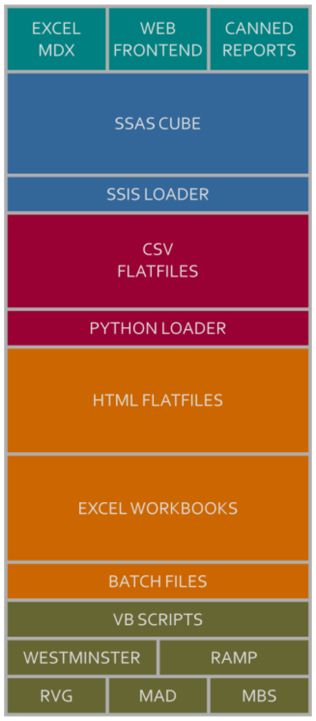


Figure 6.1.2 – Prototype System

While it is clear from both Figure 6.1.1 and 6.1.2 that there are more elements to the prototype, this was a consequence of the time constraints and design requirements of the system. That is, there was no need to reinvent the wheel – Risk Viewer already had the data required to populate the prototype and prove that the new technologies worked. Risk Viewer in the case of the prototype served as the "extraction" component and provided information from source systems at the firm. Additional scripting and code external to the prototype's BI stack was required to make the conversion from the Risk Viewer output to a format which the Integration Services loader could easily understand.

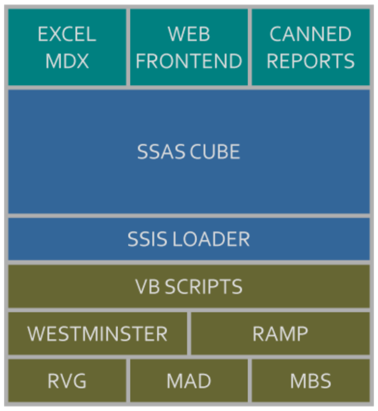


Figure 6.1.3

By removing the Risk Viewer and adaptor components from the technology stack (Figure 6.1.3), the process could be potentially simplified to just the back-end systems, the newly developed BI stack, and a select few presentation-layer components. Instead of using Risk Viewer as the extraction layer, custom components would be written within the ETL tool – in this case, Integration Services in the Microsoft BI stack.

6.2 Impact

Following user demonstrations and experience with the prototype BI solution, it was gathered that the analytics database, when combined with new automated loading tools and presentation techniques, had the potential to better fit user and company needs than existing systems. Such a potential fit is based upon further development of the solution and its components in a user-centric process, largely in concert with the recommendations set forth in this paper.

Key to further development is the previously mentioned (in Section 6.1) adjustment to the structure of the system. The current design allows for a migration in parts by placing the new storage and presentation layers on top of Risk Viewer and automating the import process. Over time and as resources become available, the Risk Viewer layers could be removed and the functionality added to the new system's ETL layer.

7. Recommendations

The project's recommendations are intended to improve the potential fit of a production system based upon the prototype solution discussed in the Results and Conclusion sections of this report. The reasoning behind providing these detailed transition plans and "next steps" is to establish the usefulness and application of the prototype designed in the course of this project.

7.1 Direct Transition Path

In the immediate transition to a production-ready version of an analytics database and other supporting business intelligence (BI) elements, certain steps must be taken. These steps are designed to prepare the full BI solution for users and IT support, and are made in consideration of ordinary development limitations and capabilities of the selected prototype components

7.1.1 Implementing Multidimensional Storage

The first step towards a production BI system is implementing the storage layer, the analytics database which contains all of the risk numbers, descriptions of portfolios, and other key details. While the current storage layer, SQL Server 2005 Analysis Services, performs adequate tasks for demonstration purposes, the design of the cube itself must be extended further to be useful in real-world scenarios.

First, the cube design would need to include all of the currently available risk measures, in the proper groups, that are needed by risk management and other elements of the firm requiring access to the data. This means not only the different greek risks, but also the various computations of risk by different proprietary models. The prototype only included two risks (Δ, γ) on portfolios, and only used a single model for the data. Ideally, a production level system would offer all three risks (Δ, γ, ν) as measure groups comprised of the different models' calculations of risk.

Additionally, the dimensions provided must fully describe their data points through the use of attributes. For portfolios, including information on trader(s), hierarchy data, and the portfolio's trading desk (or the trading desks of the traders) are all valuable pieces of information. Other dimensions such as the tenor of a given product, or the product dimension itself, may also require further attributes be added as the business needs are better determined through repeated demonstrations of new features or capabilities. The ability for additional information to be provided in-context about dimensions' data slices within interfaces should not be overlooked, particularly with regards to PivotChart-like constructs with tabular layouts.

7.1.2 ETL Design and Migration

One of the greatest potential strengths of the prototype when pushed into production is the ability to move loading scripts and functionality currently contained within spreadsheet math and macros into SQL Server Integration Services.

Creating custom components within Integration Services would allow data to be pulled directly from proprietary data systems into scripts and transformations before being added to the analytics database. These components could later be re-used without any changes to the underlying code. The same tasks which are currently run through the spreadsheets and other methods could be moved into these components and other parts of Integration Services to run server-side, with less overhead than interpreting the HTML files from the current Risk Viewer software.

7.1.3 Reporting and Data Access

Customizing the way users interact with the BI system is key to ensuring adoption of the new platform. While initial discussions with users revealed an interest in providing simple, direct interfaces to the system (such as the direct query interface using MDX), there was significant interest in more user-friendly approaches in the final demonstrations. Additionally, certain presentations of data were the most common in any of these scenarios.

