



WPI



BNP Paribas: Enterprise Architecture

A Major Qualifying Project Report

Submitted to the faculty of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements of the
Degree of Bachelor of Science

by

Zhen He, Computer Science and Actuarial Mathematics
Joe Servi, Mathematics

Project Sponsor: BNP Paribas

Submitted to:

On- Site Liaisons: Scott Visconti
Arjun Kohli

Project Advisors: Professor Arthur Gerstenfeld, Department of Management
Professor Dan Dougherty, Department of Computer Science
Professor Jon Abraham, Department of Mathematics

Submitted on

Thursday, January 12, 2012

Abstract

The goal of the project, sponsored by the Global Equities and Commodity Derivatives (GECD) at BNP Paribas, was to create a graphical management application that allows the group to better visualize a wide array of system flows. It was required for the application to have bi-directional communication with internal BNP databases and clearly show any possible problems with the data, such as an overloaded server. The outcome of the project is an integrated application that graphically displays and interactively manages business flows and their underlying data and builds performance/capacity dashboards.

Acknowledgement

There were many people involved with the Enterprise Architecture project both from BNP Paribas and Worcester Polytechnic Institute. Foremost we would like to thank Mr. Scott Visconti, our sponsor, who oversaw the project from the BNP Paribas side. Also Mr. Arjun Kohli, the project manager, who managed the day to day operations of the project. We want to extend our gratitude towards Professor Arthur Gerstenfeld, director of the Wall Street Project Center, for creating such an energizing experience. Along with Professor Gerstenfeld, we want to thank Professors Dan Dougherty and Professor Jon Abraham, our faculty advisors, for their continued support and advice and assistance in our final report and presentation. Finally we thank everyone in the GECD division of BNP Paribas who lent their advice, support and opinions concerning the eventual destination of the project. Thank you.

Authorship

At the tool evaluating phase of the project, Zhen He took full credit of the development in ApEx and Graphviz and Joe Servi took full credit of the development in Visio and Data Modeler. At the formal development stage of the project, Zhen He and Joe Servi made equal contributions to the development in the final product of this Enterprise Architecture project.

The writers are listed to the right of the corresponding sections of this report.

Abstract – Zhen He and Joe Servi

Acknowledgement – Joe Servi

Executive Summary – Zhen He

Introduction – Zhen He and Joe Servi

Background – Zhen He

Requirement – Zhen He

Business Uses – Zhen He

Development Phase I – Zhen He and Joe Servi

Development Phase II – Zhen He and Joe Servi

Results – Zhen He

Conclusion – Joe Servi and Zhen He

Appendix – Zhen He

References – Zhen He

Since Zhen He is a double major, she will be doing additional 1/3 unit of work related to this project in the next academic term (January 2012 – March 2012).

Executive Summary

BNP Paribas' Global Equities and Commodity Derivatives (GECD) division offers a variety of industry leading financial products. To keep its businesses running successfully, the division relies on various internal applications (systems) and hardware that conduct and support many business operations to function correctly and efficiently. IT managers and infrastructure teams need to have a firm control over the health of these applications and hardware within the context of different business flows. Whenever there is a performance problem, they should be informed right away.

For years, however, managers kept track of business flows and performed capacity planning manually. It was extremely costly and inefficient.

To achieve automation and increase efficiency in this area, BNP Paribas IT managers wanted to visualize their entire "enterprise architecture" and perform capacity management accordingly. At the same time, Human Resources managers and Business managers also sought a better visualization of the organization for their own purposes. Obviously, the existing database level of presentation of the data could not meet their need.

With this said, this project aimed to help BNP Paribas to visualize and manage their "enterprise architecture". The goal of the project was to develop a well-integrated application which addresses two key problems – visualization and capacity management. It should have four basic components of implementation. The first component is to automatically extract data from the database and feed it into a data management interface. The second component is to create the visualization. The third and fourth are graph interactivity and capacity analysis respectively. The final component can be further broken down to two parts – to reflect the health of any entity on the graph and to build a detailed performance dashboard for application or hardware.

The project entailed two phases. At the first phase, which is a tool evaluating phase, a variety of tools and technologies were reviewed. Two major visualization tools – Graphviz and Microsoft Visio were assessed at this phase. Oracle Application

Express (ApEx) was tested as a potential application interface. Major components of implementation were explored on each of these tools as thoroughly as possible. Key functions were implemented and demos on dummy data were given to the stakeholders of the project.

Research on the tools showed that Graphviz has its strength in representing structural information (Gansner, Emden R., Eleftherios Koutsofios, and Stephen North). It is powerful in positioning optimization and rich in graphical attribute options. The simplicity of the input file Graphviz takes in to render the graph also made it easier to automate the whole process. These were all crucial merits that would help the final build of the application.

Microsoft Visio, by contrast, is strong in graph interactivity including drag and drop. It was relatively difficult to automate the graphing process with Visio, however, and testers had to optimize positioning explicitly in the development of the tool. While graph interactivity was good to have in the product, it was not a necessity. Complexity in automation and extra work for positioning nodes and links in the graph, on the other hand, represented significant flaws in the tool.

As the single tool tested as an application interface builder, ApEx performed well. It is web-based, specialized in database applications, fast to develop, and easy to deploy and administrator.

At the end of the first phase, testers of the project concluded that the combination of Graphviz and ApEx seemed to be the next step to go given the timeframe. Based on the analysis provided by the testers, the management agreed with this conclusion. The project then moved to the second phase. It was a formal development phase where full capacity of the two tools was explored and the final product was built.

By the end of the project, a well-integrated visualization and data management application was built. The application has the following features: Visualization based on User Selection, Graph Interactivity, Capacity Metrics, Summary View & Detailed View, Processes Sub-graph, and Data management. It also has other functionalities including User Log, Saved Diagrams, and Graphical Attribute Management.

The application is a good blend of different technologies. Three major tools, six programming languages, and other technical concepts were deployed and practiced. Graphviz, SQL Developer, and Apex, as the graphing tool, the programming environment, and the application interface developer respectively, were heavily used and well integrated to function as a unit. Towards the end of the project, the visualization was incorporated into the application interface with all the behind-screen functionalities generated by Graphviz still supported. At this point, Graphviz and Apex were completely merged into one application.

One important part of the project is the process data to be visualized. It consists of two parts – enterprise entities that constitute the business flows and relationships among entities that preserve business logic. There are five types of entity data – Application, Hardware, Organization Unit, People, and Process. These five enterprise entities, together with the intricate relationships among them make up the various hierarchical diagrams the visualization generated.

There are many business uses of the product. In general, the uses can be grouped into three categories.

- Performance Management (Capacity Analysis)
- Personnel Management (Organization Chart)
- Technical Management (Processes Sub-graph)

In the future, the management can conduct efficiency analysis, optimize resources allocation, and expand or retract businesses based on the information.

Table of Contents

.....	
Abstract.....	i
Acknowledgement	ii
Authorship.....	iii
Executive Summary	iv
1. Introduction.....	1
2. Background	4
2.1. BNP Paribas and GECD	4
2.2. Data of Interest	5
2.3. Technologies Review	7
2.3.1. Tools.....	7
2.3.2. Programming Languages	8
3. Requirements	10
4. Business Uses.....	13
4.1. Performance Management.....	13
4.2. Personnel Management	13
4.3. Technical Management	13
5. Development Phase I: Tools Assessment, Prototyping.....	15
5.1. Apex + Graphviz – in-house development	15
5.1.1. Pre-analysis on the tool.....	15
5.1.2. Technology used	15
5.1.3. Database and Sample data	16
5.1.4. Development in Graphviz	16
5.1.5. Development in Apex.....	20
5.1.6. Two way interaction.....	26
5.1.6. Evaluation	29
5.2. Visio – in-house development	33
5.2.1. Hierarchical Design	33
5.2.2. Free Form Design	33
5.2.3. Semi-Free Form Design.....	33
5.2.4. Methodology	34
5.2.5. Bi-directional Interface	37

5.3.	Oracle Data Modeler – in-house development.....	39
5.3.1.	SQL Developer Investigation.....	39
5.3.2.	SQL Developer Conclusions.....	39
6.	Development Phase II: Formal Development of Final Product.....	40
6.1.	Detailed Design Plan.....	40
6.1.1.	User Interface Design	40
6.1.2.	Database Design.....	42
6.2.	Technology Used.....	43
6.2.1.	Software	43
6.2.2.	Programming Language.....	43
6.2.3.	Technical Support	44
6.3.	Architecture.....	44
6.4.	Data	44
6.5.	Methodology	45
6.5.1.	The implementation of user preferences.....	45
6.5.2.	Improvement on graph interactivity.....	45
6.5.3.	Embedment of graph in the application.....	46
7.	Results.....	48
7.1.	Major functions and capabilities implemented	50
7.2.	Summary of functions	55
8.	Conclusion	57
	Appendix A.....	58
	Appendix B.....	60
	Appendix C.....	61
	References.....	63

1. Introduction

BNP Paribas' Global Equities and Commodity Derivatives (GECD) division offers a variety of industry leading financial products and services including equity derivatives and commodity derivatives, indices and funds, and research and brokerage services. To keep its businesses running successfully, the division relies on various internal applications (systems) and hardware that conduct and support business operations to function correctly and efficiently. IT managers and infrastructure teams need to have a firm control over the health of these applications and hardware within the context of different business flows. Not only do they need to get alerts right away when a system or a server signals a performance problem, but also to understand how an over-capacity server can impact any particular flow.

Previously, managers of BNP Paribas kept track of business flows and perform capacity planning manually. A few years ago, right before this project was conceived, the company had once attempted to perform capacity analysis on applications and hardware. In order to draw useful conclusions, process data (data that constitutes a business flow and usually can be visualized as a flow diagram) had to be manually entered into *Casewise*, an external tool BNP used for business process analysis, management, modeling, enterprise architecture etc. This took the company nearly a year for the data to be in place. However, the results went out of date shortly after useful analysis was done. It was too costly and inefficient. After that, the company abandoned the way and sought for more automation in this area.

On May 6th, 2010, the financial world was briefly shaken by what has come to be known as the '2010 flash crash.' In a hectic day of trading, the Dow Jones Industrial Average plunged 998.5 points in a few minutes only to recover its losses several minutes later. No conclusive reason was ever agreed upon as to the cause of this hyper-volatile string of events, but the day sent a reverberating pulse through the financial industry calling for something to be done.

As a consequence, it became more important for BNP Paribas IT managers to

manage their entire “enterprise architecture” to prevent another ‘flash crash’ from happening again. Managers wanted to get a better sense of how the organization, business application, and infrastructure components participate in various business flows. However, the existing database level presentation of the data could not meet their need. Process data was still presented in raw database tables. Relationships between the data were still referenced by index. By scanning rows after rows of a table, one could hardly tell anything useful in terms of how the two business entities are connected, not to say perform any useful analysis on their performance. It was difficult to piece together the entire network of interconnected data and to manage capacity was almost impossible.

With this said, the project aimed to help BNP Paribas visualize and manage their “enterprise architecture”. The goal of the project was to develop a well-integrated application which addresses two key problems: an all-in-one-place visualization, and convenient capacity analysis. The application graphically displays business flows and interactively manages their underlying data. It also builds performance/capacity dashboards on business entities within the context of any flow.

The application has four basic components of implementation. The first component is to automatically extract data from the database and feed it into a data management interface. The process consists of making queries to the database and transforming its raw relational form into a more presentable format displayed in the user interface. The second component is to graph the data. The graph should be able to establish relationships at different levels. For example, if it currently displays a general view with processes as elements, after drilldown, it should reveal what is going on underlying each of the processes. The third component is graph interactivity. The graph should allow for simple user interactions. Based on user selection and preferences, the graph should collapse or expand a node, hide part of the picture, or roll up or roll down at a particular level. The final component is capacity analysis. This component involves two parts. One is to reflect the health of any entity on the graph so managers spot the problem at the first sight of the picture. The other is to

build a detailed performance dashboard. The dashboard lists performance statistics on certain capacity attributes measured against predefined thresholds.

To implement the application, the plan of the project entailed two phases. The first phase is a tool evaluating phase. Two major visualization tools – Graphviz and Microsoft Visio were assessed at this phase. Oracle Application Express (ApEx) was also tested as an application interface. After some analysis, the combination of Graphviz and ApEx proved to be the right solution. The project then moved on to the second phase which is a formal development phase. Full capacity of the two tools was explored at this phase and the final product was built.

2. Background

In this section, background of the company, BNP Paribas and its division, GECD are briefly discussed. The data to be visualized in this project is introduced and candidate tools and technologies that were used to build the application are reviewed.

2.1. BNP Paribas and GECD

A leader in global banking and financial services, BNP Paribas is one of the six largest banks in the world. The US site of the company has a very strong Corporate & Investment Banking business. BNP Paribas Global Equities & Commodity Derivatives, an arm of BNP Paribas Corporate and Investment Banking, offers custom-made derivatives on equity and commodity underlyings worldwide. It brings together three complementary business lines of Structured Equity, Flow & Financing and Commodity Derivatives.

- **Structured Equity:** provides structured solutions to a broad group of personal and business customers, banking networks, insurance companies and pension funds. It provides customized or exchange-traded structured products to meet their needs in capital protection, yield and diversification.
- **Flow and Financing:** covers products and services required by institutional investors to implement their investment, hedging, and portfolio optimization needs in a multitude of markets and underlyings. These products and services encompass flow derivatives, stock lending, prime brokerage and execution, as well as Asian equities research and brokerage.
- **Commodity Derivatives:** offers a full range of price risk management solutions on underlyings including energy, metals and soft commodities. The OTC (Over-The-Counter) group provides liquidity and market-making services, while the Futures team provides global clearing and financing tools for listed commodity futures and options to the corporate and institutional client base of BNP Paribas.

To support these important businesses, over 1400 front office staff work under GECD, numerous applications (systems) and servers keep running, and hundreds of

business flows are going behind.

2.2. Data of Interest

One important part of the project is the process data to be visualized. Process data contains valuable information about business flows. It consists of two parts – enterprise entities that constitute the business flows and relationships among entities that preserve business logic. There are five types of entity data.

- Application – system where business activities are carried out. A typical example is High Volume Trading Platform.
- Hardware – infrastructure unit that supports applications, i.e. switch ports, servers.
- Organization Unit (OU) – business component and department that is either a group of business users or an IT support team of applications or hardware, i.e. GECD, CIB. OU is usually an intermediate node in an organizational hierarchy.
- People – individual employees. They are usually end leaves in an organizational hierarchy.
- Process – collection of Applications, Hardware, and OUs. Process is an integral and independent subpart of a business flow.

The most important entities are Application (system), Hardware (server), and OU. They make the majority of a business flow. The first two are also main targets for capacity analysis. Capacity dashboards are built around the performance of an *Application* or *Hardware* entity.

Another thing notable about an *Application* or *Hardware* entity is the distinction between its physical instances and the master entity itself. The master entity is a conceptual existence representing one particular type. For example, *MktFeed* is a type of *Exchange Feed* application. However, there is no single application called “MktFeed” running in reality. What actually perform the type of business operation are its physical instances. They are *NYCE feed*, *ARCA Feed*, and *Nasdaq Feed*, each responsible for one market. Each physical instance might be used by a different OU

and participate in a different business flow. Therefore, it makes sense to distinguish them to manage their capacity separately. It also makes sense to look at all instances as one master entity in a summary view.

Process is a relatively special entity among the five. It is a composite entity made up of the interconnected entities. The elements of a process act as a whole for one business purpose.

