Risk Reporting Techniques

A Major Qualifying Project

Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirement for the
Degree of Bachelor of Science

Submitted on
Friday, 1/13/2012

Project Sponsor: **Bank of America – Merrill Lynch**

Submitted to:

On-Site Liaisons: **Ron Toam** (Vice President; Capital Markets Front Office)

Project Advisors: **Professor Daniel Dougherty**, Department of Computer Science, WPI
Professor Jon Abraham, Department of Mathematical Sciences, WPI

Submitted By: **Fernando Martell**, Computer Science
Yunwen Sun, Mathematical Science

Abstract

The purpose of this project was to create an application that can help the Global Credit Mortgage Distressed team at Bank of America with generating risk reports in the new firm-wide framework called Quartz. After completing an in-depth analysis on deals models, how data was being stored in Quartz, and how fields were mapped between Quartz and an existing database called Vienna we developed an application that extracts information from Quartz, and replicates risk reports from a current application called Realm. The creation of this application in our project is an important step in the process of firm-wide migration of applications to Quartz.

Acknowledgements

We would like to thank Ron Toam, Jin Lee, Michael San Jose and Hyun Lee for their support and guidance throughout the competition of this project. In addition, we also want to thank Ekaterina Ratcheva, Charis Yang and the other members of the Global Credit Products Technology division. We also wish to express our gratitude to professors Daniel Dougherty, Jon Abraham, Justin Wang and Arthur Gerstenfeld for their comments and suggestions that ultimately made our project a success.

Authorship

Section	Author(s)
Abstract	All
Acknowledgements	Fernando
Executive Summary	Yunwen
Introduction	All
Historic Background of Bank of America	Yunwen
Analysis & Results	…
Understanding Quartz	Yunwen
Credit Default Swap Products	Yunwen
The Application	Fernando
The Objects	Fernando
Conclusion	Fernando
Glossary	Yunwen
Appendix A	Fernando

Table of Contents

Table of Figures

Table of Equations

Executive Summary

The Global Credit Mortgage Distressed team at Bank of America is in the process of firm-wide migration to a new infrastructure framework called *Quartz*. This involves moving trade data, analytics, and a variety of tools used by various groups into Quartz. With the trade data and analytics in Quartz the next step is to develop tools to view risk – commonly called "Risk Reporting". Our end product was an application that could replicate trade reports on the new platform - Quartz. By creating this application we provided the team with a tool to replicate and view risk reports based on data stored in the Quartz database and to write risk reports to a spreadsheet or a temporary database.

The project consisted of three main parts:

- Understanding the Credit Default Swap trade data – the data in the sample report we replicated
- Analyzing Quartz Deal Models and how different these are from deal models used in the current system and identifying how data was stored in Quartz
- Developing the Risk Reporter Application

First, we researched and learned about the Credit Default Swap (CDS) valuation, including modeling credit using a reduced form model, bootstrapping hazard rates, and valuing Default Leg and Premium Leg of a CDS. We gave a presentation on CDS Valuation to help developers in the team to understand the valuation.

Second, we created a mapping spreadsheet to map the fields between *Sandra* (the database in Quartz) and *Vienna* (the current system) after comparing the Quartz Deal

Model and the Vienna Trade Model. To accomplish this, we used applications such as Database Browsers and QzDev, tools that were already implemented in Quartz.

Finally, we created the Risk Reporter Application that could view and replicate a trade risk report in Quartz and write the replicated reports to a spreadsheet or a temporary database. The final product was a tool to replicate and view trade risk reports on the new infrastructure platform.

As the company shifts into Quartz, the implementation of this application will provide developers with an example and the needed documentation on how the migration process can be accomplished.

1. Introduction

Bank of America, like any other financial institution, relies on their technology groups to develop applications that will support their complex trading businesses. Currently there are over 100 databases, more than 40 trading systems, 20 plus risk systems, and multiple back office systems within Bank of America. Despite the large number of tools, the current infrastructure does not allow proper integration between them. This harmed efficiency as employees had to manually translate and copy data between tools.

Applications in the current system, Vienna, have inconsistent data sources, data lineage, as well as write and read data from over 100 database systems. A large amount of time is wasted in the process of salvaging data from one system or another, time that could be otherwise used in analyzing it. In order to address this problem, Bank of America is working on a new infrastructure framework called Quartz. It is a consistent platform with unified cross-asset front to back pricing and risk management and sales/trading. All the applications in Quartz will have a common data source called Sandra, an object oriented database. Sandra will be the one system that contains all trade, market, analytics and risk metric data across all assets. Front to Back office developers will use the same code base, deal model and coding standards. Everyone will get to use and see data real-time; an update will propagate to everyone immediately. The migration to Quartz will provide a resilient and fault tolerant framework where developers and traders can manipulate data more efficiently.

The main goal of the project was to explore various solutions to reproduce risk reports based on positions and analytics in Sandra and Quartz from sample reports in

Realm, the current system. In order to accomplish this, we closely worked with business analysts and developers at Bank of America in order to analyze the different ways in which we could integrate the ability to generate risk reports into the framework. Additionally, it was also necessary to understand the differences between the deal models of Quartz and Vienna in order to understand how instruments are being stored in each system. Another goal was to identify how market data is being stored in Quartz and provide the BA team with documentation for future reference. The information in Quartz is being stored in objects; by identifying the objects that contained data relevant to the report, we could narrow down the object's attributes for future implementation.

Risk Reports provide multiple sensitivities and details on the different trades in a portfolio. Bank of America employees use these reports to assess the business risk for an entire portfolio or a single trade. Our implementation focused on a single risk report, the trade credit details by trade. The report is generated by users on demand and it provides information about the trade details and risk measures.

Credit Default Swap is the product in this trade risk report, it is also the major product the employee team at Bank of America is specializing in. One of the applications that The Global Credit Mortgage Distressed teams is seeking to migrate is called Realm. This application is in charge of producing various reports that are used throughout the company. For example, one of the projects the team worked on determined how much capital was needed to meet Basel II. This set of recommendations on banking laws and regulations was issued by the Basel Committee in 2004 to create an international standard for banking supervisors (Basel II). Realm extracts most of the information in the reports from the Results Database in order to make Basel II calculations and perform other tasks.

This database is nourished by many applications and contains data on trades and their sensitivities among other data. Because the data needed to generate the reports in Realm is coming from many places, part of the migration process is to understand how data is being stored in Quartz. However, most of the team members are not familiar with the new infrastructure.

2. Historic Background on Bank of America

Bank of America is one of the world's largest financial institutions that provide banking, investing, asset management, and other financial and risk-management products and services to individual consumers, small and middle market businesses as well as large corporations. It has approximately 57 million consumer and small business relationships with approximately 5900 retail banking offices, more than 18000 ATMs and more than 29 million active online banking users. (Bank of America Merrill Lynch)

Bank of America was formed in late 1920s as the result of a merger between the San Francisco based Bank of Italy and the Bank of America, whose business interests were solely based in Angeles at that time. The expanded back underwent a rebranding process and was renamed as BankAmerica. (Bank of America)

Due to a number of bad loans in Latin America between 1986 and 1987, Bank America went through significant losses which led to a decrease in the value of BankAmerica stock. BankAmerica had to offload parts of its business in order to avoid of being taken over by a number of interested financial institutions, which further decreased the value of its stock. In the end, BankAmerica avoided the takeover and even survived the stock market crash of 1987, and start growing its fortune around 1992. (Bank of America)

The acquisition of Merrill Lynch &Co. Inc., the world's largest and mostly wide recognized brokerage in 2008 lead Bank of America become the biggest U.S bank in terms of assets. It was renamed to Bank of America Merrill Lynch and it offers everything from fixed income trading to credit card lending after the acquisition. (MSNBC, 2008)

Merrill Lynch was facing with the danger of bankruptcy in 2008. It's share dropped from 98$ at the beginning to under 17$ in September 2008. Bank of America already had an investment bank division before acquiring Merrill Lynch but it was never strong. This combination of Bank of America and Merrill Lynch brought together their strength to make a stronger bank and save Merrill Lynch from bankruptcy. (MSNBC, 2008)

3. Analysis & Results

3.1 Understanding Quartz

In order to address the business need, we had to analyze the structure of the Quartz framework. More specifically, we needed to address the way that data was being stored in the Sandra database and understand the difference between the deal models in both systems.

3.1.1 Vienna Trade Model and Quartz Deal Model

The deal model represents how data is being stored by the system. In both systems, data is stored in computer objects that are instances of classes or entities that denote trade assets and/or containers for those. In Vienna, the deal model has only three entities: a Book, a Portfolio and a Trade. The Book is a container of Portfolios and the Portfolio a container for Trades. The trade is just the representation of an asset or a contract.

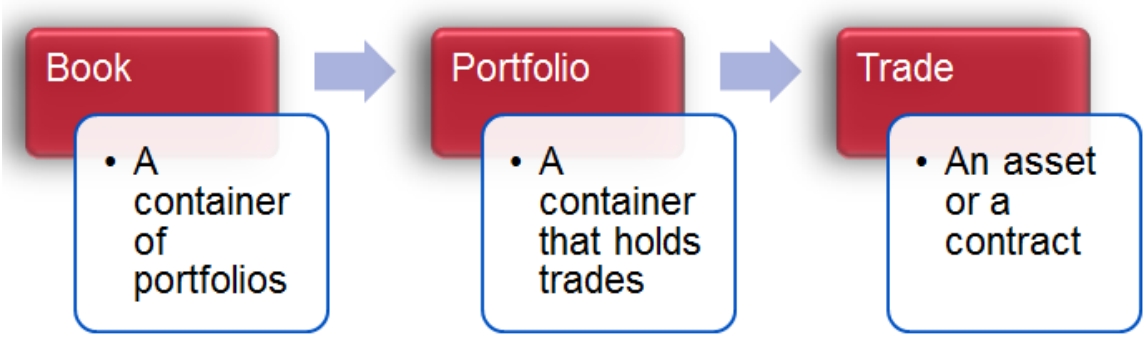


Figure 1 Vienna Trade Model

The Quartz deal model is far more complex, containing many more objects and relationships between them. Instruments in Quartz are similar to trades in Vienna in that

they represent assets or contracts. Instruments can be of various types, for example: Credit Default Swaps and Cash Flows. All of these objects live in the Sandra database and can be accessed by Quartz applications or by writing code in Python to evaluate the multiple attributes in them and printing the data to a shell.

3.1.2 Mapping between Vienna and Quartz

All of the different types of objects that could potentially have data that will be useful for us in replicating a Vienna report were analyzed. After identifying these objects, we created a table that had the Realm report column header name, the object type in Quartz holding the data, the attribute name in that object where the data was being stored, and the description of what that data represented. This map could serve as a document where future developers could look for references when trying to develop new applications in Quartz.

