Project Number: MQP MXC-W076

Process Controls and Improvements For Deutsche Bank's Job Execution Framework

A Major Qualifying Report: submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the

Degree of Bachelor of Science

By:

Christopher Hammers
chammers@wpi.edu

Danielle Kane
dmkane@wpi.edu

Brendan McLaughlin
blaughin@wpi.edu

In cooperation with:
Deutsche Bank

December 15, 2006

Professor Michael Ciaraldi, CS Advisor

Professor Arthur Gerstenfeld, IE Advisor

Professor Jon Abraham, MA Advisor

Abstract

This report, prepared for Deutsche Bank's Global Exchange Services (GES) division, is focused on assessing the current job scheduling systems used to execute and deliver reports throughout the bank's futures and options business. Through a series of interviews, research, and analysis, we developed recommendations to improve the future application, the Job Execution Framework (JEF), and established controls to help regulate the job submission process.

Acknowledgements

Our group would first like to thank Gregory Friel for making this project possible at Deutsche Bank. Without his continued support and personal dedication, we would not have had the opportunity to work at one of the world's top investment banks. In addition, we appreciate the use of the bank's office and services throughout our time here in New York.

We would like to thank our colleagues in London, particularly Ian Ramsay and John Hawkins. Their expertise and excitement toward this project has helped us present effective and strong arguments. In addition, they provided us with access and guidance to valuable technical resources and personal contacts. We appreciate the time they set aside for this project including many hours for conference calls to help us along the way.

We would also like to thank Antonella Allaria for working closely with us throughout this project. Her help coordinating meetings between London and New York made adjusting to the global atmosphere here at Deutsche Bank much easier. Also, the input and background perspective she provided was invaluable to the success of our project.

Special thanks also to Andrea Rudnick, Vilas Hirani, Pete Lindsay, Jens Balkmann, Dennis Klocke, and Iliana Dimitrova for their contributions to our project. Their input through interviews and discussion provided us with important findings to help build the most encompassing and useful recommendations.

Finally, we would like to thank our advisors at WPI, Professors Art Gerstenfeld, Mike Ciaraldi, and Jon Abraham. Their constant support, revisions, and general input to this project helped us to present the best final product. Also, their efforts to coordinate a project center of this magnitude with the world's top companies have not gone unnoticed, and have provided us with an excellent learning experience.

Executive Summary

Within Deutsche Bank's Global Exchange Services (GES) division, thousands of trades are conducted on a daily basis. Responsible for trading and clearing derivatives within the global Futures and Options markets, GES depends on a set of job scheduling systems to facilitate the flow of data. Currently, several different scheduling systems are used to meet the company's operations throughout the 73 nations that Deutsche Bank is located. The largest system, TaskMan, is primarily used in the United States and England and to date contains over 2,500 individual jobs, far more than originally conceived. Additionally, two other systems – RANtask and the web-based Asia-Pacific Perl Scripts – are implemented in offices including Frankfort, Germany and Sydney, Australia respectively. However, in an effort to keep up with the ever-growing demand for these systems worldwide and provide Deutsche Bank with one all-encompassing job scheduling system, GES IT developers are producing a replacement system known as the Job Execution Framework (JEF). JEF is being developed as a multi-threaded system with a main goal to allow for future expandability, and is planned to officially replace the current systems by late 2007, early 2008. Prior to its deployment however, developers must identify and address all the current problems, and adjust JEF accordingly. This project focused on identifying and analyzing the current issues in the job scheduling systems at the bank, notably the large TaskMan system, to present recommendations to improve and safeguard JEF against similar problems.

In order to accomplish this goal and provide Deutsche Bank with the most effective recommendations, this project achieved the following four main objectives:

- Identified Job Submission process and problem areas
- Established controls and new submission process to apply to JEF
- Determined job migration steps and controls to identify active jobs to transfer to JEF
- Recommended features to implement in JEF

To accomplish these tasks, we first conducted research on the current job scheduling systems (primarily TaskMan) through a series of interviews with employees in GES IT, Operations, and Management. By utilizing the different perspectives and expertise of each person within these divisions, it was possible to piece together the complete job

submission process. With a step-by-step process documented, we easily identified problem areas and potential improvements that could be made to the Job Execution Framework. In particular, we constructed a comprehensive list of limitations and issues that plague the current applications used, including TaskMan, the Global Incident Management System (GIMS), RANbase, and RANtask. These issues – the basis of the majority of our recommendations – were prioritized in order to allow GES IT to easily address the problems with the greatest impact. Examples of TaskMan problems include:

1. A lack of controls
2. No direct link between GIMS ticket and TaskMan
3. Users cannot view or search what jobs are already running
4. No standardized testing system established prior to job submission
5. No method for users to track the status of a job to see if it is working properly

Many of these issues are attributed to the fact that TaskMan was originally constructed to simply run eight to ten Structured Query Language Scripts (referred to as jobs/queries). However, in order to run the 2,500 jobs now in the system, TaskMan was modified in an ad-hoc manner with minimal regard to future issues and expandability. Therefore, GES IT must understand the benefits, limitations and functionality of the current job scheduling systems to improve the features of the Job Execution Framework.