Various relationships exist between enterprise entities to indicate business logic. Relationships include hardware to application, application to application, hardware to organization unit, and organization unit to organization unit. Relationships make up chains and chains make up networks that are part of the entire business hierarchy. The simplified hierarchy can be illustrated by the following diagram:

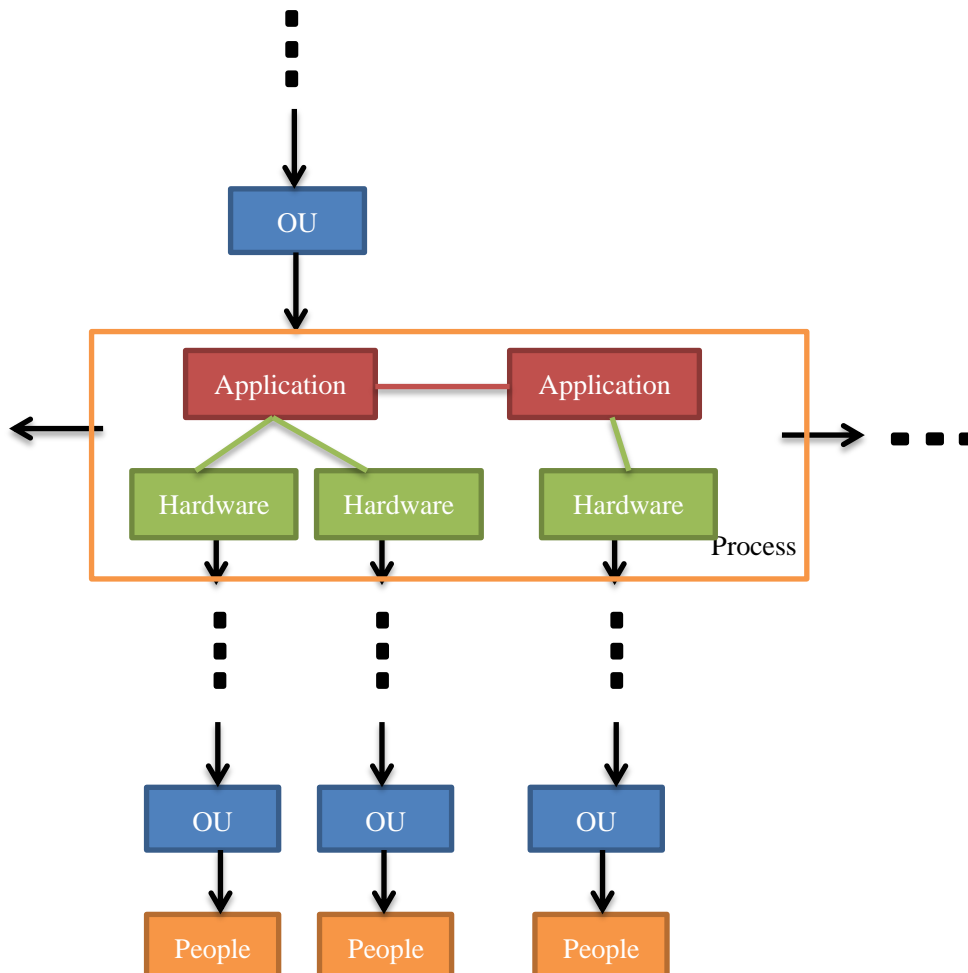


Figure 1: Hierarchy.

The hierarchy starts with organization units, for example, BNP Paribas or GECD. These *OUs* usually take charge of some business process, which consists of

interconnected applications and hardware that support them. The process can flow into other processes or business entities at the same level. The process can also flow down into another group of *OUs*, usually the infrastructure teams that support the hardware in the process. The hierarchy keeps running down until reaches the ends where individual employees under a particular OU reside.

To see samples of real data, readers can refer to table *Computer System, Infrastructure, ProcessID* (for entities) and *Relation Classification, Map Infrastructure to System* (for relationships) in Appendix C.

2.3. Technologies Review

A variety of technologies contributed to the development of the project, either at the tool assessment stage or in the formal development phase. Five tools and six languages are included here. Below is a literature review to each of them.

2.3.1. Tools

Graphviz

Graphviz, short for Graphviz Visualization Software, is a set of open-source tools provided by AT&T Labs Research for drawing graphs and networks. The software takes in the description of the graph in a simple text language called DOT, which can then be rendered into a variety of useful formats, such as JPEG and SVG (for web pages), with different layout options including “spring model”, radial, and circular layouts. In the market, Graphviz is known as industry-standard graph visualization software. It has many important applications in networking, bioinformatics, software engineering, database and other technical areas.

ApEx

Oracle Application Express (Oracle ApEx) is a fast application building tool based on the Oracle databases (Oracle Application Express). It requires no installation on the user side and can be used as pure web-based development. It can be used for building departmental-style applications with a dozen users, but can also scale up to handle thousands of users (Oracle Application Express). The framework itself adds very little overhead to each page request so the performance is only affected by the

efficiency of the SQL queries built in the application.

Microsoft Visio

Microsoft Visio is a commercial diagramming program for Microsoft Windows that uses vector graphics to create diagrams. It has developer capabilities which allow the application to integrate with Visual Basic to produce automated responses. Visio comes with a large number of shape templates that correspond to different popular designs such as implementing an employee hierarchy. Visio focuses a variety of simplistic measures while maintaining a graphically pleasing display.

Oracle Data Modeler

SQL Developer Data Modeler is a unique, free data and database modeling tool, which provides a full range of utilities to support all data modeling needs. Data Modeler uses the .XML coding language to produce a graphical diagram. Data Modeler displays relationships between various entities, and allows the user to make simple data manipulations from the graphical level.

2.3.2. Programming Languages

Python

Python is a general-purpose, high-level programming language. It is designed to emphasize code readability. Its use of indentation for block delimiters is unique among popular programming languages. It has remarkable programming power which includes large standard library and comprehensive modules. Similar to Scheme, Ruby, and Perl, Python is also a dynamic language. It is often used as a scripting language, although it can be applied to non-scripting contexts. The reference implementation of Python is free and open source software and has a wide community-based development model.

DOT Language

DOT is a plain text graph description language. It is a way to describe graphs. The files that use DOT language usually end with *.gv* or *.dot*. Many programs can process DOT files, including *dot*, *neato*, *fdp*, and *circo*. Most of the programs are part of the Graphviz package.

SQL

SQL (Structured Query Language) is a programming language for managing data in relational database management systems (RDBMS). It has a variety of capabilities including data insert, query, update and delete, schema creation and modification, and data access control. It is the most widely used database language.

PL/SQL

PL/SQL (Procedural Language/Structured Query Language) is Oracle Corporation's procedural extension language for SQL and the Oracle relational database. It is designed specifically for the seamless processing of SQL commands. Server-side PL/SQL is stored and compiled in Oracle Database and runs within the Oracle executable. As one of the three key languages embedded in the Oracle Database, it automatically inherits the robustness, security, and portability of Oracle Database.

HTML

Technically speaking, HTML (Hyper Text Markup Language) is not a programming language. It is a markup language that uses tags to describe web pages. HTML documents are equivalent to web pages. Web browsers read these documents, interpret the content by HTML tags, and display them as web pages.

JavaScript

JavaScript is a prototype-based scripting language. It is primarily used in the client side, implemented as part of a web browser to provide enhanced user interfaces and dynamic websites. Besides web pages, it has other applications such as PDF document.

3. Requirements

The requirement of the project came from three major areas:

Dashboard on System (Application) Performance

- System Monitor

List of Systems

- System Cartography
- Systems by Function
- Systems by Business Unit

List of Servers

- Server Cartography
- Servers by System

(BNP Paribas. “Enterprise Architecture Overview.”)

The first requirement was to build dashboards to monitor system (application) performance. This was the step stone to capacity management. In order to build performance dashboards, the systems and the hardware (servers) that support them needed to be understood with the context of business flows. That was where the second and third requirements came in – to visualize the list of systems and servers. The visualization was not necessarily a messy picture with everything included but could be part of the entire organization within certain contexts. Systems could be viewed with the context of business function or business unit while servers with the context of system.

The requirements also included what the visualization ideally should look like. The following picture is a business flow of systems with performance dashboards.

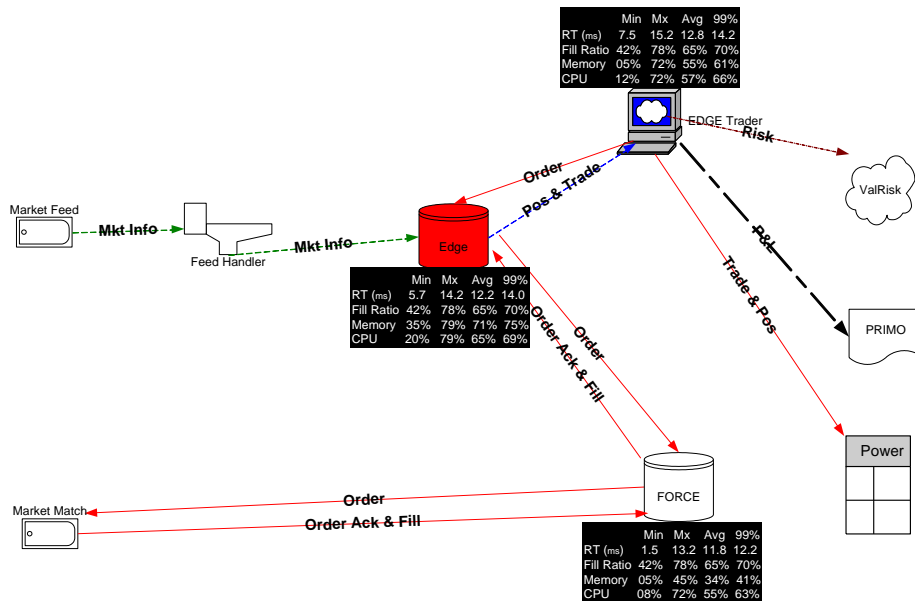


Figure 2: Graph Example from Requirements.

(BNP Paribas. “Enterprise Architecture First Phase – a tool for capacity management.”)

The example shows systems (nodes) and feeds (links) between them. Note that the three applications, *Edge*, *FORCE*, and *EDGE Trader* have their corresponding capacity dashboards (metrics) displayed by side. The dashboards summarize how the application performs within a period of time, measured by key aspects like Memory and CPU. One of the applications, *Edge*, is color-coded in red because based on the performance statistics it is diagnosed as currently over capacity.

Recall that the background section elaborated on the difference between a master entity of an application or a server and its physical instances. Correspondingly, requirements of the project also emphasized on the distinction between a summary graph and a detailed graph. The application that was to be developed was required to implement both of the two.

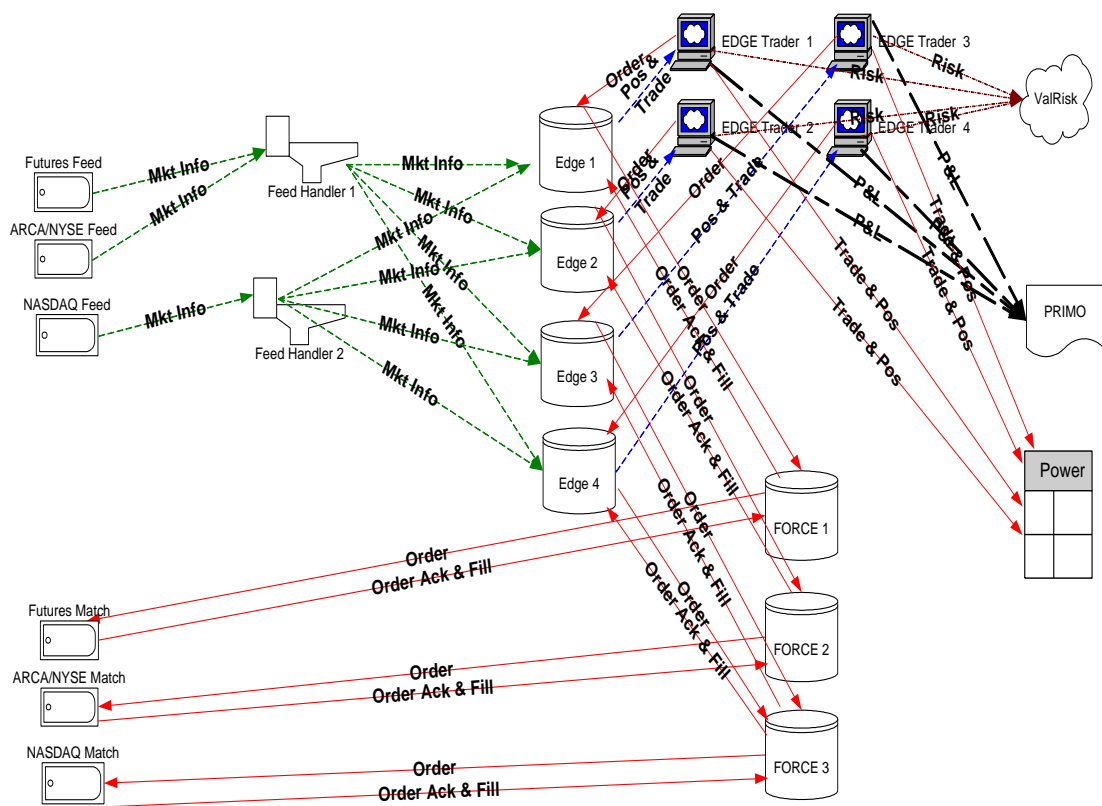


Figure 3: Detailed View. A detailed graph would display every individual instance. (BNP Paribas. “Enterprise Architecture First Phase – a tool for capacity management.”)

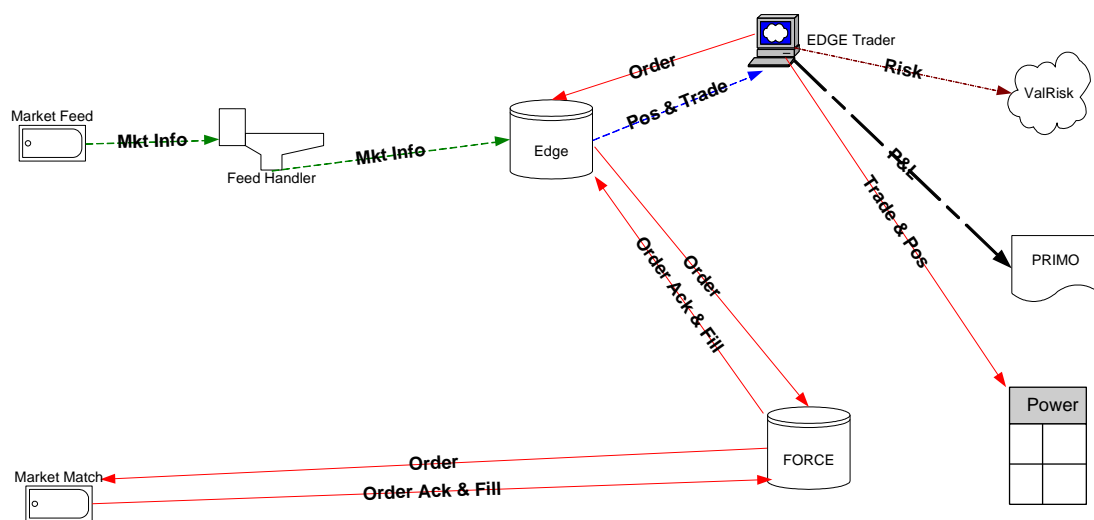


Figure 4: Summary View. A summary graph only shows the master entity. (BNP Paribas. “Enterprise Architecture First Phase – a tool for capacity management.”)

4. Business Uses

The requirements came from stakeholders each with their own need from the same application. In general business uses of the application can be grouped into three categories.

- Performance Management
- Personnel Management
- Technical Management

4.1. Performance Management

This is the most important business use of the three. If a server is over capacity, it gets reflected on the graph by the color. The manager sees the alert and wants to see what is going on with that server. He would be able to drill down to see the performance metrics (dashboards) built for that server. It is a log showing detailed information about how the server performs over a period of time. Any problematic record is highlighted so the manager could spot it at the first sight. Since the server is graphed within the context of a business flow, the problem is understood with reference to other entities the server is connected to. This use can be applied to systems (applications) and hardware (servers).

4.2. Personnel Management

One of the interested parties of the data is Human Resources. They are interested in organizational charting to improve personnel management. The graph shows a clear relationship between organization units and people under them. It is easy for HR people to see what position each employee falls in within the entire organization. Any personnel update can be reflected in the graph. The graph also shows what application or infrastructure component any employee currently has access to. If an employee moves from one department to another, what to deprive him of and what to grant him is easy to see from the graph. This helps to make access granting in a huge organization more regulated.

4.3. Technical Management

Similar to people in an org chart, processes can be shown in detail on the graph.

Managers can analyze the elements and the flow between them in a particular process. They can also see how one process interacts with other business flows. Together with performance management, managers are able to conduct efficiency analysis, optimize resources allocation, and expand or retract businesses.

A full list of interested parties and business uses can be seen in table *Interested Parties for Data* in Appendix B.

5. Development Phase I: Tools Assessment, Prototyping

5.1. Apex + Graphviz – in-house development

In this part of the tool analysis, Graphviz was tested as a graphing engine and Apex as a data management interface. In order for the two separate tools to act as an integral application, their interactivity with each other was also tested.