3.2 Credit Default Swap Products

In Global Credit Mortgage Distressed team at Bank of America we worked with Credit Default Swap data on a regular basis for generating reports. As mentioned in the introduction, the risk report our team wanted to replicate is a Credit Default Swap trade report. To better accomplish this project on risk reporting techniques, it was important for our team to not only acquire and apply the technical skills but also understand the financial side of the project with critical thinking.

3.2.1 Introduction to Credit Default Swap

Credit Default Swaps (CDS) are the most widely used type of credit derivatives and a powerful force in the world markets. The first CDS contract was introduced by JP

Morgan in 1997 and increased in use after 2003. (Investopedia) By the end of 2007, the outstanding CDS amount was $62.2 trillion, falling to $26.3 trillion by mid-2010, according to the International Swaps and Derivatives Association, an association created in 1985 that aims to help to improve the private negotiated derivatives market by identifying and reducing risks in the market. (Investopedia, International Swaps and Derivatives Association - ISDA)

A CDS contract can be used as a hedge or insurance policy against the default of a bond or loan. An individual or company that is exposed to a lot of credit risk can shift some of that risk by buying protection in a CDS contract. To illustrate, imagine that we have the following example in the financial market.

In our hypothetical example, there is a Pension Fund which manages the retirement money of teachers in New York City. In order to achieve a good return on investment, the Pension Fund could lend assets to corporations in need of capital. Suppose there is a Corporation that needs $1b to build a new research laboratory. The Pension Fund could enter in to an arrangement with the Corporation by lending it $1b and receive 10% interest on it annually, and then receive $1b back at the end of the lending period. (Khan Academy)

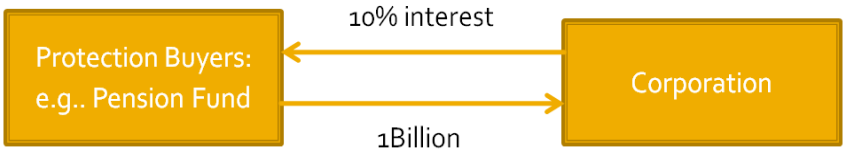


Figure 2 Lending and borrowing between two counterparties

However, the Pension Fund manages retirement money on behalf of many individuals, and it has an obligation to make sure that the money is invested prudently; it is therefore only allowed to lend its money to Corporations that have an AA rating or better. This constraint will likely be part of the Pension Fund's formal investment policy, designed to balance the desire for a high return on investment with a tolerable level of assumed risk. In our example, let's assume the Corporation receives a BB^- rating from one of the credit rating agencies which makes it not so safe to lend money to. See the figure below. (Khan Academy)

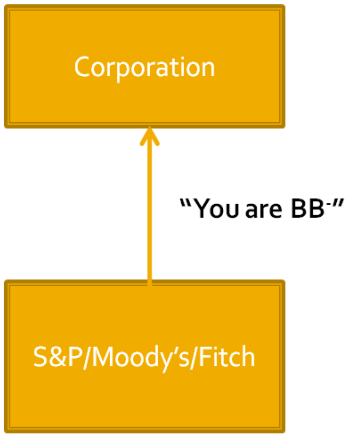


Figure 3 Corporation rated by credit rating agnecy

With a rating of BB^-, this Corporation viewed as more likely to default. If it suffers a credit event and/or eventual default, the Pension Fund may not be able to get its entire loan back. In order to shift some of that risk, Pension Fund can buy insurance, or protection, on this transaction by making a contract with a Bank or an Insurance company that are considered safe. In our continuing example, the Pension Fund will locate a Bank/Insurance Company who has been rated as AAA by credit rating agencies. Out of the interest Pension Fund receives annually from the Corporation, 5% is paid to the

Bank/Insurance Company on an annual basis (in our example, 5% of the 10% annual interest is 0.50% or 50 basis points). If the Corporation defaults in a credit event and is not able to return the money to Pension Fund, the bank/insurance company cover the loss and pay the money back to the Pension Fund. In other words, by paying 50 basis points of the loan amount to the protection seller each year, the Pension Fund's investment is insured by a bank/insurance company. We call this contract a Credit Default Swap Contract. The figure below illustrates this CDS contract. (Khan Academy)

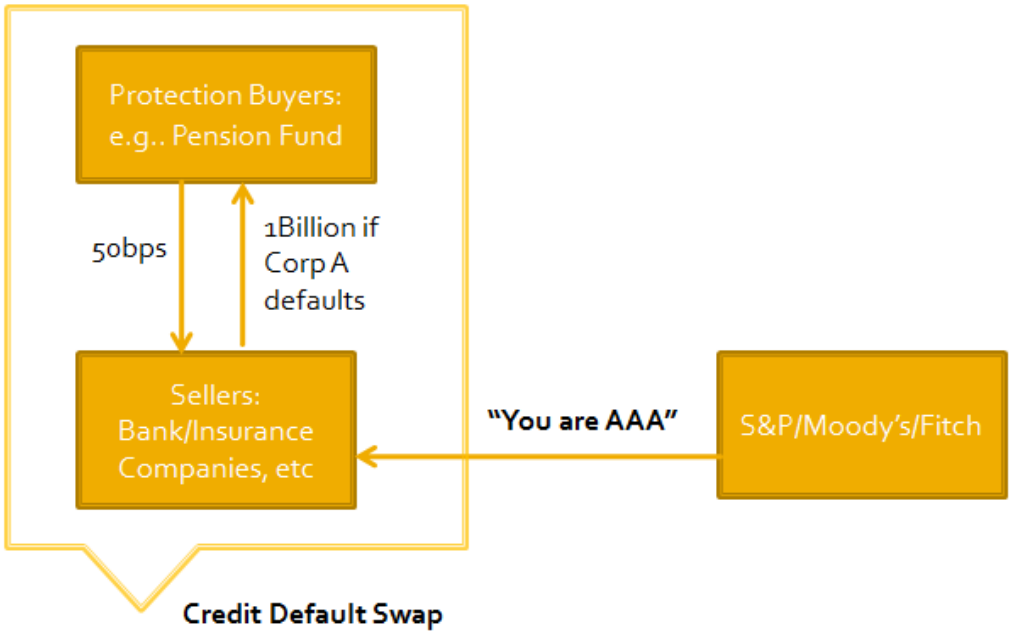


Figure 4 Credit Default Swap Contract

To combine those steps together, the next figure describes the process we have just gone through in order to understand a credit default swap contract. The Corporation here is called the reference entity; it can be also a government or other legal entity that issues debt of any kind. The 50 basis points is called the CDS spread, which is the annual amount that the protection buyer must pay the protection seller over the length of the contract.

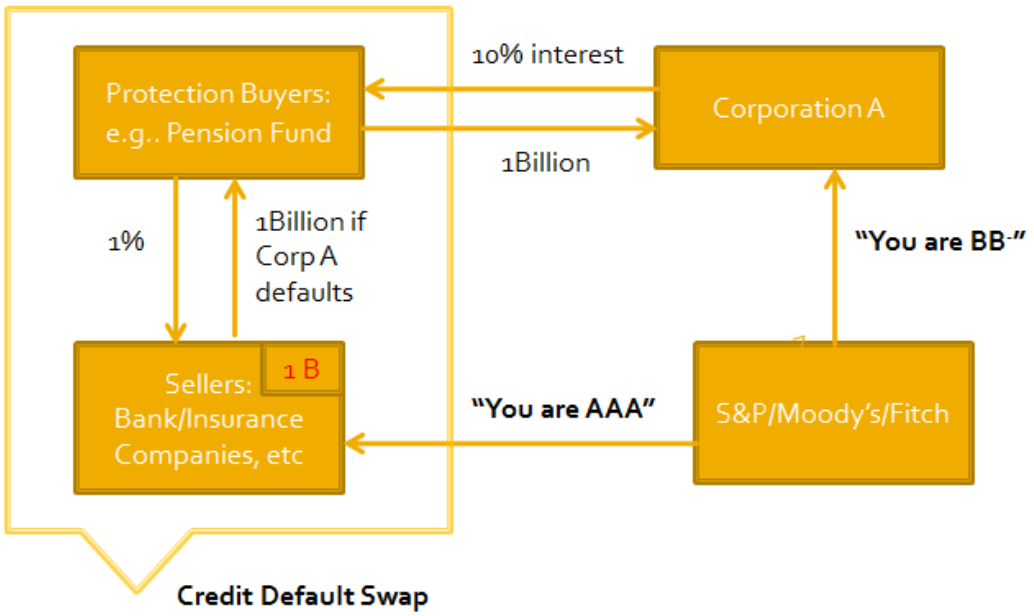


Figure 5 A process of the creation of a Credit Default Swap Contract

Once a credit default swap contract is made, there will be two outcomes: either a credit event occurs or it does not. The figure below is the first case when the associated credit instrument suffers no credit event. The buyer purchased a CDS at time t0 and makes regular premium payments (the 50 basis points in our earlier example) at times t1, t2, t3, and so on. The buyer continues paying premiums until the end of the contract time tn when no credit event occurs. (Wikipedia)

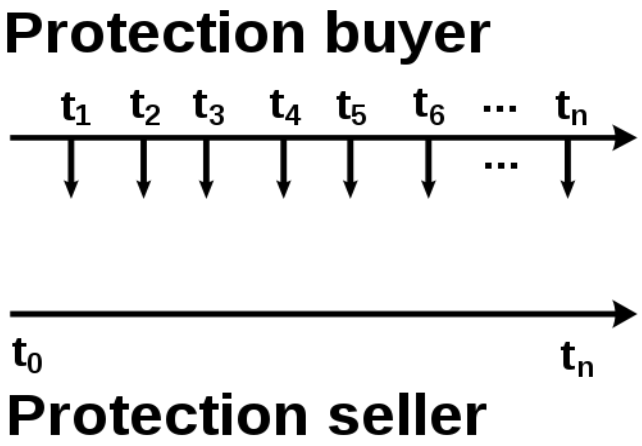


Figure 6 Transactions between protection buyer and protection seller under no credit events

However, if a credit event occurred to the associated credit instrument at t5, then the buyer would stop paying premium payments to the seller and the seller pays the buyer for the loss. If the credit instrument is a bond, the protection seller will pay the par value of the bond to the buyer, and the buyer will physically delivers the bond to the seller. This settlement is an example of physical settlement. The figure below describes the default case. (Wikipedia)

