Fall 2012

Data Integration: a Case Study in the Financial Services Industry

Louis Epie
Regis University

DATA INTEGRATION: A CASE STUDY IN THE FINANCIAL SERVICES INDUSTRY

A THESIS
SUBMITTED ON 14TH OF December, 2012

TO THE DEPARTMENT OF INFORMATION SYSTEMS

OF THE SCHOOL OF COMPUTER & INFORMATION SCIENCES

OF REGIS UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF MASTER OF SCIENCE IN

SOFTWARE ENGINEERING AND DATABASE TECHNOLOGIES

BY

[illegible]

Louis Epie

APPROVALS

Darl Kuhn

[illegible]

Faculty of Record

Donald J. Ina

Ranked Faculty Name

[illegible]

Abstract

Current economic conditions result in banks participating in multiple mergers and acquisitions. This results in banks inheriting silo and heterogeneous systems. For banks to remain competitive, they must create a strategy to integrate data from these acquired systems in a dynamic, efficient, and consumable manner. This research considers a case study of a large financial services company that has successfully integrated data from different sources. In addition this research investigates endeavors that experts in the field have undertaken to develop architectures that address the problems with data integration and appropriate solutions.

Acknowledgements

To my wife Augusta and daughter Alicia – Thank you for your support, understanding and sacrifice for the last two years.

To my thesis supervisor and facilitator, Darl Kuhn and Prof. Donald Ina, for their advice and guidance throughout the development of this thesis.

To Pioneer Investments IT staff for giving me the opportunity to work with some of the best minds and technologies in the financial industry.

To the staff and students of both NUI Galway and Regis University's MScSIS program 2010 – 2012 for giving me the opportunity to learn

Table of Contents

List of Tables

Table of Figures

Chapter 1 Introduction

Data integration is the set of procedures, techniques, and technologies used to design and build processes that extract, restructure, move and load data either in operational or analytical data stores either in real time or in a batch mode. Financial services spend a great deal of money and time in data integration, combining data from different sources and enhancing the quality of data consumed. Data integration has been frequently cited as being one of the most expensive challenges for information technology companies. "Market-Intelligence firm IDC estimates that the market for data integration and access software (which includes the key enabling technology for information integration) was about $2.5 billion in 2007 and is expected to grow to $3.8 billion in 2012" (Bernstein & Haas, 2007). Data integration could take the form of moving data from one application database to another, translating messages between businesses and providing access to structured data.

To address the issues surrounding data integration, various types of data integration methods will be investigated including works done at an architectural level by experts in the field. A case study is introduced in Chapter 3 to look at a data integration project that was taken in a financial institution showing how integration was achieved, which is the outcome of this project and analysis.

Types of Data Integration

There are many forms of data integration architectures in existence today with many companies and vendors looking at better ways of improving the data integration process. However, most of these architectures have either evolved from traditional integration approaches such as silo architectures or from a combination of modern architectural approaches to data integration. Listed next are some examples of data integration architecture types.

Enterprise application integration (EAI). EAI provides the possibility to integrate transactional data from disparate system sources. In a perfect world, EAI would have a simple architectural pattern. An application will create a transaction, review and update the data for the transaction and commit these transactions. These application environments will consist of enterprise resource planning (ERP) package applications from vendors such as SAP and Oracle to name a few, including internally developed custom applications. Because many companies will have multiple ERP applications internally developed, the act of creating, populating, and committing a transaction is a complex event. EAI is so complex because of the need to bring together disparate technologies to work efficiently together. Figure 1 shows an example of an EAI architecture consisting of a SAP general ledger application, an Oracle order entry application for order entries all sitting on an IBM server.

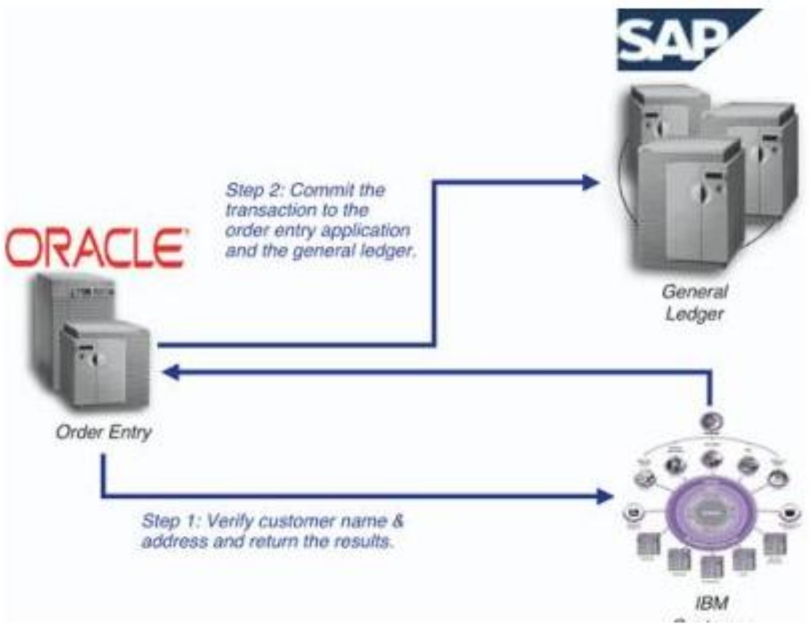
Step 2: Commit the transaction to the order entry application and the general ledger.
General Ledger
Order Entry
Step 1: Verify customer name & address and return the results.
IBM

*Figure 1.*EAI Data integration architecture. From “Data Integration Blueprint and Modeling Techniques for a Scalable and Sustainable Architecture” by A. Giordano, 2011. Copyright 2011 by IBM.

Integration by Federation. Federation is a data integration pattern that has been used in the industry since the 1980s. In Federation, disparate data is combined into a common logical data structure called a virtual database. This virtual database created by federation does not contain the data itself but contains information about where the actual data resides. Federation is advantageous in the sense that it provides a means of integrating an organization's data that is stored off site or that is stored with a third party such as a cloud service provider. Figure 2 below shows an example of a federated architecture with order information hosted on an Oracle database and customer information hosted on a DB2 database.

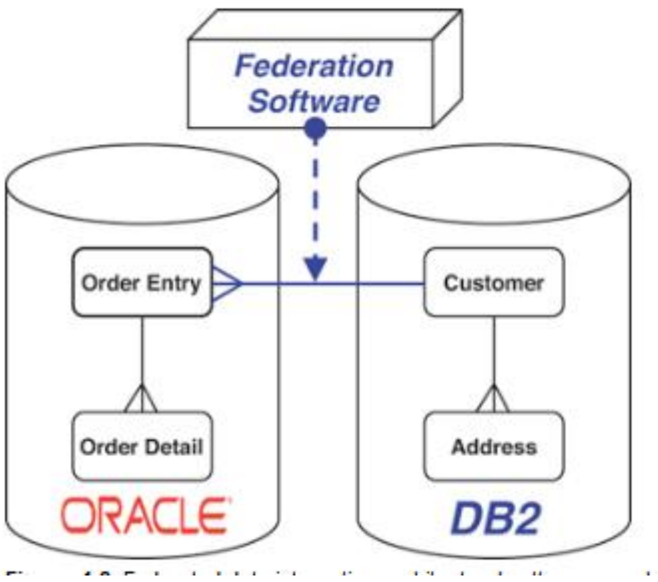

Figure 2. Federated data integration architecture. From "Data Integration Blueprint and Modeling Techniques for a Scalable and Sustainable Architecture "by A. Giordano, 2011. Copyright 2011 by IBM.

Extract, Transform, Load (ETL). ETL is the aggregation and the collection of transactional data. In ETL data is extracted from multiple sources and captured in a database for reporting and analytics. The complexity and cost of data integration in ETL occurs in the bulk of data movement. ETL has had a growth explosion in the past 15 years (Giordano, 2011). In the mid 1990s processing 40GB of data a month was considered significant, but such a volume is nothing as compared to what is happening in the 21st century where processing a terabyte of data a day is commonplace. Apart from flat files and relational data formats, the integration environments in ETL should be able to process XML and unstructured data formats.

ETL also has its complexities, for example integrating data from different source systems with different batch processing times makes it difficult to extract or capture such data after the batch run causing latency issues. In addition traditional ETL designs had just a single data integration process for extracting, integrating, validating, and loading data causing synchronization issues. ETL architectures are best for non-real-time transactional data especially where there is a lag between the times when transactions are needed and when they are created. Figure 3 shows ETL integration architecture from the extraction of customer information and to when it is loaded and combined.

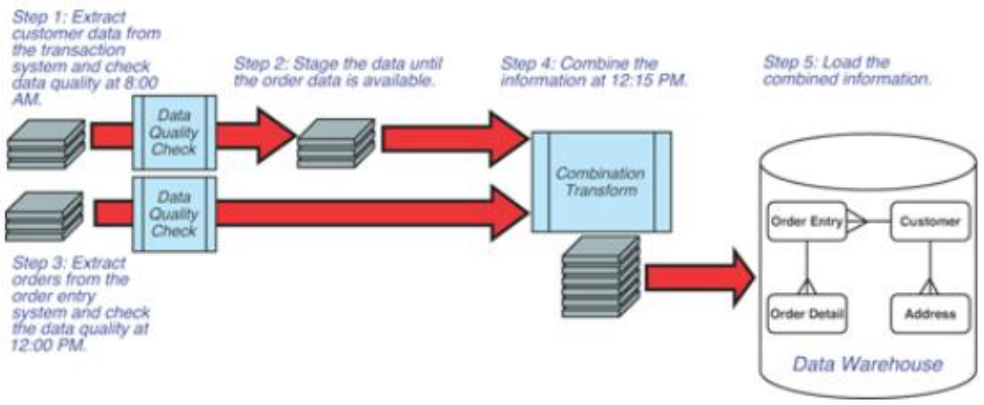
Step 1: Extract customer data from the transaction system and check data quality at 8:00 AM.
Step 2: Stage the data until the order data is available.
Step 4: Combine the information at 12:15 PM.
Step 5: Load the combined information.
Data Quality Check
Data Quality Check
Combination Transform
Step 3: Extract orders from the order entry system and check the data quality at 12:00 PM.
Order Entry
Customer
Order Detail
Address
Data Warehouse

Figure 3. ETL Data integration architecture. From “Data Integration Blueprint and Modeling Techniques for a Scalable and Sustainable Architecture “by A. Giordano, 2011. Copyright 2011 by IBM.

Service-Oriented Architecture (SOA). Service-Oriented Architecture (SOA) is a transactional data integration architecture in which messages are routed to instantiate objects that will perform at different levels, offering services on a common network interface called a service bus (Hurwitz, 2007). These objects are usually representations of functional business components usually instantiated at different levels of granularity (Rosen, 2008). SOA provides a set of guidelines used during systems development and integration. SOA is an architecture that packages functionality as a service. These services can be used by other organizations even when their client systems are substantially different, this architecture is depicted in Figure 4 showing services being extensible in the service bus.

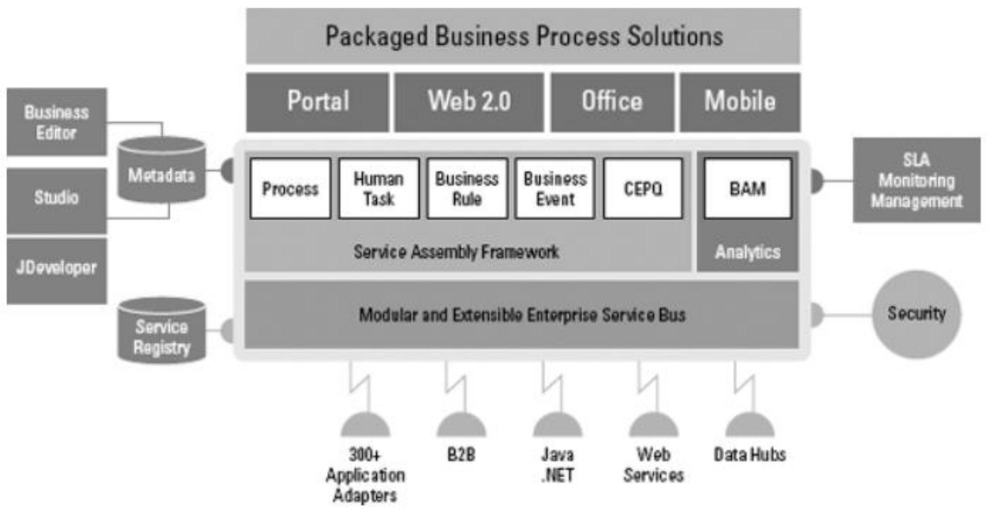

Figure 4. SOA view of the IT world. From “Service Oriented Architectures For Dummies” by J.Hurwitz, 2007. Copyright 2007 by Wiley Publishing.