5.1.1. Pre-analysis on the tool

Candidacy of Graphviz

- Embedded with optimized positioning algorithm
- Standard in representing structural information
- Rich in custom options

Candidacy of Apex

- Fast development
- Specialized in building applications from database
- Web-based
- Easy to deploy (end users access the application through a URL)
- Easy to administrator (scalable for thousands of users)

5.1.2. Technology used

Programming Languages

- Python – automated the generation of input files in the DOT language.
- SQL – embedded in the Python script to make database queries. It was also widely used in Apex as the source for many standard components such as Interactive Report.
- PL/SQL – a major developing language in Apex together with SQL. It provided the source for Apex Processes and communicated between database references and application references.
- HTML – customized the displaying features both in Graphviz and Apex.

Software

- Python 2.7 – recommended bugfix release of Python.
- Graphviz-2.28 – latest stable release of Graphviz.

- Oracle Application Express 4.1 – latest release of Oracle Application Express
- Oracle SQL Developer – manages the data in database from a development level.

5.1.3. Database and Sample data

For the purpose of evaluating the tool, an Oracle database called “GECD_EnterpriseArchitecture” was set up, along with nine tables filled in. There were four types of Enterprise Entities: Application, Server, RoleTeam, and Employee and five types of relationships: Application to Application (AA), Application to RoleTeam (AR), Application to Server (ASERVER), Server to RoleTeam (SR), and RoleTeam to Employee (RE). There were two major relationship chains, one for business user teams, and the other one for infrastructure teams. They can be illustrated by the following flows.

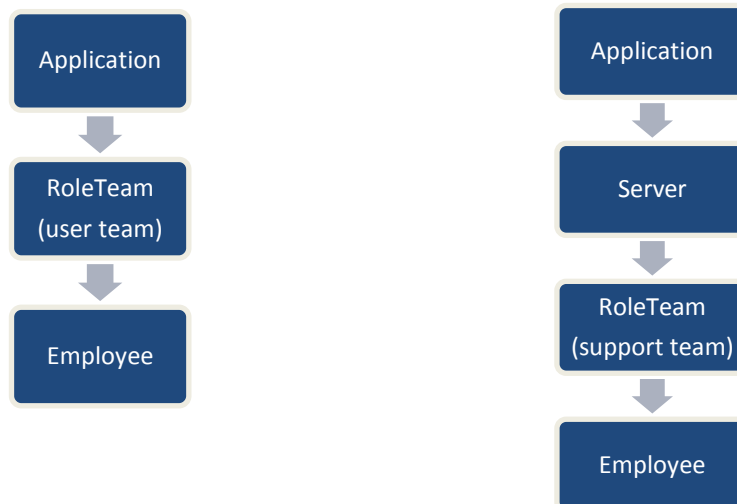


Figure 5: Two Flows in Graphviz testing.

Although the sample data used at this stage was largely simplified from the real data, it preserved the essential hierarchical structure of the business flows and sufficed for the testing purpose.

5.1.4. Development in Graphviz

5.1.4.1. Database connection

cx_Oracle, a Python extension module, allows access to Oracle and provides Python database API (Application Programming Interface). The following code used the module to establish the connection and fetch the data.

```

Import cx_Oracle
conn = cx_Oracle.Connection ("EA_APP/EA_APP@GECD_EnterpriseArchitecture")
Cursor = conn.cursor()

```

5.1.4.2. DOT language and the graph

This section is to illustrate what basic format of file Graphviz takes as input and what the outputted graph looks like. In sample.txt file, write a piece of code as follows:

```

digraph myGraph {
nodeA;
nodeB;
nodeA ->nodeB;
}

```

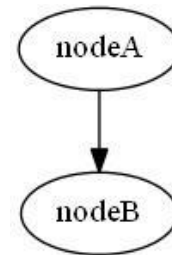


Figure 6: Sample Graph I

In the example, myGraph is declared as a digraph, which means links are directed with arrows pointing the direction. Inside ‘{}’ is where the nodes, links, and their attributes are defined. Here two nodes, nodeA and nodeB, and one link from nodeA to nodeB are created in the graph.

Instead of manually inputting the data and formatting it in the text file, python code automated this process and generated the graph systematically.

5.1.4.3. Automation Procedure Architecture

The following flow shows the automatic drawing process that was coded in the Python script.

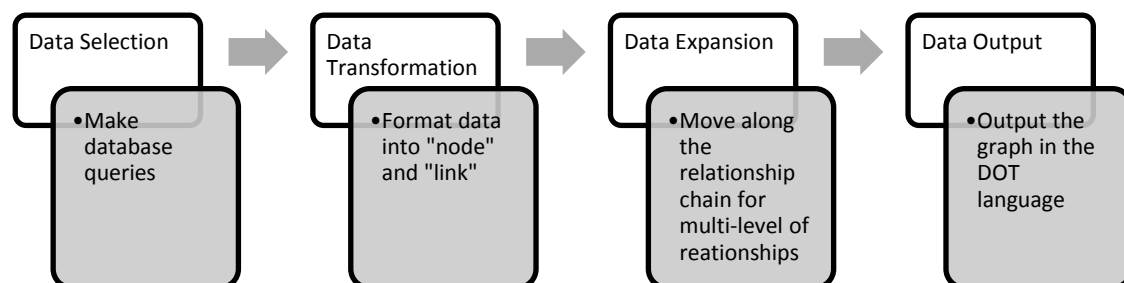


Figure 7: Automation Procedure Architecture.

5.1.4.4. Implementation Design: move along different levels of relationships

Structural information of interest does not usually consist of one level of relationships. As to graph a multi-level relationship correctly, for example the

relationship of application-server-role/team, a way to pass the result of the first SELECT statement (for application - server) to the second statement (for server-role/team) was necessary.

There were two possible ways to pass the variables. The first one was to select all the data needed for graphing in one single chunk of code. This was equivalent to creating a master table with all the entities and linkage information assembled. The second option was to have each relationship in its individual SELECT statement implemented in separate functions in Python and pass the variables in function calls. In this way, the 'variable passing' was done dynamically at runtime.

The first way was technically simpler. However, it had the major drawback of inflexibility. Simply by varying the number of levels to visualize in the graph and the structure of the relationship chain could produce many different graphs. Writing SELECT statements for each of them was not a good practice of software engineering. There was a lot of repetition involved. The code was also hard to maintain in the long term and scale poorly. In contrast, the second way was much more flexible. It dealt with one level of relationship at a time and generates multi-level graphs by reusing each level and put them together. The variation was encapsulated at a higher level where it was easier to accept user input and dynamically created new combinations.

The second design option was implemented. Each relationship had its own function where only one level of data was selected. The function prepared for next level data selection by returning the list of destination node ids. When the next level function took in the list as a parameter, it used the list as part of WHERE clause and began another cycle of selection. Here is some sample code:

```

1
def selectionAR(appIds, appNames = None):
    idstring = formatIDs(appIds)
    namestring = formatNAMES(appNames)
    return"""
        select
        A.NAME,
        A.A_ID,
        (SELECT NAME FROM ROLETEAM R WHERE R.R_ID=AR.ROLE) TEAM,
        (SELECT R.R_ID FROM ROLETEAM R WHERE R.R_ID=AR.ROLE) NEXT
    from APPLICATION A
    LEFT JOIN AR
    ON A.A_ID= AR.APP
    where A.A_ID = ANY {0} OR UPPER(A.NAME) = ANY {1}""".format(
        idstring, namestring)

```

Figure 8: Selection Function. The function took application ids as input, formatted them and used them in the selection statement.

```

1
def drawAR(cursor, wfile, hilite = False):
    nextIds = [];
    for col1, col2, col3, col4 in cursor.fetchall():
        wfile.write(drawApplicationNode(col1, col2))
        wfile.write(drawRoleNode(col3, col4))
        wfile.write(drawLink(col1, col2, col3, col4))
        nextIds.append(col4)
    if hilite:
        wfile.write(highlightNode(col3, col4))
    return nextIds

```

Figure 9: Process function. The function recorded this level of information to the output text file and returned the nodes for next level.

5.1.4.5. Functions

By the end of coding in python, the following aspects were explored and (or) implemented.

1) Customized colors and shapes for different levels.

In terms of varying the appearances of nodes and edges, Graphviz has a variety of options to choose from including shapes, colors, and styles (Node Shapes). Here is a simple example to illustrate how it works.

```

digraphmyExample {
nodeA [shape = box,color = red,style = filled];
nodeB [shape = ellipse, color = blue, style = unfilled];
nodeA ->nodeB;
}

```

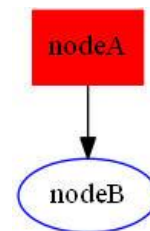


Figure 10: Sample Graph II.

The Dot language provides over 100 types of attributes to manipulate with (Node,

Edge and Graph Attributes). The full list can be found the documents in Graphviz.Org.

2) ‘Search’ functionality

Users could select the items they wanted to start the visualization with by providing the python program with command line arguments. The program implemented a complete set of Application Programming Interface (API) ready to use. No matter what type of application would be used in the future to accept user selection, as long as the type and an id list of nodes were supplied, the program could generate the desired graph automatically.

3) Expand in three directions: upward, downward and in the same level

Starting from any level with any node, the graph was generated systematically by expanding upward, downward, or remaining at the same level.

5.1.4.6. Picture Rendering

Graphviz supports a variety of graph formats. To generate a specific type of format, for example, the JPEG format, navigate to the directory where the dot executable sits and use the following command:

```
dot -Tjpg sample.txt -o samplePic.jpg
```

For other formats, substitute `-T{jpg}` with other `-T` options. For example, Svg files can be generated by using `-Tsvg`.

5.1.4.7. Sample code and pictures generated

See Appendix A.

5.1.5. Development in ApEx

Very different from the development in Python, development in ApEx involved little build-from-scratch work. Most of the time, developers clicked on buttons, specified attributes from dropdown boxes, and wrote small piece of code. ApEx did all the underneath “dirty” work and provided developers with fancy-looking in-built components with powerful capacities ready to use.

5.1.5.1. Development Preparation

It is very easy to develop in ApEx. All a developer needs to do is request a

workspace from Oracle and get the username and password set up. After that, login in from the login page on Apex.Oracle.com website and start developing.

5.1.5.2. Most important features in Apex

The following is a walk-through of the most important features provided by Apex.

Interactive Report

Perhaps the most important and useful feature in Apex is Interactive Report (IR). It displays a table of records stored in database with columns as the developer chooses. The content of the table comes from a single SQL SELECT statement. The Report is very easy to create. A couple of button clicks and attribute specifications produce a very standard looking report. Functionalities like search/filter, sort, and edit each record come automatically with nice looking interface.

The following SELECT statement made a report as below:

```
SELECT AA.ID,
       AA.SOURCE,
       (A1.NAME) FROMNAME,
       AA.DEST,
       (A2.NAME) TONAME
FROM AA,
     APPLICATION A1,
     APPLICATION A2
WHERE AA.SOURCE = A1.ID AND AA.DEST = A2.ID
```

Id	Source	Fromname	Dest	Toname
65	4	FORCE	2	ExchangeMatch
81	1	MktFeed	3	ValRisk
1	1	MktFeed	2	ExchangeMatch
2	2	ExchangeMatch	3	ValRisk
101	2	ExchangeMatch	1	MktFeed
41	3	ValRisk	2	ExchangeMatch

Figure 11: Interactive Report. The report displays all the records in table AA (Application to Application) with their names aside.

The major part of the report, the table-like assembly of records, closely resembled the table AA in the database with more user friendly interface.

The Interactive Report has three inherent functions.

- Search and Filter.

A search on any column of the report can be performed in the search bar below.



Figure 12: Search Bar.

Click on the magnifier to select the column name and type in the string to be contained in the search results. For example, if the user wishes to search for all the records whose *fromname* contains 'exc', choose *fromname*, type in 'exc', and click the button *Go*:

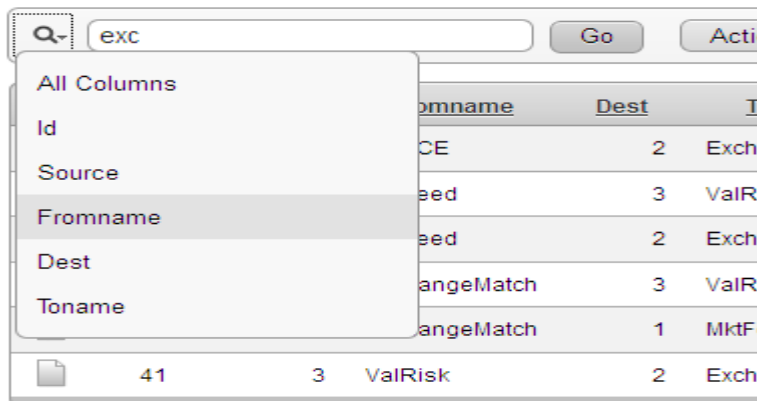


Figure 13: Search.

Now the report only displays the hits from search result.

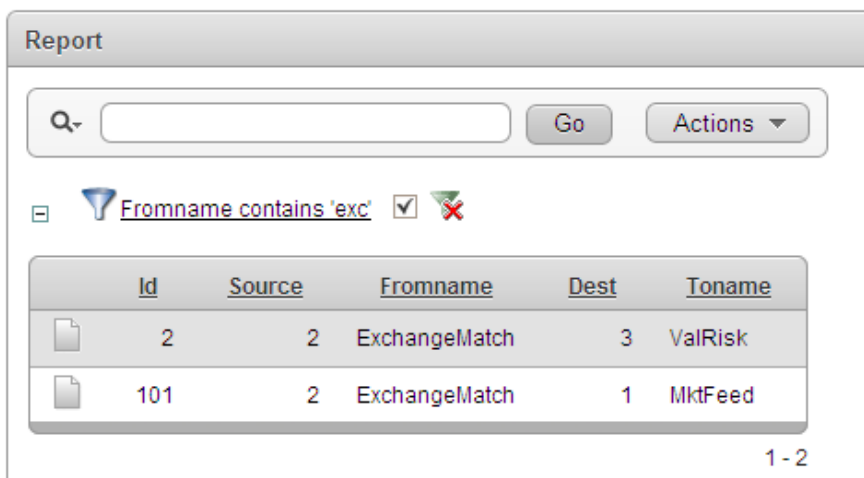


Figure 14: Interactive Report after Search. It only displays hits from the search.

If the user wishes to perform a more precise filter on the data, he /she can go to

Actions and select *Filter*. This opens up a new Filter window where the user can specify the filtering criteria.

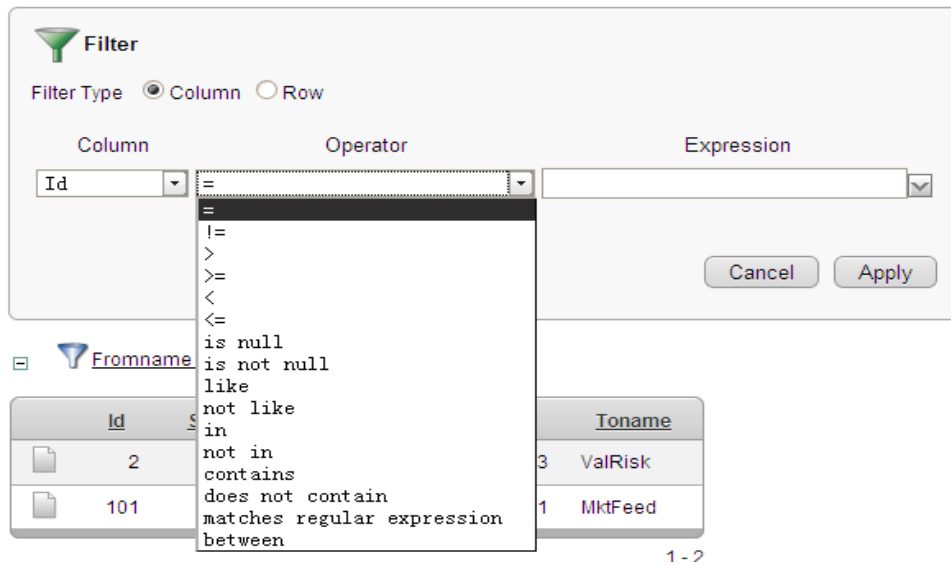





Figure 15: Filter.

- Sort

Sorting on any column is easy. Click on the column name and select  or  for ascending or descending sorting.

- Edit on the records

Click on the icon  and the user is directed to another page with all the detailed information about that particular row of record assembled and ready to edit. This is closely related to another powerful component in Apex - Form.

