Basel II Enterprise Architecture: Guidelines for Regulators and Banks

Umar Rafi[1]

ABSTRACT

Basel II implementations are one of the largest projects ever to be carried out by banks. However, the national governance and regulatory authorities of many countries have been unable to provide architecture guidelines covering Basel II technical best practices; specifically to banks, which do not have the required background to carry out such large scale projects. In some cases, the national governance and regulatory authorities, themselves, have limited knowledge of this space. This paper addresses this problem by describing a practical solution for implementing enterprise architecture for Basel II implementations, through the lessons learned from successful client engagements.

Keywords – Basel II, Enterprise Architecture, Data Warehouse, ETL, Regulators

1. Introduction

Basel II is the latest buzzword in the Risk management arena, both for banks, as well as for Information Technology (IT) companies, which specialize in this space. Due to this, there is some up and coming literature available on the *how-to's* of implementing Basel II projects; covering everything from Business Process Re-engineering (BPR) of banks' risk management processes to enterprise-level changes in banks' IT architecture(s). The Bank of International Settlements (BIS) has defined, in detail, the *banking*[2] side of Basel II. This includes detailed frameworks, describing Basel II requirements, with regular updates to these frameworks. In addition, the regulatory authorities of various countries regularly provide detailed guidelines to banks on the intricacies of Basel II, from a risk management side. The above is further supplemented by the expertise available in banks, through their internal risk management groups.

However, there is one area, where banks have been forced to work on a *hit and trial* method, relying solely on *on the job training*. It is the area of Basel II enterprise architecture, i.e. the changes banks need to make in their enterprise architecture to ensure their IT software infrastructure can successfully support the business-related risk management aspects of Basel II. To clearly comprehend the importance of a robust, extensible and, most of all, practically workable Basel II architecture, one needs to understand the costs related to any Basel II implementation. The largest thirty banks in North America will spend between $50-150 million on their Basel II implementations.[3] The author's experience at client sites indicates that, for very large banks, these projects will run for two to four years with the involvement of 350+ employees, at various stages of the project. While the targeted goals of these projects and their business side complexities are primarily in the arena of risk management, more than 50% of the staffing and costs of the actual

[1] i-flex Solutions, Dubai, UAE (Email: uxrafi@hotmail.com, Telephone: + 971 50 145 3185)

[2] The terms *banking* and *business* will be used, interchangeably, in this paper, to describe the non-technical portions (primarily in the area of risk management) of Basel II Implementations

[3] Riskwaters.com

implementations will be in the IT arena. The purpose of this paper is, thus, to provide architectural guidelines to governance bodies, regulators, and banks for possibly successful Basel II enterprise architectures. This is based on the author's involvement in three separate Basel II projects with tier-1 North American banks. All three banks have successfully moved their Basel II solutions into production.

2. Basel II Architecture Framework

Before proceeding further, it is important to establish a common vocabulary regarding the various terms that will be used, throughout this paper: In this paper, Application architecture will refer to the software architecture of each of the applications that solve specific business problems, within singular functional groups and without crossing multiple systems. This paper will not concentrate on application architecture beyond this point. It will only concentrate on Enterprise architecture.

Thus, the key point to understand is the boundary between application architecture and enterprise architecture. Enterprise architecture denotes, "*Both an entire enterprise, compassing all of its information systems, and a specific domain within the enterprise. In both cases, architecture crosses multiple systems, and multiple functional groups within the enterprise* (The Open Group, 2006)."

There are many different enterprise architecture frameworks that are accepted in the industry. Some examples are the Zachman[4] Framework, E2AF (Extended Enterprise Architectural Framework), TOGAF (The Open Group[5] Architecture Framework), AGATE (France DGA Architecture Framework), etc. We will detail enterprise architecture using the TOGAF definition.

TOGAF divides enterprise architecture into the following four categories. Going into the details of these is beyond the scope of this paper; however, brief explanations are provided below (The Open Group, 2006):

- Business (or Business Process) Architecture – This defines the business strategy, governance, organization, and key business processes
- Data Architecture – this describes the structure of an organization's logical and physical data assets and data management resources
- Application Architecture – this kind of architecture provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization
- Technology Architecture – this describes the logical software and hardware capabilities that are required to support the deployment of business, data, and application services. This includes IT infrastructure, middleware, networks, communications, processing, standards, etc.

Out of these four parts of the TOGAF enterprise, the architectural tasks for a Basel II implementation, primarily, fall into the category of data architecture and application architecture. We will, thus, concentrate on these two areas, while only briefly, below, touching upon business and technology architectures, in this paper.

Basel II Business Architecture

Figures 1 & 2 describe the business decision making processes of a typical bank – pre-Basel II and post-Basel II. In a nutshell, Basel II involves moving the risk management decision-making process from a tactical level (e.g. branch manager interacting with the client etc.) to a strategic level (executives providing regulators with comprehensive economic capital information), and then making capital allocation decisions based on that information.

---

[4] John Zachman, of IBM, is considered by many to be the father of enterprise architecture.

[5] The Open Group is a vendor-neutral and technology-neutral consortium, concentrating on open standards and global interoperability of information flow

The architecture required to support the pre-Basel II setup involves software systems that can provide the end users (branch managers etc.) efficient access to the bank's data, thereby allowing the end user to make efficient decisions regarding capital at a tactical level. The architecture required to support the post-Basel II setup involves software systems that accumulate the bank's data, from all business lines, and, then carry out comprehensive risk management calculations, across the bank at a strategic level[6].