Reasons for the Challenges of Data Integration

Data integration is one of the most complex disciplines in the financial industry. This complexity is a result of having to combine multiple and distinct source systems into a single and consistent data store for the use of the business and technology users. Fundamental changes in the financial markets industry is the main cause for the challenges of integrating data in the banks. Some of these challenges include the following:

- **The Issuance of New Securities***:* Because of the need for investment banks to offer compelling products to their clients, many financial securities have been created. There are currently more than eight million securities each requiring timely, accurate, and detailed information. These financial securities and their complexity are a challenge for managers of financial information in these institutions (IBM.com, 2004).

- **Changes in the Market***:* The market of today has so many participants executing trades both on the buy and the sell side. Many of these firms use trading and other algorithmic execution models. Decimalization and program trading has led to a high reduction in trade size with a corresponding high volume, these factors have led to a strain on data management platforms because they require delivery of a high volume of data (IBM.com, 2004).
- **Regulation and Compliance:** Regulation and compliance considerations have forced banks to place a high priority on the production of accurate, timely data to feed the internal risk management systems. Because of this institutions are now forced to meet a more stringent and fiduciary responsibility to provide correct data to regulatory agencies.
- **The Expanding Role of Data Vendors:** Vendors of data such as Bloomberg, Reuters and Factset are now providing disparate security attributes and pricing information to investment banks as a result of a need for more attributes and analytics from these banks who are in need of more insight into the markets. However, managing multiple sources of data by these banks creates cost and consistency issues that must be addressed. These challenges have created significant stress on data management systems resulting in technological changes and business processes.
- **Similar Data Platforms:** Most banks historically have built and maintained their own security and client master database in isolation from other market participants. As these banks have expanded through mergers or organic growth, data silos have emerged that match each line of business. These data platforms are similar in style and content across the bank, they are maintained through a combination of automated

data feeds from external data vendors, in-house applications and manual entries and adjustments.

- **Difficulty in Data Cleansing:** Most often, the approach to cleansing data is to compare data from multiple vendors or data sources. This process is tedious due to the lack of standards across the financial industry. Every data provider has a proprietary way of representing data due to the lack of standards to govern data presentation. These difficulties have resulted in a high probability of data error as these errors can occur at the original source, the data vendor or any number of in-house data applications (Zins, 2009). Figure 5 below shows an architecture with multiple data sources.

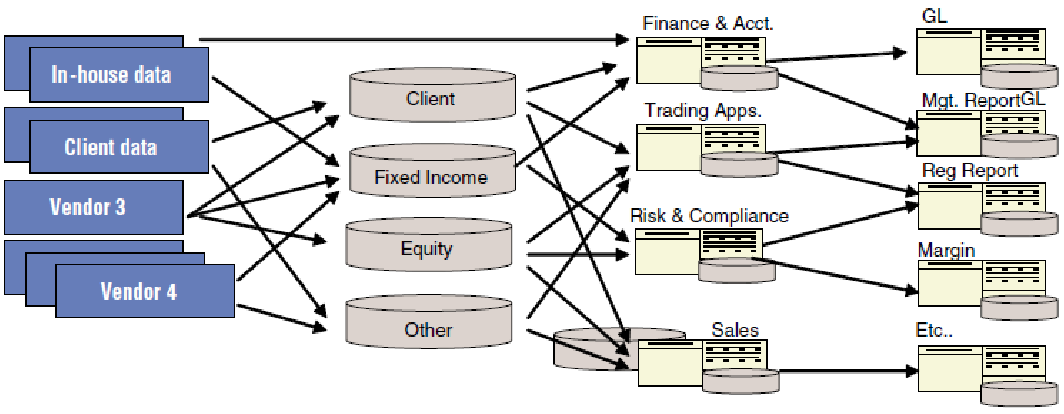

Figure 5. Multiple data sources. From "Transforming Data Management in the Financial Markets Industry" by IBM Business Consulting Services, 2009. Copyright 2009, by Author.

Thesis Statement

Integrating data in the financial industry presents complex challenges from a technical, design, and operational perspective that requires expertise to implement a system that collects and centralizes information in a manner that can be efficiently leveraged by the business users.

Scope

Integrating and managing data is an activity that almost every industry that relies on data must undertake. The scope of this work is limited only to the integration and management of data between the back, middle, and front office of an investment bank.

Significance of the Study

Successfully managing and integrating data plays an important role in investment banks as this gives them the ability to compare data from numerous sources and choose the best, latest, and most up to date data thus yielding higher investment returns.

Research Methodology

This thesis uses a two-phase approach. Phase one will examine existing literature in the general IT domain and the financial industry domain. This literature review will examine existing academic and industry research, books in the field of data quality, master data management, IT architecture and reference data systems.

Phase two will examine the design, implementation and ongoing operations in a financial services company with respect to integrating data.

Success Criteria

This paper will provide an overview of data integration issues in the financial industry and architectures currently used in integrating and managing data. It will also explore the benefits of utilizing these methods and technologies and how such an exercise was achieved within an investment bank in the sample case study.

Chapter 2 Literature Review

Financial institutions particularly investment banks have always been in need for the best source of information and data in order to provide customers and clients with the optimal service to increase profitability and to outperform competitors. This need for higher profits, a better customer service, and a need to outperform competitors has led to enormous pressure on the IT department of these institutions in the area of data integration.

The purpose of this chapter is to look at research and studies that has been done in the area of data integration architectures in the financial industry and in similar industries and how this can be incorporated by financial institutions particularly investment banks. More specifically, this chapter will explore what data integration is, the operations of financial institutions around data, the issues affecting data integration, the available data integration architectures and data store houses.

An Overview of the Operation of Investment Banks

Large global investment banks are typically composed of several business units, these include: Investment Banking which is concerned with advising public and private corporations; Research is concerned with producing reports on the valuation of financial products; and Sales & Trading, which is more concerned with the buying and selling of products on behalf of the bank and the client's of the bank. Banks make money by undertaking risk through proprietary trading, which is done by traders of the bank and they seek to maximize profitability for a given amount of risk.

Investment banks are split into three main areas, these are the Front Office, Middle Office and Back Office with the Front Office considered to have the highest-calibre employees and the Back Office having the least (Essvale, 2006).

Front office. This department of an investment bank helps customers raise funds in the Capital Markets and advises on mergers and acquisitions. Investment bankers in the Front Office prepare ideas which they bring to meetings with their clients with the hope that their efforts will help secure clients in a particular investment opportunity. Once such an investment opportunity has been agreed upon, the bank prepares all the documents necessary for the execution of such a deal, this may involve subscribing investors to the issue of a new security, negotiating with a merger or coordinating with bidders.

Financial Markets are split into four main divisions: sales, trading, research, and restructuring.

- *Sales and Trading*: This is the most profitable area in an investment bank as it responsible for most of the revenue. Traders in this division will buy and sell financial products with the goal of making an incremental amount of money on each successful trade. The function of the sales team in this division is to call on institutional and high-net-worth investors, suggesting trading ideas to them, and taking orders. Sales desks then send forth their clients' orders to appropriate trading desks to price and execute trades.
- *Research:* This division reviews the prospects of other companies and gathers reports. Although this division does not generate any revenue, their resources are used in assisting traders in trading, suggesting ideas to the sales force in other to capture the interest of clients. In the last few years the relationship between investment banking and research has become highly regulated reducing the importance it once had to investment banks (Essvale, 2006).
- *Structuring:* This is a relatively new division for investment banks with the emergence of Derivative products. This division of investment banks employs highly

technical and numerate employees who create complex structured products that offer greater margins and returns.

Middle office. This division houses the risk management team which is responsible for analyzing the risk taken by traders when conducting their daily trades, in addition they are also responsible in making sure that their balances are in order on the balance sheet. This division is also responsible for making sure that traders have the right amount of capital to trade in order to prevent “bad” trades having a detrimental effect on the bank.

Back office. The operations in this division involve checking the data of trades that have been conducted over a certain period of time, making sure that they are not erroneous and that the required transfers are transacted correctly. The operations in this division will move smoothly if data from both external and internal sources are well integrated together. It is this area of the bank where the majority of data integration work is done. Figure 6 shows a layout of the Front, Middle and Back offices.

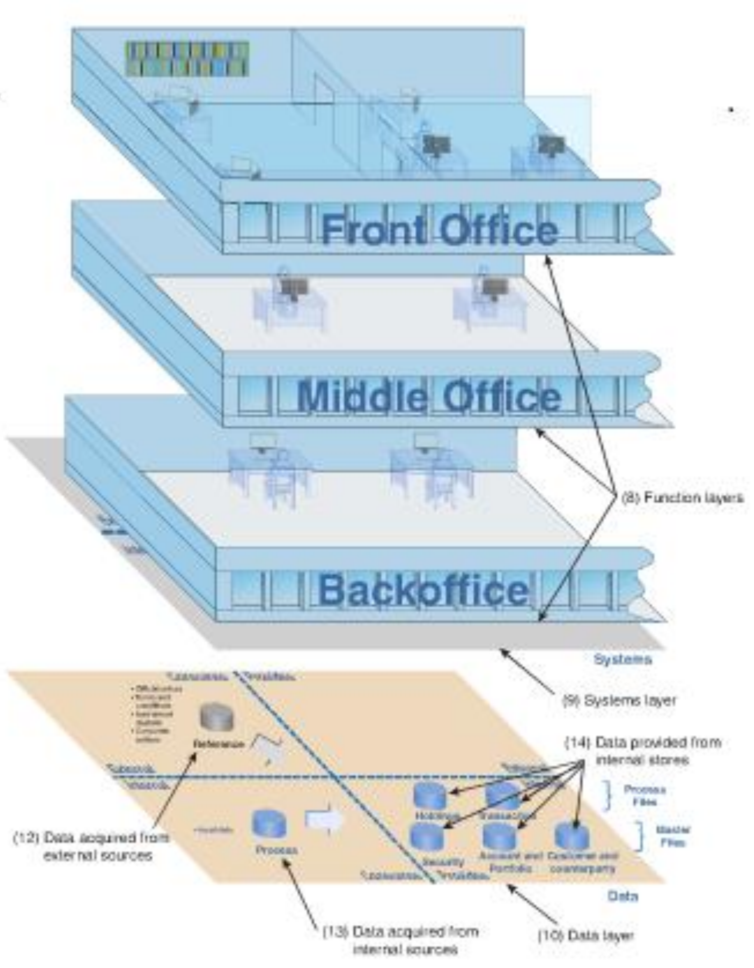

Figure 6. Data supporting buy & sell side of an investment bank. From “An Introduction to Trading in the Financial Markers Technology: Systems, Networks, and Data” by R. T. Williams, 2011. Copyright 2011 by Academic Press.

The Data Situation in Investment Bank

The changing landscape of the trading of securities in investment banks has put issues surrounding data management in the spotlight. With the advent of electronic trading, there has been high-speed and high volume of data for almost everything, from market quotes to the confirmation of payments. Increased regulations have put pressure on market participants to store large volumes of data for reporting and auditing.

Data accuracy has always been very important at every stage of the trading lifecycle, from transaction flows, analytics and risk management, accounting and reporting affecting every part of the front, middle, and back offices. In the past, issues with data have created bottlenecks in workflows, with data being inaccurate, untimely, and inaccessible.

> “According to a survey published by Tower Group in 2005 institutions reported that 9% of all trades failed automated processing routines. Inaccurate reference data was the cause of 36% of these failed trades (or it caused fails in 3.3% of all trades), and poor data management processes led to an additional 43% of failed trades”.

Due to competition and shrinking profits, investment banks cannot afford the cost that comes from data error and are rethinking the importance they have traditionally put on data as an asset. Data is now looked upon as a key revenue and profit generation in the firm. These days it is common to find titles such as CEO (Chief Executive Officer), CFO (Chief Financial Officer), CIO(Chief Information Officer) and the Chief Data Officer (CDO) who have been given the task to provide accurate integrated data that is accessible in real-time, 24/7, from anywhere in the world (Khanna, 2008). Despite changes in the market and the creation of roles and titles to manage the data that is being influenced by these market changes, the industry is still struggling to create a coherent strategy for data management requirements and is ill prepared to meet the recent challenges in the market.

For many years these banks have haphazardly tried to accommodate new demands to accommodate large volumes of data types, with more applications needing access to this data and the need for greater speed to access this data. Databases have been added in an ad hoc manner to serve the needs of specific departments even when the data already exists in some other locations. Some firms have been known to have over 50 databases storing the same data in different formats with varying levels of accuracy. Others have created their own databases, taking data from various and multiple data vendors such as Bloomberg, Telekurs and Reuters. Because there is no industry standard for representing reference data, data vendors have been known to supply data in disparate formats using different naming conventions.