Form

Forms are usually used to edit one single record from a database table. They can be created in bundle with an Interactive Report for a table (*form page on table with report*), or by themselves. In either way, there is a way to tell the form page what record to load from the report. A Form typically looks like this:

The screenshot shows a dialog box titled "Form on AA". At the top right, there are two buttons: "Cancel" and "Create". Below the buttons, there are two input fields. The first field is labeled "* Source" and the second is labeled "* Dest". Both fields are currently empty.

Figure 16: Form for Creating. This is for creating a record.

The screenshot shows a dialog box titled "Form on AA". At the top, there are three buttons: "Cancel", "Delete", and "Apply Changes". Below the buttons, there are two input fields. The first field is labeled "* Source" and contains the value "4". The second field is labeled "* Dest" and contains the value "2".

Figure 17: for Editing. This is for editing a record.

If the form page is not created together with the report page, the developer has to establish the link himself. In the *Link Column* section of *Report Attributes*, choose the *Target as Page* in this Application and specify the page number of the form page. In the *Name and Value* fields, specify the Item name on the form page and the corresponding value to send from the report page. In this way, the form page knows which row of record is coming and what information should be loaded. Normally, the primary key of the table is the “messenger” and all the other fields of information are filled in automatically.

The analysis on the Interactive Report and Form showed Apex as a promising data management interface builder. IR displayed all the records of a table and Form allowed creating and editing any particular record for that table. The combination of these two fulfilled the need as to see and change the data. The powerful search/filter ability of IR also enabled managers to quickly locate a single record or a group of records without much effort. The selected records could later be used as the input for Graphviz to begin graphing with.

5.1.5.3. Advanced Interactive Report implementation

In terms of how managers want to use the Interactive Report to select some of the data to visualize, the standard IR lacks a way to indicate the selection.

Research showed that the right solution to this challenge was to use checkboxes as a separate column in the report. A PL/SQL package called `htmlldb_item` was used for this (APEX Check Boxes in Reports Regions). The package creates form elements dynamically based on a SQL query. The following code demonstrates how to incorporate the `CHECKBOX` function into SQL statements.

```
SELECT HTMLDB_item.checkbox(1,id) CheckBox,  
       r.id,  
       r.name  
FROM ROLETEAM r
```

Note that in the first line, the `CHECKBOX` function took in two parameters. The first one was an index determining which `htmlldb_application` global variable was used. For example, *1* indicated variable *F01* and *2* *F02*. The `htmlldb_application` global variable was the variable that kept track of the column of the report and could be referenced elsewhere in the application. The second parameter was the value of the checkbox, in this case, the `id` for table `ROLETEAM`. Below is the screen shot of the `RoleTeam` report with checkbox added.



Figure 18: IR with CheckBoxes.

To illustrate how the checkboxes work, imagine the user checks records with `id` 1, 2, and 4. Since there are three records selected, `htmlldb_application.F01.count` is 3.

The *htmldb_application.F01 (1)*, *htmldb_application.F01 (2)*, and *htmldb_application.F01 (3)* are 1, 2, and 4 respectively. In this way, the selection was well indexed and could be referenced somewhere else in the application.

The Checkbox solution provided an intuitive way for the user to select the records and for the interface to accept selection. By using the *htmldb_application* global variable, selected records could be referenced and prepared for the next step of action.

5.1.5.4. More Research on Apex

This section devotes to additional exploration in Apex at the stage of tool testing.

1) More page Item types

The default item types Text Field and Number Field were not so useful because users had to remember the entire string in order to edit. In such cases, Select List, Popup List of Value and Text Field with autocomplete were more helpful.

- Select List - a non-editable dropdown box
- Popup List of Value - a Select List with a pop up window instead of a dropdown
- Text Field with autocomplete - gives hints while users type in the string.

All three of them helped to make the application more user-friendly.

2) Print out dynamically changing text

When the developer tested on the usage of the checkboxes, there was a need to print out the values of *htmldb_application.F01* variables. It turned out that Apex has a region type called 'dynamic content region' where the dynamically changing source can be displayed by function *htp.p*. The region was also used for debugging purposes.

5.1.6. Two way interaction

5.1.6.1. From Graphviz to Apex

The biggest obstacle that prevented Graphviz + Apex from being the right tool for the project was the static nature of the graph generated by Graphviz. It was believed by all the testers and developers that the graph provided no way of user interaction until the midpoint of the project.

The major breakthrough came from the finding that a URL link could be added to a node in the same place where customized attributes are defined. It was a very faint connection between the Graphviz and the outside world but it was also a very brilliant idea.

At the first phase of the exploration, a simple URL (www.google.com) was added to every node. The modified sample script looks like this:

```
digraphmyExample {
nodeA [shape = box,color = red,style = filled, url = "www.google.com"];
nodeB [shape = ellipse, color = blue, style = unfilled, url = "www.google.com"];
nodeA ->nodeB;
}
```

Research showed that the “hot region” for a node is defined as the entire area bounded by its shape. However, not all of the graph formats are clickable. As the Graphviz documentation from Princeton.Edu (Graphviz – QED) points out, URL links are only supported in ps2, cmap, imap and svg formats. Svg format was chosen later because it could be opened in a web browser.

The URL link was then customized based on the id of the node being clicked. After it was found that it was possible to embed the id in the link, the URL was changed to the path of a particular Apex page. The idea was to direct the user to the editing page for that particular node.

ApEx page URL

In order to pass in the parameters to the right place of a URL, it is important to understand the constitution of an Apex page URL. Note that most of the Apex application URLs have a part starting from f?p. There are usually 9 parameters that can be specified after.

f?p = 1:2:3:4:5:6:7:8:9

- 1- The application number. To access the current application id, use &APP_ID..
- 2- The page number.
- 3- The session. To access the current session, use &SESSION..
- 4- The request to pass to the destination page.

5- DEBUG. Decide whether the page is run in debug mode or not. either YES or NO

6- Two values, 'RP' for indicating resetting pagination and the other for pages where cache is reset.

9- Printer friendly. (URL Parameters)

Parameter 7 and 8 are very important. Parameter 7 is a comma separated list of page items ready to be assigned the values from the comma separated list of values specified in Parameter 8 (URL Parameters). As in the case here, the URL attached to any node could be

http://nycs00057970:8080/apex/f?p=100:9:&SESSION:::NO::P9_NODEID:4 (Oracle Application Express APEX: Passing Values between Screens)

Here the URL linked the node to page 9 of application 100. An Item on that page called P9_NODEID was set to 4. The page 9 was a form page for editing a node. By passing the value 4 to the node id on that page, the information of that record could be pulled out and loaded into the fields of the page, ready for editing. In practice, the value 4 could be substituted by any variable holding the id. Setting the Parameter 7 and 8 of a page URL like this was extremely critical and useful for the communication between Apex and the outside world and within Apex itself.

Therefore, by manipulating page URL manually and incorporating it in the input dot script for Graphviz, the one-way interaction from the graph to the data management interface was established.

5.1.6.2. From Apex to Graphviz

The ideal way of user interaction from Apex to Graphviz was to have a button trigger the drawing procedure. The procedure took the records selected as input and ran an OS command based on that. A button and a triggering process were not difficult to implement in Apex. However, how to invoke the OS command from Apex was a problem. There were basically two ways to do this. One was by external procedure call, and the other was by dbms_scheduler job. In both cases, the command

would be executed on the same server as the Oracle database resided. After some research, external procedure calls turned out to be more complicated. Therefore, `dbms_scheduler` was chosen.

The possibility to achieve two-way interaction was critical in the assessment of the tool combination of Apex and Graphviz. After this founding, for the first time the tool combination was proven a feasible option.

5.1.6. Evaluation

Graphviz and Apex have many attributes that made them strong candidates for the project.

Merits on Graphviz

Based on the pre-analysis, Graphviz was proven to have:

1) Embedded positioning algorithms

By simply specifying in command argument major layout to use (Using GraphViz, a Brief Tutorial | Orient Lodge), the user gets a nicely displayed graph. Graphviz does all the intelligent work behind the scene. This largely reduces the amount of programming work that needs to be done.

2) Great scalability

Graphviz is the industry standard in representing structural information. Its positioning algorithms optimize the arrangement of the nodes and minimize the overlapping of the links. This largely increases its scalability for a huge amount of data, which is the amount the developers would be faced with.

3) Various attribute options

As shown in sample code and graph, Graphviz provides a variety of attributes including colors, fonts, line styles, and custom shapes that help distinguish between different types of nodes and links.

Based on the python program, Graphviz was found to assist in automating the process of feeding data and drawing. As opposed to many other graphing engines, it requires no more special file format than `.txt` as input. The syntax of its language,

DOT, is also simple and programmatic. Any programming language can generate the format automatically without much effort. Furthermore, the separation of the input, which contains the pure master data, and the rendering process, which outputs master data together with attribute data, makes the automation even simpler. Developers can plug their own data at the point of providing the input file without affecting the step of rendering. Recalling from the requirements, one of the basic components is to automatically pull out data and graph it. Graphviz satisfies the requirement easily.

Merits on ApEx

ApEx was chosen to complement Graphviz both as a data management tool and a control panel. Apart from those listed in the candidacy section, the following points are stressed here:

1) Standard database application builder

ApEx is designed for database applications. The source of many components in the application is a single SELECT statement. This specialty easily satisfies the need to build a data management interface above the underlying database.

2) Easy deployment

No installation on the user end is required. As soon as a username and password are granted by the administrator, the user can login in and play with the tool. This shortens development cycle and allows easy enhancements even after the application is in use.

3) Easy development

ApEx provides standard components (i.e. the Interactive Report) that come with powerful functionalities. All the developers need to do is specify attributes of the component by filling out forms and selecting from dropdown boxes. It is much faster than the standard programming way of building up bit by bit in a programming script.

4) Powerful administrative management

As the requirements pointed out, the project was designed to incorporate the need from different groups of users. In terms of this, ApEx has powerful administrative and security management capacity. In the *Administration* section of the application,

different users can be granted different set of permissions. Many components of the application also have a *Security* section which prevents certain group of users from using it.

5) Multi-language supported

ApEx supports not only one language. To interact with database, it uses SQL and PL/SQL. To render pages within the application, it supports html and javascript. The latter two languages are also widely used in tweaking the appearance of a standard component and defining processes or functions to link components.

Merits on the combination

In terms of the combination of the two tools, the bi-directional interaction of Graphviz and ApEx was established one way by hyperlinks and the other by stored procedures and `dbms_scheduler`. In this way, at least on a preliminary stage, Graphviz and ApEx could be integrated into one united piece of software.

Drawbacks on the combination

There were certainly some downsides in this option. The most obvious one was the limited user interaction that can be performed on the graph. Most of the desired actions like create a node, delete a link, and drag and drop could not be done on the graph. Although the hyperlink approach guaranteed at least some communication between the graph and its underlying data, the way it worked was still the least intuitive. The other drawback lay in the development in ApEx. On the one hand, doing all the dirty work for developers saved development time and was good for creating standard-looking components. On the other hand, however, it hid necessary details from developers. Since developers did not know the code that implements functionalities like search/filter in the IR, it became much more difficult for them to extend and customize the functionality. In most cases, developers did not even know where to change some of the attributes although there were actually options in the dropdown boxes. This could affect the learning curve of the developer and become a problem at the early stage of the development. The third problem of Graphviz and ApEx was their separation. Although they could be connected, the combination was

still not functioning as an all-in-one-spot application. Binding them strongly and embedding the graphs in Apex page was difficult but definitely the next step to go.

5.2. Visio – in-house development

5.2.1. Hierarchical Design

The second application investigated Microsoft Visio. Visio was chosen because it has the capability of heightened user interaction and increased graphical displays making it a more viable option. Visio has several interfaces that vary in terms of functionality and automated display. Each interface corresponds to a different popular template. For example, one can make an employee hierarchy with the ‘Hierarchy Template’. Initially a hierarchical design, which was set up to process flows and nodes in a user friendly design, was tested. However, the hierarchical design did not provide the user with the level of freedom that would be needed to complete future stages of the project.

5.2.2. Free Form Design

Once the hierarchical design was deemed too limiting, a free form design was investigated. This free form design gave the user complete freedom. With this, a VBA macro was developed that allowed the user to automatically display a node and evenly placed and connected dependent nodes around it. However, flaws were found in this freeform design as well. The free form method would make it difficult to process large amounts of information, and the information that was processed would be confounded to a point that didn’t require unlimited user freedom.

5.2.3. Semi-Free Form Design

Finally an option was developed that allowed the user to place nodes freely and had several organized ways of connecting nodes.

A tutorial can be viewed at:

<http://support.microsoft.com/kb/309603>

This tutorial demonstrates how to create a template for programmability with Visio. It creates a simple program that opens up a Visio document, pastes an object on the screen, saves the documents and quits. Visio VBA language was the main component of this option. Initially this presented a challenge because there is not much information available pertaining to this language, aside from the tutorial discussed earlier. However, the VBA Visio application allows the programmer to

record macros. Therefore, the user has the ability to see a script of their actions within the programming language which is a key component for developing this application further. This feature also allowed for the discovery of the key VBA Visio commands which were implemented throughout the rest of project.

5.2.4. Methodology

5.2.4.1. Algorithm For Application Development

To develop the VBA Visio application, a table with employees and their direct superior was saved in an Excel file. The information from this file was then imported into Visio so that it was incorporated into a hierarchical display. To ensure that the employees were displayed in a tight format that avoids overlap an algorithm was developed. The original algorithm found the root of the hierarchy and gave that employee a level of 1. The algorithm then went through subordinates and gave each node a level which corresponds to its distance from the root. Next, the algorithm would evenly space the most populous level and apply the space to the entire hierarchy. It then connected all subordinates of the most populous levels, readjusting the superiors to avoid overlap. Finally the algorithm went through the data, level by level, placing superiors above the average of its subordinates until the boss was placed in the top middle. This algorithm provided a visually sound diagram of the hierarchical structure of the company.

5.2.4.2. Data Presentation

The next stage of VBA Visio development was to take a more comprehensive set of tables from a sample database that we created, process them through Microsoft Queries into VBA Visio and present them with Visio. This new set of data contained nine tables saved in an Excel file: 4 tables listing nodes (Applications, Servers, Teams and Employees) and 5 tables listing links (Application to Application, Application to Server, Application to Team, Server to Team and Team to Employee). The highest level would be the Application level. Applications had servers and teams associated with them. Servers had teams associated with them and teams had employees associated with them.

The goal of this stage was to give the user the ability to search for an individual node of interest. The Visio application would automatically display all of its dependents and superiors, and for applications it would show other applications that the chosen application interacted with.

To achieve this goal the first step was to create a master table using Microsoft Queries that incorporated all the information from the nine tables discussed earlier. Each row of the master table displayed every employee's name and ID with the names and ID of their team members, server (if applicable) and application. The query would run through the table and the table would be saved as an Excel table which was then called upon by the VBA Visio application.

After the specific excel table, constructed based upon the initial query, was imported by VBA Visio, the information from the query needed to be streamlined directly into the VBA Visio code. A connection was established to allow VBA Visio to run query scripts and then each line of output from the query was saved into a master array in VBA Visio. A complimentary step involved replacing the home of the data tables from Excel to Oracle. A connection had to be established between Microsoft Queries and Oracle and the queries had to be tweaked to accept data from a different source.

5.2.4.3. Algorithm Optimization

The next step was to update the pre-existing algorithm to handle the new dataset. Fortunately, the levels were predefined by node type making the update possible. Initially, if an employee worked for two teams he would only be displayed once and he would ideally be placed in a way that made sense for both sub structures. However making this ideal situation a reality was very difficult because it involved creating an overlap adjustment algorithm which shifted the position of overlapping nodes while trying to minimize the width of the entire graph. After experiencing many different complications with this overlap adjustment algorithm, it was decided that if a node was part of multiple superiors it would be displayed multiple times on the graph. This allowed the process to be considerably simplified and was more user-

friendly.

5.2.4.4. Increased User Ability

In the original algorithm, the user only had the capability to select an application and the application and its dependents would be displayed. The updated algorithm involved going through the master array and assigning a 'y location' to each node associated with the selected application, based upon the node type (employee = 1, team = 2 etc.). Then, the algorithm would go through all employees associated with the application and assign them 'x locations' in sequential order. Since the master array was sorted in the query, this was a relatively simple process and there was no chance of crisscrossing nodes. Next, the teams, servers and applications were placed above the average of its dependents similar to the original algorithm. Finally, a rectangle was created for each node and placed in its respective place given its x and y location.

Along with effectively organizing nodes, the Visio application offers several types of linking options. Links can be completely manually created by giving an x and y coordinate to the start and end points. The links can snap into the bottom, top, left or right sides of two nodes or a new node and a link can be created as a relationship to an existing node. It was determined that the snap method was the most effective and provided the most freedom while not over complicated this part of the project. This feature was then incorporated into the algorithm so that went through each node starting with employees and snapped a link between its superior and the node.