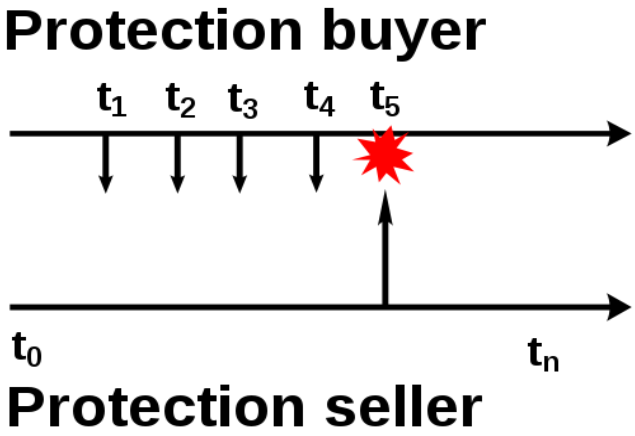


Figure 7 Transcations between protection buyer and protection seller under credit events

We have mentioned at the very beginning of the introduction that CDS has two uses. Its first use, as an insurance policy against the default of a bond or loan, was

discussed above. A second use is for speculators to "place their bets" about the credit quality of a particular reference entity. With the value of the CDS market much larger than the underlying bonds and loans that the contracts reference, it is clear that speculation has grown to be the most common function for a CDS contract. CDS provide a very efficient way to take a position on the credit of a reference entity. An investor with a positive view on the credit quality of a company can sell protection and collect the payments that go along with it rather than spend a lot of money to load up on the company's bonds. An investor with a negative view of the company's credit can buy protection for a relatively small periodic fee and receive a big payoff if the company defaults on its bonds or has some other credit event. (Pinsent, 2008)

3.2.2 Modeling Credit Using Reduced Form Model

Reduced form models describe default by means of an exogenous jump process, more precisely, the default time τ is the first jump of an important kind of stochastic process, the Poisson process. In a Poisson process, events occur continuously and independently of one another. With this model default is not triggered by basic market observables but has an exogenous component that is independent of all the default free market information. The first jump of a Poisson process obeys roughly the following:

$$\mathbb{Q}(\tau \in [t, t+dt) | \tau > t, \text{market info up to } t) = \lambda(t)dt$$

Equation 1 Hazard Rate

The equation tells us that having not defaulted before time t, the probability of defaulting in the next dt instant is λ(t)dt. We call λ(t) the hazard rate, or intensity. The binomial tree below is another way to describe the modeling of default in which the tree terminates and makes a payment K at default. At each node there are two arrows

representing two possibilities. One means that the credit instrument defaults with a probability of λ(t)dt, the tree terminates and make a payment K. The other arrow means that the credit instrument suffers no credit event with a probability of (1- λ(t)dt).

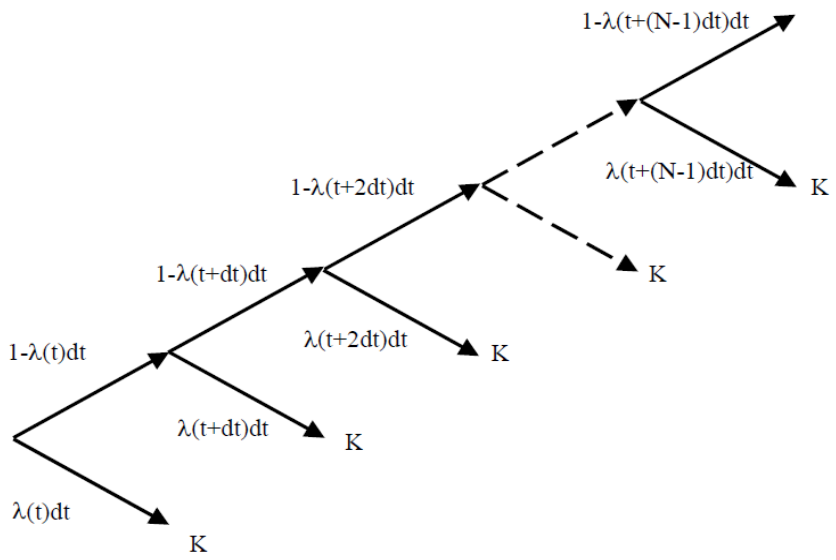


Figure 8 Binomial tree to describe the modeling of default

Define also the cumulative hazard rate:

$$\Lambda(t) := \int_0^t \lambda(u)du,$$

Equation 2 Cumulative Hazard Rate

$\Lambda(t)$ is also called the cumulative intensity, or Hazard Function. One of the important facts about Poisson process is a property of the transformation of the jump time τ according to its own cumulative hazard rates Λ. We have:

$$\Lambda(\tau) =: \xi \sim \text{exponential standard random variable}$$

Equation 3 Cumulative hazard rate as an exponential standard random variable

We could think of the cumulative hazard rate up to time τ as an exponential random variable. This exponential random variable is independent of all other variables (interest rates, equities, intensities themselves, etc.) By inverting the last equation, we have: $\tau = \Lambda\text{-}1\ (\xi)$. Recall that the cumulative distribution function of a random variable with an exponential distribution is:

$$\mathbb{Q}(\xi \leq x) = 1 - e^{-x}$$

Equation 4 Cumulative distribution function of an exponential random distribution

The chance of $\tau > t$ is the same as the chance of the cumulative hazard rate up to τ exceeding the cumulative hazard rate up to t. By replacing $\Lambda(\tau)$ with ξ and apply the cumulative distribution function of exponential random variable, we have:

$$\mathbb{Q}\{\tau > t\} = \mathbb{Q}\{\Lambda(\tau) > \Lambda(t)\} = \mathbb{Q}\{\xi > \Lambda(t)\} = e^{-\int_0^t \lambda(u)du},$$

Equation 5 Applying cumulative distribution function of an exponential random distribution

Therefore, the probability that default occurs after time t is:

$$\mathbb{Q}\{\tau > t\} = \mathbb{E}[e^{-\int_0^t \lambda(u)du}],$$

Equation 6 Probability that default occurs after time t

3.2.3 Bootstrapping Hazard Rates

Bootstrapping is a common method used to calculate the zero-coupon yield curve from market figures. It is also used for constructing the term structure of hazard rates. The figure below is a screen shot of the process of bootstrapping hazard rates.

	A	B	C	E	F	G	H	I	J
1									
2	Tenor (Years)	CDS spreads (in BPS)	CDS spreads (in%)		0-1Year	1-2 Year	2-3 Year	3-4 Year	4-5 Year
3	1	20	0.20	t	0	1	2	3	4
4	2	35	0.35	T	1	2	3	4	5
5	3	45	0.45	CDS spreads (in USD)	0.0020	0.0035	0.0045	0.0050	0.0055
6	4	50	0.50	DiscountFactor	0.9091	0.8264	0.7513	0.6830	0.6209
7	5	55	0.55	HazardRate (λ)	0.003328	0.008570	0.011347	0.011372	0.013464
8				SurvivalProbability	0.99668	0.98817	0.97702	0.96598	0.95306
9	DiscountRate	10%	Make cells F14 to J14 equal to ZERO in SOLVER by changing cells F7 to J7						
10	RecoveryRate	40%		PV_ProtectionLeg	0.001812	0.006030	0.011056	0.015583	0.020396
11									
12	Notional (USD)	1		PV_PremiumLeg	0.001812	0.006030	0.011056	0.015583	0.020396
13	Protection (USD	0.6							
14			Constant	Difference	0.0000	0.0000	0.0000	0.0000	0.0000
15	Exponential (e)	2.7182							
16									
17	Frequency of premium payments	Annual							

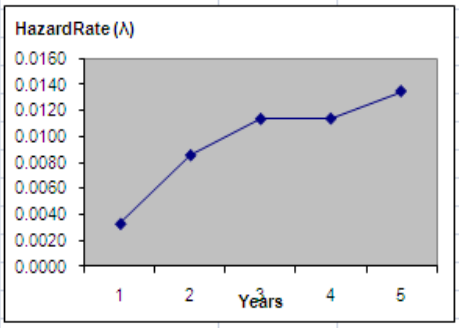
Figure 9 Bootstrapping process in excel

It starts with taking the shortest maturity contract and using it to calculate the first survival probability. In this case, the 1Y credit default swap has to be used to calculate the value of $\lambda_{0,1}$. Assuming an annual premium payment frequency, and assuming premium accrued is not paid, this is achieved by solving: PV_ProtectionLeg = PV_PremiumLeg for the value of $\lambda_{0,1}$. This procedure is then repeated to solve for $\lambda_{1,2}$ and so on until the final maturity default swap is reached. (O'Kane & Turnbull, 2003)

3.2.4 Explaining an implied negative hazard rate

If CDS spread is changed for some particular year, then you need to run the Solver for that particular year and also the subsequent years. For example, the CDS spread is changed for Year 2, then the Solver should be run for year 2, year 3, year 4, and year 5. Cells F14 to J14 should be made zero by changing cells F7 to J7 in the Solver.

		Year_1	Year_2	Year_3	Year_4	Year_5
CDS spreads (in BPS)		20	35	45	50	55
HazardRate (λ)		0.003328	0.008570	0.011347	0.011372	0.013464
SurvivalProb		99.668%	98.817%	97.702%	96.598%	95.306%



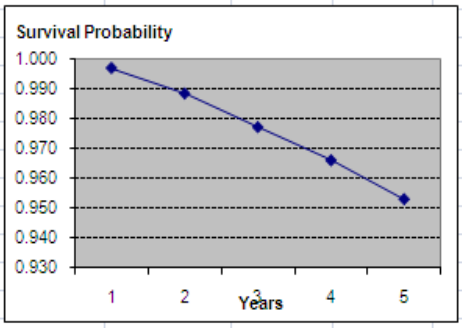
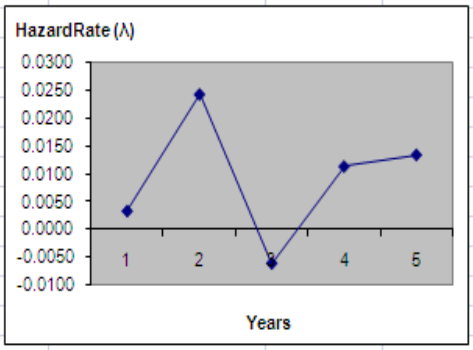


Figure 10 Results of bootstrapping hazard rate scenario 1

We have noticed from the hazard rate chart and survival probability chart that increase in CDS spread will increase the hazard rate but reduce the survival probability. In other words, the higher the CDS spread, the riskier the reference entity is. This makes perfect sense: the higher the CDS spread, the more likely the reference entity is to default by the market, since a higher fee is being charged to protect against this happening.