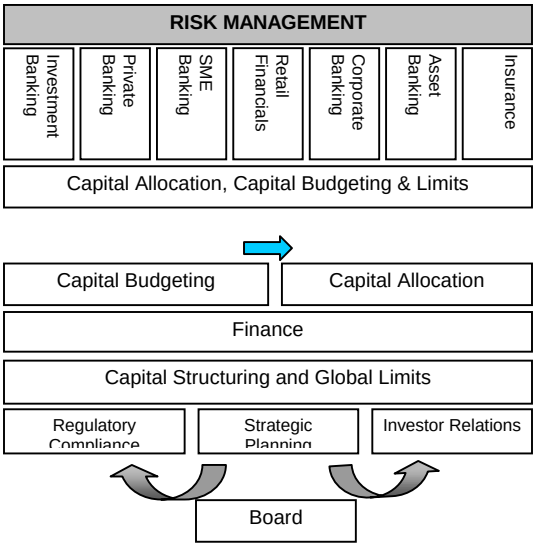

Figure 1: Pre-Basel II (Belmont, 2004).

Basel II Technology Architecture

While the horsepower of the hardware/networking equipment used for Basel II may be different from that used for any type of enterprise architecture, it will still consist of the standard set of components – servers, firewalls, routers, client machines etc. These can be provided, in a comprehensive manner, by any well-established data center or by the banks' own data centers.

The list of software infrastructure applications required to implement a successful Basel II solution is detailed in Section VI of this paper. The software infrastructure products are, generally, abstracted in the enterprise architecture, through the application software that is built around them and/or on top of them. .

3. Theory vs. Real-World

Everything mentioned in this paper, until now, is nothing new. There is plenty of literature in the industry that describes the above-mentioned topics in detail. However, this is point where this paper will attempt to differentiate itself from all the other literature present in the industry, on this subject.

Theoretical Literature on Basel II Architecture

Much of the architectural definitions for Basel II are actually given by risk managers, and not by software architects. Primarily because, very few software architects are able to comprehend the complexities of risk management from a business point of view, as they rarely have the required banking subject matter background needed to translate complex Basel II requirements into IT architectures. At the same time, the

[6] An off-shoot of this is providing the bank with the capability to use Risk Adjusted Return on Capital (RAROC) to make economic capital allocation decisions

risk managers, due to their lack of technical knowledge and experience are only able to define the architecture at a high-level. This has created a gap, where the technical details of Basel II enterprise architectures remain undefined and untested.

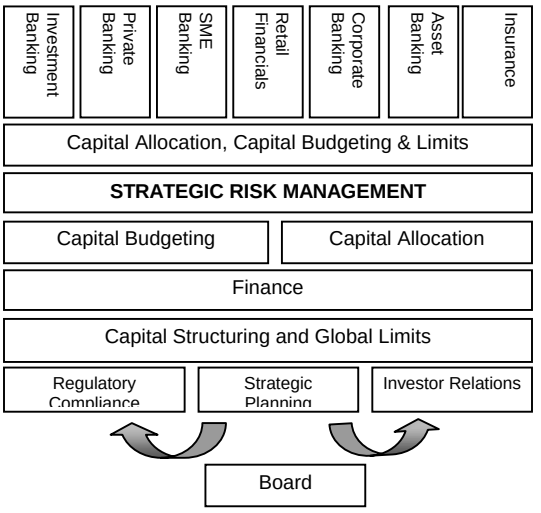

Figure 2 : Post-Basel II.

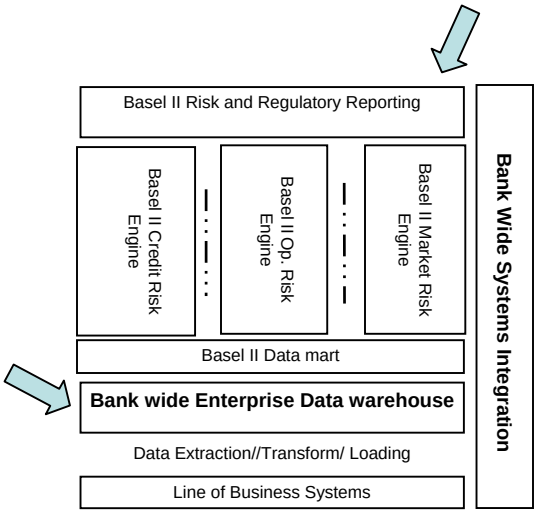

Figure 3: Basel II Abstract Architecture.

In addition, amongst the banks, while the relationship on sharing information on the business side is mutually *co- operative,* the relationship on the technical side is mutually *competitive*. At the end of the day, banks are businesses operating in a cut-throat arena. Like all businesses, banks are in the business of increasing their own business (not to mention, also in the business of putting other banks out of business.

In addition, to the best of knowledge of the author, none of the governance bodies and national regulatory authorities has published standard enterprise architectures/guidelines for the banks in their respective jurisdictions. BIS has also refrained from delving into this arena.

Hopefully, this paper will fill the above-described gap.

Basel II Enterprise Architecture

At an abstract and simplistic level, as shown in Figure 3, Basel II is basically a gigantic Data Warehousing and Systems Integration exercise, touching, literally, every system in every Line of Business (LOB) of a bank.