Additional logic was added to allow the user to have the root be something other than application. This new logic also gave the user the ability to display the application to application level visually. A smaller query was installed into VBA Visio that made an array containing application to application information and each node was given an x location that corresponded to its sequenced ID number and centered around the selected application. Finally, each node was given a color that corresponded to its node type. This optimization completed the second phase of the project.

5.2.5. Bi-directional Interface

The next stage of the project was to allow the user to make changes to the graphical display. These changes must be designed to change the information stored in the database. It was thought that this would be promising for the Visio application because it has highly presentable interface. To make Visio a bi-directional application, three different possibilities were investigated.

The first option involved creating a control panel. This control panel was based on the display produced by the algorithm. The user is able to add, remove or edit a node or connection, or even re-center the graph around another node. When a control is selected it would reprint the graph, while simultaneously updating the database. The second option was a scanning method. The user would be able to make changes to the visual interface and press a button that scanned the graph. The program would compare all the objects to the pre-scanned graph and update the database with all changes. The final option was to record keystroke events taken by the user and decipher what the user changed and update the database.

The control panel option was dismissed, because it would be easier to create a control panel using ApEx/Graphviz. Therefore it was determined that this was not worth pursuing. The scanning method would only be viable if there was a small amount of data, otherwise it would be too slow to be an effective method. The key stroke option proved to be the most optimistic as it did only the necessary amount of work to complete the task. However, it is important to note that if the database was smaller, the scanning method would be easier to implement. More information about the possibilities of a small database application is mentioned in the conclusion of this section for future use.

Based upon the assessment above, the most effective way to record keystroke events was to use the already built in 'record macro function.' The record macro function allows the user to make changes to the interface while simultaneously saving VBA Visio code that would update to include the changes that you made. This means that if the user reverted back to the pre-change graph and ran the recorded macro it

would produce the same results as the manual changes did. The user could press a button that would save the old version, press record macro, make necessary changes, stop the recording and, press a button to save the new information. Ideally this could be shortened to pressing a button that saves and starts recording, make changes and press another button that stops recording and saves information. However, those capabilities were not reached. Instead, when recording was stopped, the user presses a save button that saves the recorded macro into a text file and then imports the text file into VBA Visio. VBA then searches the text file for triggers that tell the program that the user added, removed or edited nodes or connections. It would then take this command and update the database.

Work was done to explore the display ability of this application. When the user searches for the initial node structure, the nodes would be placed behind four translucent different colored screens corresponding to each node type. If the user wants to manipulate the data, they must click on the 'prepare to edit' button which saves the file as it is and brings the white nodes over the translucent barrier. This enables the user to move, delete and change the nodes and connections. When changes were done being made, the user can press 'commit' and the database would be updated and the nodes were again placed behind the barrier.

Finally, when a node was not in edit mode, the user is able to click an invisible button located above barrier and node itself. Clicking on this button would bring up all necessary information concerning the node as well as collapse, expand and bring to front options.

Although Microsoft Visio has many advantages like display ability and high functionality, it wasn't chosen as the graphical application for the Enterprise Architecture problem. This was because it is not a web based application, and can only deal much smaller amounts of data than needed for this project. Visio would be a great application for a more containable project involving a single team or process.

5.3. Oracle Data Modeler – in-house development

5.3.1. SQL Developer Investigation

SQL Developer was the first application used to achieve this project's goal. SQL Developer was chosen for this project because it allows the user to drag nodes (processes and applications) onto a screen and connect them using flows. After the user makes changes to the information of the screen it will automatically be saved as a .xml file. The user has the ability to to manipulate the .xml code to hide and show nodes, delete and create links, and change their source and destination through manipulation of the .xml code.

5.3.2. SQL Developer Conclusions

Although this application was more interactive with the user, it was not capable of deleting and creating nodes through the .xml code. These are two major flaws in this program which makes the application less desirable. After investigation into the SQL Developer application, it was determine that the user could not control the graphic interface through the basic code and as a result was not pursued further.

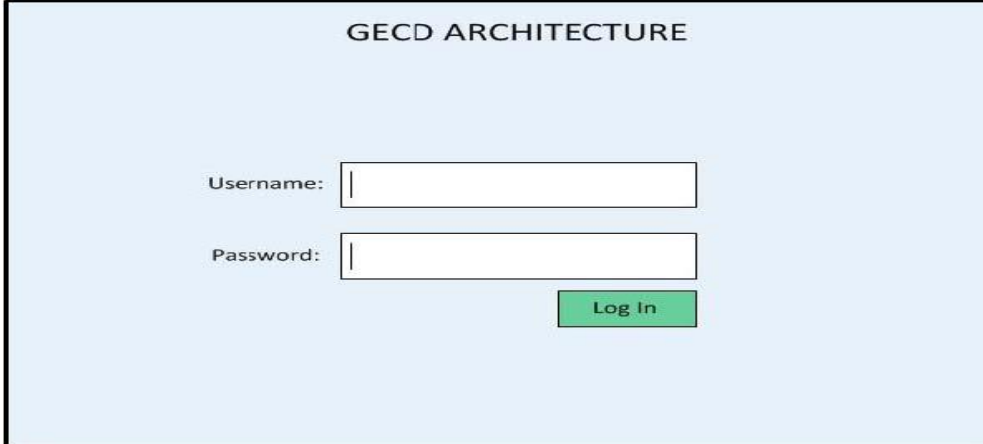
6. Development Phase II: Formal Development of Final Product

6.1. Detailed Design Plan

6.1.1. User Interface Design

Based on the requirements, the following user interface was designed to be implemented in Apex.

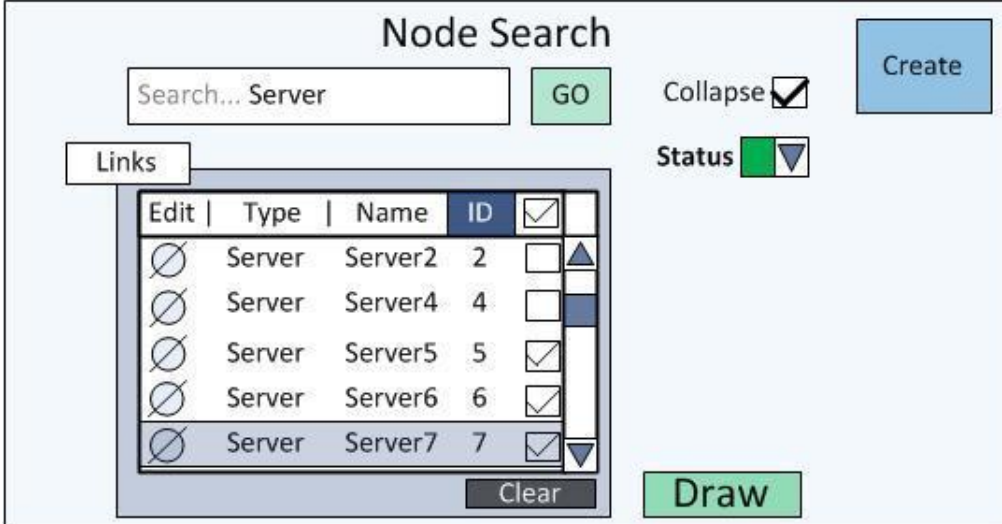
Page 1: Login page



The login page features a light blue background with the title "GECD ARCHITECTURE" centered at the top. Below the title, there are two input fields: "Username:" and "Password:". A green "Log In" button is positioned below the password field.

Figure 19: Login Page Design.

Page 2: Main page with the visualization on the bottom



The main page is titled "Node Search" and includes a search bar with the placeholder text "Search... Server" and a green "GO" button. To the right of the search bar are a "Collapse" checkbox (checked) and a "Status" dropdown menu (set to "Green"). A blue "Create" button is located in the top right corner. Below the search bar is a table with the following data:

Edit	Type	Name	ID	
<input type="checkbox"/>	Server	Server2	2	<input type="checkbox"/>
<input type="checkbox"/>	Server	Server4	4	<input type="checkbox"/>
<input type="checkbox"/>	Server	Server5	5	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Server	Server6	6	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Server	Server7	7	<input checked="" type="checkbox"/>

Below the table is a "Clear" button. To the right of the table is a green "Draw" button.

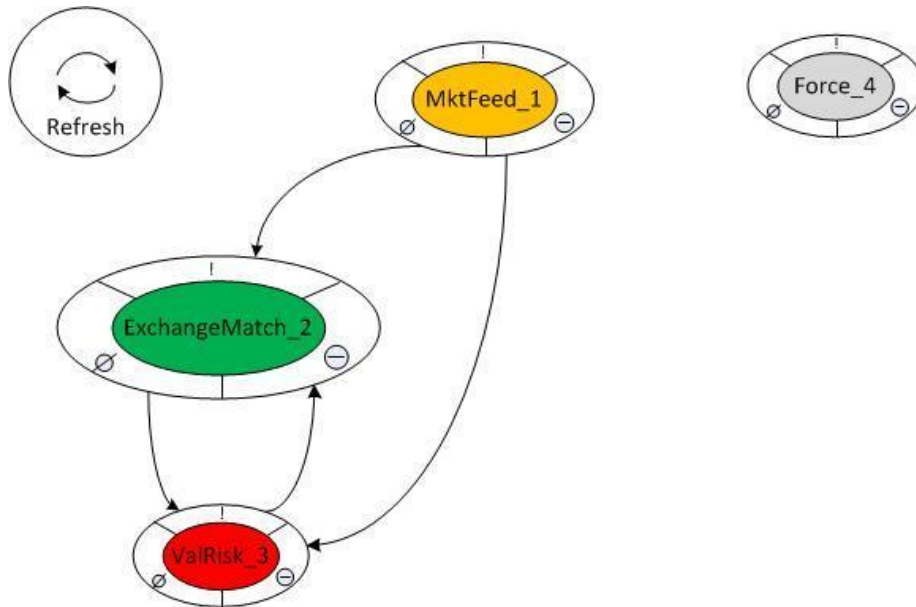


Figure 20: Main Page Design.

Page 3: Node Editor

Node Editor

ID 14 Cancel Delete Apply

Node Name

Node Type ▼

Links

Edit	ID	Link Type	Name
⊗	4	Server	Server4
⊗	8	Server	Server8
⊗	2	Server	Server2
⊗	3	Team	Supp3

Create

Figure 21: Node Editing Page Design.

Page 4: Capacity Management

Capacity Management							
Server/App	ID	Name	Metric	Value	Units	Threshold	Timestamp
Application	2	ExchangeMatch	Tx/Sec	3,000	#	2,000	11/15/2011 11:38:29
Application	2	ExchangeMatch	Latency	8	ms	4	11/15/2011 11:38:32
Server	4	Server4	% Cap	95%	%	75%	11/15/2011 11:38:35

Figure 22: Capacity Management Design.

Page 1 was the login page. By providing username and password, users could get into the application. Page 2 was the main panel displayed after the user logged in. All the information of databases was listed here in the master interactive report. The page also implemented controllers to interact with the graph. Ideally, the graph was displayed right below the main table. By clicking on the Edit icon of any record, the user was directed to Page 3, a *form* page, where detailed information of that record was ready to edit. For each record of Application or Hardware, there was a corresponding Page 4 showing capacity thresholds and metrics.

6.1.2. Database Design

There are six types of enterprise entities in the visualization. Five of them are represented as “node”, Application, Hardware, Organization Unit (OU), People, and Process, and one as “edge”, Link. Link is a source-to-destination two dimensional record, which preserves the relationship of the other five types of entities. Process is a composited type of node entity. It can not only be displayed as a single node to illustrate how different Processes interact with each other, but also as a group containing all the sub entities to show what a particular Process consists of.

How to store these six types of entities in the database was very important. One idea was to have a master table containing all the records of node type entities and another table for Link. It was convenient to extract information for the master interactive report on Page 1 in this way. It was also easier to make references since all the information is stored at one spot. However, different entities might have totally

different columns. It was hard to accommodate all the needs from different entities in one table. Furthermore, data of entities were provided by a variety of departments in the company. For example, data for Hardware was fed by infrastructure teams while information of People came from an organization chart. Organizing different entities into separate tables made it much easier to manage data from various sources.

With the analysis above in mind, the database was designed as follows. Each entity of node type (OU, Hardware, Application, People, and Process) had its individual table (EA_OU, EA_HARDWARE, EA_APPLICATION, EA_PEOPLE, and EA_PROCESS).

Link had its own table EA_LINKS with records that establish a relationship from entity of reference 1 to entity of reference 2. Each Link record had a nullable field called Process_ID, which indicated whether the relationship belongs to a particular process.

To feed all the data into the master report on Page 1, EA_APP_VIEW was made to compile all the records from the six tables and create a temporary “view” of the entire database. It was created solely by a SELECT statement that put all the data together. It was not a real table but could be used the same way as a regular table in the source of the interactive report.

Six tables and one “view” was a very important design decision made in the project. It realized both the generalization in the master report and the customization in each of the edit forms.

6.2. Technology Used

6.2.1. Software

Graphviz – Picture Generator

ApEx – Application builder

SQL Developer –Programming environment for stored procedures and database management

6.2.2. Programming Language

Dot Language – primary language used in input script for Graphviz.

SQL –primary language used to feed data in ApEx components. It was also heavily used in SQL Developer for database management.

PL/SQL – major language used in conditional expressions and process sources in ApEx. It was also critical for functions and stored procedures in SQL Developer.

HTML – secondary language used in input script by Graphviz. It also played an import role in customizing page appearances in ApEx.

Javascript –secondary language used in functions for ApEx.

6.2.3. Technical Support

ApEx was locally installed in a server (nycs00057970:8080) of BNP Paribas. It was necessary for dbms_scheduler to work because the scheduler needed to execute OS commands. It was also more convenient to reference local files.

Databases were also moved to the same server.

Oracle package dbms_scheduler was used in a stored procedure to trigger the graphing process.

6.3. Architecture

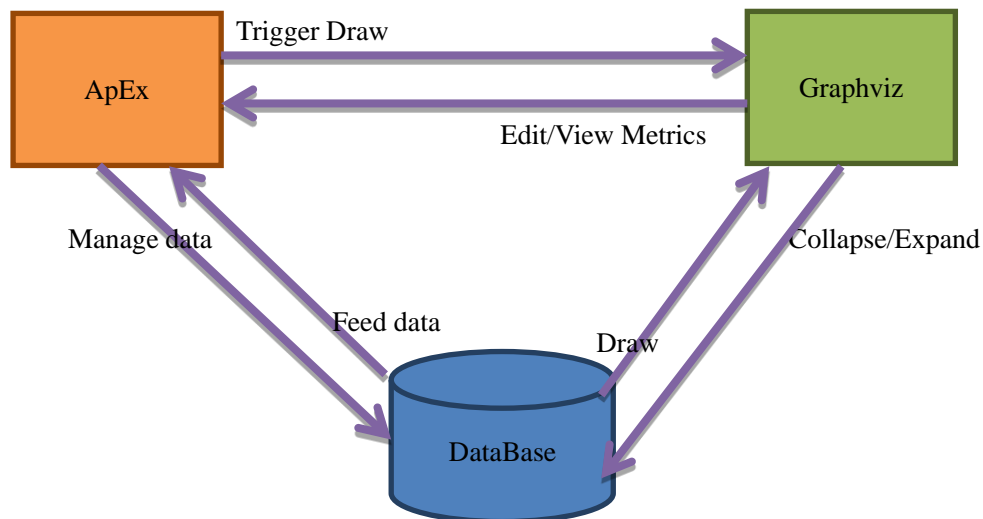


Figure 23: Architecture.

6.4. Data

In the formal development of the application, real data was plugged into the database.

6.5. Methodology

This is the section that explains the implementation of the most important features in the application.

6.5.1. The implementation of user preferences

User preference was a concept that users only saw the part of the data they selected displayed in a way they prefer. Data could be selected altogether in a pre-saved diagram or one by one through the checkboxes discussed in Phase I of the project. The way the data was displayed, either expanded or collapsed, could be specified in a separate checkbox.

Each graph had its own set of preferences and one user could have multiple graphs. Therefore user preference could not be stored together with the master data. Instead, two tables, EA_USERPREFERENCES and EA_USERPREFERENCEDETAILS were used to keep copies of the master data. Each copy captured the preferences set by a particular user for a specific graph.