		Year_1	Year_2	Year_3	Year_4	Year_5
CDS spreads (in BPS)		20	80	45	50	55
HazardRate (λ)		0.003328	0.024308	-0.006087	0.011351	0.013443
SurvivalProb		99.668%	97.274%	97.868%	96.764%	95.472%



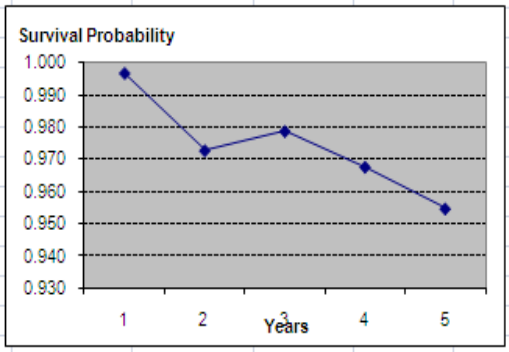


Figure 11 Results of bootstrapping hazard rate scenario 2

When the CDS spread curve is inverted, it is sometimes found that this implies negative hazard rates, for example, in the figure above the second year's CDS spread is changed from 35 to 80 which leads to an inverted spread curve and a negative hazard rate. The negative hazard rate in the second scenario is clearly incorrect as a probability can never be negative. The figure below compares the two scenarios.

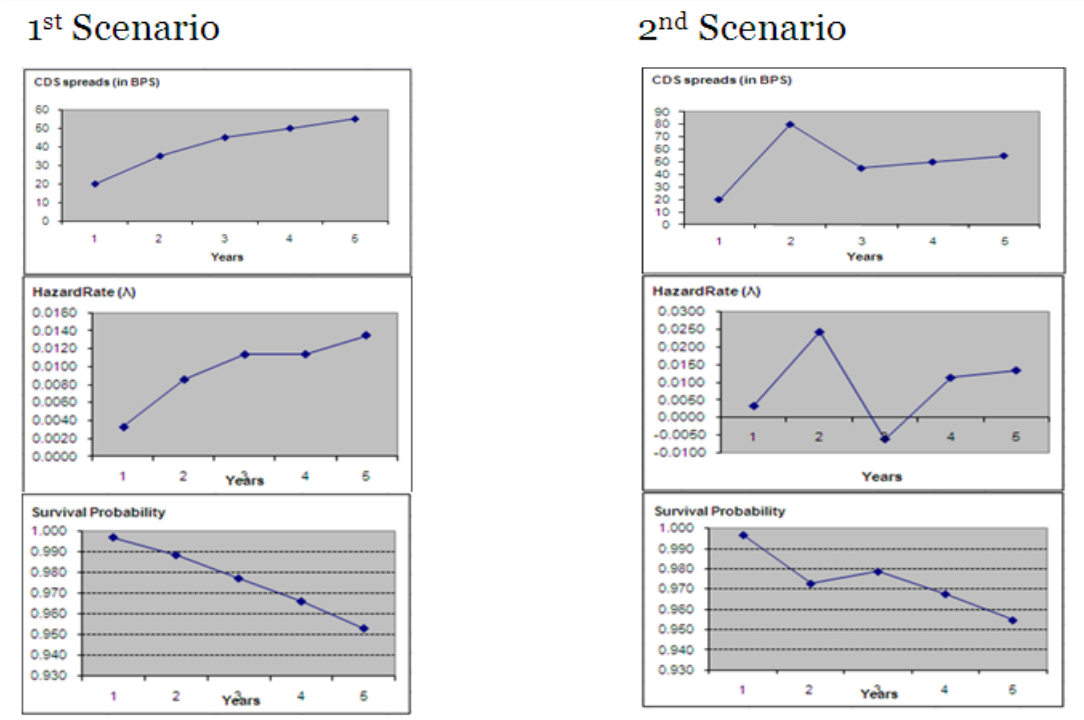


Figure 12 Comparison between results of Scenario 1 and Scenario 2

The negative hazard rate problem can be better understood by using an analogy. Let's imagine a bus ticket example. For example, if bus passes cost $30 for a 3 days pass, and you could buy a five-day pass for $40, the implied price per day of days 4-5 in this case would be $5. We could say that for just $5 more per day, you can upgrade from a 3-day pass to a 5-day pass. In this case, we might quote prices as:

- 3-day pass: $10/day
- 5-day pass: $8/day

One way to look at it would be days 1-3 cost $10 each and days 4-5 cost $5 each, now let's change the numbers to:

- 3-day pass: $10/day (or a total price of $30 for a 3-day pass)
- 5-day pass: $4/day (or a total price of $20 for a 5-day pass)

In this case days 4-5 cost -$5 each, which means not only the passenger doesn't need to pay for the ticket, but the bus driver or company has to pay the passenger to take the bus. This is clearly not reasonable, and same logic could apply to credit default swap.

In reality people wouldn't pay 80 bps per year for two years of CDS when they could instead pay 45 bps per year for three years. And if you did agree to pay 80 bps per year for two years, the implied cost of third year would be forty five times three minus eighty times two, which leads to -25. It doesn't make sense when a negative amount is paid because paying a negative amount of money means you would receive that amount of money, and in a CDS contact, the protection buyer will never receive premium payments from protection seller. Therefore the implied negative hazard rate doesn't make sense in the real world.

However, the negative hazard rate implies an interesting topic called arbitrage. The second scenario where a negative hazard rate has been caused by 1 2-year spread level at 80 bps followed by a 3-year spread level at 45 bps. In this case, we can buy 3Y protection at 45 bps and sell 2Y protection at 80 bps. As long as there is no credit event in the first two years, we receive 35 bps in both years and it is 70 bps in total. We use these to fund the payment of 45 bps in the final year, after that we still have 25 bps on hand. The presence of this sort of apparent "free money" opportunity is call arbitrage – and the usual expectation is that markets will quickly adjust to eliminate these opportunities. (O'Kane & Turnbull, 2003)

3.2.5 Valuation of Credit Default Swap

3.2.5.1 Valuing the Default Leg

The default leg (also known as "contingent leg" or "protection leg") can be thought of as a single payment in the event of a default. This payment is made by the protection seller as a compensation for the protection buyer's loss from the credit event. It is calculated as the notional amount minus the amount that is assumed can be recovered. In other words, the absolute payment is the product of the notional amount, and 1 minus the expected recovery rate. However, this payment is only made in the event of default, and can occur at any point in time up to the maturity date. To explain, imagine that default could only occur on one of the two special dates T_1 and T_2 with known probabilities P_1 and P_2, otherwise the reference entity will survive to maturity. In this imaginary situation, the survival curve is 1 up to T_1, jumps down to 1- P_1 after it, and jumps down further to 1- P_1- P_2 after the second date. For pricing purposes the expected payment on T_1 is the absolute payment time its probability P_1 and the expected payment on T_2 is the absolute payment times its probability P_2. The value of the protection leg is the sum of the Net Present Value (NPV) of the expected payments, which means multiplying the expected payments by the discount factor on each of the two payment dates, and adding them up. (International Swaps and Derivatives Association)

However, in reality, default could occur on any date, so it is necessary to consider every possible default date and weight each possibility according to the probability of default on that date. Rather than a sum over two dates, the value of the protection leg then becomes a sum over a continuum of possible times. Computationally,

it is easier to make the time increment infinitesimally small, rather than daily, and in the limit the sum becomes a mathematical integral, or an area under a curve.

The function to be integrated is the probability of default (for each infinitesimally small period of time) times the discount factor that applies during that small time interval. To illustrate, take a look at this time line below. The different color in the equation corresponds to different point of time and events in a life of credit default swap. The green part in the function is the probability that the default occurs after t. The red part represents that the reference entity then defaults in the next small interval with probability$\lambda(t)dt$. And finally the yellow part in the equation is to discount payments back to today using discount factor$Z(t)$.

$$x = N(1-R)\int_0^T \lambda(\tau)Z(\tau)e^{-\int_0^\tau \lambda(t)\,dt}d\tau$$

Equation 7 Present value of a default leg

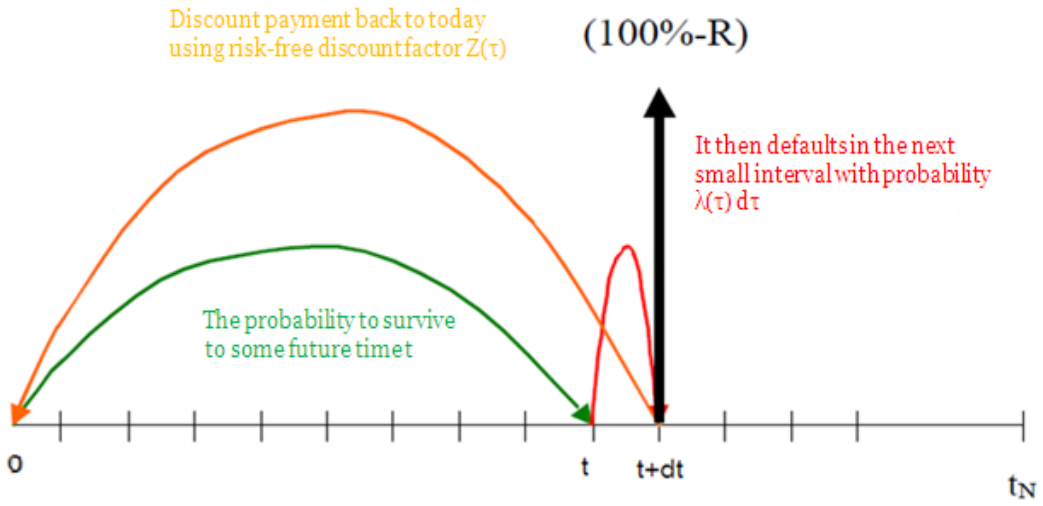


Figure 13 Default leg valuation process

3.2.5.2 Valuing the Premium Leg

The premium leg is a regular series of payments, made by the protection buyer, which continues as long as the reference entity has not defaulted, or until the maturity of the CDS, whichever is sooner. Each payment is the product of the notional amount, the premium rate (or spread), and the accrual factor for that period (e.g., around 0.25 for quarterly payments). However, this absolute payment is only made if the reference entity has not defaulted, so for pricing purposes the expected payment (in the statistical sense) is the absolute payment times the probability of survival at the end of the period. This probability needs to be interpolated from the survival curve using the assumption of the piecewise exponential shape. The contribution of this payment to the CDS price is the NPV of the expected payment, which means multiplying further by the discount factor on the payment date. The value of the premium leg is then calculated by summing the contribution from each payment. (International Swaps and Derivatives Association)

$$FC = \sum_{i=1}^{M} Z(t_i)\ \alpha\ (t_i - t_{i-1})\ e^{-\int_0^{t_i} \lambda(t)dt}\ d\tau$$