We will drill down one level lower and divide it into the following five different areas, using the remaining two TOGAF architectures that have not yet been discussed in this paper – data architecture and application architecture:

- Enterprise Data Warehouse
- Basel II Data mart
- Extraction/Transformation/Loading (ETL)
- Risk Engine
- Reporting Engine

4. Basel II Data Architecture

From the list above, the following two areas, and related tasks, fall into the category of Basel II data architecture:

- Enterprise data warehouse
- Basel II data mart

4.1 Designing a Enterprise Data Warehouse

While it is possible to implement a Basel II solution without an enterprise data warehouse, however a data warehouse-centric solution will be far more elegant and extensible. To the extent that we recommend to all banks to implement/utilize an enterprise data warehouse as the central data repository of the bank. This data warehouse will support all systems of the bank, and will not be exclusive only to the Basel II application.

Data warehouses are multi-year projects and need to be planned carefully. Most large banks already have data warehouses in place, prior to Basel II. The details of a designing a data warehouse are far beyond the scope of this paper.[7] However, there is one point, which needs to be emphasized significantly: a data warehouse is the heart of a bank's enterprise architecture. It stores and pumps out the blood of the bank's enterprise architecture, i.e. the data. It is, thus, imperative that the banks' management ensure they understand, in detail, the business and architectural aspects of the data warehouse that they have in place in their bank, or are in the process of putting in place, prior to moving into the, until now, uncharted waters of Basel II enterprise architecture. If the bank already has a data warehouse in place, then the challenge is, firstly, to ensure that it takes as feeds, the data from all the LOBs, which are needed for Basel II. These feeds could be coming from data marts, owned by different departments of the bank (and/or by the bank's subsidiaries), or they could be coming directly from the source systems, with no data mart(s) in between. The schema for the already existing data warehouse of a bank was, more than likely, not designed keeping Basel II in mind. This should not be a major issue, as the purpose of the data warehouse is to efficiently accumulate the source data, from all systems across the bank. A separate Basel II-specific data mart will be placed after the data warehouse (i.e. data will move from the data warehouse to the Basel II data mart), which will, eventually, act as the actual Basel II data source.

Since a bank-wide enterprise data warehouse is essential for a Basel II solution, vendors in different categories (risk engine, reporting etc.) tend to sell their data repositories as *possible* data warehouses. Generally a bank's first data repository will be a part of its core banking system. This can be used as the initial enterprise-wide data repository. Afterwards, when a bank purchases a risk engine, it will come with its own data warehouse-*type* repository, which can be used as a temporary enterprise-wide risk data mart or repository. For banks that have been around for a while and have a large amount of legacy data, but no data warehouse, the issues become more difficult. As they are not starting out from scratch, they cannot simply use their core banking etc. repositories as temporary enterprise data repository. There is no elegant and systematic solution for them. Due to tight deadlines, obviously, such a bank cannot, first, wait to complete its enterprise data warehouse and then begin its Basel II implementation. One solution is to use the Risk engine repository, initially, as a Risk data mart, with feeds coming into it from other data marts. This can, eventually, grow and become the Enterprise data warehouse. Alternatively, the Risk data mart can become a Dependent[8] data mart, once the bank develops a separate Enterprise data warehouse.

4.2 Designing the Basel II Data Mart

This is a Dependent data mart that will use the bank's enterprise data warehouse as a feed. This data mart is not *enterprise* in the sense that it is specifically designed as a storage mechanism for the Basel II

[7] They are also beyond the area of expertise of the author

[8] A Dependent data mart takes its feed form a data warehouse, while and Independent data mart takes its feed from LOB systems (and subsequently, may feed into a data warehouse)

implementation and not for all the applications of the complete bank. It should be owned by the Risk Management department of the bank. The question arises: why is a separate storage mechanism needed for Basel II, when the organization has already put into place an enterprise data warehouse, where all the data of the bank is stored in an organized manner? The answer to this question is quite simple: The data in the enterprise data warehouse is for every department/system in the whole bank (CRM, HR, Finance, Risk Management, etc.). It is not specifically organized in a manner which is efficient for the requirements of Basel II. Every single table and row in the data warehouse will not be needed by the Basel II data mart. At the same time, the Basel II data mart may require data which is generated through the combination of various types of data that is present in the enterprise data warehouse.

However, the biggest reason for suggesting a separate Basel II data mart has more to do with management issues than anything else. A good enterprise architect is not someone who can, simply, provide the best technical solution for a bank's architecture. Such solutions are well documented, through various frameworks, and thus can be provided by any enterprise architect with any level of experience. A good enterprise architect is someone who can design the enterprise architecture for a bank; keeping in mind how the enterprise architecture will assist in solving the short and long term business needs of the bank. As well as how the architecture maps to the organizational and management (and political) structure of the bank It is due to these reasons that it is important to have a separate Basel II data mart. As the bank and the bank's enterprise data warehouse grows, its design/schema(s) will be influenced by the management of various departments of the bank; each wanting to modify it to suit its own need. After a few years, it is quite possible the bank's enterprise data warehouse will be only a shadow of its original design. This is bound to have a negative affect on the risk management department and the data it requires. Hence it is a good idea for the risk management department to have its own data mart, under its own management, thereby isolating it from the management decisions regarding data, made by the other the groups in the bank.

The exact breakdown of the architecture of the Basel II data mart, itself, can take many forms. The Basel II data mart may be a subset of a larger data mart, covering enterprise risk management, as a whole, for the bank, with the later consisting of risk management data, beyond Basel II. Alternatively, and/or in addition, a bank's risk management group could have separate data marts for Credit, Market and Operational risk data. The major vendors, in this space, provide pre-defined schemas that can be used as the Basel II data mart schemas. Such schemas, normally, need to be extended, by the bank, to map to its internal requirements.