Apart from having redundant data and the costs associated with redundancy, data vendors sometimes provide different and incomplete data for the same record of information (Loshin, 2011). This means that for a particular security, two distinct values could be held for it in different databases in the same firm, in addition to that if one of the vendors had provided an incomplete record, it is most likely to be completed by the Middle Office manually which brings in the possibility of human error.

For decades, most investment banks have had decentralized and fragmented reference data acquisition and storage processes containing poor quality reference data which is inconsistent among its data stores. Figure 7 below depicts an architecture with data stuck in silos systems.

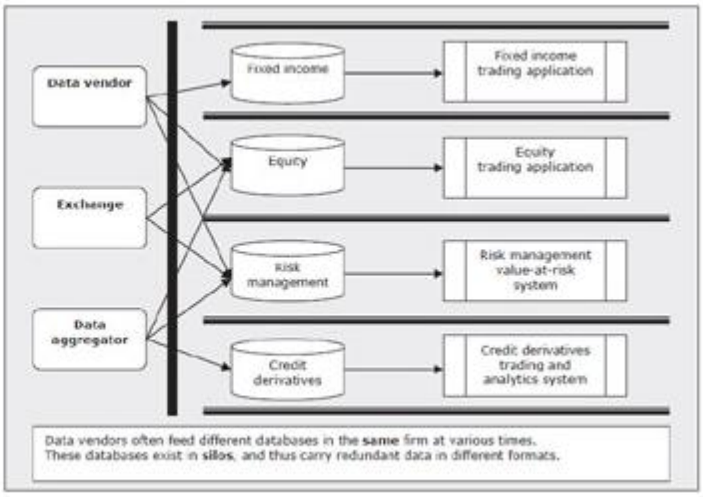
Data vendor
Exchange
Data aggregator
Fixed income
Equity
Risk management
Credit derivatives
Fixed income trading application
Equity trading application
Risk management value-at-risk system
Credit derivatives trading and analytics system
Data vendors often feed different databases in the same firm at various times.
These databases exist in silos, and thus carry redundant data in different formats.

Figure 7. Data stuck in silos. From “Straight Through Processing For Financial Services. The complete Guide.” By A.Khanna, 2008. Copyright 2008 Academic Press.

Due to the prior mentioned issues affecting data, especially its architecture, it is worth looking at the various architectural approaches that are central to the creation of a robust data integration solution.

Data Integration Architectural Approaches

There are many architectures that have been used for data integration not only within the financial industry but in other industries such as the pharmaceutical industry, the motor industry, and telecommunications industry. Due to these various and varying architectures it was worth reading up on some of these architectures to have an understanding of the various methodologies available for data integration.

Web services and enterprise information integration architectures. Pan and Vina (2004) considered an alternative architecture that has already been deployed in several cases. Their architecture is based on two emerging paradigms in the industry, which are Web services and Enterprise Information Integration (EII). According to Pan and Vina, Web services are becoming the de facto standard for the interoperation between different software applications. EII systems are based on a wrapper-mediator architecture, where data from various sources is transformed into a new central repository. This means data always remains at the source and the EII system provides users with a unified virtual view of the source data. When a query is received by the mediator, the query is decomposed into sub-queries from the data sources, which are then executed in real time and the integrated sub-query results is obtained to form a global unified query result set. Figure 8 depicts an architecture with a wrapper creator at its core.

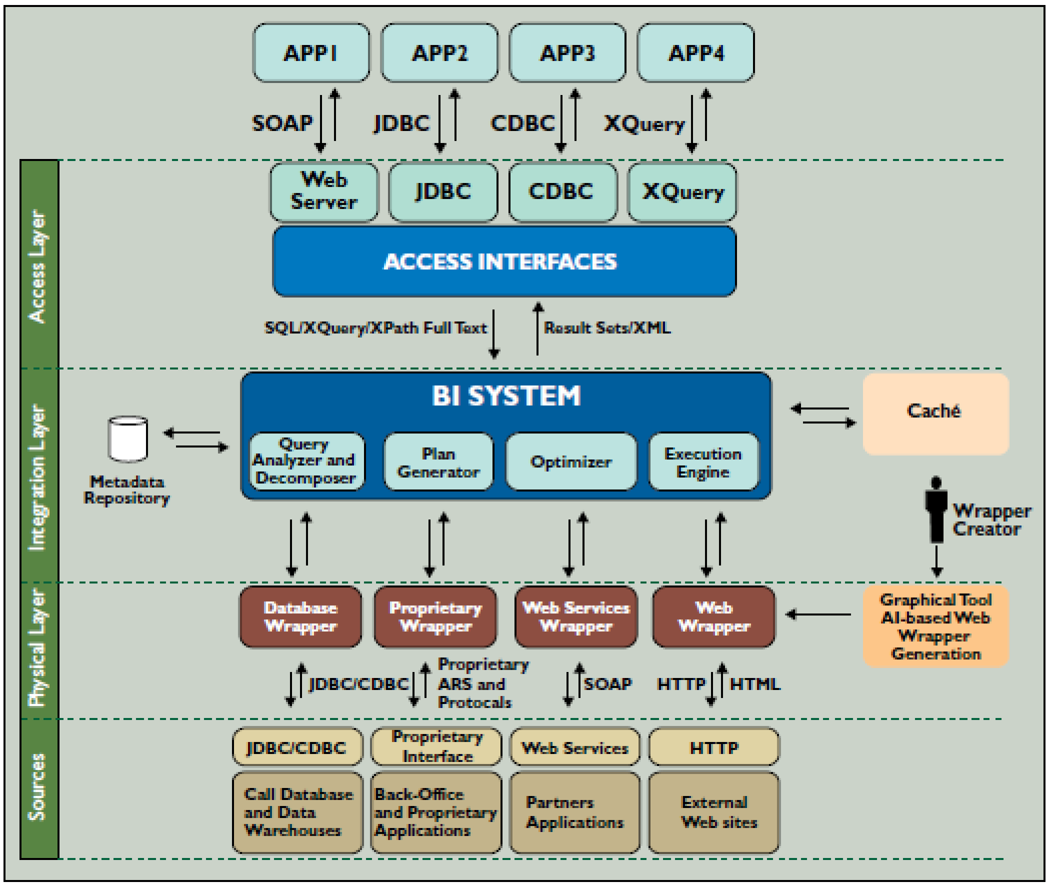

Figure 8. Data integration architecture. From "An Alternative Architecture for Financial Data Integration" By A.Pan and A.Viña 2004, *Communications of the ACM, 47*(5), 37-40. Copyright 2004 by ACM.

This proposed data integration architecture is based upon EII wrappers and Web Services as shown in Figure 8. The sources of data in the above architecture are internal databases, proprietary applications, systems from partner companies or from other departments accessible through Web services and autonomous Web sites from external organizations. In this architecture, the physical layer is comprised of wrappers that make data sources conform to a common model. Wrappers that are easily configurable, Web services, and popular back-office applications are included as standard components for this architecture.

Despite the benefits of this architecture, the disadvantage with it is that integrating data from external Web sources requires screen-scraping techniques that can be error prone. The integration layer of this architecture provides a unified view over data from sources that could be queried using a database-like language such as XQuery or SQL. This allows users to obtain answers to queries that are similar to conventional databases.

The prior architecture proposed by Pan and Vina (2004) has a number of advantages over traditional integration techniques:

- Data is retrieved from the sources at query time, which allows for real-time data integration and is an asset to financial institutions.
- This architecture is inexpensive to build and less difficult to maintain as compared to a central repository. They can be scaled up to a large number of sources, which can be ideal when a fast deployment is needed for a unified system after a merger or acquisition of a bank.
- External data sources can be easily dealt with since data resides with the external sources and the owner of the source data can decide which data can be accessed.
- Using Web wrapper techniques gives the user the illusion that data from the external websites is stored in a local database.

Despite the prior advantages, it is hard to judge the performance of an application because no mention is made of performance times in any particular setting. It would also be good to know how this architecture will perform in a batch publishing setting because most financial institutions rely on batch jobs running at various times of the day.

Model driven architectures. Model driven architecture is another method considered for data integration by Kim, Zhang, Oussena and Clark (2009). This approach of data integration utilizes metadata across the data integration process. By having data and metadata decoupled by using a Model Driven Data Integration (MDDI), the complexity involved in data integration is eliminated according to this study.

A case study adopted by (Kim et al., 2009) looked at adopting Meta Data Architecture (MDA) technology as a framework for data merging and data customization in data mining. This case study was done for the Mining Course Management Systems (MCMS) Project. In the case study, 3 years of institutional and historical data was collected to build a data warehouse to predict individual student performances and the dropout rate as well. The data source for this study included the following:

- A student record system that has held information about students' background, examination results, and course enrollment.
- An Online Learning System, which has allowed tracing students' activities in a virtual class, included the downloading of class materials and joining online group discussions.
- A library system, which has provided information on student loans on academic material and associated literature.
- A reading list system has been hosted on the library system server in with a separate database. This is to help track how often students borrow books from the recommendation list.
- An online resource system to store student logs of access to online resources such as e-books and online journals.
- A specification document to provide information on the course. Text Ming was applied to find out the title of the course, the methods of teaching and much more.
- A text data source, which has provided information on the module study guide. All details of the modules were contained therein.
- A course marketing system for marketing the course.
- An online test system, which has enabled students to take an online entrance skill check.

The overall architecture of this case study can be seen in Figure 9.

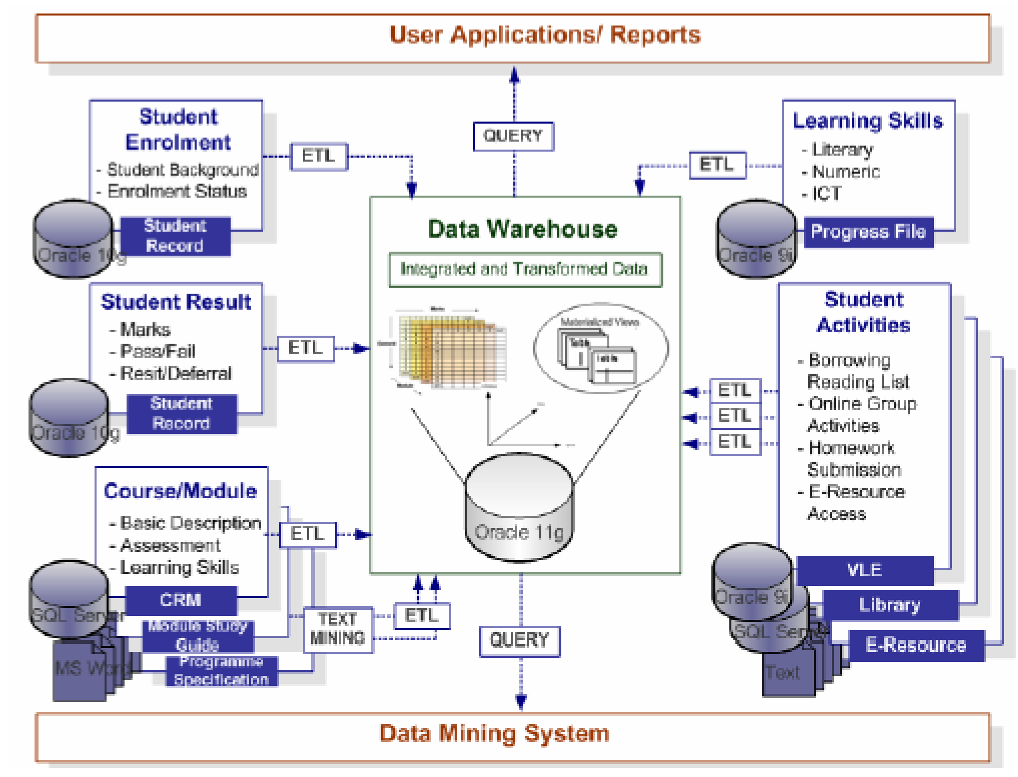

Figure 9. MCMS system overall architecture. From "A case study on model driven data integration for data centric software development". By Kim et al.,2009. DSMM '09 Proceedings of the ACM first international workshop on Data-intensive software management and mining, 1, 1-6.

The data integration process of this project was divided into four phases, which were as follows:

1. Analyzing and understanding of data*:* In this phase business requirements were gathered, the quality of these requirements were assessed and associated with the source system across multiple other systems. Data models were gathered from each data source systems.
2. Data collection and extraction: The gap between available data and its quality based on the requirements of the business were assessed in this step. For this the data extraction models for each data source was defined using a UML class diagram.