Equation 8 Present value of premium leg with full coupon payments

However, the premium leg also includes the payment of premium accrued from the previous premium payment date until the time of the credit event. To include partial premium accrued, we have the following equation. In the previous equation we determined the probability of surviving from the valuation date to some future time t in the premium period, suppose it then defaults in the next small time interval dτ. The probability of this is given by $Q(\tau>t)\lambda(\tau)d\tau$. Then multiply it further by the discount factor to discount it back to last premium payment date.

$$PC = \sum_{i=1}^{M} \int_{0}^{t_i} \lambda(\tau)\ Z(\tau)\ \alpha\ (\tau - t_{i-1})\ e^{-\int_{0}^{t_i} \lambda(t)d\tau}\ d\tau$$

Equation 9 Present value of the partial coupon payment in the premium leg

By adding in partial premium accrued, the value of the premium leg on valuation date is calculated as:

$$P = NS(FC + PC)$$

Equation 10 Present value of premium leg

3.3 The Application

The Global Credit Mortgage Distressed team required the creation of a small application that could replicate a Realm report in the new framework. The report that was going to be replicated is the Trade Credit Details by Trade Report. This report contains information about Credit Default Swaps, the counterparties involved in the trade, the trade sensitivities and information about the portfolio and the trader. The application needed to replicate the report and display the data to the user as well as write it to a temporary database for future queries. The end user will simply have to provide the application with a portfolio(s) that they wish to generate the report for and whether they wish to write the data to the database or not. The application will then query Sandra, and display the report as a Graphical User Interface (GUI). If the write to the database failed or there was some error in extracting the data, the application will notify the user with a pop up notification.

We decided on a simple design where a controller module will be in charge of extracting the data from the database and assembling the report. Adaptors will then use

that report and perform data with it, such as displaying it to the user or writing to the database. We decided to do it this way to provide flexibility for any future expansion and to encapsulate the logic of generating the report to a single module.

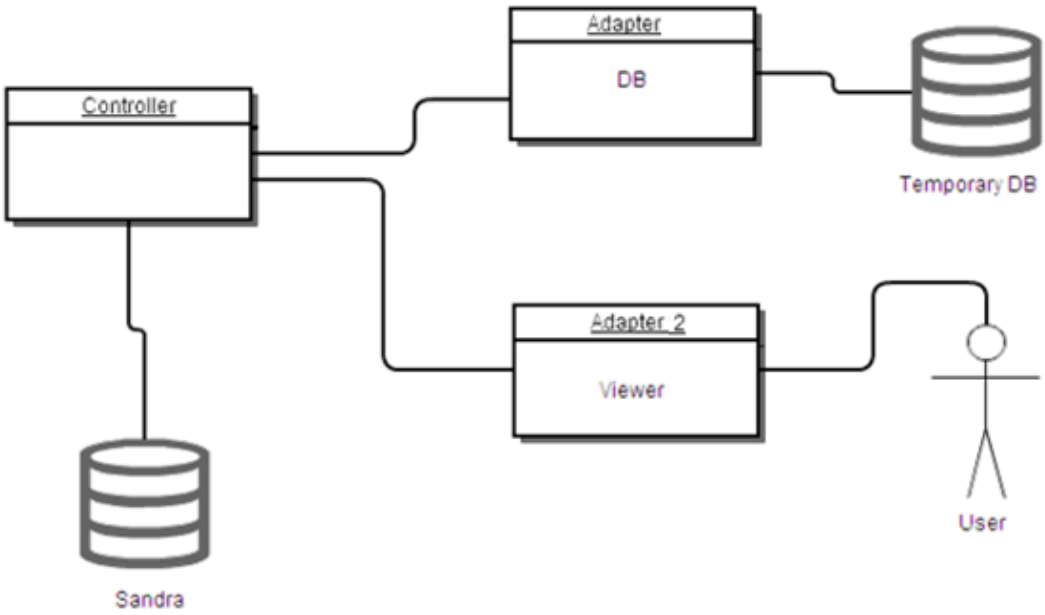


Figure 14 High Level Entities

3.4 The Objects

By encapsulating their behavior, the objects have modularity which makes maintenance and changes easier as the objects are decoupled from other parts of the system. Our application consisted of four main objects that are in charge of replicating the report and a number of "adaptor" objects who adapted that report for display or to write to the database. Consider this simple factory analogy where there is a leader that will get an order, for example a car; it will ask some other person in the factory to gather the pieces of that model. Once the pieces are gathered the leader will then pass the pieces to a dedicated assembly line for that model. In the assembly line they will put the pieces together and assemble the car. Once the car is complete the leader will send it to get

painted and finally send it to the sales department who will sell it to customers. In our case the Controller object will be the factory leader. The object gathering the pieces needed for the job is the Instrument Populator. The Mapper and the Risk Reporter will work in lieu to assemble the report. Finally, the adaptors can be viewed as the sale department where the final product is displayed. Below we provide a description on each object and the data flow in the application. Detailed information on the class structure can be found in the design document in the Appendix A.

3.4.1 Controller

This object is in charge of initializing all the objects that are part of the application. It contains local copies of all these objects and is constantly calling methods on them to pass and receive data. Therefore, this object handles the flow of information from receiving a list of portfolio names from the GUI to returning the report for display. This object serves the same purpose as a keystone in an arch. It is the central supporting element; all the other objects receive the data that they need to work with from it and, after that data is processed, and they return the results to it.

3.4.2 Instrument Populator

This object will process a list of portfolio names and generate a table structure that we decided to call a Population Table, where each row will represent an instrument in that portfolio. In other words, if the portfolio that we are trying to process has five trades in it, the table generated by this object will have five rows. Each row has four columns: the book object, the instrument object, the instrument notional and the deal object. All these objects are extracted from the Sandra database and from them, and the other objects that they are referencing, we can extract the data that will be in the report.

Going back to our factory analogy, this Population Table represents the parts needed to assemble the requested car. After all the instruments in the given portfolios are processed, the Population Table is returned to the controller.

3.4.3 Risk Reporter and Mapper

The Risk Reporter is in charge of generating the report. It does so by taking in the Population Table generated by the Instrument Populator and a list of all the headers that will the report will contain. The list of headers are passed by the GUI to the Controller with the request to generate a report is received. The Risk Reporter works with the Mapper object to know which attribute to call on which object. The Mapper, as the name suggest, contains the map of all the attribute names, their parent objects and the column headers in the report. The map is a dictionary where they key is a tuple that contains the object type and subtype. The object type can be a Book, Deal or Instrument, while the subtype can be 'None' or the short name of an instrument. This short name is an attribute that only exist in instruments, it denotes what type of instrument we are dealing with, like a Credit Default Swap or a Forward Cash Flow. Since only the instruments objects have a short name, the subtype of a Book and a Deal is 'None'. The tuple key has as a value another dictionary that holds the Column Header of the report and the method calls needed to be called on the object to retrieve the value.

With the help of the Mapper, the risk reporter then enters on a loop to evaluate all the objects in the Population Table that the Instrument Populator created. If we go back to our analogy of the factory, the Risk Reporter can be viewed as the assembly line, where all the pieces needed to assemble a car are put together. The Mapper is like a set of instructions on how to assemble all those parts. The end result of the loop is a table that

contains the replicated report. After the loop is done, this table is returned to the Controller.

3.4.4 Adaptors

The adaptors are objects that are in charge of performing an action on a replicated report. They all accept tables with replicated reports as input and do something with that table. The actions on this table can range from translating the table to a view module for display, or writing the data to a database. The adaptors contain all the logic to perform those actions. For example, they know the schema of the database or they know how to create a view module that the application GUI can use.

Because Adaptors are objects that accept a table with the report and do something with it, developers can add more adaptors in the future without tampering without the need to modify the application.

3.5 Data Flow

When the user sends a request to the GUI to generate a report from a given portfolio name, that request is received by the Controller. The Controller then sends the list of portfolio names to the Instrument Populator who accesses the Sandra database to extract the necessary objects to create the Population Table that contains all of the objects needed to generate the report. When that table is ready, it is sent back to the Controller who sends it to the Risk Reporter. As soon as the Population Table is received, the Risk Reporter generates an empty report with the column headers that will be included in the report. For each of those headers, it retrieves the data by calling methods in the Mapper, which will return the object type where the value for that particular header can be

accessed and the method calls that have to be executed on the particular object. The results are then appended to an array that will represent a row of data in the report. After the row is complete, the array of results is appended to the table that represents the report and the loop continues for the next row in the Population Table until the table is depleted. After the last row in the Population Table is evaluated, the Risk Reporter returns the table containing the report to the Controller. The table will then be processed by the View Adaptor to create a view module with the table. That view module can then be used by the GUI to create a new tab where the report will be displayed. The table can also be sent to the Database Adaptor who will then attempt to write it to a Temporary Database. After this entire process is completed, the Controller is ready to accept a new request to generate a new report.

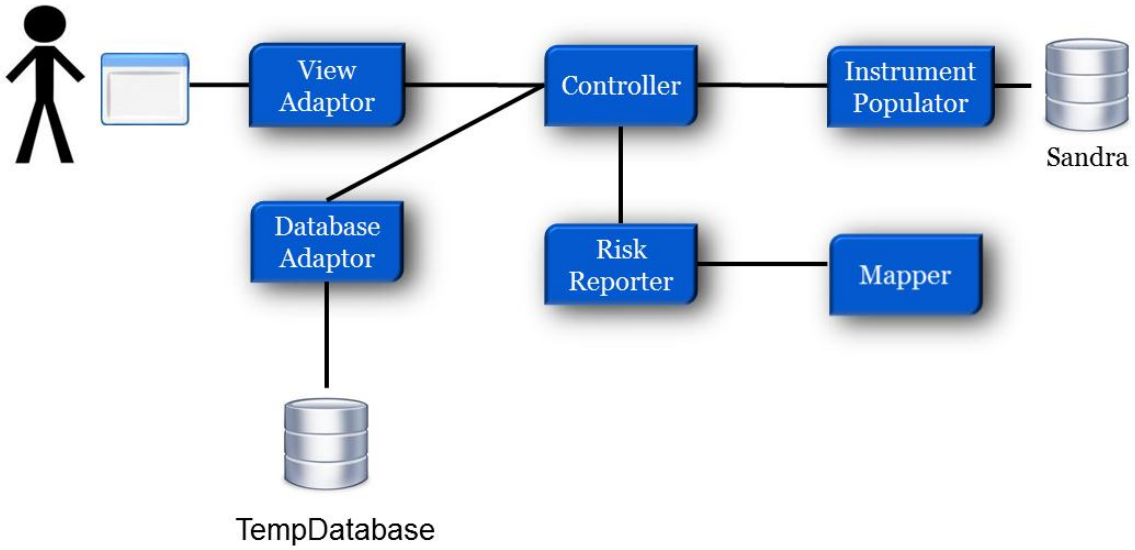


Figure 15 Data Flow in the Risk Reporter Application

3.6 The Graphical User Interface

Once we had a working copy of the application, we designed a Graphical User Interface to tie everything together and provide the user with a nicer way of visualizing the data. The application interface contained:

- A text box where the user can write the portfolio name. As the user types the applications queries the Sandra for the book and provides the possible matches as a list below (just like an autofill box in Windows)
- List boxes with the possible report headers. This provides the user with the ability to limit the visible column headers and generate custom reports.
- A tabbed view where reports that are generated are stored. This means that the user can generate multiple reports and all of them can be viewed under the same window.