5. Basel II Application Architecture

The following three areas and related tasks fall into the category of Basel II application architecture:

- ETL
- Risk Engine
- Reporting Engine

5.1 ETL

Of all the tasks involved in implementing a Basel II architecture, ETL is, by a very large margin, the most difficult to implement. Due to this, the largest portion of this paper is being dedicated to ETL. It is the one aspect of the Basel II implementation, which is totally dependent on the bank's internal teams. In this case, ETL will include the *collection* of data, as well. TD Bank, of Canada - the tenth largest bank in North America, with 50,000 employees, 14 million customers and close to Can $400 billion in assets – is one of the first banks in the world to initiate its Basel II project (in 2002). According to its Associate Vice President of Corporate Technology Solutions:

"*Having access to quality data is the single most important aspect of Basel II. It is clearly what will make*

or break your Basel II implementation. It has stepped up the requirement for quality data beyond anything anyone had thought of. (IBS, 2006)".

Furthermore, a KPMG survey in 2002, covering 190 banks in 19 countries also indicated data collection to be the largest challenge in a Basel II implementation. Figure 4 also highlights the challenges of collecting and managing data for Basel II projects (Starley, 2004). Having said that, it is difficult to decide whether ETL is a part of the data architecture or whether it should be a part of the application architecture.

Normally, it would be considered a part of the data architecture, as it involves moving data in and out of databases. However, in the case of Basel II enterprise architecture, we have decided to place it in the

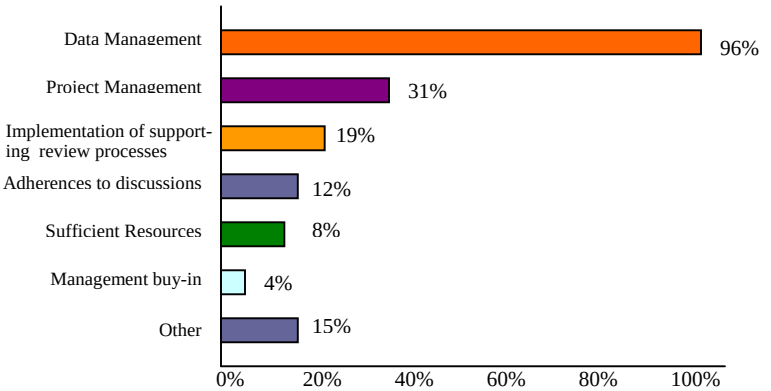

Figure 4: Major Basel II challenges being faced by Asian financial

application architecture category. The reason being that, in this case, ETL, not only performs its traditional tasks for *Extraction/Transformation and Loading*, it also acts as a systems integration mechanism. However before we discuss how ETL will be used in Basel II implementations, we need to get a clear idea of how systems integration is currently carried out in enterprise architectures, and where ETL fits in with respect to various solutions. If data is the blood in a Basel II system, and the data warehouse is the heart of the system, which stores and pumps the blood, then the system integration software is the veins and arteries[9], which take the blood from the various parts of the bank to the data warehouse and back to the various systems.

Over time, there have come into existence three different ways to integrate disparate software systems. These are as follows:

- Extraction Transformation & Loading (ETL)
- Enterprise Application Integration (EAI)
- Service Oriented Architecture (SOA)

These integration mechanisms rely on three different methods of communication:

- Batch Processing (ETL-based)
- Synchronous (Application Programming Interface (API))-based
- Asynchronous (Messaging-based)

[9] In certain cases, system integration software also acts as the, *kidney* of the architecture, in that it cleans and filters the data

5.1.1 Integration using ETL

ETL, as the name applies, has three steps: data is *extracted* from one system; it is *transformed* into a format that is acceptable to another system and is, then, *loaded* into this other system. ETL is the simplest, though least flexible method of integrating systems. It is simple because it relies on a set of scripts running queries against one database and extracting data. This data is stored in a temporary repository and is, then, passed through a set of transformation scripts, which change the syntax/format of the data. After this, another set of scripts load this data into a second database.

Within Basel II enterprise architecture, this process is run, sequentially, moving the data from one database into another. Each database acts as the data repository for the application that runs *on top* of it, e.g. the LOB systems will have their own respective database(s), the data marts are a database, the bank-wide overall enterprise data warehouse is a database, the Basel II data mart is a database, the risk engine will have its own database as will the reporting engine. Separate ETL processes will be needed to move data between each one of these databases.

ETL is the least flexible mechanism for systems integration, because it, generally, works in batch mode, i.e. huge chunks of data are *ETL-ed* through specific scheduled jobs - hourly, daily, weekly etc. In addition, it is primarily a point-to-point solution, connecting one system directly to another. It does not act as a central hub. Hence any new system that is added, to the enterprise architecture, will have to be directly connected to each one of the systems, with which it needs to share information. Along with this, ETL is a completely data-centric and data-driven solution. The extraction and loading scripts run straight against the database(s) and not through the API of the application. The data is thus directly exposed to the scripts. Anytime the data structures change, the extraction and/or loading scripts need to be changed.

ETL solutions are the best option, when the business logic of the systems being integrated does not need to be accessed by the integration mechanism as well as when the information needs to be passed from one system to another system (not to multiple other systems), in a sequential manner. ETL solutions can be directly coded using scripts, e.g. PL/SQL, VB etc. However, the more popular way is by utilizing third-party ETL tools, which abstract out the scripting through graphical user interfaces, and provide built-in intermediary databases for rapid transformation of data, as well as other features like schedulers etc.