3. Data merging and cleansing: These merging models were designed to combine each data extraction model into a unified model while considering data cleansing at the same time.
4. Data customization: This was the final step, and was designed according to an application *scenario.*

The prior four phases are depicted in Figure 10.

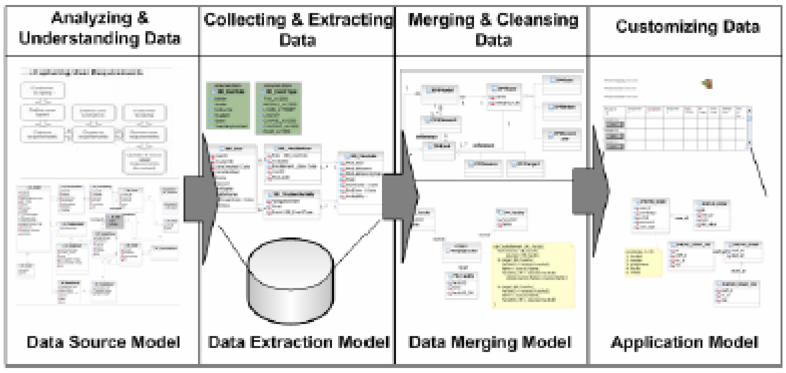

Figure 10. MDDI modelling phases From “A case study on model driven data integration for data centric software development”. By Kim et al.,2009. DSMM '09 Proceedings of the ACM first international workshop on Data-intensive software management and mining, 1, 1-6.

The transformation of the logical models provided guidance on how to integrate actual data and an automated generation of real code for execution from an MDA viewpoint. Most of the data was derived from real data sources through automated reverse transformation using Rational Data Architect (RDA) in this project. Figure 11 shows the transformation of these logical models.

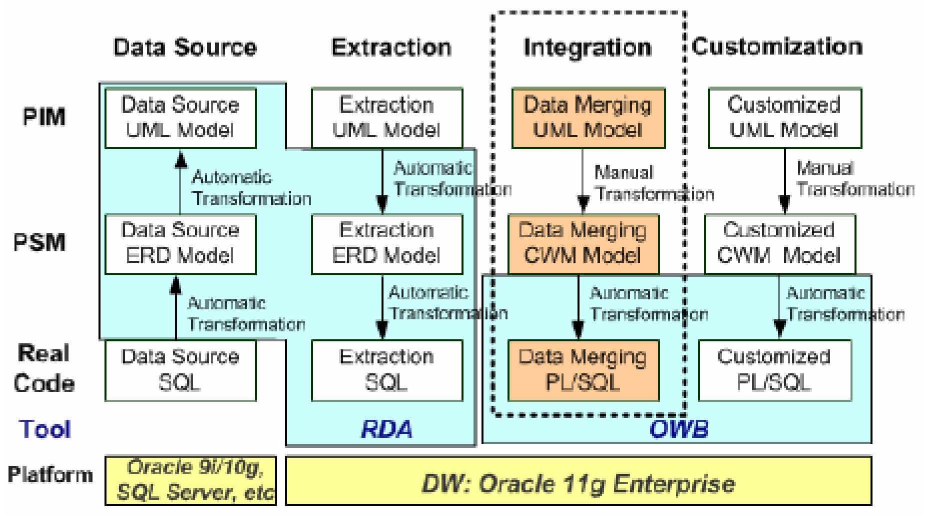

Figure 11. Transformation and models for MDA based on MDDI modelling phases, From "A case study on model driven data integration for data centric software development". By Kim et al.,2009. DSMM '09 Proceedings of the ACM first international workshop on Data-intensive software management and mining, 1, 1-6.

To achieve data merging, relationships were defined between elements from different models at the conceptual level. This enabled data mapping from each source data to the target data. This case study showed a very rigid and successful process of integrating data using data modelling techniques. However, it would be fair to question how such an approach will work in a fast-paced environment where there could be 20 to 40 or more heterogeneous systems or applications at any one time. A modelling approach may be time consuming resulting in such an architectural approach having to compete with other approaches that are not as model intensive as the above-mentioned architecture. Also most off-the-shelf data integration products these days will have their four modelling phases already integrated into their applications as functions, with institutions having only to understand the functionality of such an integration product before purchasing it as a solution.

Data delivery in a service oriented architecture. Service Oriented Architecture (SOA) has been a trend the enterprise that has excited the IT world. Carey's (2005), in his study, using the BEA AquaLogic Data Services platform, explored the opportunities created by the emergence of Web Services and XML standards for enterprise application integration. BEA leverages the W3C XML, XML Schema and XQuery recommendations to deliver a standard based foundation for data services.

The BEA AquaLogic Data Services Platform (ALDSP) is aimed at developers of composite applications that need to access information from a range of enterprise data sources. These data sources include Web services, relational databases, packaged applications, views, delimited files, and Java-fronted custom data sources. ALDSP looks at data from a service-oriented point of view and models an enterprise from a set of interrelated data services. Each data service provides a set of service calls that an ALDSP client can use to access data with most of these services being made up of a mixture of read and write methods.

To show how a service is built using ALDSP, suppose there is a need to build a logical data service that enables applications or other services to access customer information and that this information resides in several different enterprise information sources. Now imagine that one relational database contains the CUSTOMER and ORDER table and the other database contains CREDIT CARD information. Although there is a WEB service hosted within the enterprise that provides credit rating information, Figure 12 shows the physical data service model because of having ALDSP introspect these data sources and make them available by creating a composite data service.

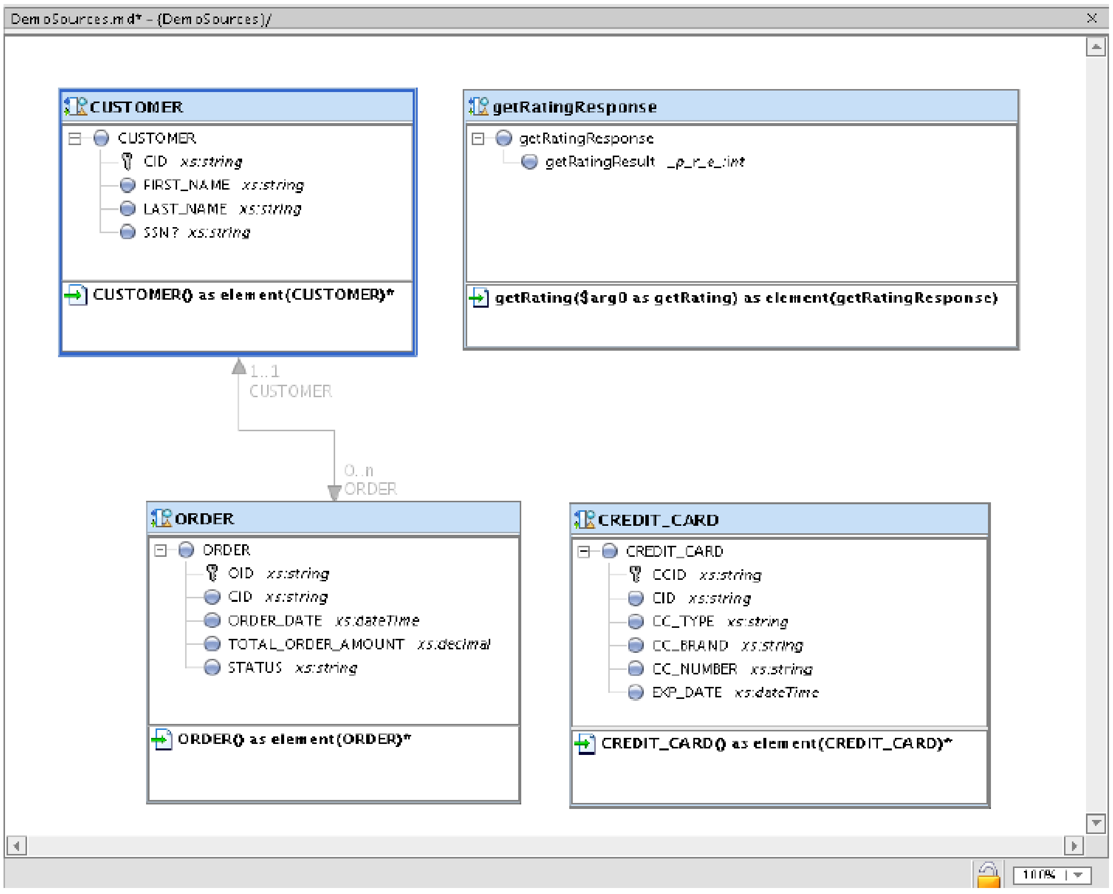

Figure 12. Physical data services. From "Data Delivery in a Service-Oriented World: The BEA AquaLogic Data Services Platform" by M. Carey, 2005, *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 695-705. Copyright 2006 by ACM.

Central to this service are instances of the customer data, order data, nested credit card data and the credit rating for a given customer. The left hand design view lists the read methods which include getProfile(), to get integrated profile for all customers, geProfileById() to get the profile of a customer by customer id, getProfileByName() to get the integrated profile of all customers with a specified name and getProfileByRating() to get profiles for all customers with a specified credit rating. The design view of the physical data services is depicted in Figure 13 below showing the ALDSP design view of the integrated customer profile data service

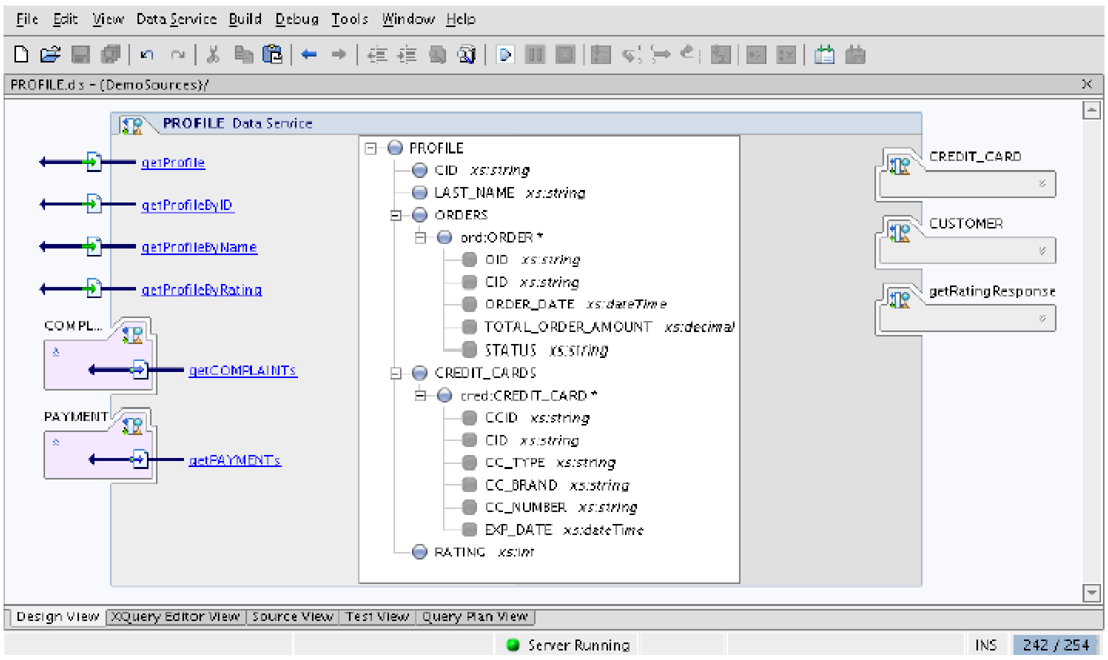

Figure 13. Design view of customer profile data service. From “Data Delivery in a Service-Oriented World: The BEA AquaLogic Data Services Platform” by M. Carey, 2005, *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 695-705. Copyright 2006 by ACM.

The aim of ALDSP is to make it possible for developers to develop data services which can be developed once and reused anywhere and everywhere for data integration. However, for this to happen, the following criteria needs to be met:

- Every data service needs to be efficient once executed; it needs to be efficient as naturally written code.
- Any data service call which is based on XQuery needs to be efficient with a runtime price not being negligible.
- A query or data service call that uses only a subset of its results from an underlying logical data service call must still be efficient.

Only when these assurances are actually provided will developers be able to use ALDSP to build data services in a fashion that offers the greatest benefits in integrating data.

The thought of having an application such as ALDSP being used by developers to build services once as a set of functionality to be used anywhere seems to be an approach worthy of consideration when integrating data (Dong, Halevy, & Yu 2009). The reason for this is other applications or clients that are consumers of data may only have to subscribe to these services for the consumption of their data. This approach definitely seems to be less model intensive as compared to that considered by the Model Driven Architecture and easy to implement giving the scale of applications that are found in most financial institutions.