The user preferences table kept track of the relationship between each graph and its owner. Each record in the table stored high-level information about a graph with a reference into the table EA_USER, where a specific user could be looked up. Detailed information of a graph was kept in the user preference details table. Records with the same reference id which pointed to the same graph made a full copy of the master data with SELECT/DESELECT and EXPAND/COLLAPSE columns set to user preference.

On the user interface side, the main report was dynamically populated by the data copy from the user preference details table based on what saved diagram the user selected to see. This set of user preferences could then be further tweaked by checkboxes. By checking the corresponding boxes, the SELECT/DESELECT or EXPAND/COLLAPSE column of a record was set and the user preference was saved back to the database. These two fields were critical for the “CRAWL” procedure to figure out how to generate the graph.

6.5.2. Improvement on graph interactivity

Hyperlinks embedded in sections of a node were more than web page URLs.

They could be used to launch a stored procedure located on the server specified by the URL. However, embedding stored procedures the same way as regular web page URLs had the side effect of opening extra windows.

Completely getting rid of the side effect was difficult given the way how URLs work. The compromised solution was to pop up the target window and immediately destroy it. This required several steps. Firstly pop up a second window with the target URL to launch the stored procedure. Secondly destroy the new window. Thirdly go back to the previous page in the main window since the main window has automatically changed to a blank page during the first and second steps. By the third step, it achieved the effect of “staying at the same page”.in the main window.

There were two possible places to implement the changes: the text file taken by Graphviz as input and the svg file outputted by Graphviz. Research showed that Graphviz has limited html options (Node, Edge and Graph Attributes). For example, it lacks the onclick attribute common in html script. This forbade the use of “on-click” event where the new solution could be possibly implemented. Graphviz also has no way to render JavaScript, which meant it was impossible to include multi-step arguments in one single URL.

The new solution was first implemented in JavaScript by directly altering the svg file Graphviz generated. New arguments were added among the xml code. Later, however, it was found that although Graphviz did not understand JavaScript, it treated the code as plain URLs without complaining. When Graphviz rendered the script, the code got passed on into the xml code in svg. This way it saved another round of processing of the graph before it got displayed.

At this point, the outgrowth of graph interactivity - the piled-up extra windows was replaced with a flash of pop-up. This largely improved user experience.

6.5.3. Embedment of graph in the application

Originally, users had to switch between browser pages to interact with both the ApEx application and the graph. Embedment of graph in the application finally realized the notion to build an all-in-one-spot application with all technologies well

integrated.

The graph was first included behind the “src” (source) tag of an *html* region in the application. The graph was displayed without a problem - graph interactivity was fully supported within the page. However, the display did not synchronize well with what was actually stored in the database. Whenever an update was made to the graph, the svg file was regenerated correctly; however, the display was not always up-to-date. This posed a serious problem because users never knew if the current visualization was what they looked for. Many methods were tried including refreshing the page but none worked. It was believed that there was caching going on behind the screen. What was specified at the source was not always re-fetched each time the graph itself was changed.

The solution to this caching problem was to use *PL/SQL Dynamic Content Region* to contain the graph. This is a special region provided in Apex which automatically refreshes itself whenever its source is changed. The PL/SQL procedures to generate the graph and load the image were specified in the source. This way the graph displayed in the page was always the latest version.

The solution brought in striking functionalities to the application surprisingly. One was “live” graph interactivity. Originally by clicking on a node in the graph to indicate expanding or collapsing its children, the graph would not instantly change itself. Users had to go through an intermediate step to redraw the graph in order to see the updated version. The approach of *Dynamic Content Region* together with the hyperlink approach embedded in the graph, however, made the graph live. There was no extra step to refresh the graph. The graph vividly expanded or collapsed within seconds of user clicking.

7. Results

At the end of the project, a well-integrated visualization and data management application was built. This section is a full presentation of the final product.

Following are the most important application pages. These pages were implemented closely by the detailed design plan.

- 1) Login page. Users are prompted to this page when the application is launched.



Figure 24: Login Page Implemented.

- 2) Main page. The main body is the search region where users can specify user preferences.

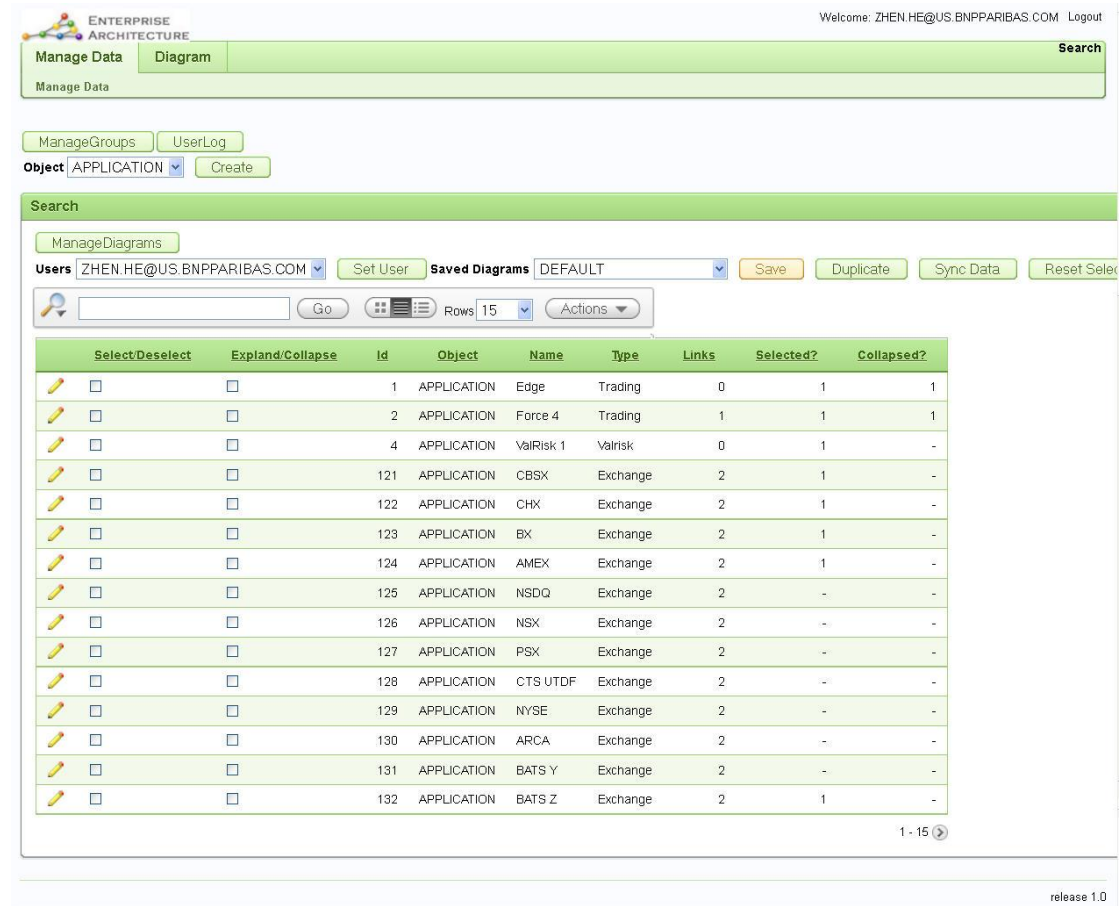


Figure 25: Main Page Implemented.

- 3) Visualization. There is a separate tab called *Diagram* that is used to display the graph.

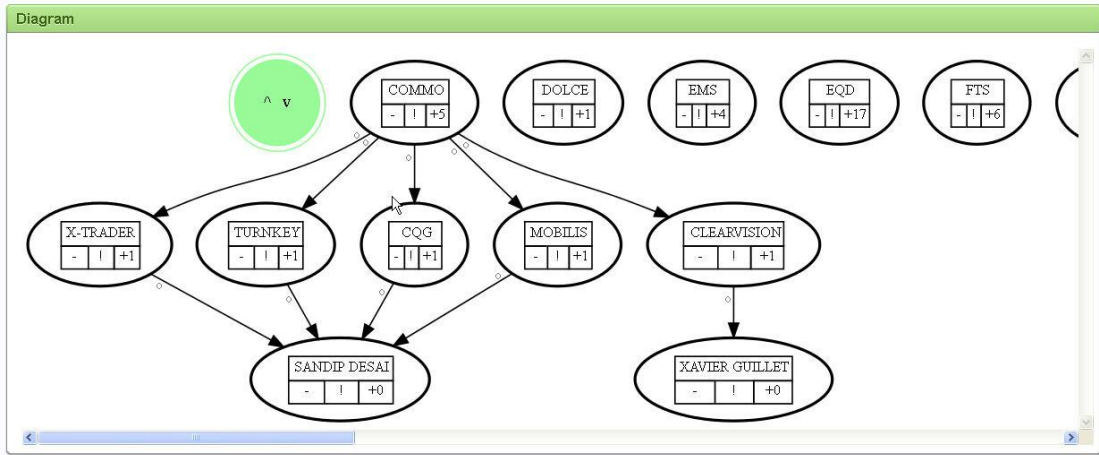


Figure 26: Visualization Page Implemented.

4) Performance Metrics page. It includes both diagrams and charts.

Id	Refid	Name	Type	Comments	Unit	Critical	Warning	
Select	5	2	My app latency	Latency	-	ms	20	7
Select	6	2	mem hog	Memory Usage	-	%	80	50

1 - 2

ID	REFID	TS	VALUE
25	5	21-NOV-11 05.44.26.978000 PM	11
26	6	21-NOV-11 05.44.27.072000 PM	21
27	5	21-NOV-11 05.44.27.165000 PM	2

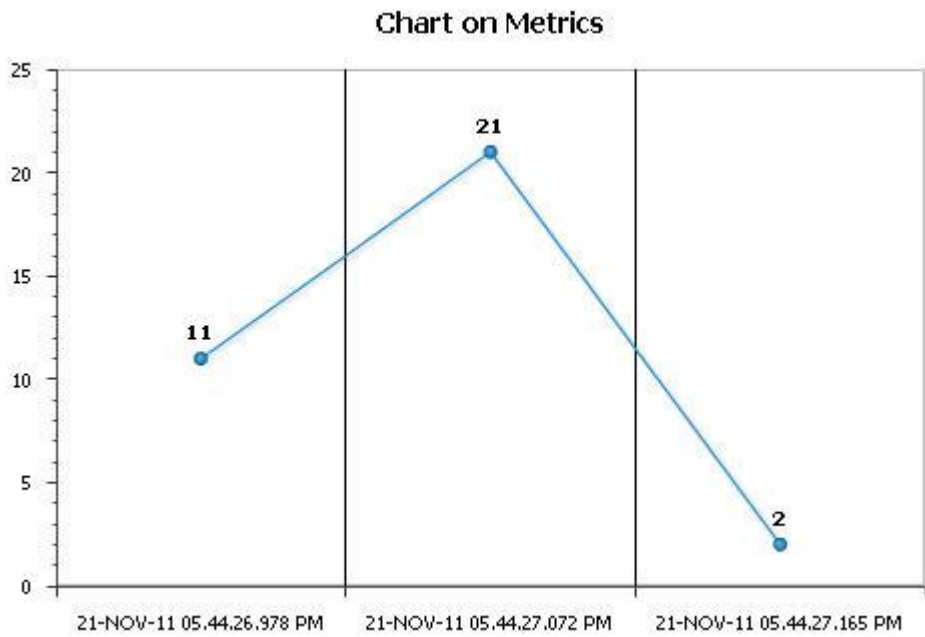


Figure 27: Capacity Page Implemented.

7.1. Major functions and capabilities implemented

The below is a description of major functions implemented on the pages just displayed.

Database Visualization based on User Selection



	Select/Deselect	Expand/Collapse	Id	Object	Name
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1	APPLICATION	Edge
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	2	APPLICATION	Force 4
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	4	APPLICATION	ValRisk 1
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	121	APPLICATION	CBSX
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	122	APPLICATION	CHX
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	123	APPLICATION	BX
	<input type="checkbox"/>	<input type="checkbox"/>	124	APPLICATION	AMEX

Figure 28: User Selection Implemented.

Within the context of a saved diagram, users can select/deselect through checkboxes the entities they want to include in their visualization. They can further indicate if the entity is expanded downward to include child entities or collapsed as a single node. The visualization of the selected entities is then displayed.

Graph Interactivity

The entities on the graph are clickable in four sections, the name section, the “+” and “-“, and the metrics section. By clicking on the name section, users are able to edit the entity.

By clicking on the “+” and “-“sections of a node in the graph, users are able to expand and collapse the node in the next generation of the graph. By clicking on the metrics (!) section, users are able to view how an *Application* or *Hardware* entity performed over time against the threshold of an attribute (i.e. latency, memory etc.)

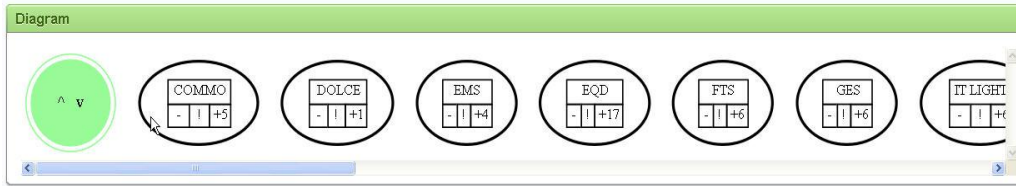


Figure 29: Diagram collapsed.

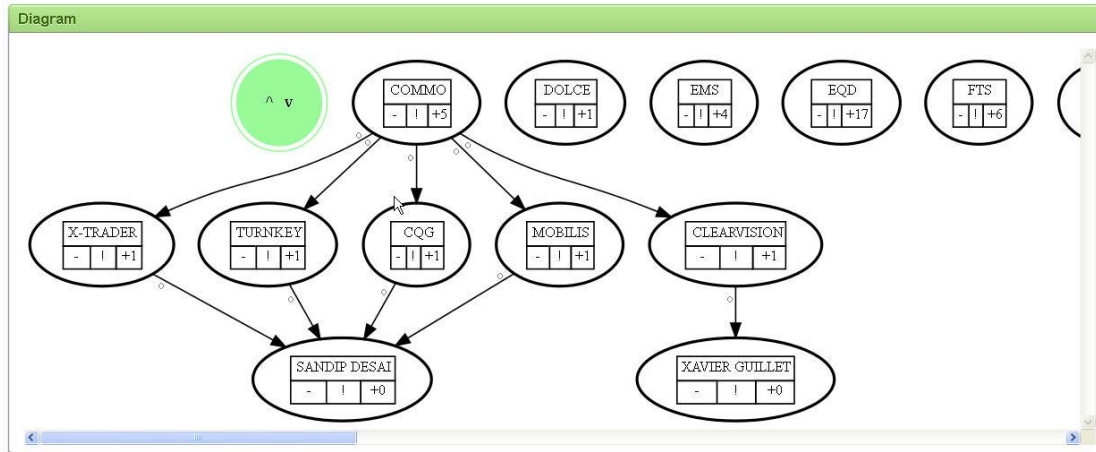


Figure 30: Diagram expanded.

Capacity Metrics

Users can view the capacity metrics of an *Application* or *Hardware* entity either from the editing page of the entity or from the metrics section of the node in the graph. The metrics are a time-based series of performance statistics measured against certain attributes (i.e. latency, memory etc). They are presented both in a table and in a line chart. In the table, the records that pass the thresholds (critical or warning) are color-coded (red or orange) to highlight the problem.

Summary View vs. Detailed View

Users can choose to see a summary view or a detailed view of their data. A summary view collapses all the physical instances of an entity into one single node. It hides unnecessary details from users in the graph. In contrast, a detailed view rolls down to show every instance of an entity. It is for a thorough examination of the entity if the summary view signals a problem.