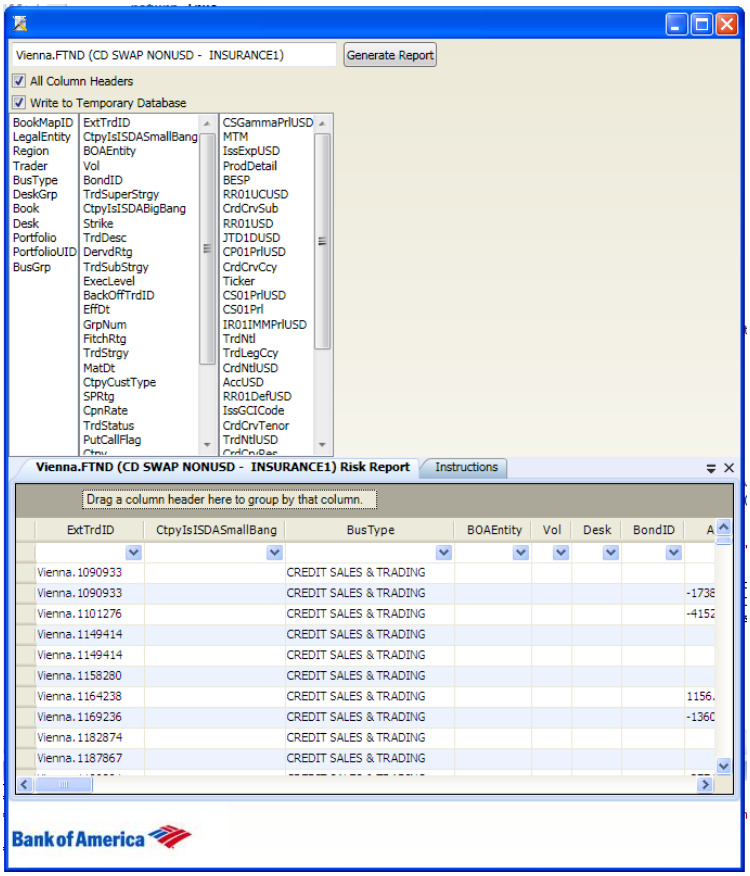


Figure 16 The Graphical User Interface

4. Conclusion

Creating a risk report in the Vienna system requires pulling out information from a specialized database that was being nourished by a myriad of applications. These application were not well-integrated, communication between them required manually copying data from one application to another or modifying the data to a state where it was salvageable for one of the application. Bank of America's solution, Quartz, will provide integration to handle data more efficiently.

The implementation of the Risk Reporter Application provided developers in the Global Credit Mortgage Distressed team with the ability to replicate a risk report entirely from Quartz. Additionally, the extensive documentation that was created along with the application will provide developers with useful information for the future migration process. The final state of the application successfully replicates the risk report and displays it in a customizable GUI spreadsheet powered by Quartz. The application is also fault tolerant; it catches any possible runtime errors at any point of the execution and provides the user with their details. The way that Adaptors are integrated into the application makes it easy for any developer to incorporate new functionality into the application. The new Adaptor just needs to accept a risk report, in the form of a Quartz table, and perform the desired operation on it.

Finally, the code that we wrote for generating the GUI can help developers understand the QuartzUI library. More specifically, it helps developers understand how the library components and dynamic data interact. As more applications are being created or migrated to Quartz, it is important to create view modules that are not only visually

appealing, but extensive in functionality. By analyzing our project, future Quartz developers can find a starting point, a base to support writing and displaying risk reports.

5. Glossary

5.1.1 International Swaps and Derivatives Association – ISDA

ISDA is an association created by the private negotiated derivatives market that represents participating parties. This association helps to improve the private negotiated derivatives market by identifying and reducing risks in the market. It was created in 1985 and it has members from institutions around the world. (Investopedia, International Swaps and Derivatives Association - ISDA)

5.1.2 Reference Entities

The reference entity is essentially the party upon which the two counterparties in the transaction are speculating. The seller of the transaction is selling protection against the default of the reference entity. The buyer of the securitized credit derivative believes that there may be a chance that the reference entity will default upon their issued debt and is therefore entering the appropriate position. The reference entity bears the credit risk of the contract, and can be a corporation, government or other legal entity that issues debt of any kind. (Investopedia, Reference Entity)

5.1.3 Credit Spread

The "spread" of a CDS is the annual amount the protection buyer must pay the protection seller over the length of the contract, expressed as a percentage of the notional amount. The higher the CDS spread, the more likely to default by the market, since a higher fee is being charged to protect against this happening. Note that these spreads are not the same type of concept as “yield spread” of a corporate bond to a government bond. Rather, CDS spreads are the annual price of protection quoted in bps of the notional value, and not based on any risk-free bond or any benchmark interest rates. Periodic premium payments allow the protection buyer to deliver

the defaulted bond at par or to receive the difference of par and the bond's recovery value. (Nomura Fixed Income Research, 2004)

5.1.4 Recovery rate

The recovery rate is the percentage of the notional amount that the defaulted credit instrument recovered from the credit event. To be precise, it is the expected price of the CTD (Cheapest-to-deliver) obligation into the protection at the time of a credit event. (O'Kane & Turnbull, 2003)

5.1.5 Premium Leg

This is a regular series of payments, made by the protection buyer, which continues as long as the reference entity has not defaulted, or until the maturity of the CDS, whichever is the sooner. It also includes the payment of premium accrued from the previous premium payment date until the time of the credit event.

5.1.6 Default Leg

The default leg is the contingent payment of (100%-R) on the face value of the protection made following the credit event. This payment is made by the protection seller as compensation for the protection buyer's net loss, and is calculated as the notional amount minus the amount that is assumed can be recovered.

5.1.7 Trigger Event

ISDA's standard documents for CDS provide for six kinds of trigger events. However, market participants generally view the following three to be the most important: (Nomura Fixed Income Research, 2004)

- Bankruptcy
- Failure to pay
- Restructuring

5.1.8 Physical Settlement

In a physical settlement, the protection seller buys the distressed loan or bond from the protection buyer at par. Here the bond or loan purchased by the seller of protection is called the "deliverable obligation." Physical settlement is the most common form of settlement in the CDS market, and normally takes place within 30 days after the credit event. (Nomura Fixed Income Research, 2004)

5.1.9 Cash Settlement

The payment from the seller of protection to the protection buyer is determined as the difference between the notional of the CDS and the final value of the reference obligation for the same notional. Cash settlement is less common because obtaining the quotes for the distressed reference credit often turns out to be difficult. A cash settlement typically occurs no later than five business days after the credit event. (Nomura Fixed Income Research, 2004)

5.1.10 Python Programming Language

Python is a dynamic programming language that is used in various application domains. Python has very clear syntax and emphasizes high readability. It is often compared to Tcl, Perl, Ruby, Scheme or Java. (Python Software Foundation (PSF))

5.1.11 Object

Object are entities that can be called by using a programming language in computer science. It can be a value, variable, function, or data structure. In Sandra, which is an object oriented database, objects can be manipulated by the commands of Python programming language. (Python (programming language))

6. Appendix A

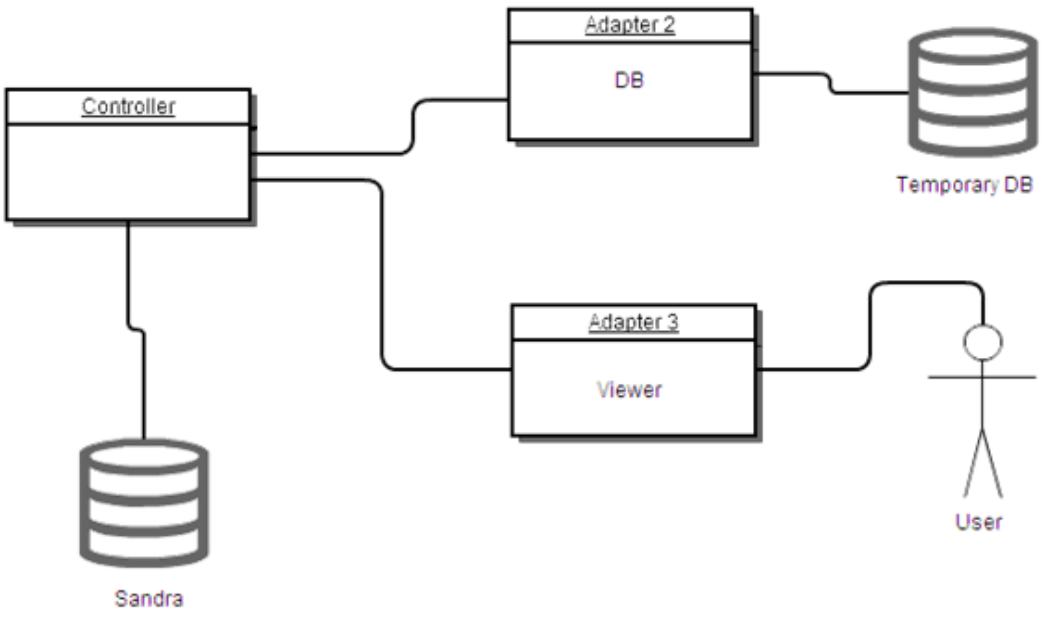
6.1 Design Document

Purpose

To design an application that will replicate a trade credit details by trade report from Realm using data in Quarts. The software is just a starting point where other developers can use as a reference to understand Quartz Positions in the future.

The application must display the replicated report to a user interface in the form of a spreadsheet or table and/or to a database system (temporary database).