A master data model integration using SOA. Service-oriented architecture (SOA) represents a model where automation logic can be decomposed into smaller distinct units of logic (Erl, 2005). It is also a model that allows for the integration of other data sources as services to other applications or systems thereby allowing for data integration in a heterogeneous environment.

Krizenvick and Juric (2010) showed how a Master Data Model (MDM) is used in SOA architecture to achieve data integration. In their study, the main idea of an MDM is to be able to isolate key data from existing applications into a centralized repository that will provide a full data management and a set of services by which applications can access and manage these data. One of their main goals was first to build a data service layer which is a main requirement in adopting a SOA architecture. Krizenvick and Juric discovered that implementing master data services is usually very complicated since changing master data is usually restricted with the use of policies. Using SOA architecture provided the needed flexibility for the building of a master data service because of the following functionalities:

- The ability of message routing
- Easy Transformation of data formats and communication protocols
- Adapters for accessing data from heterogeneous data sources.
- Human workflows for multi-level confirmation of master data changes.

- The availability of mechanisms for monitoring and managing SLA and (Quality of Service) of data services.

With such an approach based on SOA architecture, key business data (master data) was stored in the MDM repository. The existence of an Enterprise Service Bus (ESB) provided useful functionalities such as message routing, conversion between transport protocols and data formats and much more. There was also the flexibility of implementing complex master data services using Business Processing Extraction Language (BPEL). This approach hid the complexity of the MDM architecture and provided a “virtualized golden record” at the business level. Figure 14 shows the implementation of a master data services layer.

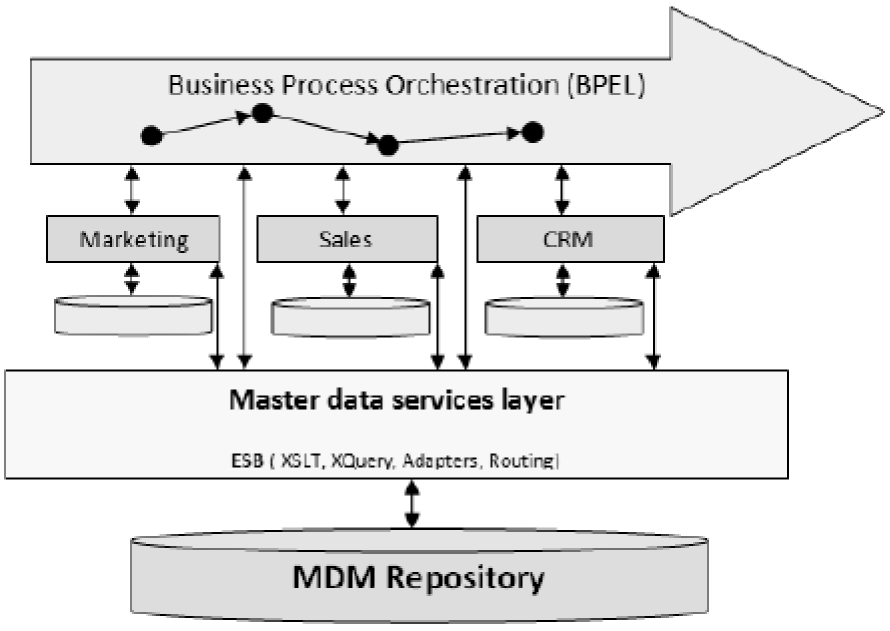
Business Process Orchestration (BPEL)
Marketing
Sales
CRM
Master data services layer
ESB (XSLT, XQuery, Adapters, Routing)
MDM Repository

Figure 14. Implementation of master data services layer. From “Improved SOA Persistence Architecture Model,” by M. Krizenvick and M. Jurik, 2010, *ACM SIGSOFT Software Engineering Notes*, *35*(3), 1. Copyright 2011 by ACM.

A unified data access from various heterogeneous data sources is provided at the service layer in this architecture. It is built in such a way that tight coupling between the business level and data sources are avoided using Service Data Objects (SDO).

SDO is a data programming architecture and an API for data application development (Krizenvick & Juric, 2010). With SDO there is no need to be familiar with a technology specific API such as (JAXB, DOM or JPA) to access data. All operators need to know is the SDO API, which allows them to work with data from multiple data sources including relational databases, entity EJB components, Web services, and XML pages (Yoder & Johnson, 2006).

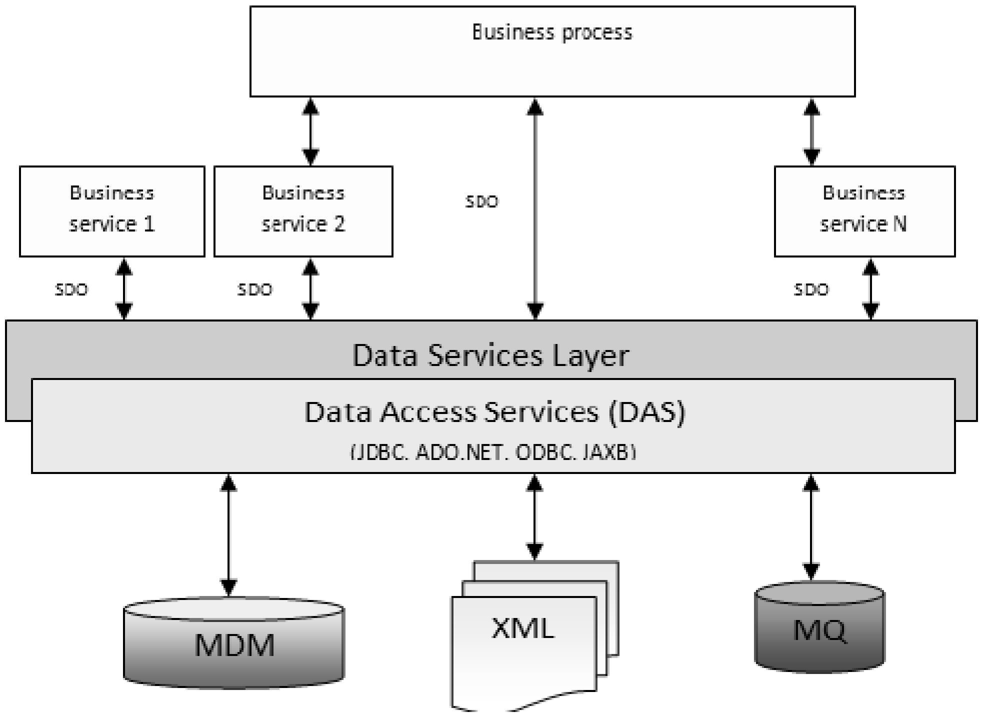

Figure 15. A Data services layer supporting SDO. From "Improved SOA Persistence Architecture Model", by M. Krizenvick and M. Jurik, 2010, *ACM SIGSOFT Software Engineering Notes, 35*(3), 1. Copyright 2011 by ACM.

As can be seen in Figure 15, the data services and business services exchange data using SDO data graphs are envelopes around the data object. This makes it easier building data services because there is a uniform way of transferring and using heterogeneous data, regardless of the data source.

This proposed architecture will no doubt sit well in a banking environment where data is sourced from so many systems having various formats, and kept in a central repository as a "Golden Copy" before being published to other applications. However, the proposed architecture as mentioned by the author comes with a few challenges such as architecture complexity, higher start-up cost, difficulty in calculating the Return on Investment (ROI), which could be a put-off when considering such an architectural approach as a data integration solution.

A business object model integration. Refactoring patterns are also an alternative to achieving data integration. Refactoring patterns look at business object models of external data sources and find out how these could be integrated into a process driven architecture.

Henritch & Zdun (2006) looked at three main refactoring patterns: Wrap Service As Activity, Restructuring Specific Business Model and Synthesize Business Object Models. According to Henritch and Zdun, these three refactoring processes explained alternatives for refactoring a single business object model into a process-oriented architecture. However, they mentioned that for large systems it is important to consider multiple refactoring of the business object models and their interdependencies.

Wrap service as activity. Sometimes the external data source systems do not have the requirements for a process driven architecture. With most banking architectures being SOA oriented nowadays it is often difficult to integrate an external data source if the interfaces it offers are fixed state interfaces, or if the required communication protocol is not supported by the external system. Flexible interfaces to external systems are inevitably required in such scenarios to assemble processes involving external system invocations from within a process design tool. However, loose coupling would be difficult to achieve if the external system is hard code dependent on fixed interfaces.

Hentrich & Zdun (2006) suggested refactoring the external systems and process driven SOA architecture by defining one or two services for each entity in the external system, and defining a special service activity type in the process engine that will wrap invocations to external services. The main function of the service will be to translate all service based invocations into the interface of the external system and to also translate the response back into a service based reply. Figure 16 below shows refactoring using Wrap Service as Activity.

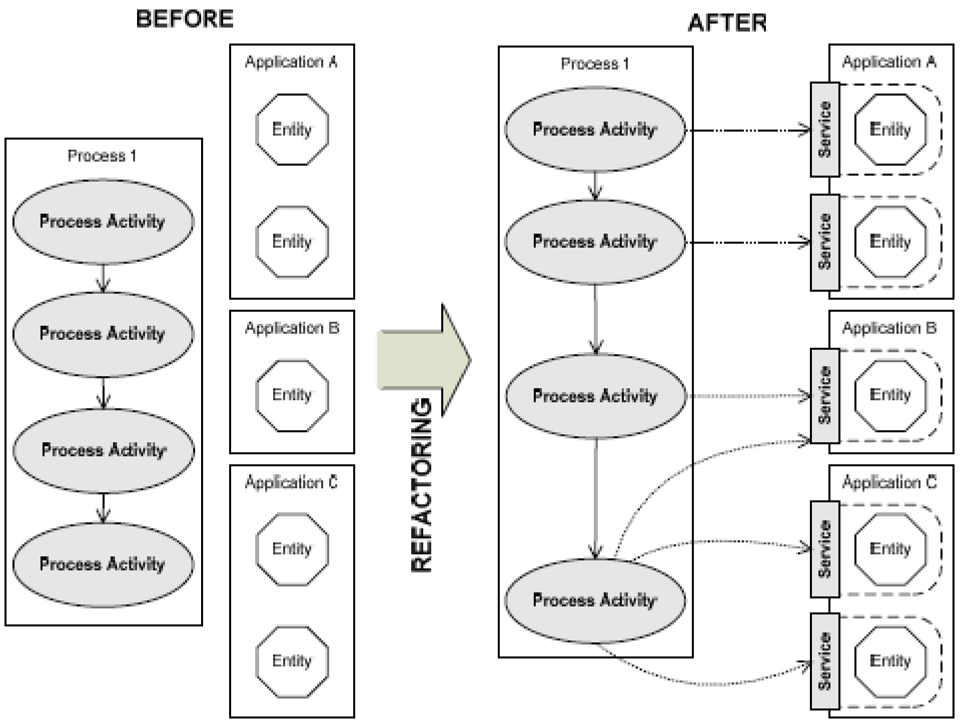

Figure 16. Refactoring services wrap service as activity. From "Patterns for Business Object Model Integration in Process-Driven and Service-Driven Architectures," by C. Hentrich and U. Zdun, *Proceedings of the 2006 Conference on Pattern Languages of Programs*, No. 23. Copyright 2010 by ACM.

Restructuring specific business object models. Wrap Service as Activity should be the first step when integrating systems into a process-driven architecture. However, this will not always work as the external legacy system may not be structured in a suitable way to allow for an object model through services. Also, the business process may require an integration of data from two or more business object models that are application specific and the service access is not good enough to deal with problems of integration. When faced with two external systems that are not compatible it is better to opt for restructuring rather than performing individual mappings.

For restructuring to take place, one has to make sure that the evolution of the system is non-intrusive as much as possible. Existing client code should not be broken and substantial portions of the system should remain unchanged. If such an assessment is positive then the application specific business model of the external system can be restructured by evolving the system to meet the new requirements of the process-oriented architecture. Once this is done, service interfaces can be offered so that the business process can access the evolved external system.