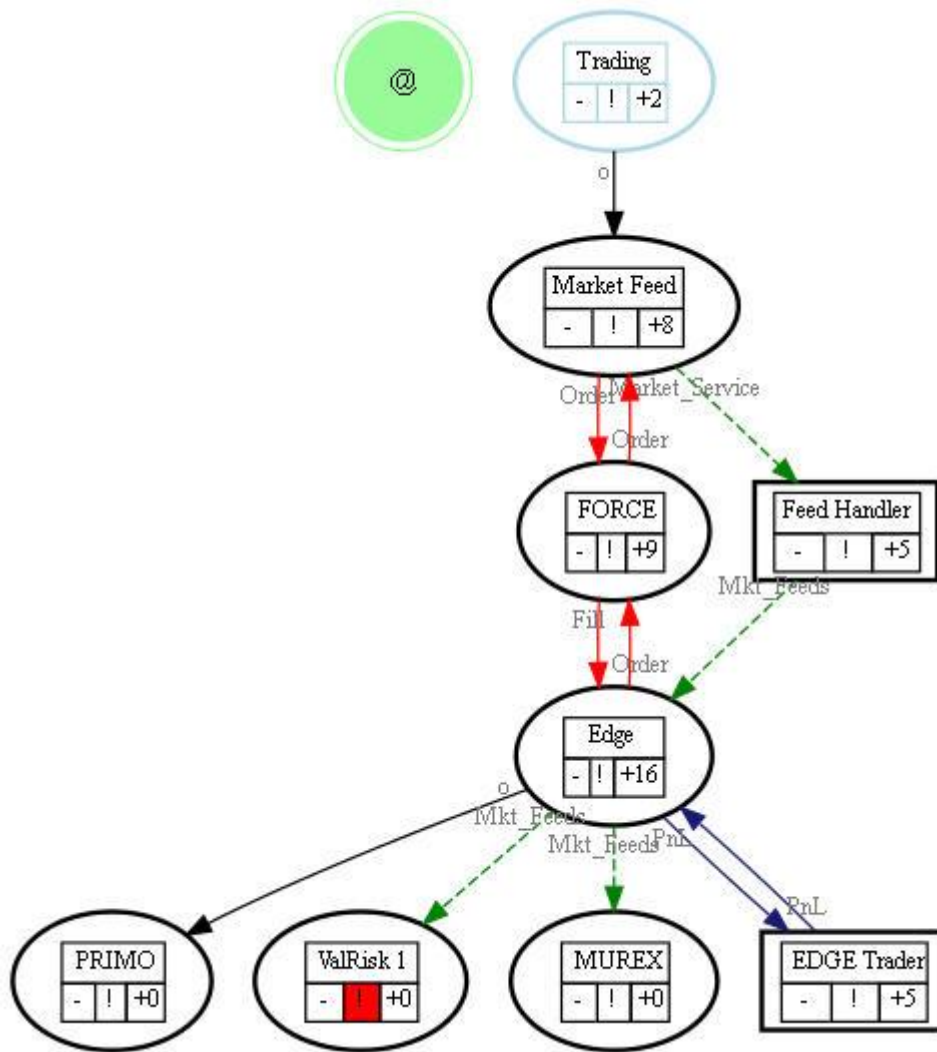


Figure 31: Diagram Rolled up.

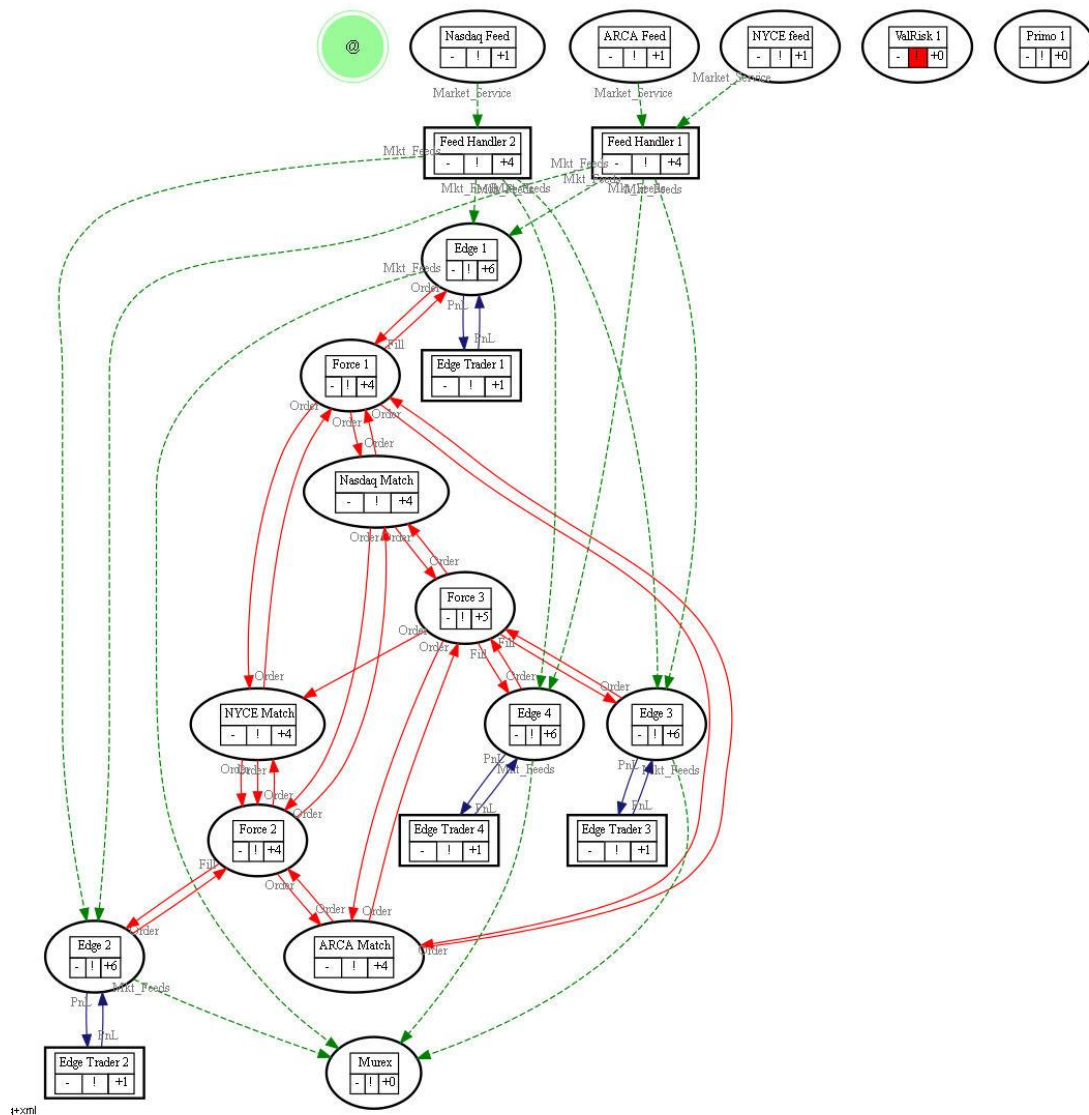


Figure 32: Diagram Rolled down.

Processes Sub-graph

Graphviz lends the user the ability to superimpose smaller graphs on top of the general one using the sub-graph functions. In our project, processes are shown in sub-graphs, and are shown with a black rectangular box. Entities outside the process can still connect to links within the sub-graph box. Sub-graphs are used to give the user information about both the operational hierarchy and process diagrams while maintaining graphical clarity.

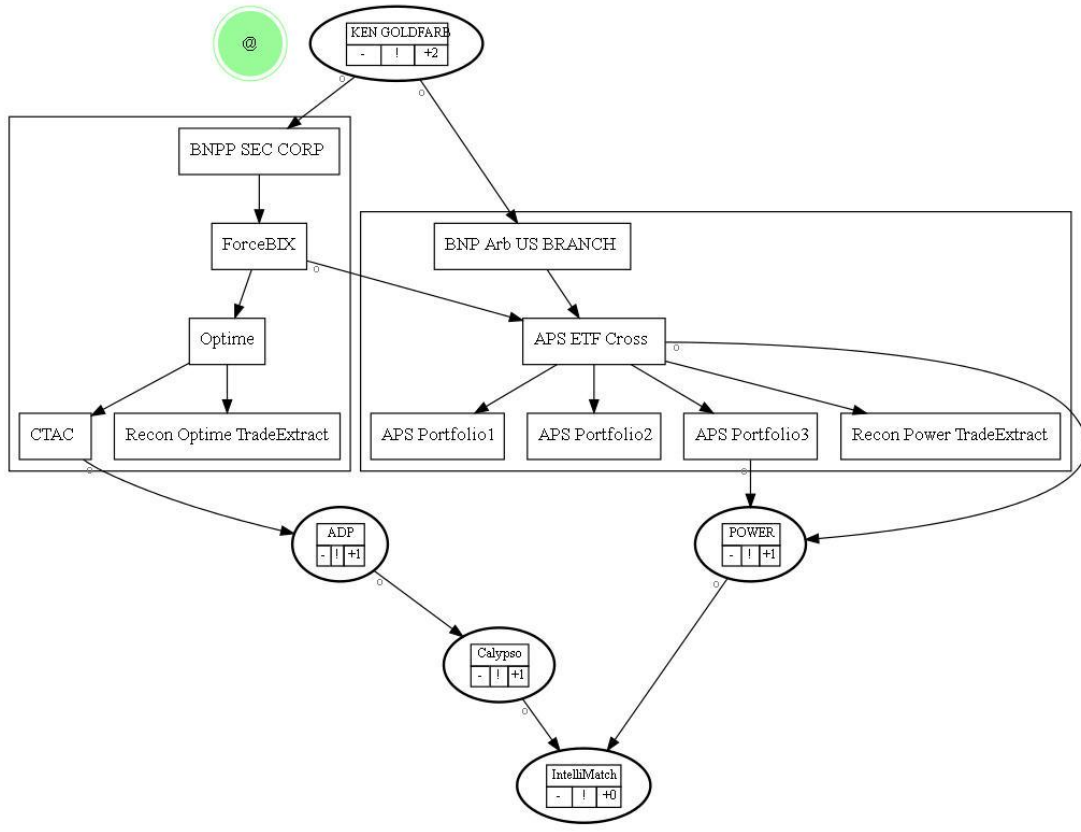


Figure 33: Diagram with Composite Processes.

Database management

Users are able to view the entire database in one master table, create/edit entities, and create/edit relationships between entities through a data management interface. They can also choose to synchronize their copy of data with master data if needed.

7.2. Summary of functions

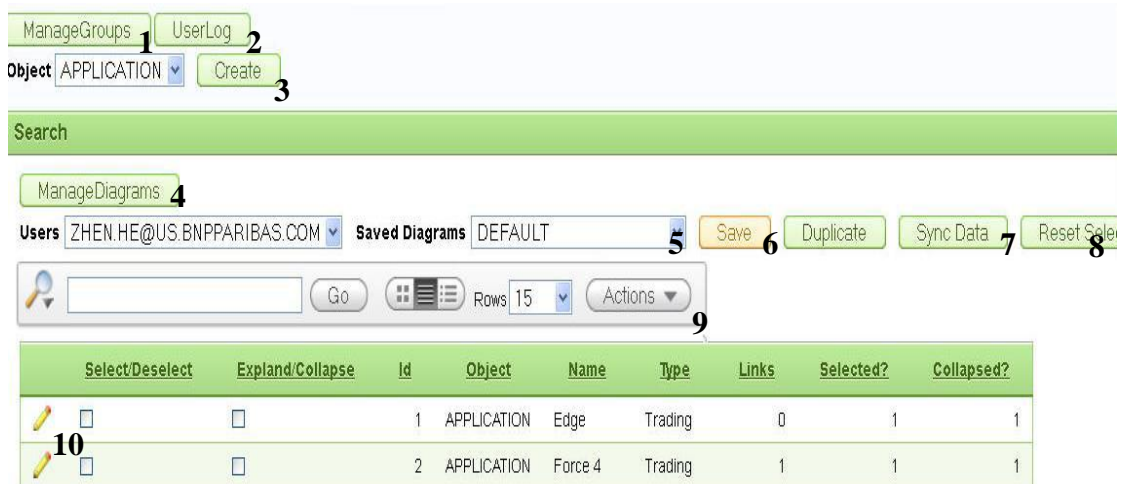


Figure 34: Summary of functions in ApEx interface.

1. Manage the graphical attributes of an entity or a relationship in the graph.
Users can assign colors, shapes, and styles to a node or link.
2. Keep a log of user actions made to the data. The actions include insert, delete, and update and can be sorted by date.
3. Create an entity of type Application, Hardware, OU, People, or Process.
4. Create a new diagram or edit an existing one. A diagram is a copy of the master database with a specific set of user preferences. It is equivalent to a graph in its database format.
5. Go back to any saved diagram under the current selected user.
6. Save any user preferences modification made to the current diagram.
7. Synchronize the current user copy with the master database. The mismatch between the two comes from create/edit/delete an entity or a relationship. The synchronization has nothing to do with user preferences.
8. Reset user preference.
9. Search for any particular record(s).
10. Edit an existing entity. Relationships that involve that entity can be edited from the editing page for that entity.

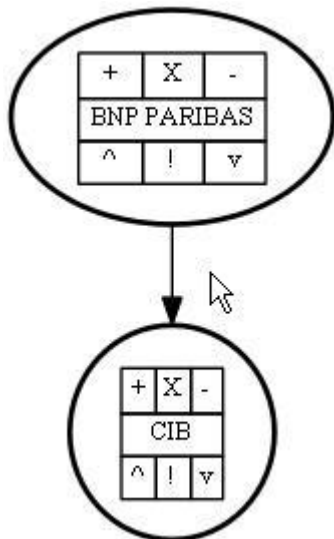


Figure 35: Summary of functions on Super Node*

+: Expand a node so that children of that node are shown in the graph.

-: Collapse a node so that children of that node are hid.

X: Hide the current node.

[Node Name]: Prompt the user to the editing page for the node.

^: Roll up the graph so it displays a summary view.

v: Roll down the graph so it displays a detailed view.

!: Prompt the user to the performance metrics page for the node. This section is color-coded according to the health of the entity.

*Note that by the end of the project, Super Node as shown above was not fully supported in the application.

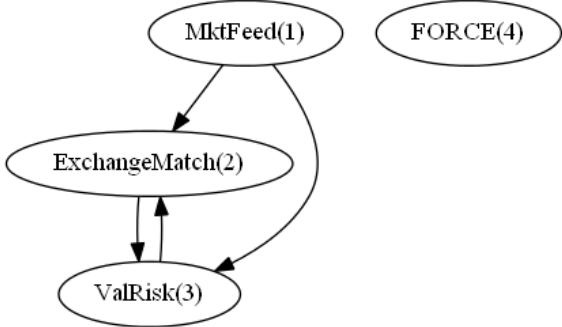
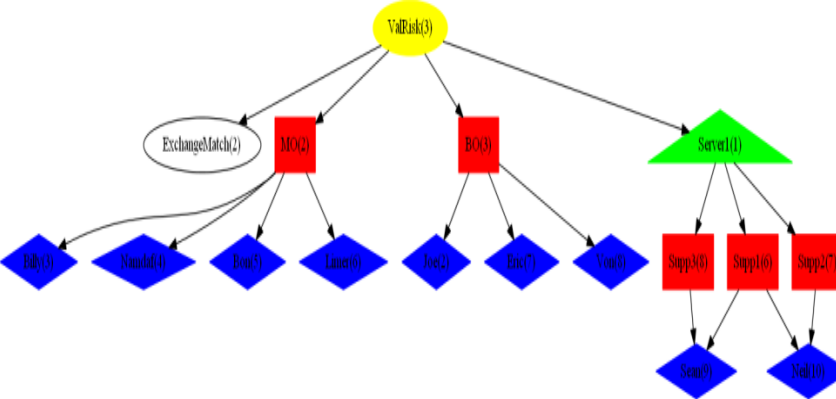
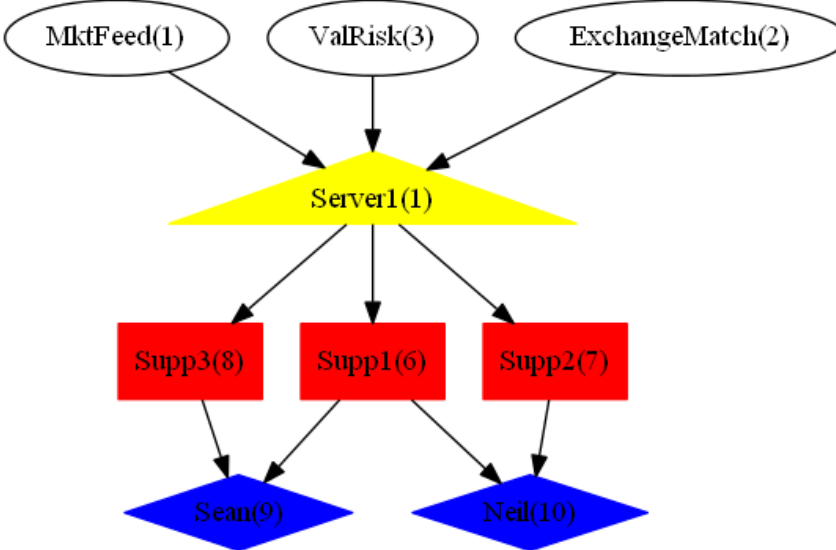
8. Conclusion