5.1.2 Integration using EAI

Around ten years ago, during the mid-nineties, real-time messaging based integration solutions started establishing themselves. These solutions provided three distinct benefits. They allowed:

- Real-time asynchronous systems integration
- Centralized hub-based integration (as opposed to point-to-point)
- Ability to integrate multiple disparate systems (e.g. SAP, Seibel, Oracle, flat files etc.) though pre-defined message format translation

EAI systems came into existence in the form of high performance messaging brokers, which initially used CORBA ORBs (as well as proprietary brokers) as the broker of choice, with C++ (and various other) APIs. They then, moved onto standardizing around JMS as the messaging service, with Java APIs. EAI products support multiple messaging formats through connectors for the different systems that require integration. Over time, the messaging format(s) are standardizing around XML. Currently, EAI is the most elegant, scalable, and most of all, flexible mechanism for integrating systems. It is also, by far, the most complex. It requires a high level of skill-set, as well as the day to day maintenance of extremely sophisticated messaging brokers.

EAI should be the systems integration mechanism of choice when multiple disparate systems need to be

integrated in a real-time basis, in a non-sequential manner, i.e. data needs to pass, in a real-time fashion, from one system to many systems and vice-versa in any random manner. In addition, EAI solutions are very useful, if it is expected that new systems will be regularly added to the bank's enterprise architecture.

5.1.3 Integration using SOA

The systems integration domain has matured from being centered on TOGAF's data architecture (ETL), to the application architecture (EAI). Now it is attempting to move towards centering on the business architecture (BPM – Business Process Management). SOA lies somewhere between EAI and BPM. SOA is a computing system that standardizes on *services* to integrate software applications, using an Enterprise Service Bus (ESB). An ESB can be described as *EAI+* - an EAI system that relies on Web Services. This makes SOA the new (est) holy grail of systems integration. It has, over the past two years, gained popularity as the integration architecture of choice for new greenfield banks.

While EAI is *reactive*, SOA is *proactive*. EAI systems are designed to be brought into architectures which have stand-alone applications in place, with no well-defined industry-standard (or even enterprise-standard) interfaces. Thus, EAI presumes there does not exist systems integration standardization in the bank's enterprise architecture, i.e. each system purchased by the bank has its proprietary interfaces, languages, messaging formats etc. e.g. a bank that has Peoplesoft HR, Seibel CRM, mainframe-based systems, flat files, DB2 databases, Oracle Financials, etc.; all of which need to talk to each other, in a real-time basis. The EAI system becomes the hub of the complete enterprise. All systems talk to this EAI hub and not to each other. The hub takes care of standardizing all the differences between these systems.

SOA, on the other hand, proactively solves the system integration problem by looking at the enterprise as a whole, and designing the integration mechanism across the *services* provided by the products, which are being integrated. It is a more loosely coupled integration than EAI. This frees (though, not totally) the enterprise from concentrating on the integration of systems at a technical level, and allows concentration on integration at a business level. Integration moves from integrating the interfaces of software applications to an orchestration of services offered by each software system in a coordinated manner.

SOA is the *present* state for new application-level systems being developed by vendors (and banks' own internal teams). However, it is still the utopian *future* as a systems integration mechanism for any large bank, with legacy systems. As an example, the readers need to keep in mind that many, if not most, large banks in North America still maintain 50% or more of their data on mainframe systems, which pre-date even EAI by decades, what to talk of SOA!

5.1.4 ETL as the Systems Integration Solution

Based on the above, our recommended solution for passing data from one system to another, in Basel II enterprise architecture is ETL. The large tier-1 North American banks, which form the basis of the research for this paper, independently of each other, settled on ETL as the preferred mode of passing data from one system to another. This further validates our decision.

It is important to understand that EAI is still the most practically elegant and scalable system integration solution in the industry (in theory, SOA is a better solution; however, it has practical limitations, as explained in the previous section). If a new bank is to come into existence, then they will be well-advised to make an EAI system the center of their enterprise architecture from the get-go. However, most banks are decades to centuries old, with systems from all timeframes, which are tightly linked together, streamlined and optimized in a point to point manner. Attempting to rip out all such integrations and to replace them with an elegant, but extremely challenging, EAI solution is a task which is more ambitious than the Basel II

implementation, itself[10].

Basel II enterprise architecture is suited for the ETL-based solution for additional reasons, as well: data collection, for Basel II, is an extremely arduous task. It does not make sense to make it more difficult by attempting complicated solutions, which provide little or no extra benefit. Also, systems integration for Basel II is a data-centric task: the challenge is in collecting and transforming the data. Beyond that, the data just needs to be passed from one database to another in a batch format – tasks for which ETL is ideally suited. In addition, Basel II systems are integrated in a sequential manner: data moves from one singular system to another singular system, in a well-defined and exact sequence. Data does not (barring a few cases) need to go *backwards* to the source application, nor does it need to go, randomly, to multiple systems in a real-time manner. Basically, data starts from the LOB systems and ends up in the cells of the final regulatory reports; in between getting transformed by very sophisticated software applications, which load the transformed data into their own database. From there, it needs to be passed onto the database of the next application; so on and so forth.

Hence, ETL will suffice not only for extraction, transformation and loading of data, but also for playing the role of the primary systems integration mechanism for the Basel II enterprise architecture. It solves the problem, without adding extra unnecessary complexity.

5.1.2 ETL and the Three Pillars

It is important to understand how the ETL processes relate to the three different Basel II pillars. Data collection, internal to the bank, in Basel II, is primarily needed for the credit risk and operational risk areas of Pillar 1 (exclusively for the IRB approach; for the Standardized approach it will be combination of internal data and data provided by external third parties). For Market risk, much of the data will be aggregated from external third party sources.