In a transition strategy, for instance, it would be helpful to reproduce a version of the current Risk Viewer reports through a server-side reporting interface such as SQL Server Reporting Services and have it publish to a company intranet site. This would allow for a near-seamless transition from the current system, and allow for the new cube-backed storage to provide the information for the reports earlier in the transition. Users may not even notice the new system taking over the publishing of the reports from the old Excel-backed process, and will be able to see test reports or other presentation schemes run with live data used to compile the legacy reports.

7.2 Extending the Concept

Up to this point, what has been presented is a straightforward transition strategy to improve the user experience in dealing with fixed income risk in a known and effective manner. There are also a number of extensions to this strategy requiring further research into execution, user adoption, and implementation time; however they represent new directions which could lead to further uses for business intelligence platforms in the enterprise, as well as better solutions delivered to the users. These extensions fit into two categories – those which build upon the current direction of the proposed system, and those which apply to different uses for the principal components of it.

7.2.1 Room for Improvement

In terms of extending the proposed project, a Risk Viewer replacement, a few possibilities are immediately evident – including near-real-time data analysis, more granular data access, and novel approaches for analyzing the data. These would improve the experience of fixed income users and provide a proof of concept for future implementations using these features.

Ordinarily, analytics databases capture information at a single, usually infrequent, point in time and process it into a cube for access. This, by design, presents older data to the user as the “one truth” of the information available to the firm – what the analytics database provides is ideally the most accurate available. However, if the analytics database contains risk or position data, having access to historic data and near-real-time simultaneously allows a risk manager or other stakeholder to see a more updated perspective on their portfolios or trades. This improved granularity in the time dimension, whether it’s by “floating” a real-time set of values or providing a higher frequency of data points, could help key stakeholders in managing risk.

The Microsoft-based system implemented as a prototype in this project is capable of supporting such a design with proper consideration of business needs and the accuracy of the data in (near) real-time aggregations and analysis. However, it would also be important to clearly define the intended uses of a real-time BI solution to maintain a crisp and useful user experience within the system. Just like a relational database, the analytics database used as part of the BI suite could be applied to any number of problems – it is critical to clearly define the problem to be solved to avoid a confusing tool which tries to do everything at once.

7.2.2 Further Applications

Basic risk measures are not the only data which could be useful within a business intelligence solution for a financial firm such as BNP Paribas. Risk data and market data can be combined to yield expected profit and loss reports, for example, to be compared against actual profit or loss on a portfolio. Data mining techniques could be applied to this profit and loss data to better interpret risk positions over time and in changing market conditions.

7.3 Applying Agile Development

Any system which intends to replace or supersede Risk Viewer at BNP Paribas is in a unique position to act as a proving ground for new and innovative approaches to software development within the firm. The small current group of advanced users, combined with broad possibilities for new applications, lends itself to development strategies such as Agile, where user feedback at all points in the software lifecycle are key to the success of the project. Users in the Risk Viewer replacement project see themselves as valuable stakeholders and in a position to give useful feedback, helping to drive changes and improvements relevant to their needs.

The importance of adopting agile development strategies in the Risk Viewer replacement is evident in the replacement itself. Due to a combination of technology not being readily available and ordinary enterprise technology strategies, users many years later indicated that the system did not fully meet their day-to-day needs in risk management. Agile efforts would tend to catch these shortcomings earlier in the development process, ideally in the first prototype stage, such as the one fulfilled by this project, and documented through process standardization such as CMMi.

In the end, the biggest benefit to Agile and its brethren is not about creating better software – it's about creating better, more functional relationships between business and technology resources in a firm. Historically, technology resources lament that users never "ask the right questions" or "ask for the right features". Bringing users, developers, and project management all into the same development cycle allows them to see first-hand what they should be asking about or asking for in software projects. While some elements may be beyond the scope of the project at hand, such as handling fixed income risk data in this case, often the questions from users will be relevant and suggest small improvements which can yield significant improvements in user adoption and their experience.

7.4 On the Subject of Open Source

While this project used commercially produced and supported software to create the prototype of a business intelligence platform to meet the needs of BNP Paribas, it is necessary to consider the open source projects currently offering (and further developing) components of business intelligence solutions. Both community and vendor-sponsored open source platforms are available as of the writing of this report, and were considered for use in this prototype. However, in the interests of development time – yielding a working example of the software was more important than optimizing the TCO – a commercial, off-the-shelf solution was selected to meet the interim needs of the firm.

However, this interim decision does not consider future development efforts in the relevant open source communities or the possibility that BNP Paribas would not require the full gamut of options or features present in the Microsoft offering used in this project. Furthermore, an option available in enterprise development is the corporate sponsorship of open-source development efforts through purpose-established non-profit foundations or other means. These organizations offer further flexibility in selecting software components of an effective business intelligence stack in the future, by spearheading development efforts on custom platforms, such as an analytics stack which would be customized towards financial firms.

8. References

Ballard, Chuck, et al. Dimensional Modeling: In a Business Intelligence Environment.IBM Redbooks, 2006.

Fry, Ben. Visualizing Data. 1st ed. Sebastopol: O'Reilly Media, 2008.

Database Design for Mere Mortals. 2nd ed. Boston: Addison Wesley Professional, 2003.

Humphries, Mark, et al. Data Warehousing: Architecture and Implementation. Upper Saddle River: Prentice Hall, 1999.

Melomed, Edward, et al. Microsoft SQL Server 2005 Analysis Services. Sams Publishing, 2006.

Segaran, Toby. Programming Collective Intelligence. 1st ed. Sebastopol: O'Reilly Media, 2007.

Solution Matrix. "Total Cost of Ownership Analysis (TCO)." Solution Matrix. 3 November 2008 <http://www.solutionmatrix.com/total-cost-of-ownership.html>.

Tate, Kevin. Sustainable Software Development : An Agile Perspective. Upper Saddle River: Pearson Education 2005.