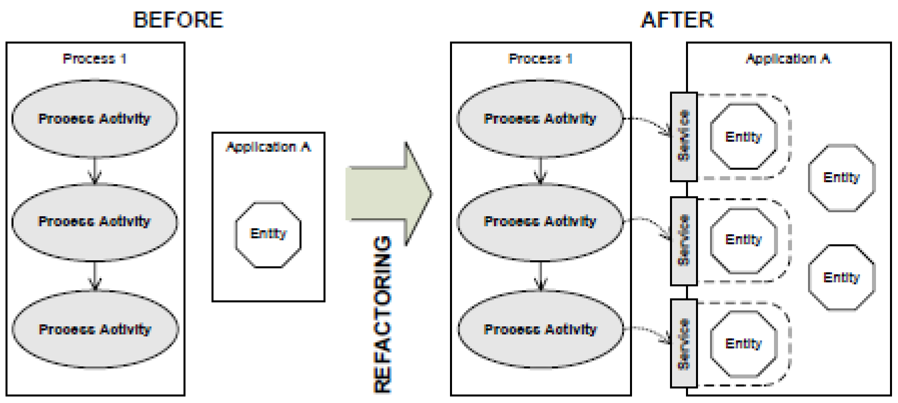

Figure 17. Refactoring by restructuring. From “Patterns for Business Object Model Integration in Process-Driven and Service-Driven Architectures,” by C. Hentrich and U. Zdun, *Proceedings of the 2006 Conference on Pattern Languages of Programs,* No. 23. Copyright 2010 by ACM.

Figure 17 illustrates a refactoring based on the construction of an application-specific business object model. Refactoring is achieved by having one monolithic entity being split into a number of entities with some of them being split as services. There are other ways the restructuring process can be achieved. The aim, however, is to preserve existing code as much as possible.

Although the above refactoring process described by Henritch & Zhun sounds easy to execute, it is difficult to see how this restructuring will fare when it involves a variety of heterogeneous systems or applications with hundreds of entities. Therefore, the task of having each monolithic entity reviewed and split will be a very daunting task, prone to error, and time consuming when dead lines have to be met. However it may be feasible when the entities involved are minimal.

Synthesize business object models. This process of refactoring occurs when using Wrap Service as Activity fails because of data integration issues and the restructuring of Business Object Models proves to be infeasible or difficult to achieve. To begin the process of synthesizing the business object models the following steps must be followed, design a synthesized business object model that consolidates the structures of the involved business model, map relevant application specific business object models into a synthesized business object model, then perform the task of data integration at a global level.

Application specific business object models that have services are usually integrated into a process flow using Wrapper Services and are invoked by activities in the process flow. Figure 18 shows a business process design and two applications that can be accessed via a service interface, assuming that the two applications cannot be changed and there are data integration issues, the diagram illustrates a refactoring process using synthesized business object models.

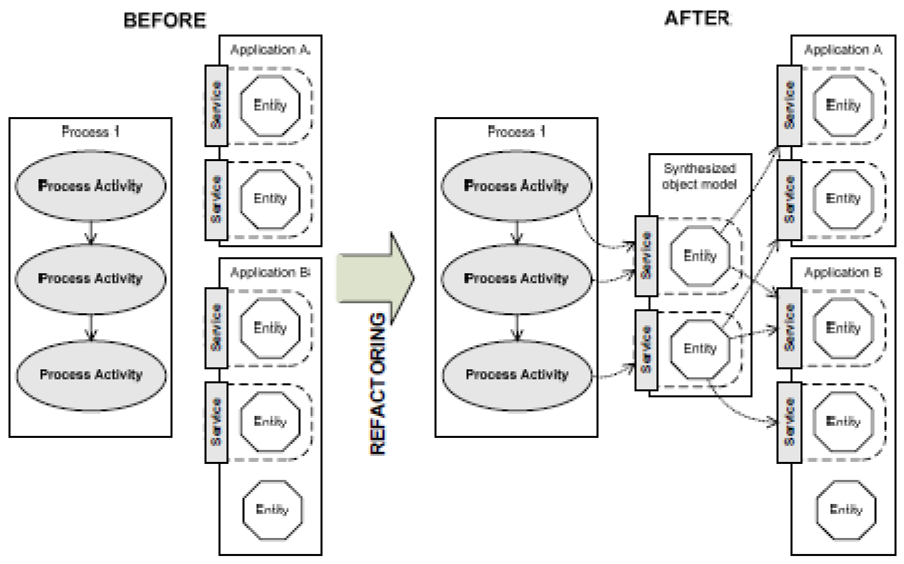

Figure 18. Refactoring by synthesizing. From "Patterns for Business Object Model Integration in Process-Driven and Service-Driven Architectures," by C. Hentrich and U. Zdun, *Proceedings of the 2006 Conference on Pattern Languages of Programs,* No. 23. Copyright 2010 by ACM.

The various refactoring patterns mentioned offers alternatives for refactoring design decisions for the integration of specific business models. These patterns will work only when there is an existing SOA based architecture already in place. With an SOA-based architecture in place, the advantage is having endless possibilities of patterns that could be used in integrating other applications or systems. However, the use of a Service Data Object (SDO) as discussed by Krizenvick and Juric (2010) seems to be a much more appealing than having to manipulate the business object model as the process of integration leaves the current business model as they are. The business model also in the financial industry does not change that much to warrant a manipulation of it, it could also be time consuming needing the skill set of a business analyst.

New data integration trends. Although there have been so many architectures that have been developed and are still being developed regarding data integration, there are currently a few new architectures that are being explored by major manufactures and vendors in the IT industry. Mohania and Bhide (2008) looked at a few data integration approaches currently being studied in the IBM Research lab in India. One of the approaches of data integration they looked at was the Context Oriented Integration Approach. Two of the studied Context Oriented Integration Approach were SCORE (Symbolic Context Oriented Information Integration) and EROCS (Entity Integration in the Context of Structured Data).

SCORE. SCORE is an information integration approach with the main purpose of integrating structured and unstructured data. The retrieving of unstructured data and structured data by SCORE is achieved by using a SQL query on the structured data which is then translated into keywords to use in the retrieval of relevant unstructured data. In an example given by the author and reflected in the diagram below, an application submits a query asking for the names of a company with the maximum stock price variation in the past week. With such a query input in the SCORE application, the application looks at the results of the query including the neighbouring tables that have related information and identifies the keywords that are most relevant to the profiles of the stock. These keywords are used to retrieve relevant news and reports from the search engine. Figure 19 shows an integration process using SCORE.

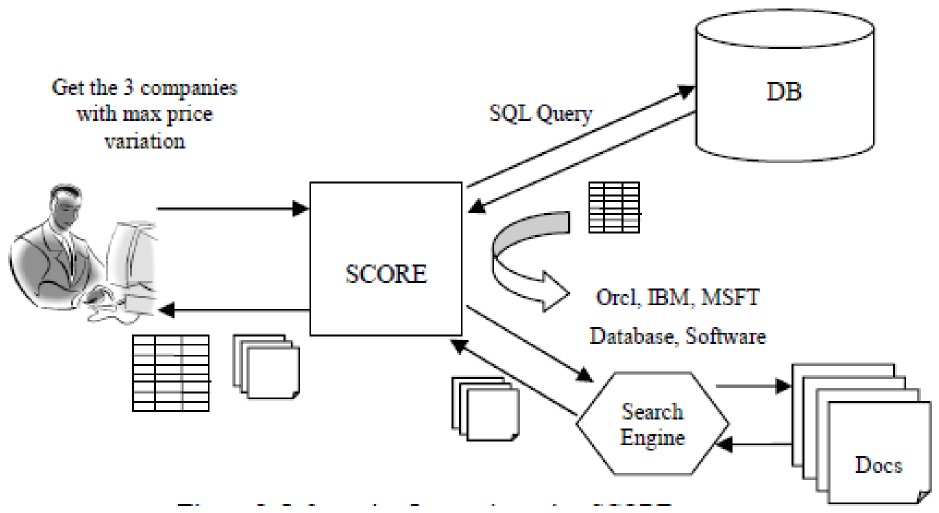

Figure 19. Information integration using SCORE. From "New Trends in Information Integration," by M. Mohania and M. Bhide, 2008, *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication* , 74-81. Copyright 2008 by ACM.

The challenge faced by SCORE included its handling of numeric data because the handling of numeric data requires a technique to map numeric input to unstructured text, which poses to be problematic. With such an issue, a few doubts have surfaced about considering such a solution in an investment bank setting where numeric data is an essential core to their daily transactions. Therefore, a system that has a problem integrating numeric data would have no value in any banking environment where numbers are a core to their daily transactions. Also although the above approach seems to handle unstructured data very well, the same result also now achievable using XML, which is available in most modern relational databases such as Oracle 11g.

EROCS. The EROCS integration technique works by linking a given text document with relevant structured data. Structured data is viewed by EROCS as a predefined set of entities that best match the entities in a given document. EROCS however goes a step further by being able to identify an entity in a document even when the relevant entity is not explicitly given in the mentioned document. EROCS works by taking a document as its input and then filters it to retain the most relevant terms. The result set, on the other hand, is viewed as a set of predefined entities with associated context information. Given such an input like a text document, EROCS matches the context information of the candidate entities within the document. Figure 20 shows a text document viewed by EROCS.

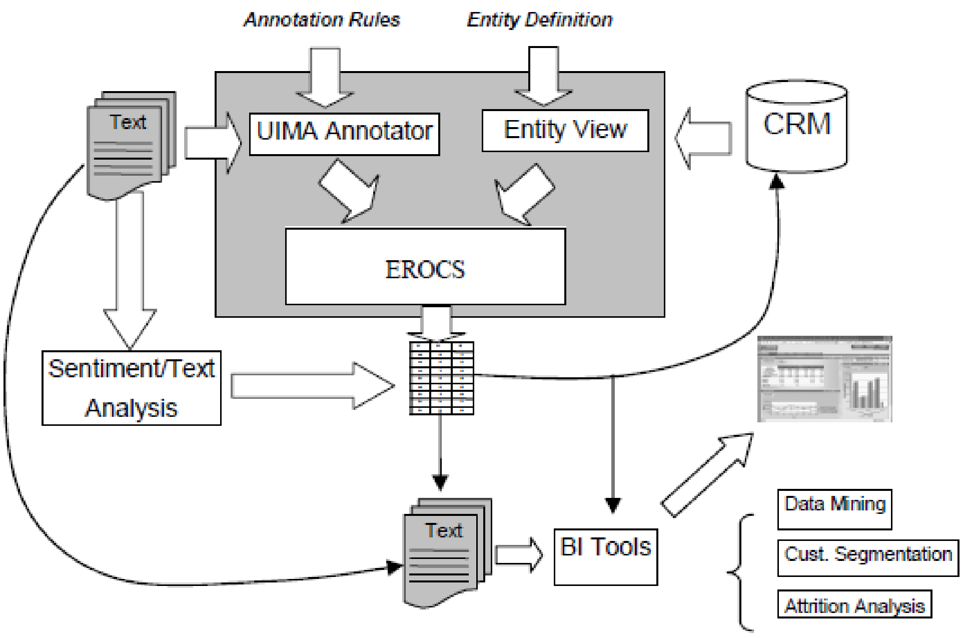

Figure 20. EROCS Overview. From "New Trends in Information Integration," by M. Mohania and M. Bhide, 2008, *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication* , 74-81. Copyright 2008 by ACM.

The major challenges of the EROCS approach is that EROCS assumes the presence of a template to find the right entity, another issue not presented is how it will handle multiple templates if provides as an input. A major issue not mentioned about EROCS is how it integrates data coming from other applications with variant output formats, in other words not necessarily text since a good data integration application should easily be able to work with other systems or applications which would either serve as clients or subscriber applications.

Summary

From the literature review, it is evident that there are a countless number of approaches available to investigate in regards to data integration. However, the one that stands out the most seems to be implemented using a service orientated approach since the idea of services and loose coupling between integrated applications seem to appeal to most organizations (Sunderaraman, n.d).

Chapter 3 Research Methodology

This chapter presents the research methodology that will be used in testing the thesis. These phases are (a) a case study of a data integration project within a major financial services organization and (b) an examination of the effects and lessons of carrying out such a project. It would have been ideal to carry out a survey after the project however due to time constraint and the paper work involved in having an approval for such a survey only the two phases mentioned above will be examined.

Holter and Kanneberg's depiction of the attributes of a research as seen in Figure 21 (as cited in Stoveland, 2008) will be adapted to demonstrate the reliability and validity of the qualitative techniques used in this thesis. The triangulation approach will be used to confirm the case study findings and to provide correlation between the literature review, project experience and the case study.