Credit risk data will involve storing a bank's credit-loss possibilities. For corporate, sovereign and bank exposures, this involves collecting a minimum of five years of underlying history for PD estimates, and a minimum of seven years underlying history of LGD and EAD estimates. For retail exposures, it involves collecting a minimum of five years of underlying history for PD, LGD and EAD (OSFI, 2006).

Within the LOBs, the data will have to be collected from different source systems – loan processing systems, collateral management systems, limits management systems, exposure management systems etc.

The categories, under which the data falls, are as follows (Belmont, 2004):

- Reference Data – the supporting data required for risk calculations. It includes:
  - Market data (end-of-day, historical time series and real-time market prices, volatilities and trading volumes)
  - Credit data (historical default and recovery data, internal and external rating data, customer demographic information, current values of collateral, macro economic data, and exposures with consideration of netting rules and guarantees)
  - Operational Data (key risk indicators, internal and external operational loss data, results of control self-assessments, and audit points)
  - Limit hierarchies (credit, market, ALM, operational-risk limits, as well as position limits)

[10] The author has seen one or two such ambitious attempts by very large banks. They failed, due to the under-estimation of the tasks involved. A better approach would have been to gradually move towards an EAI-based enterprise architecture, in stepwise manner (while keeping SOA in mind, as the future) instead of going to EAI in one big bang

  - Scenario data (pro-forma permutations of risk factors to enable daily stress and scenario testing)
- Transaction data – details of current, past and potential transactions
- Results data – all final risk-measurement results and by-products of risk calculations

5.2 Risk Engine

A risk engine is the brain of the Basel II enterprise architecture. This is where the actual calculations take place. In fact, the whole aim the rest of the architecture is to get the data to the risk engine, in an acceptable format, so it can carry out the credit, market and operational risk calculations. The details of these calculations are beyond the scope of this paper, nor does the enterprise architect need to have detailed knowledge of them (beyond the stage of package selection, where the architect will need to rely heavily on the expertise of the risk management business group of the bank).

Thus, the calculations carried out by the risk engine remain internal to the engine, as far as the enterprise architecture is concerned. These fall into the following categories:

- Capital Computation
- RWA Computation
- PD/LGD Computation
- Credit Rating

All risk engines have their own data repositories. These repositories have their own schemas, which are designed to store the data needed by the risk engine. From an architecture point of view, what needs to be understood is the type of data, the format, which is needed by the risk engine, and the format in which it is spit out by the risk engine.

When purchasing a risk engine for Basel II, different banks, at differing levels of maturity, will have differing needs. Mature banks, normally, have some level of risk management automation already available to them. This may include third-party or proprietary systems for Limits Management, Credit Ratings, Collateral Management, Workflow Systems, Analytical Modeling etc. Such banks will only require a Basel II product that specializes in Basel II, primarily concentrating on Pillar 1 capital calculations for credit, market and operational risk.

However, smaller banks need to understand that Basel II is one aspect of an overall enterprise risk management system – both from a business and technical point of view. Such banks will need to ensure the risk engine application they purchase can also cater to their enterprise risk management needs, which fall outside the Basel II space, as well. In addition to the three primary Basel II risk areas – credit, market and operational – these include trading risk, treasury risk, strategic risk, energy risk, asset & liability risk, KYC (know your customer), compliance and governance etc. In addition, such banks may need to use the data repositories of their risk engines as a risk management data mart or even a temporary enterprise data warehouse, as mentioned in section IV, part A/B of this paper.

5.3 Reporting

Reporting is the final stage of the Basel II solution. The ETL and processing cycles are internal to the bank. What the regulator will see is a final report (actually, a series of schedules, representing different types of reports). While the Basel II reports for all banks, across the world, are derived from very similar type of financial data and calculations, the formats of the reports are defined uniquely by each country's regulator and may be based on different reporting standards, e.g. in Canada they are defined by Office of the Superintendent of Financial Institution (OSFI), while in Europe they are defined by Committee of

European Banking Supervisors (CEBS). We will use Canadian Basel II reporting, as defined by OSFI, as an example in this paper. Canadian banks are required by OSFI to publish three types of Basel II reports:

- Basel Capital Adequacy Report (BCAR) – A Pillar I report that divides the financial information along the lines of exposure and asset classes, e.g. amount of exposures that fall into the category of securitization
- New Credit Risk (NCR) – Provides information, along geographical and industry lines, at portfolio level for wholesale/retail activities, and also at the transaction level for wholesale activities. It is only applicable to portfolios measured with IRB approach and above Can $1 MM threshold, e.g. how exposed are the bank's assets in the province of Saskatchewan or how exposed are the bank's assets in the aerospace industry
- Pillar III – These reports fall into the category of Market Disclosure. While the bank will disclose all its confidential information to the regulators, it will not be required to disclose all its confidential information to the Market

On the face of it, Basel II reporting may seem like a simple task; at least from an architectural point of view. However, the sheer size of the data, and the calculations on the data, as well as the number of schedules in each report, makes this quite a complex task, e.g. there are thirty-one schedules for the BCAR reports, and twenty-seven for NCR. The total number of cells that need to be populated in all three categories of reports, to satisfy the Canadian regulator, is roughly 15,000. Each one of these cells has a calculation, behind it.

Once the risk engine finishes calculating and storing the data in its own application database, another ETL step takes place: the data has to be ETL-ed into the data repository of the reporting engine[11]. The reporting engine will, then, run queries on its database, capture the data, run it through its business logic and will populate the cells in the reports.