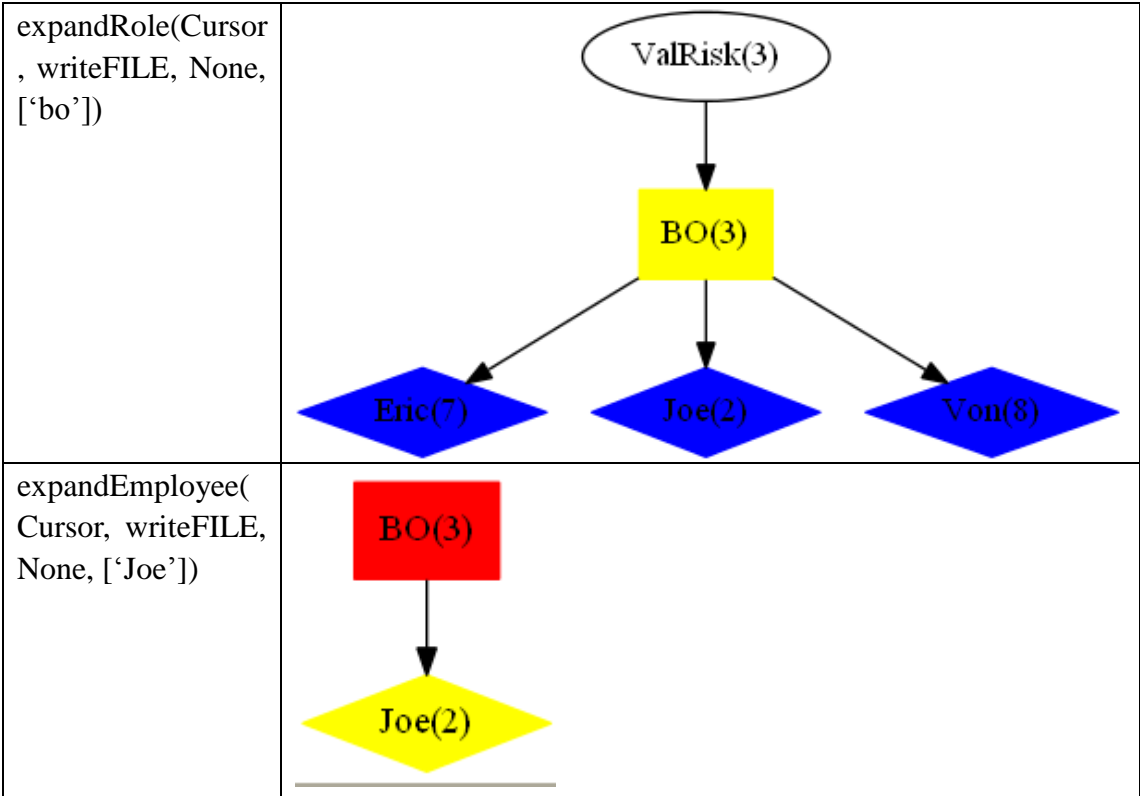
BNP Paribas, an international bank with headquarters in Paris, France, sponsored the Enterprise Architecture Project. The goal of the project was to create a graphical management application that allows the Global Equities and Commodity Derivatives group to better visualize a wide array of system flows. It was required for the application to have bi-directional communication with internal BNP databases and clearly show any possible problems with the data, such as an overloaded server.

The initial stages of the project involved finding the right set applications to complete these objectives. A variety of options were explored at this stage. Eventually it was determined that the combination of Graphviz and ApEx was the tool to use and all remaining resources were allocated to this strategy.

Throughout the project term, the application was transformed from a concept to a working prototype that will serve as a basis for further development. During the formal development portion of the project, many requests for additions to the project from the stakeholders of the project were translated into code and presented to the group. At the conclusion of the project, a team was put in place at BNP to continue development of the tool.

Appendix A

Input (Python code)	Output (Graphviz graph)
<pre>default(Cursor, writeFILE)</pre>	 <pre> graph TD MktFeed(1) --> ExchangeMatch(2) ExchangeMatch(2) <--> ValRisk(3) ValRisk(3) --> FORCE(4) </pre>
<pre>expandApplication (Cursor,writeFILE, None, ['VALRISK'])</pre>	 <pre> graph TD ValRisk(3) --> ExchangeMatch(2) ValRisk(3) --> MO(2) ValRisk(3) --> BO(3) ValRisk(3) --> Server1(1) MO(2) --> baby(4) MO(2) --> Nando(4) MO(2) --> Jon(5) MO(2) --> Lamer(6) BO(3) --> Jose(2) BO(3) --> Eric(1) BO(3) --> Vane(3) Server1(1) --> Supp3(3) Server1(1) --> Supp1(6) Server1(1) --> Supp2(7) Supp3(3) --> Sean(9) Supp3(3) --> Neil(10) Supp1(6) --> Sean(9) Supp1(6) --> Neil(10) Supp2(7) --> Neil(10) </pre>
<pre>expandServer (Cursor, writeFILE, None, ['SerVER1'])</pre>	 <pre> graph TD MktFeed(1) --> Server1(1) ValRisk(3) --> Server1(1) ExchangeMatch(2) --> Server1(1) Server1(1) --> Supp3(8) Server1(1) --> Supp1(6) Server1(1) --> Supp2(7) Supp3(8) --> Sean(9) Supp3(8) --> Neil(10) Supp1(6) --> Sean(9) Supp1(6) --> Neil(10) Supp2(7) --> Neil(10) </pre>



Appendix B

Interested Parties for the Data (BNP Paribas. “Enterprise Architecture Overview.”)

	United States	Europe
Organizational charting	<p>PeopleSoft : Ted Stephens/Michael Bastock HR Would like to have a better graphical source to maintain. This has to be up-to-date for end-of-year</p> <p>BEN: Manoj Tejwani/Tim Mahler This system has simple org chart tools.</p>	<p>OREF: Brice Degromard/Tarik Slassi/Reda Hadjouti</p> <p>Organizational charting system</p>
<p>System Graphing</p> <p>Hardware Mapping</p>	<p>John Crocker Core Integration Service Mngr BAMM Plus: IT manually inputs systems data.</p> <p>John Crocker Core Integration Service Mngr CA Asset Manager : This takes an inventory at 1pm of all servers, how many cpu's, disk space, primary applications run CA Network System Monitoring – This measures cpu utilization, provides detailed performance reports</p>	<p>ServiceNow : Martin Saldarriaga/Reda Hadjouti Tool for managing the production activities (incidents, problems, changes, etc.) aligned with ITIL. ServiceNow contains a module to store the configuration items and their relationships. A configuration item can be an application, a server, a network card, etc</p>
<p>System Entitlements</p> <p>Analytics</p>	<p>MyIT Stuart Lincoln</p> <p>Gerald Sebastien – Planning for Infrastructure Dashboard Scott Visconti/ Christophe Poulmarc'k -Planning for system dashboard</p>	<p>STAF: Brice Degromard/Tarik Slassi/Reda Hadjouti Staff Assignment Follow-Up Patrice Piron – GECD Functional Architect</p>

Appendix C

ComputerSystem

SystemID	System	Description	Process	Sys	GroupTo	GroupLabel
1	NYCE feed	NY Futures Exchange Feed	null	9	1	Mkt Feed
2	ARCA Feed	Arca NYSE Exchange Feed	null	9	1	
3	Nasdaq Feed	Nasdaq Exchange Feed	null	9	1	
4	NYCE Match	NY Futures Exchange Market	null	9	4	Exchange Match
5	ARCA Match	Arca NYSE Exchange Market	null	9	4	
6	Nasdaq Match	Nasdaq Exchange Market	null	9	4	
7	Feed Handler 1	NYSE & ARCA Feed Handler	null	10	7	Feed Handler
8	Feed Handler 2	NASDAQ Feed Handler	null	10	7	
9	Edge 1	Dir Trdng 1		6	1	Edge
10	Edge 2	Dir Trdng 2	null	1	9	
11	Edge 3	Dir Trdng 3	null	1	9	
12	Edge 4	Dir Trdng 4	null	1	9	
13	Edge Trader 1	Dir Wrkstn 1	null	1	14	Tdr Wrkstn
14	Edge Trader 2	Dir Wrkstn 2	null	1	14	
15	Edge Trader 3	Dir Wrkstn 3	null	1	14	
16	Edge Trader 4	Dir Wrkstn 4	null	1	14	
17	Force 1	Force Dir 1	null	1	17	Force
18	Force 2	Force Dir 2	null	1	17	
19	Force 3	Force Dir 3 & 4	null	1	17	
20	Murex	Back Office	null	3		
21	ValRisk	Risk	null	5		
22	Primo	P&L	null	6		

Infrastructure

InfraID	Name	Type
1	AppServ1	App Server
2	AppServ2	App Server
3	AppServ3	App Server
4	DB Serv1	DB Server
5	DB Serv2	DB Server

ProcessID

Process ID	ProcessName
1	Equity Swaps
2	Forward Trading
3	Security Trading
4	Stock Borrow & Loan
5	OTC OPTIONS
6	Prop AQS

Relation Classification

Relation ID	Relation Name	Line Link
1	Trade	
2	Position	
3	Referential	
4	Market	
5	Risk	
6	P&L	
7	Order	

Map Infrastructure to System

App	Infra
17	1
17	4
18	2
18	4
19	3
19	5

(BNP Paribas. “Enterprise Architecture First Phase – a tool for capacity management.”)

References

- Angel, James J., Lawrence E. Harris, and Chester S. Spatt. "Equity Trading in the 21st Century." 23 Feb. 2010. Print. 25 Sept. 2011.
- "Apex – Change Application-Level Default Templates « My Favorite Software Tools." *My Favorite Software Tools*. 31 Jan. 2008. Web. 29 Nov. 2011. <<http://stewstools.wordpress.com/2008/01/31/apex-change-application-level-default-templates/>>.
- "APEX Check Boxes in Reports Regions." *Oracle Consulting, Oracle Support and Oracle Training ByBC Oracle Consulting*. Web. 03 Nov. 2011. <http://www.dba-oracle.com/html/db/t_html/db_check_boxes_reports_regions.htm>.
- "APEX CSS Repository: How to Include Background Images Easily Using Data URIs." *Random Insights into Oracle*. 11 Sept. 2011. Web. 1 Dec. 2011. <<http://oracleinsights.blogspot.com/2011/09/apex-css-repository-how-to-include.html>>.
- BNP Paribas. "Enterprise Architecture Overview." Print. 24 Jan. 2011. Nov.2011.
- BNP Paribas. "Enterprise Architecture First Phase – a tool for capacity management." Print. Nov. 2011.
- "Cross-Platform Dynamic Graphics and GIS Solutions for Real-Time Data Display." *GLG Toolkit: High Performance Dynamic Graphics for C/C ,AJAX, Java and .NET*. Web. 03 Nov. 2011. <<http://www.genlogic.com/products.html>>.
- Gansner, Emden R., Eleftherios Koutsofios, and Stephen North. "Drawing Graphs with Dot." Web. 7 Nov. 2011. <<http://www.graphviz.org/pdf/dotguide.pdf>>.
- "GLG Toolkit: Graph Layout Java Demo." *GLG Toolkit: High Performance Dynamic Graphics for C/C ,AJAX, Java and .NET*. Web. 02 Nov. 2011. <http://www.genlogic.com/java2/graph_layout.html>.
- "Graphviz - QED." *Main Page - QED*. Web. 09 Nov. 2011. <<http://qed.princeton.edu/main/Graphviz>>.
- "Linking – SVG 1.1 (Second Edition)." *World Wide Web Consortium (W3C)*. Web. 07 Dec. 2011. <<http://www.w3.org/TR/SVG/linking.html>>.
- "Manipulating Database Objects Using Oracle Application Express 4.1." *Oracle / Hardware and Software, Engineered to Work Together*. Web. 01 Nov. 2011. <http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r41/apexstart_a/apexstart_a.htm>.

- "Nevron Diagram for .NET." *Nevron Web Examples*. Web. 02 Nov. 2011.
<<http://examplesaspnetdiagram.nevron.com/>>.
- Nevron .NET Vision - Online Documentation*. Web. 02 Nov. 2011.
<<http://helpdotnetvision.nevron.com/>>.
- "Node, Edge and Graph Attributes." *Home / Graphviz - Graph Visualization Software*. Web. 09 Nov. 2011. <<http://www.graphviz.org/doc/info/attrs.html>>.
- "Node Shapes." *Home / Graphviz - Graph Visualization Software*. Web. 15 Nov. 2011.
<<http://www.graphviz.org/doc/info/shapes.html>>.
- "Oracle Application Express 4.1." Web. 31 Oct. 2011.
<<http://apex.oracle.com/i/index.html>>.
- "Oracle Application Express APEX: Calling Reusable Pages in Separate Applications." *Oracle Consulting, Oracle Support and Oracle Training ByBC Oracle Consulting*. Web. 14 Nov. 2011.
<http://www.dba-oracle.com/html/db/t_calling_pages_between_applications.htm>.
- "Oracle Application Express APEX: Passing Values between Screens." *Oracle Consulting, Oracle Support and Oracle Training ByBC Oracle Consulting*. Web. 09 Nov. 2011.
<http://www.dba-oracle.com/html/db/t_passing_values_between_screens.htm>.
- "Oracle Application Express." *Oracle / Hardware and Software, Engineered to Work Together*. Web. 31 Oct. 2011.
<<http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html>>.
- "OTN Discussion Forums : Change Colour of Row - Oracle APEX SQL Report." *OTN Discussion Forums*. Web. 08 Dec. 2011.
<<https://forums.oracle.com/forums/thread.jspa?threadID=883110>>.
- "Python: Lambda Functions." Web. 28 Oct. 2011.
<http://www.secnetix.de/olli/Python/lambda_functions.hawk>.
- "Scripting – SVG 1.1 (Second Edition)." *World Wide Web Consortium (W3C)*. Web. 07 Dec. 2011. <<http://www.w3.org/TR/SVG/script.html>>.
- "SQL INNER JOIN Keyword." *W3Schools Online Web Tutorials*. Web. 26 Oct. 2011.
<http://www.w3schools.com/sql/sql_join_inner.asp>.
- Stubbs, Paul. "Office Add-Ins: Develop Add-Ins For PowerPoint And Visio Using VSTO." *MSDN / Explore Windows, Web, Cloud, and Windows Phone Software*

- Development*. Feb. 2007. Web. 03 Oct. 2011.
<<http://msdn.microsoft.com/en-us/magazine/cc163471.aspx>>.
- "Turn Report Rows into a Multi Column Format through Templates!" *APEX Development*. Web. 08 Dec. 2011.
<<http://application-express-blog.e-dba.com/?p=55>>.
- "URL Parameters." *Oracle Application Express*. 20 Nov. 2009. Web. 09 Nov. 2011.
<<http://www.oracleapplicationexpress.com/tutorials/55>>.
- "Using GraphViz, a Brief Tutorial | Orient Lodge." *Orient Lodge | An Eclectic News Site*. Web. 26 Oct. 2011. <<http://www.orient-lodge.com/node/3408>>.
- Vardi, Moshe Y., Ian Barland, and Ben McMahan. "Logic and Database Queries." Rice University, 31 Aug. 2006. Web. 4 Oct. 2011.
<<http://www.cs.rice.edu/~tlogic/Database/all-lectures.pdf>>.
- "Visio 2010 Add-Ins Overview." *Microsoft Vision 2010*. Microsoft Corporation, 2009. Web. 3 Oct. 2011.
<<http://visiotoolbox.com/2010/PDFs/visio-2010-add-ins-overview.pdf>>.
- Wang, Xiaoyun, and Tu Lai Huan. "BNP Paribas: Equity Smart Order Router." *Electronic Projects Collection*. Worcester Polytechnic Institute, 29 Nov. 2010. Web. 4 Oct. 2011.
<http://www.wpi.edu/Pubs/E-project/Available/E-project-112910-121011/unrestricted/Equities_Smart_Order_Router.pdf>.
- Wolf, Patrick. "Opening URL in a New Window." *Inside Oracle APEX by Patrick Wolf*. Web. 28 Nov. 2011.
<<http://www.inside-oracle-apex.com/opening-url-in-new-window/>>.
- Wolf, Patrick. "Oracle APEX 4.0: Cascading LOVs/Select Lists." *Inside Oracle APEX by Patrick Wolf*. Web. 18 Nov. 2011.
<<http://www.inside-oracle-apex.com/oracle-apex-4-0-cascading-lovsselect-lists/>>.