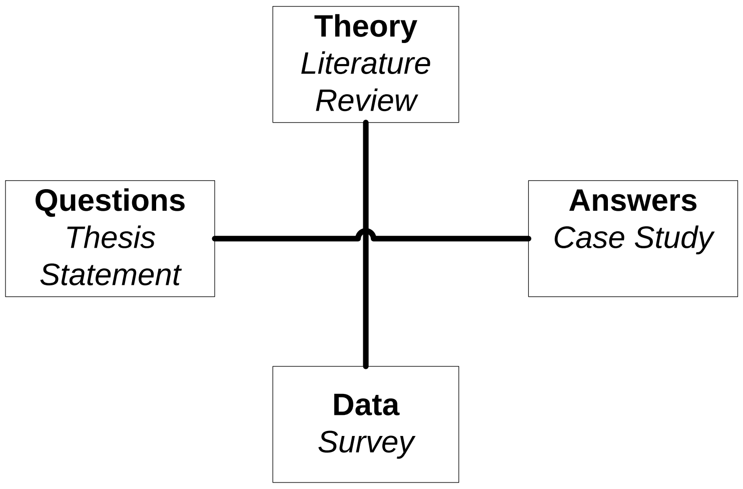

Figure 21. Elements of a research project. From "Managing Firm-Sponsored Open Source Communities A Case Study of Novell and the OpenSUSE Project," by J. F Stoveland, 2008, Master's thesis. Copyright 2008 University of Oslo.

A Data Integration Case Study

Pioneer Investments is a global asset management company with headquarters in Boston, Milan, and Dublin. In December 2011 it was reported to have $209 billion in assets under management (AUM). The company manages mutual funds and institutional accounts using a global platform for the purchase and sale of securities. Up till about 2004, each location internationally acted almost independently having separate trading, reporting systems and processes. As a result, it was a daunting task by managers and project leaders to collect, identify, and report critical information to meet internal and external mandates.

Technical Setting

Benchmark data in Pioneer Investments was always available to the fund managers via a Business Objects application. This data was used by the fund managers in order to measure the performance of their various investments or those of other fund managers. The benchmark solution had been tailored to capture and collate Index data from various Index providers in various formats. This data was then stored in the benchmark database prior to being imported for data merging by an import functionality. This database consisted already of 45000 index securities. Also available was another system called the FBA (Fund Benchmark Analytic) system, which was used in processing the weights or compositions of securities and then blended with index data. The blending of the benchmark data was processed in such a way that it merged data from both the FBA system and Index data into a refined benchmark Management Information System (MIS) database.

The Challenge

An integration process between the Benchmarks database and LZ (Latent-Zero) back-end database system was needed. The infrastructure for the data integration was going to be built to handle both Index data and FBA data into the Benchmarks database before being blended and published into LZ. An additional 15000 Index security data was also going to be handled by this new proposed solution. The solution was potentially going to expand the possibility of an increased interface from various front-end applications connecting to LZ for the back-end data.

There is going to be a need in future for the Bloomberg POMS (Portfolio Order Management System) application to source some of this Benchmark data that was fed into the LZ application. However the focus now is to have this information into LZ as priority as this was the primary trading application for Pioneer Investments.

Proposed Solution

For this project the decision was to use a product that was in line with a service-oriented architecture, able to support web services and message-based technologies. This is because Pioneer Investments had merged with so many companies in the past and had inherited systems and applications from these companies that were never integrated.

The product of choice for this project was the TIBCO Enterprise Message Service (EMS). TIBCO was preferred to all other products such as Polar Lake, as it did offer more return on investment from an IT management point of view. TIBCO was also chosen because it was able to bring together different IT asset and communications technologies on a common enterprise backbone to manage real–time flow of information. With TIBCO, logic could be integrated in the middle tier for all subscribing applications. Some other benefits that were considered:

1. TIBOC offered a reduced cost to complexity and time to manage and integrate disparate systems by using a common backbone that supports open standards existing in IT systems and leading technologies such as Java EE and .Net, which were all used in Pioneer.
2. TIBCO was thought to be reliable, using a breath of communication protocols and fine-grained administrative controls. This offered the advantage of having the development and administrative work to be done in different geographical locations and also for audit reasons.

TIBCO Messaging Service

The TIBCO messaging service supports the Java Messaging Service (JMS) models which are Point-to-point (Queues), Publish and Subscribe (Topics) and Multicast (Topics.)

Point-to-Point: This style of messaging uses a queue to store messages until they are received. The message producer sends message to the queue, while the message consumer retrieves messages from the queue and sends acknowledgement once the message is received.

Publish and subscribe. In a publish and subscribe message system, producers address messages to a topic. In this model the producer is known as a publisher and the consumer is known as a subscriber.

Multicast. Multicast messaging allows one message producer to send a message to multiple subscribed consumers simultaneously. As in the publish and subscribe messaging models, the message producer addressed the message to a topic. Multicast is highly scalable because of the reduction in bandwidth used to broadcast messages.

In Pioneer Investments a mix of all the above messaging services were used depending on the number of subscribers. Figure 22 and Figure 23 show the use of the TIBCO EMS queues and EMS servers.

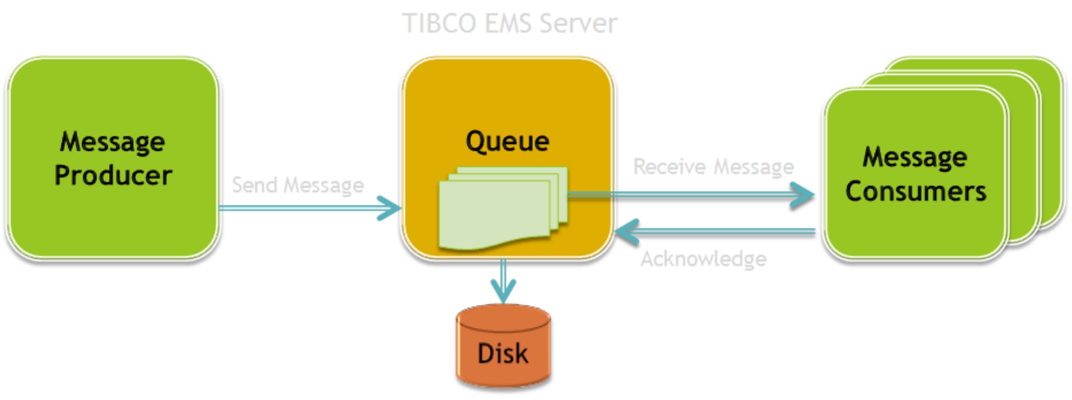

Figure 22. TIBCO EMS server using a queue

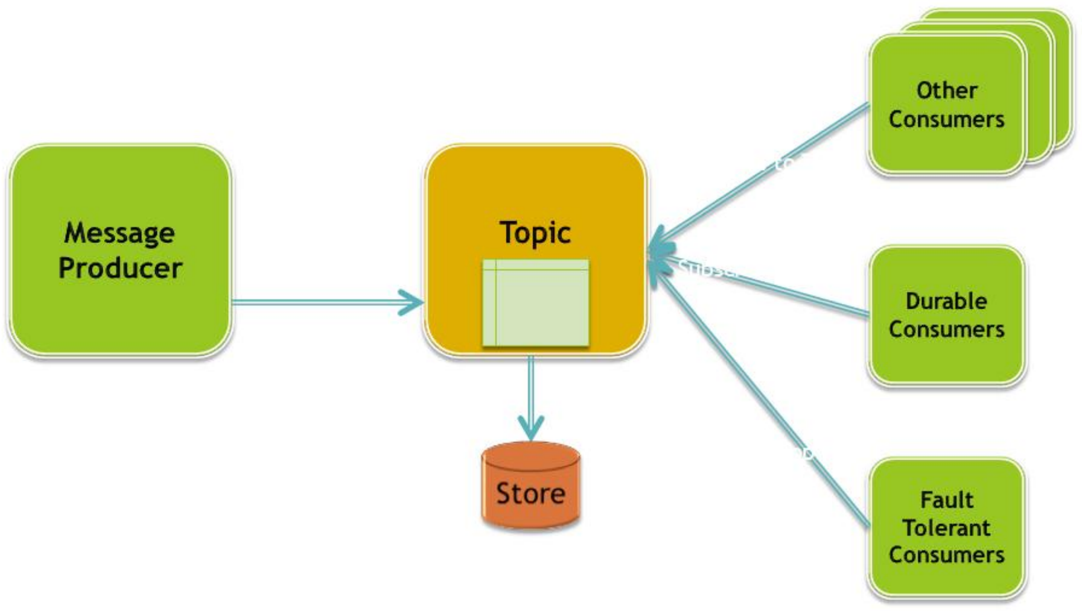

Figure 23. TIBCO EMS server using a topic.

Solution Implementation

Pioneer already had an architecture called the Pioneer Enterprise Security Bus (PESB), which was going to be used by the TIBCO process so that portfolio managers and system users would be able to gain access to the Benchmarks data using multiple interfaces such as the LZ and Bloomberg POMS application. Prior to implementation a few schema definitions were agreed upon, these included the following:

1. A Benchmark Schema Definition: This entailed the business logic for converting Integrated Data Definition Format (IDDF) to an LZ format with embedded tables for the translations of certain source fields/values from the Benchmarks to destination fields in LZ.
2. An IDDF Schema Definition: This was the root node of the IDDF schema definition for the benchmarkData, containing the header and bechmarkDefintion defined as “one too many”. The header contained general information which included the messageType, messageID, dateTime, sourceSystem, schemaVersion

and a drilled down consumptionsHints that held regionCode which was a marker as to the region the message was going to be consumed in. Figure 24 shows a high level structure of the IDDF schema definition.

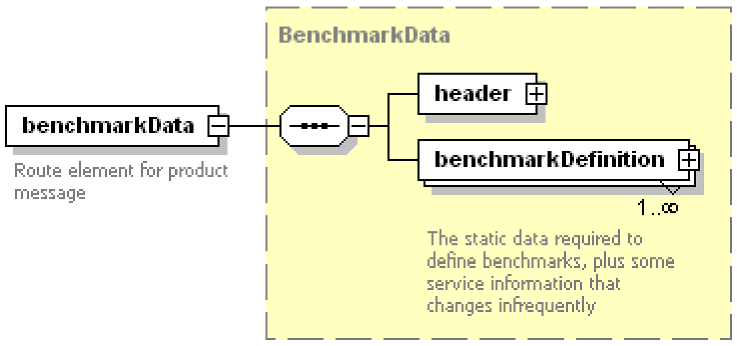
BenchmarkData
benchmarkData
Route element for product message
header
benchmarkDefinition
1..∞
The static data required to define benchmarks, plus some service information that changes infrequently

Figure 24. IDDF schema definition

The first step was to develop a stored procedure to build a record set of Index data. An ETL process was also used to extract the Benchmark security data from the Benchmark database into a new table containing all Benchmark security data. This was then presented into a PIPE delimited file for publishing. An upload process was written in a UNIX shell script to upload these files unto the TIBCO integration process environment.

The next step of the process was to configure the TIBCO EMS server to listen for Benchmark files using a listener. Once these files were available in a PIPE delimited file format, a few validation checks were done to the files to check if all the securities for the Benchmarks were received. These validation checks were written in a combination of XQuery and SQL syntax. Once these files were available, the TIBCO application merged the two files using XML and then published this out unto a new topic for consumption by the Latent Zero application. This published Benchmark XML message included a system code of "LATENTZERO" in the xml message indicating the application the message was going to be published into.

This project in total took four months from inception to completion. The advantage of this project was that, after 2 months, Benchmark data needed to be integrated with the POMS application. All that was needed in this case was not re-invent the wheel again by going through the process of sourcing the Benchmark securities as mentioned above buy by having the POMS application source this data by subscribing to the Benchmark XML topic.