Once all the required report cells are populated and the information has been validated, it needs to be transmitted to the regulator. The format of the transmission will be dependent on the format in which the regulating authority can accept the data. The current standard for business reporting is based on XBRL (Extensible Business Reporting Language). Assuming XBRL to be the format, the regulating authority will require a mechanism/application on its side, which will accept the schedules and the data in these schedules as XBRL files. The regulators will also, need to have a validation mechanism on their side. The enterprise architect should ensure that the selected reporting engine supports various file transmission formats, as some of the regulators do not use XBRL as their choice of transmission format, e.g. OSFI uses Automated Data Transmission (ADT) format.

6. Supporting Considerations

The above-mentioned thoughts, in this paper, capture the requirements for a successful enterprise architecture for Basel II. However, there are three supporting considerations; attention to which will make an enterprise architect's life easier, during a Basel II implementation. These are:

- Metadata repository
- Manual calculation mechanism
- Internationalization

6.1 Metadata Repository

Metadata is *data about data.* It includes information like the ER diagrams of the schemas of the various

[11] The data will, normally, come form other sources as well, depending on the report being generated, e.g. NCR reports require some data directly from the LOB systems and/or enterprise data warehouse, along with the data calculated by the risk engine

databases and descriptions of the business rules that act upon the data, etc. It also includes information about the ETL scripts that act on the data. Until now, having a robust enterprise metadata repository has been a luxury for banks. However, due to the new compliance initiatives (e.g. Basel II, SOX etc.) a metadata repository has become an essential item in any bank's enterprise architecture.

The Basel II reports presented to the regulatory authorities cannot be based on trust, alone. In fact, the whole origination of Basel II (and the original Basel) Accord was because regulatory authorities did not fully trust the filings of the banks. For regulators to be able to audit the reports, they need to know from where the data originated, i.e. a clear path needs to be available from each cell in each schedule of each report, all the way back through the reporting engine calculations to its database, to the risk engine and its database, through the Basel II data mart and enterprise data warehouse to the LOB systems. Basically the regulator has to ensure the calculations are kosher. However, the regulator will not have the time to figure out this path, on its own, when it audits a report. It will need the bank to provide this information. There is no way for the bank to provide this information, without an enterprise metadata repository.

Creating an enterprise metadata repository, thus, is not a trivial task. Its details are beyond the scope of this paper. However, suffice to say, it is a separate schema (or set of schemas) in a separate database that will end up storing the above-mentioned metadata. This becomes even more challenging because information on metadata is rarely, if ever, available in standard format or locations, e.g. schema information for databases is usually spread out all over the bank's IT dept. In addition, in many cases, different business groups use different names for the same type of data. Along with this, it is extremely difficult to set up a successful business process that ensures that as new systems are added to the enterprise, their metadata is seamlessly included into the metadata repository. Needless to say, banks require an enterprise metadata repository. Those that do not already have this, will be well-advised to create one; definitely for Basel II, but also because it is a best practice.

6.2 Manual Calculations Mechanism

The above-described Basel II enterprise architecture is basically an *all or nothing* deal. As explained earlier, data starts from the LOB systems, passes through multiple repositories and engines, eventually ending up in the cell of a report, which is then electronically sent to the regulator. If this system breaks (or becomes inconsistent), anywhere in between, all the subsequent calculations and information becomes useless. Thus, banks need to have internal mechanisms to validate, whether the automated systems are carrying out the calculations correctly.

There is only one way to, internally, audit these calculations: the old fashioned way, i.e. manually. Until a bank reaches a point, where they have 100% unconditional faith in the accuracy of their Basel II enterprise architecture, they will need to have groups of teams that will manually do the Basel II calculations, alongside the automated calculations. At the very least, this process will have to be implemented, while the architecture is in the Test phase[12].

At a minimum, manual calculations need to be carried out, at the following critical points, in a Basel II cycle:

- After the data is loaded into the Basel II data mart and before it is loaded into the risk engine. At this point, a team of individuals will have to manually do the risk calculations, i.e. act as a manual risk engine. It should be kept in mind that this will not be a small team. The whole automated process of a Basel II *run* – starting from LOB systems to the transmission to regulators, runs well

[12] A better practice is to continue with this practice till the first series of approvals and audits from the regulator is achieved

into hours (or even days) for large banks. Hence depending on the size of the bank this could be a team between ten to fifty individuals

- After the data is loaded into the reporting engine database and prior to when the calculations are carried out to populate the cells in the report. This would be a smaller team, however, it would have to work with the same amount of detail as the above-mentioned team to ensure that the reporting calculations are correct

6.3 Internationalization

As mentioned earlier, while the risk calculations for Basel II will generally be similar in all geographical areas, based on common variables (e.g. PD, LGD, EAD etc.), the reporting formats will be different in every country. It will make the enterprise architects' job much easier, if they use a reporting engine which implicitly handles the differences in formats across geographies (e.g. UK, Australia, Canada, USA etc.) i.e. the data can be ETL-ed into the reporting engine's schema in a one specific format by the international headquarters of a bank. After that it should be the reporting engine's responsibility to transfer the data into the appropriate formats, mapping to the reports for each country's regulator, where the bank operates.