As mentioned, a lack of controls is the most evident problem in the job submission process. There is no governance system to regulate who is qualified to submit a job into the system, which allows poorly written, unnecessary, and often duplicate jobs to enter. In addition, few controls are established to provide quality testing for each new job. The role of Operational Excellence (OE), the group within GES IT that currently inputs jobs into TaskMan, has been reduced to nearly a middleman. Rather than formally testing the performance of jobs, jobs are entered on an inconsistent case-by-case review process.

After analyzing the results gathered, we were able to form many recommendations to prevent problems from occurring as Deutsche Bank transitions to the JEF system. We focused on providing recommendations on the following areas:

1. Job Submission Requirements
2. Role of Operations and Information Technology
3. Business Objects as an Alternative
4. Access Controls
5. Job Migration to the Job Execution Framework (JEF)

6. Standardized Reporting Application
7. Wizard
8. Searchable Job Database
9. Job Execution Frameworks Improvements

These recommendations are prioritized to provide Deutsche Bank with an understanding of their potential improvements.

In order to efficiently submit and track each job that enters JEF, we recommended that each new job submission or modification contains basic standardized information. Also, all data gathered should be stored in the expandable XML database used by JEF to ensure that proper contact information and job descriptions are readily accessible in the event of future problems or maintenance. During this process, we recommended that users do not assume the responsibility of including their own SQL queries with job submission. Rather, in order to clearly designate the roles of Operations and GES IT, users should build "business requirements" for IT to interpret into the most effective SQL queries. This restructuring, along with a searchable database of jobs, will minimize the number of repeated and poorly running jobs that enter JEF. In addition, Deutsche Bank should also provide different levels of access rights for the new job scheduling system. We recommended that access levels are established including:

- Update and write access to input/change configurations (reserved for IT)
- Read-only access to view status of job (available to IT and qualified users)

In TaskMan, virtually anyone who wishes to submit a new job request can fill out a Global Incident Management ticket – regardless of their position, experience, or need for the job.

Next, to provide an appropriate method to migrate only the active, needed jobs from TaskMan and other systems to JEF, we recommended that GES IT study each job on a case-by-case basis. Only once adequate contact information and a job description are available, should a job be transferred to JEF. This report provides guidelines for the specific procedures that GES IT should follow during this migration. Also, to ensure that all of the required information is provided with each new request in JEF, we suggested developing a standardized reporting tool, a proposed wizard, and reverting GIMS back to its original purpose as a support mechanism. Currently, GIMS allows for too much individual customization during the job submission process to effectively capture the

recommended data. The wizard is also designed to help create a separate searchable library of jobs within JEF. All of the information gathered during submission, including a detailed job title and purpose, should be stored in a database to allow a user to search for existing jobs based on specified parameters. Finally, to improve the capabilities of the support team, we recommended creating a more robust logging system and more effective alert system in the JEF system. These recommendations were designed to smoothly transition to the Job Execution Framework and will help to ensure the continued success of the new job scheduling system into the future.

Table of Contents

Table of Figures

1.0 Introduction

As a leading global investment bank, Deutsche Bank conducts millions of trades each day which are spread over many different markets worldwide. Founded in Germany, the bank participates actively in markets throughout Europe, North America, and the growing investments centers in Asia, the Middle East, Latin America, Australia, and Africa. To maintain favorable annual growth and success as a global investor, Deutsche Bank places a strong emphasis on developing and maintaining superior technology to assist daily business transactions. The bank's Global Exchange Services (GES) division, responsible for trading and clearing derivatives for clients within the Futures and Options markets, is highly dependent on technology to perform daily tasks. Specifically, for this project, GES depends on a set of job scheduling systems to organize and deliver reports of data to various internal personnel and external clients. The current job scheduling systems being used, namely TaskMan, no longer meet the needs of this expanding business. To satisfy the increased business demands, developers within GES IT created a replacement system called the Job Execution Framework (JEF).