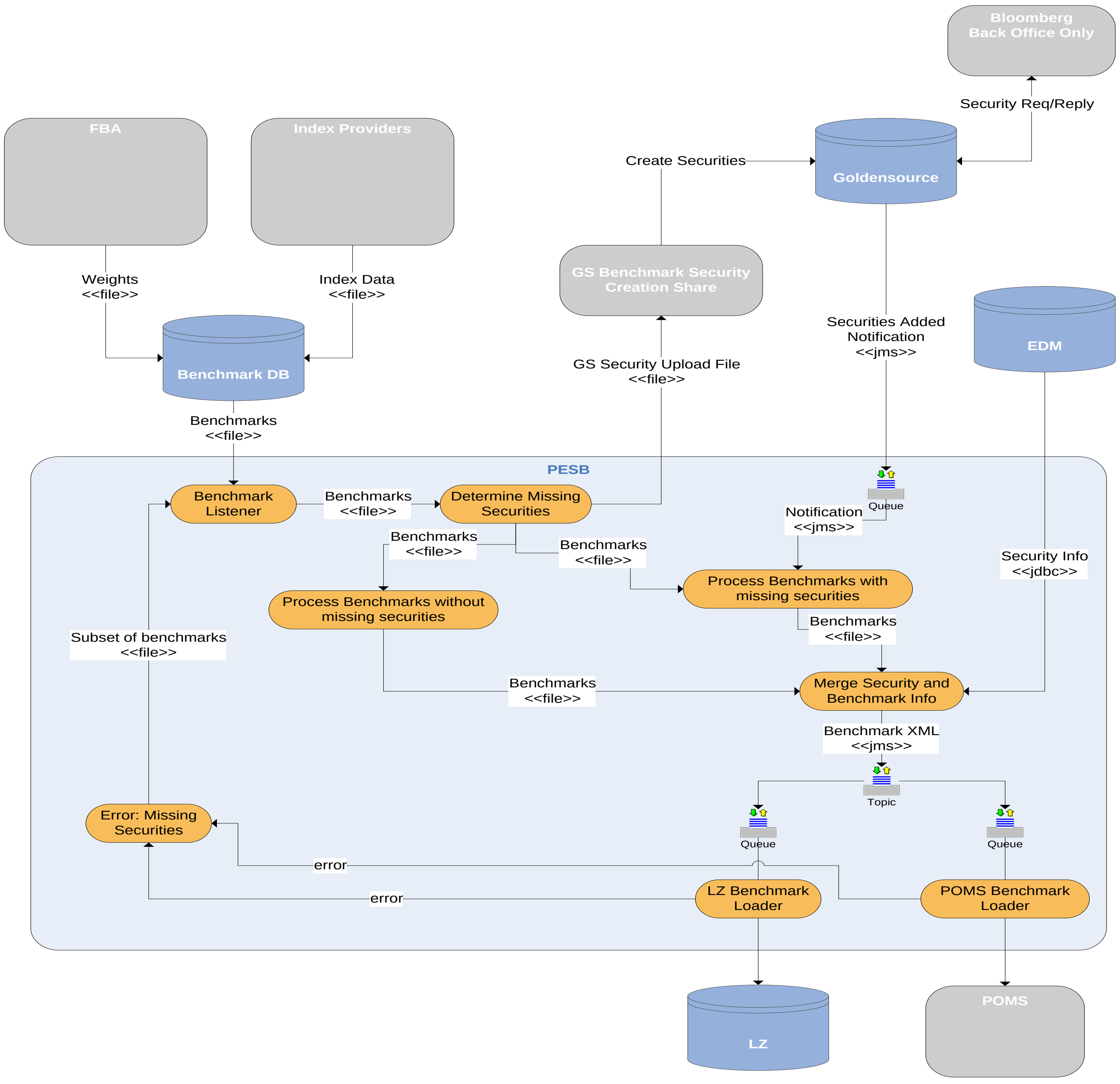

Figure 25. Pioneer enterprise service bus. Figure depicts the LZ and POMS application sourcing a merged XML message from the TIBCO topic.

Justification of Case Study

This case study represented a typical scenario in a financial institution where a data integration process was needed in a heterogeneous environment where most of the other applications had been inherited over time and at different time periods.

This case study proved that the need of having the right architecture and technology together is at the core of a successful data integration process. In our case the use of XML, JMS, SQL, XQuery, and Unix Scripts, and an Enterprise Service Bus provided a robust platform to write integration logic. For example, validation rules at the middle tier and applications subscribed to published messages using a combination of queues and topics.

When it was time for the POMS application to consume the Benchmark data, the integration process was straight forward as all that was needed was to have the POMS application pick up the Benchmark messages from the TIBCO topic which was already publishing the same message to the LZ application, this was evidence that a SOA architecture provided a “do once serve many” approach to data integration.

With such an architecture in place, there was no doubt that in future if Pioneer investments was faced with an acquisition or a merger where other applications were going to be inherited the process of data integration would not be a daunting task.

Chapter 4 Observation

Pattern of Encountered Issues

During this project there were a few issues that were encountered, one of the major problems was deciding on the elements of what was going to be consumed as a message from the Benchmark database, for example there was indecision as to the type of identifiers the messages were going to publish, while some fund managers expected to see identifiers such as the SEDOL, BBUNIQUE and ISIN in the message some other managers did not need it at the time, but thought it may be useful to have in the future.

To cater for this sort of issues it was decided to have all identifiers included in the Benchmark messages that were going into the TIBCO but have some validation where only the needed XML elements in the TIBCO message were going to be consumed for now but this could be changed in future when the other identifiers were needed.

There was also a need to distinguish between the messages that were going to be consumed by LZ and the POMS application, this was achieved by having a code scripted in the XML message at the header level indicating which application the message was intended for, for example for the LZ application the XML header contained code “LATENTZERO” while in the POMS application it had had in it the code “POMS”.

The was also the need on this project of having the right staff with the right skill set, as it was not only necessary to be able to write code but to be able to translate a business requirement into a technical solution without any of the requirements being lost in translation. This was achieved by having those staff who had liaised before with the business section of the bank to carry on liaising with them as they were used to the business terminologies.

On other data integration projects of which I have had the chance of working on, most of the challenges have really been on the business requirements and how integration could be achieved without having to redo or visit every integration process that had been done in the past in a quick and efficient manner using a SOA architecture has proven so far to be the way forward of achieving this. Also, the process of integrating, merging, or mapping data has never really been a major focus. This is because so many tools or programs can be written to achieve this using a programming language of choice or by having the merging process achieved within a database and then having the content published as an XML message.

Characteristics of Solutions

Most of the difficulties and solutions took the form of mapping data fields by determining the benchmark fields that needed to be consumed by the LZ and POMS application, writing validation rules in TIBCO and the creation of XML schemas, queues and topics and services for the publication of messages.

Chapter 5 Conclusion

Summary of Contributions

This thesis shows that data integration could best be achieved with the use of services in a SOA architecture and a blend of other technologies such as XML, JMS, XQuery, the usage of pre-packaged vendor products and databases. Such a data integration solution should be a well thought out process from an architectural perspective or point of view from the very beginning, if data is to be processed from more than one source: external or internal. Therefore, a well thought out data integration process makes it easy for future integration work. So far, data integration through services seems to minimise the work involved in data integration but does not eliminate manual intervention as there is always the need to understand the format and attributes of the data that needs integration.

Lessons Learned

Prior to under taking this research I was not aware of the various approaches of data integration especially with the use of models. Also through the case study I learnt how to go about integrating data for the future in other words making blended information ready for the future as was the case with the POMS application, which was integrated later on.

Apart from the technical aspects of data integration, resources in terms of staff and finance seem to play a major aspect in data integration. This is because you need the staff with the right skill set in several technologies and domain knowledge, in our case investment knowledge.

For example, as in the case study the POMS integration was put off because of budgeting although most of the work was done when the Benchmark integration work was done for LZ, during that time period some of the staff who were part of the LZ project were lost in other words knowledge was lost however because of proper documentation and the availability of some permanent staff it was easy to integrate data into the POMS application.

Unexpected Results and Issues

One of the issues I encountered was in the message consumption of the Benchmark security data for POMS because the Benchmark security data initially was meant for the LZ application. Therefore, the security data contained elements not needed by the POMS application. A filter had to be incorporated in the messages in other to avoid the wrong messages being consumed by the downstream applications, which are LZ and POMS. For example the XML header for each application had “LATENTZERO” for the LZ application and “POMS” for the POMS application in other to ensure that each application consumed the correct message.

Another issue I encountered was in the consumption of the integrated messages into the downstream applications (LZ and POMS). Once certain elements were allowed for consumption the size of the messages became larger thus delaying the consumption of these messages. A temporal solution was devised to eliminate any message acknowledgement once it was delivered to the downstream system for consumption. This helped to speed up the consumption process by two hours although there is a plan to look at other ways of having an increased rate of consumption while allowing message acknowledgement.

Recommendations for Future Research

There is still a good deal of manual intervention in the integration process of data, which still constitutes a great cost (Bernstein & Haas, 2008). However, there is possibility of more automation looking at the behaviour of mapping and being able to identify anomalous input data and tracing the source of query results.

Most research work on data integration has focused on problems where the goal of data integration is usually known. However problems exist in other domains, such as in science and engineering where data integration is an exploratory activity: A user integrates information, evaluates the results, and then identifies additional information to integrate.

Future research in semantic technologies and logic based reasoning engines may help in the integration task here and is an area where further research will be beneficial.

References

Bernstein, P. A., & Haas, L. M. (2008). Information integration in the enterprise. *Communications of the ACM - Enterprise information integration and other tools for merging data , 41,* 72-79. doi:10.1145/1378727.1378745

Carey, M. (2006). Data delivery in a service-oriented world: The BEA AquaLogic Data Servcies Platform. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* , pp. 695-705. doi:10.1145/1142473.1142551

Dong, X. A., Halevy, A., & Yu, C. (2009). Data integration with uncertainty. *The VLDB Journal, 18,* 469-500. doi: 10.1007/s00778-008-0119-9

Erl, T. (2005). *Service-oriented architecture: Concepts, technology, and design.* Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

Giordano, A. (2011). *Data integration blueprint and modeling techniques for a scalable and sustainable architecture*. Upper Saddle River, NJ: IBM.

Hentrich, C, & Zdun, U. (2006). Patterns for business object model integration in process-driven and service-driven architectures. *Proceedings of the 2006 Conference on Pattern Languages of Programs,* No. 23.

Hurwitz, J. (2007). *Service oriented architecture for dummies*. Hoboken, NJ: Wiley Publishing.

IBM.com/bcs (2005). *Transforming data management in the financial markets industry. IBM Business Consulting Services*. Retrieved from

Loshin, D. (2011). *The practitioner's guide to data quality improvement.* Burlington, MA: Morgan Kaufmann.

Khanna, A. (2008). Straight through processing for financial services the complete guide. Amsterdam: Academic Press.

Kim, H., Zhang, Y., Ouessena, S., & Clark, T. (2009). A case study on model driven data integration for data centric software development. *DSMM '09 Proceedings of the ACM first international workshop on Data-intensive software management and mining* , *1*, 1-6. Retrieved February 11, 2012, from the ACM Digital Library database.

Krizevnik, M., & Juric, M. (2010). Improved SOA persistence architecture model. *ACM SIGSOFT Software Engineering Notes, 35*(3), 1. doi: 10.1145/1764810.1764821

Mockus, A., Hwang, S., & Kim, S. (2009). A case study on model driven data integration for data centric software development. Proceedings of the ACM first international workshop on Data-intensive software management and mining, 9, 1-6. doi:10.1145/1651309.1651311

Mohania, M., & Bhide, M. (2008). New trends in information integration. *Proceedings of the 2nd international conference on Ubiquitous information management and communication* , 74-81. doi:10.1145/1352793.1352810.

Pan, A. & Vina, A., (2004). “An Alternative Architecture for Financial Data Integration” By A. Pan and A. Viña 2004, *Communications of the ACM, 47*(5), 37-40. doi: 10.1145/986213.986235.

Rosen, M. (2008). *Applied SOA: Service-oriented architecture and design strategies*. Indianapolis, IN: Wiley.

Stoveland, J. F. (2008). *Managing firm-sponsored open source communities A case study of Novell and the openSUSE project.* (Master’s thesis). University of Oslo, Norway. Retrieved from http://foss.xapient.net/pdf/suse%20community%20research.pdf

Sunderaraman, M. (n.d.). *Service oriented architecture in investment banking.* Retrieved from http://www.websesame.co.uk/wp9.pdf

Williams, R. T. (2011). *An introduction to trading in the financial markers technology--systems, networks, and data.* Amsterdam, The Netherlands: Academic Press.

Yoder, J., & Johnson, R. (2006). Patterns for business object model integration. *Proceedings of the 2006 Conference on Pattern Languages of Programs*, (23).doi:10.1145/1415472.1415499

Zins, C. (2009, March 1). *What is the meaning of data, information and knowledge?* Retrieved from http://www.success.co.il/is/dik.html

Appendix A

Table 1. *Definitions*

Term or Abbreviation	**Definition**
AUM	Assets Under Management
BBUNIQUE	Bloomberg Unique Identifier
Benchmark	A standard against which the performance of a security is measured
Counterparty	The other party of a financial transaction
CUSIP	An identification assigned to stocks and bonds .Committee on Uniform Securities Identification Procedures.
ESB	Enterprise Service Bus
Financial Instrument	A real or virtual document representing a legal agreement involving some sort of monetary value.
Holding	The expected time during which an investment is attributable to a particular investor.
Index	An imaginary group of securities representing a particular market or a portion of it.
ISIN	International Securities Identification Number.
LZ	Latent-Zero is a trading application.
PESB	Pioneer Enterprise Service Bus
POMS	This is a Portfolio Order Management System used for the management of client portfolios
Position	Amount of security owned or borrowed by an individual
QUEUE	A staging area that containing messages that have been sent and read to be read
Security	A financial instrument ownership position in a publicly-traded corporation (stock), a governmental body (bond),or rights to an option.
TIBCO	An integration server software for enterprises
TOPIC	A distribution mechanism for publishing messages to multiple subscribers
Trade	A transaction involving the selling and buying of a security
Vendor	Seller of financial data or information