7. Basel II Software Products

Instead of presenting a list of Basel II related vendor products, with each section above, it was decided to present it in this section, so the reader can access the information in a consolidated manner. When the term Basel II solution is mentioned, the normal assumption is to think only of a risk engine. Hopefully, this paper has made it clear that Basel II solutions require a series of products, robustly integrated together to form a comprehensive enterprise solution; of which a risk engine is only one component. Most of all, hopefully, it is now clear that the most challenging task in a Basel II architecture, is actually the ETL (even though ETL tools tend to be the simplest of the list of products needed to implement Basel II). Primarily because, ETL requires strong domain level expertise, which is specific to each bank - a task which cannot be replaced by any tool. Table 1 lists some of the products, covering the Basel II architecture described in this paper, that are available in the market. Most of these products belong to major industry players. In many cases, vendors have merely added bells and whistles to already existing products and have placed them in the Basel II product domain. While in other cases, they have actually built products from scratch to meet a specific Basel II business requirement. This paper will not make any recommendations, one way or the other, regarding the products. It should also be understood that this is not a final list and may (or may not) represent the best of breed products. Its aim is to merely point the reader in the correct direction.

8. Concluding Remarks

Risk management enterprise architecture in general, and Basel II enterprise architecture in particular, is still an immature field. This paper should, thus, be considered a first step in an area, which will remain quite fluid. As the Basel Accord(s) extend beyond credit, market and operational risks (as well as beyond the banking industry, itself) and as more and more successful Basel II implementations, move into production, across the world, the definitions of successful Basel II enterprise architectures are bound to evolve and change. Due to space constrictions, there are certain areas, which have not been covered in this paper, but will need to be tackled by enterprise architects. Some of these areas as follow, etc.:

- Reconciliation and synchronization of account systems data with General Ledger data
- Management structure of enterprise architecture team
- Enterprise architecture, across multiple countries, where the bank's subsidiaries may be following different approaches in each country (e.g. AIRB in Canada, 1A in USA, Standardized in Sri Lanka, etc.)
- Stress-testing the architecture
- Integrating the Basel II implementation into a robust

- enterprise risk management architecture, allowing a bank to move to RAROC-based decision making
- Transitioning from an ETL-based systems integration solution to an EAI-based solution implementation mechanisms

Table 1: Basel II Products

Category	Products
Data warehouse	DB2, SQL Server, Oracle, NCR, etc.
Data mart	DB2, SQL Server, Oracle, NCR, etc.
ETL	Informatica, Ascential, Ab-initio etc.
Risk Engine	Algorithmics, Reveleus, FinArch, Sunguard, SAS, etc.
Reporting	Cognos, Business Objects, FRS, etc.

- Security mechanisms to ensure data is available only to the concerned parties; security being the most difficult aspect of an enterprise architecture to implement
- Disaster recovery plans to support the architecture etc.

Basel II is to be regulated by the national regulatory authorities of each country; not centrally by BIS. The magnitude of technical tasks involved, in implementing Basel II, for IT departments of most banks, will be so huge that banks could end up being caught in a no-win situation – if they don't implement Basel II, they will not be able to keep up with the competition in the international financial industry; if they do attempt to implement Basel II, they will drown in the massive costs of incorrect technical implementations.

Hence, the governance and regulatory authorities will be doing a huge favor to their financial industries, under their respective jurisdictions, if they can provide guidelines, not only for the business aspects of Basel II, but also for the technical aspects. Indeed, the author is aware of various banks, which have purchased multi-million dollars worth of Basel II-related software and are now totally in the dark on how to move ahead in its implementation. The national governing authorities will be well-advised to create consortia of the leaderships of their banks and provide them training in the best practices of Basel II architecture implementations. This will allow the banks to correctly evaluate the various software products, which are being offered by the vendors, as well as the services being offered by consultants.

This paper makes clear the magnitude of the tasks involved in Basel II implementations; specifically for countries, which do not have the domestic technical expertise and/or excessive funds. For such countries, it becomes even more important to ensure they get their Basel II architectures correct, on the first try. This paper will, thus, have achieved its purpose, if it can assist regulatory authorities and banks, in handling the above-described technical challenges of Basel II.

Acknowledgements

The author would like to acknowledge the assistance of Samina Rizwan, Mohan Bhatia and Zahid Naeem in reviewing this paper.

References

1. Basel Committee on Banking Supervision (2001). *The New Basel Capital Accord*
2. Belmont, David P. (2004). *Value Added Risk Management in Financial Institutions: Leveraging Basel II & Risk Adjusted Performance Measurement*.
3. Singapore: John Wiley & Sons.
4. Bhatia, Mohan (2006). *Credit Risk Management and Basel II*. Bangalore, India: i-flex Solutions.
5. Chorafas, Dimitris, (2005). *Operational Risk Control with Basel II*. Massachusetts: Elsevier Butterworth Heinemann.
6. IBS (2006). *Banking Operations in Canada – Jockeying for Position*. IBS Publishing.

7. Office of the Superintendent of Financial Institutions (2006*). Implementation Notice: Data Maintenance at IRB Institutions.* OSFI.
8. P. Starley, E. Hanson. (2004). *Asia Pacific Basel II Survey.* Ernst & Young.
9. The Open Group (2006). *The Open Architecture Framework.* TOGAF.

About the Authors

Umar Rafi is a Principal Consultant, covering the Middle East region for i-flex Solutions – a world leader in providing IT solutions to the financial services industry, with more than 765 customers in over 130 countries. Prior to this, Umar managed the Canadian operations for FRS Global – a Luxembourg-based company, providing enterprise-wide risk and regulatory compliance solutions to international financial institutions worldwide. Umar has a Bachelor's degree in Applied Mathematics and a Masters degree in Computer Sciences from the USA.