High Level Entities



6.1.1 Controller Module

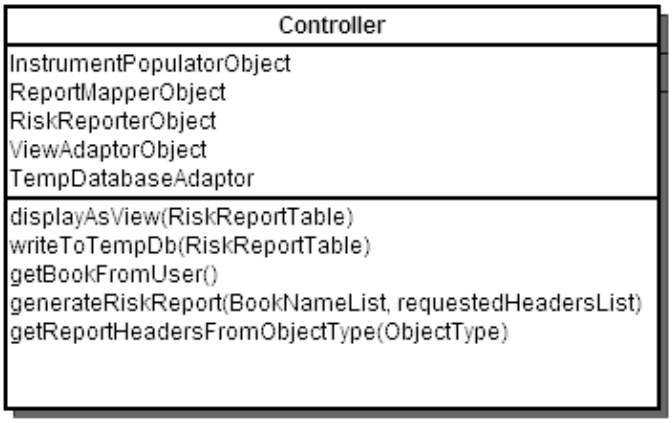
Not to be confused with the Controller class. It will be in charge of accessing the Sandra database and gathering the information that will be used in the report. The Controller Module is comprised of the Controller, Risk Reporter, Instrument Populator and Mapper objects.

6.1.2 The Adaptors

They are in charge of using the data returned by the Controller Module to perform a specific task; like writing to a the temporary database, displaying data structures as User Interfaces modules, etc. They are specialized classes with a single purpose who can take the output of the Controller Module, the replicated report, and do something with it.

Low Level Entities

6.1.3 Controller



6.1.3.3 Purpose

This object is in charge of creating all the other objects as well as serve as a link for the data transfers between these objects and the User Interface.

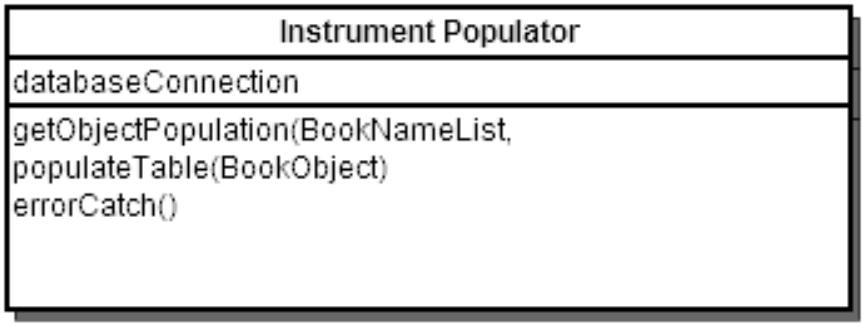
On init this object will create and store instances of all the objects in the application.

6.1.3.4 Method Description and Logic

displayAsView: this method takes in a QzTable with the replicated report and generate a qz.ui popup (using the view adaptor) to display the report. This method was created primarily for testing purposes and is independent from the main UI from the application.
writeToTempDb: this method takes in a QzTable with the replicated report and writes that information to the Temporary Databse using the TempDbAdaptor.
getBookFromUser: when this method is called the viewAdaptor will generate a qz.dialog box where the user can write in a book(s). This is captured by the adaptor and sends to the controller to be processed. This was used for testing purposes.
generateRiskReport: this method takes in a list with the portfolio names that are to be included in the report and a list of column headers that are to be included in the report. The book list is then passed to the Instrument Populator who will generate the Object Population Table. This table is then passed to the Risk Reporter along with the list of column headers to generate the report. This method then returns the QzTable(with the replicated report) that is returned from the Risk Reporter.

getReportHeadersFromObjectType: This method takes in an object type (in the form of a string) to return a list of all the report column headers that can be evaluated for that object. This method uses the mapper object to obtain that information.

6.1.4 Instrument Populator



6.1.4.5 Purpose

This object is in charge of generating an Object Population Table from a given List of Portfolio(s) Names. The Object Population (OPT) Table is a QzTable that contains object extracted from the Sandra Database for each Instrument in the Portfolio(s). Each row of the OPT will represent a single Instrument where the Instrument Object, its notional and the respective Book and Deal object will be stored.

On init the object will create the QzTable schema of the table as well as a connection variable to access the credit_dev database.

6.1.4.6 Method Description and Logic

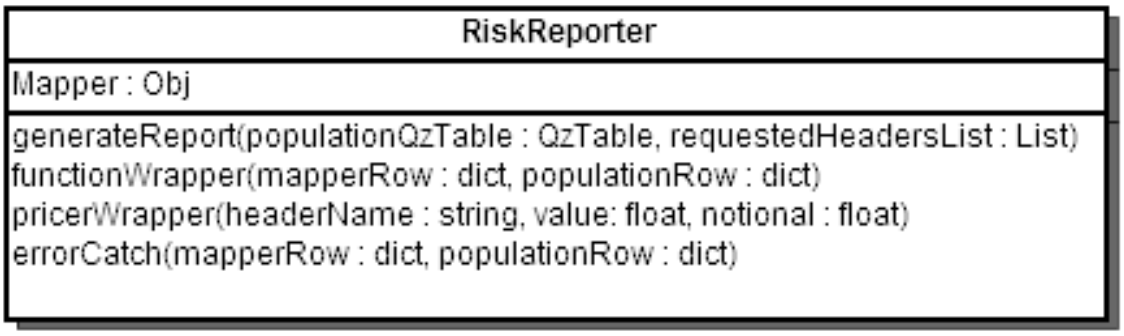
getObjectPopulation: this method will take in the List of Portfolio Names that are to be included in the report and use the table predefined QzTable schema to create the table.

For each book name it will then extract the book object from the database and pass that to the populateTable method to append the values to the table

populateTable: this method uses the given Book Object to extract the all Deals from that object. For each Deal, it extracts all the Instruments in the Deal as well as the Instrument Notional. For each Instrument, it appends to the table the Instrument Object, the Notional, the Deal and Book Object.

errorCatch: a method used to return None, if there was an error extracting one of the many objects. The None can then be appended to the table to signal there was an error in the extraction. This was done in order to prevent exceptions from the built-in method getattr and also to serve as an error tracker in the future.

6.1.5 Risk Reporter



6.1.5.7 Purpose

This object is in charge of extracting the relevant information from the various objects in the Object Population Table and returning that information as a Risk Report. It uses the Mapper object to know what methods to call from each object to get the value for the column headers in the report. It returns a QzTable with a replicated risk report.

On init this object stores a local copy of the mapper object.

6.1.5.8 Method Description and Logic

generateReport: This method uses the Object Population Table, the mapper and the function wrapper to replicate the report. It takes in as an optional parameter a list of requested Column Headers for the report. If this list is passed the report will be generated for the column headers specified by that list. If is not passed then a report containing all the column headers will be generated. For each of the rows in the Object Population Table and for each of the column headers in the report, this method will first get the object where the information is stored for the column header that we need. After the object has been identified it will identify the object subtype, if applicable, and use the mapper again to find the method call map by passing in the object type, subtype and column header. Once it has the map this will use the functionWrapper to extract the value from the object and the pricerWrapper to correctly modify the value returned from the functionWrapper for some of the column headers values from the Pricer Object in the Instruments. After a column header value is ready, it will append it to results list that will be appended to the Risk Report QzTable after all the column headers for that Object Population Table row are extracted (since this represents a row for the Report).

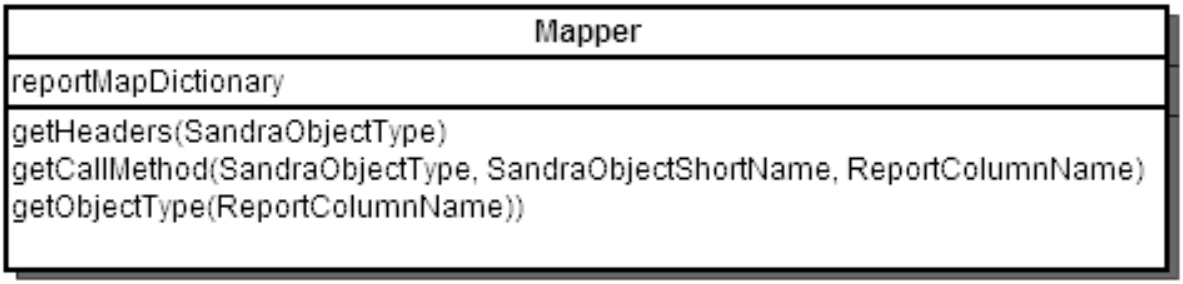
fucntionWrapper: The function wrapper takes in a Mapper Row and an Object Population Table Row as parameters. The Mapper Row contains information about the Column Header, the object type and the subtype and the method call map. The Object Population Row contains the objects that have the information needed. Using the Mapper Row, we extract the instance of the object needed to start the method calls from the Object Population Row. This object then becomes the first value in a results list. Then the method call map is extracted and we create a list of strings from the method call by

splitting the string for each point. So for example if we have ‘foo.bar’ we will get a list like this: [‘foo’, ‘bar’]. Then for each of this string we will call the method on the last value of the results list. The last value will always represent the last appended object so this will make sure that we are calling the method in the right object. We use the built-in function getattr to call the method and pass in the errorCatch function as a default parameter just in case calling that method creates an exception. After we iterated through all the items in the list of split strings we return the last item in the result list which should be the value for the column.

pricerWrapper: this method just makes sure that the value returned from the functionWrapper for some of the headers that come from the Pricer object in the Instrument Objects are correct. For example, they can be incorrect because sometimes they need to be multiplied by the notional. This just takes in the notional, the value returned by the functionWrapper and the column Header being evaluated. It returns the correct value for that column header.

errorCatch: a method used to return a string with the error specification, if there was an error extracting one of the values in the process. This was done in order to prevent exceptions from the built-in method getattr and also to serve as an error tracker in the future.

6.1.6 Mapper



6.1.6.9 Purpose

This object is in charge of storing the map statically. The map is a dictionary that will contain the method calls needed to extract the information of a specific column header from an object. Because of this, the object map contains all the possible headers that are present in an object as well as the type and subtypes of the objects where we can get the column header value. The key to the map dictionary is a tuple of strings where the first is the object type and the second the subtype. The value is another dictionary where the key is the column header and the value the call method map. This call method map is nothing more than the multiple method calls needed to be called on the object type to get the value for the column header. For example if the way to get to the Clean Price on an instrument is Instrument.Pricer().CleanPrice() then the map call method map will be 'Pricer.CleanPrice', the column header will be 'Clean Price' and the tuple for that will be ('Instrument', 'CDS') where CDS is the subtype for that instrument. We decided that the subtype will be the attribute called ShortName for the instruments. Since only instruments have this attribute, the subtype for the Deals and Book object will be None.

6.1.6.10 Method Descriptions and Logic

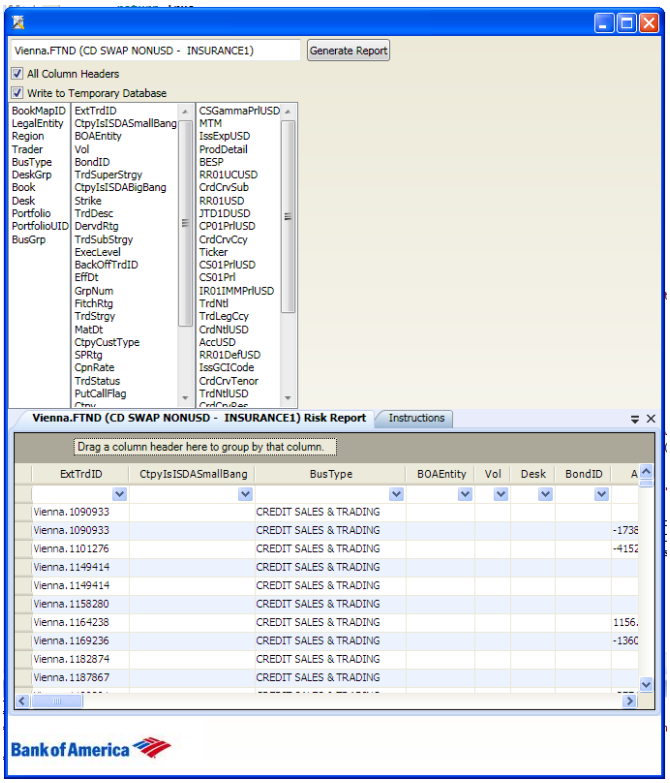
getHeaders: This method returns a list of all the possible object headers in the report for the given object type. If no object type was given, it returns all the possible column headers in the report.

getCallMethod: This method returns the method call map that matches the object type, subtype and the column header.

getObjectType: this returns the object type from the given column header. It searches for the column header for each of the object types in the map and returns the first match.

6.1.7 Application

This object generates all the User Interface that the user will interact with. Below is a screenshot of the UI:



The Application uses the qz.ui elements to render the UI. The text box at the top is the ui element called autobookchooser. This element tries to find the portfolio names from the database as you type. Below there are two check boxes. If the first one is checked, the one called All Column Headers, it will generate the report with all the column headers, ignoring the user input on the list boxes below If the first check box is not checked the report will be generated using the highlighted column headers in the three list boxes. If the second one is checked, it will cause the application to write the generated report to the temporary database. Generated reports will be displayed on the lower part of the screen where they will be contained in tabs. The tab name is the portfolio name used to generate the report. The user can drag a column in the report to group by that column.

6.1.8 Adaptors

6.1.8.11 View Adaptor

Purpose

The purpose of this adaptor was to generate the ui modules that can be used by controller to provide some functionality. Right now the application object handles most of the view modules so this class was primarily used to generate views for testing purposes.

6.1.8.12 The Database Adaptors

Purpose

The purpose of this adaptor is to provide functionality to write the report to a database. Two object were created to fulfill this purpose. The first one used the qzdbapi

library to create the connections and the execute the queries on the database. The second one used the pyodbc library to do the same. The later was the one being in use by the TempDBAdaptor because the first approach was failing (some internal error) by the end of the internship. The TempDBAdapor is a child of this object and it has its own method to execute queries on the database. It alson contains the logic to write the QzTable with the report to the database. It does this by first purging the portfolio from the database, along with all the deals and instrument sensitivities related to them deals. After that it makes a clean write. The method can only write a full report (containing all the column headers) to the temporary database.

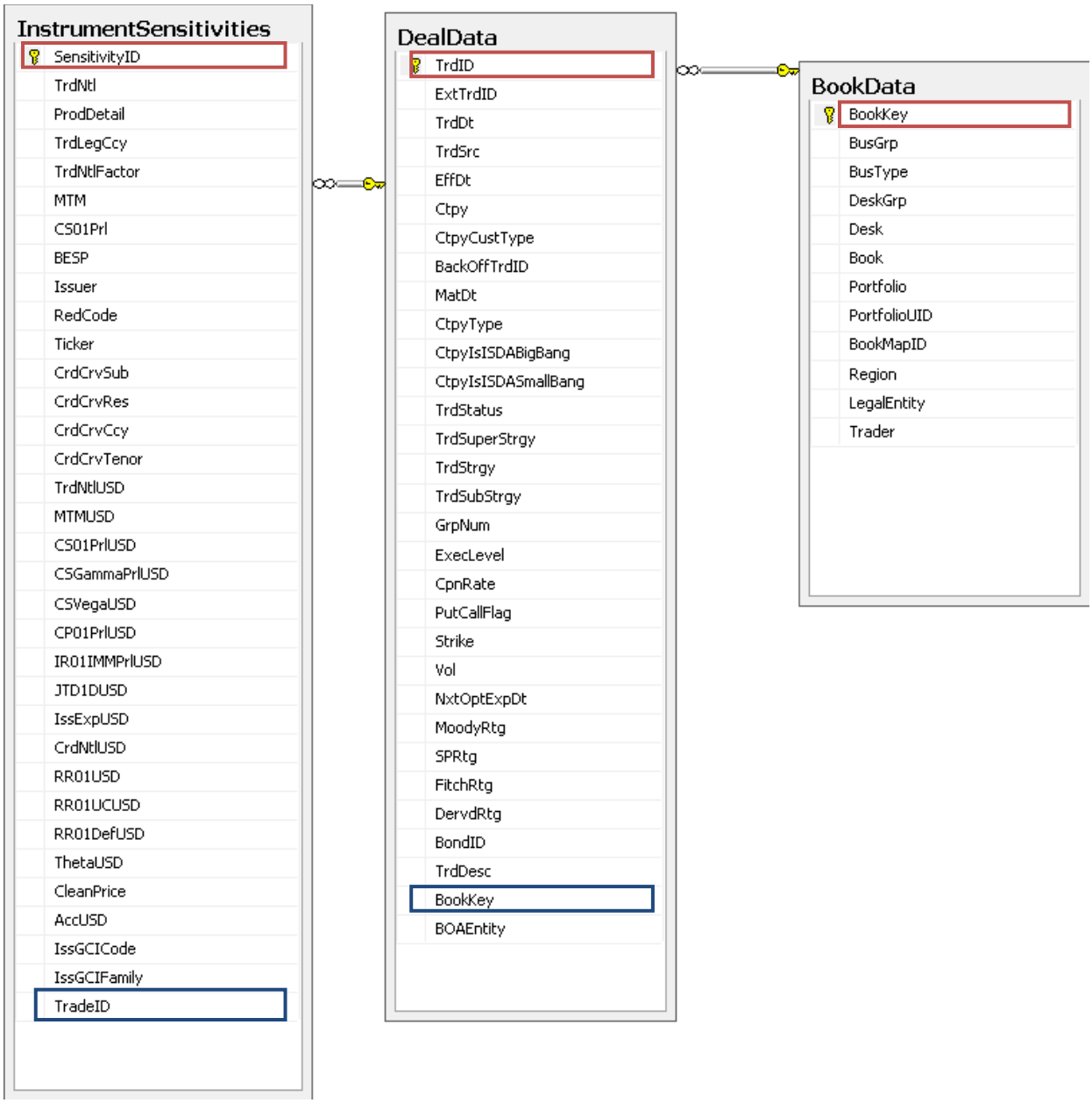
Database

A test database was created to provide a way to store the information that the objects were retrieving from Sandra.

Model

The database emphasizes the storage of information so normalization was not very important. That being said the database is based on the object model, where each attribute in the object represents a column in a table in the database.

The ERD for the database is:



Improvements

- Finishing the error tracking capabilities for the objects.
- Revising the design of the database adaptors. The work on these adaptors was completed too late at the end of the internship that we didn't have time to present

it to the team and get feedback. A more efficient and robust design can probably replace the current one.

- The method that writes the report to the temporary database right now checks that the report has all the column headers. This checks that the number of columns in the report table is 73. This number is hard wired which can lead to problems later if the number of columns in complete report changes.

7. Bibliography

Bank of America. (n.d.). Retrieved 01 09, 2011, from The Finance Owl: http://www.thefinanceowl.com/banks/usa/bank-of-america/

Bank of America Merrill Lynch. (n.d.). *Corporate Profile*. Retrieved 09 27, 2011, from Bank of America Merrill Lynch: http://investor.bankofamerica.com/phoenix.zhtml?c=71595&p=irol-homeprofile

Basel II. (n.d.). Retrieved 01 10, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Basel_II

Dubno, M. (2011, November). Future of Technology for Global Markets. New York: Bank of America.

International Swaps and Derivatives Association. (n.d.). *Modeling Assumptions behind the ISDA CDS Standard Model.*

Investopedia. (n.d.). *Credit Default Swap: An Introduction*. Retrieved 11 28, 2011, from Investopedia: http://www.investopedia.com/articles/optioninvestor/08/cds.asp#axzz1ewURiHd1

Investopedia. (n.d.). *International Swaps and Derivatives Association - ISDA*. Retrieved 11 28, 2011, from Investopedia.com: http://www.investopedia.com/terms/i/isda.asp#axzz1f1uTdhn0

Investopedia. (n.d.). *Reference Entity*. Retrieved 11 28, 2011, from Investopedia.com: http://www.investopedia.com/terms/r/reference-entity.asp#axzz1f1uTdhn0

Khan Academy. (n.d.). *Khan Academy*. Retrieved 10 31, 2011, from Credit Default Swaps : Introduction to credit default swaps: http://www.khanacademy.org/video/credit-default-swaps?playlist=Credit%20Crisis

MSNBC. (2008, 09 15). *Bank of America to purchase Merrill Lynch*. Retrieved 01 08, 2012, from MSNBC: http://www.msnbc.msn.com/id/26708958/ns/business-us_business/t/bank-america-purchase-merrill-lynch/

Nomura Fixed Income Research. (2004). *Credit Default Swap Primer.* Tokyo.

O'Kane, D., & Turnbull, S. (2003, April). Valuation of Credit Default Swaps. *Lehman Brothers Quantitative Credit Research Quarterly*. Lehman Brothers.

Pinsent, W. (2008, 06 04). *Credit Default Swaps: An Introduction*. Retrieved 11 28, 2011, from Investopedia.com: http://www.investopedia.com/articles/optioninvestor/08/cds.asp#axzz1ewURiHd1

Python (programming language). (n.d.). Retrieved 01 11, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Object_(computer_science)

Python Software Foundation (PSF). (n.d.). *Python Software Foundation*. Retrieved 09 27, 2011, from www.python.org: http://www.python.org/

Wikipedia. (n.d.). *Credit Default Swap*. Retrieved 11 29, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Credit_default_swap