TaskMan and the other job scheduling systems currently in use have many problems that must be addressed as the company prepares to shift to the JEF system by 2008. Due to a lack of process controls surrounding TaskMan, over 2,500 separate jobs have been entered into the system since its creation in 1998. Many duplicate, faulty, and unnecessary jobs are run daily as a result of the few restrictions on the job submission process. Also, problems are difficult to rectify by support personnel because of the general design and capabilities of TaskMan and its supporting applications. As the company prepares to shift to the JEF system already in production, a clear set of procedural controls must be established in order to prevent such problems from reoccurring in the future.

To facilitate the implementation and success of the JEF system, this project focused on identifying all of the problems with the current systems such as TaskMan and RANtask. A general lack of controls around who is permitted to submit jobs and the information required with each job request have made support difficult. As a result, we

focused on establishing methods to capture specific information required for each job by recommending the creation of a standardized reporting tool. In addition, the supporting applications used in the job submission process were examined and evaluated to develop recommendations to improve the entire process flow. The internal capabilities of the systems were analyzed through interviews with users, support teams, and GES IT managers to address shortcomings that should be incorporated as features in JEF. With an established process, we were able to provide recommendations to smoothly transition business critical jobs from the current job scheduling systems to the incoming Job Execution Framework.

2.0 Background

In global markets, the ability to transfer information to meet demanding schedules and deadlines is a good indicator of the success or failure of many investment companies. Due to human limitations, those in the financial industry have become dependent upon the advancement of technology. However, if companies such as Deutsche Bank allow their technology to become outdated and overwhelmed, information flow slows, investors miss opportunities, and business suffers. The purpose of this chapter is to provide a broad overview of Deutsche Bank's business and the concepts it incorporates. In particular, it focuses on Deutsche Bank's futures and options division, and provides an overview of the job scheduling systems being used to transfer raw data to various departments in the form of reports. A thorough technical analysis of the job scheduling systems and their related applications provides the reader with a basis to understand the project goals.

2.1 Deutsche Bank Organization and History

Since its establishment in 1870 in Berlin, Deutsche Bank has striven to provide a comprehensive range of financial services to individuals and businesses worldwide.

Figure 1: Global Presence ["DB at a Glance", 2006]

As a leading global investment bank, the company continues to attract a strong and profitable client franchise. As of June 30, 2006, the bank maintains branches in 73

nations around the world, employs 65,435 people, and holds over 1,058 billion Euros in assets. Operations have expanded beyond its European roots to focus on growing markets in North America and Asia, and for the first time in 2001, shares of Deutsche Bank's stock were traded on the New York Stock Exchange. Presently, however, the largest operations remain in Europe, with the largest investment banking hub located in the United Kingdom. Figure one on the previous page illustrates the main locations of Deutsche Bank throughout the world, designating the investment hubs with larger symbols.

Deutsche Bank is organized into six core businesses including:

- Global Markets
- Global Banking
- Global Transaction Banking
- Private and Business Clients
- Private Wealth Management
- Asset Management

This project focuses on exchange services for the futures and options business, which is primarily encompassed within the Global Markets business unit. The Global Market business trades heavily in the government and corporate bond markets, derivatives markets, and other emerging markets ["Our Company", 2006]. In order to run smoothly as a global competitor, Deutsche Bank organizes its core businesses into the following operational divisions:

- Corporate and Investment Bank (CIB)
- Private Clients and Asset Management (PCAM)
- Corporate Investments (CI)

The Corporate and Investment Bank division is responsible for developing, selling, and trading capital market products. Marketed toward both corporate and institutional clients, CIB is sub-divided into two corporate divisions, Corporate Banking & Securities and Global Transaction Banking. Since the beginning of 2005, Corporate Banking and Securities includes the Global Markets and Corporate Finance divisions and completes sales and trading, capital market origination, and corporate advisory and financing businesses. Global Transaction Banking deals with Deutsche Bank's cash management, trust and securities services, and trade finance businesses. Together with Corporate Finance, GTB incorporates Global Banking. The second division – Private Clients and

Asset Management – offers investment management business and traditional banking services to individuals and businesses. The third operating division, Corporate Investments, coordinates and manages Deutsche Bank's private equity, venture capital, industrial holdings, and real estate assets ["CI Operating Committee", 2006].

Functional committees are established within the bank's management structure with cross-divisional executive power. These committees serve control functions to provide "stronger implementation of resource allocation decisions and of risk management" ["Functional Committees", 2006]. As a functional committee, Global IT and Operations (in which this project is focused) provides strategic and operational support across the operating committees. The following diagram illustrates the management structure of Deutsche Bank's functional and operational committees.

Figure 2: Deutsche Bank Management Structure ["Organizational Structure", 2006]

2.2 The Global Business Environment

Assisted by the rapid development of technology, many businesses have expanded beyond the borders of a single country to provide services for the global market.

While international business offers many companies a stronger customer base and can decrease costs by outsourcing labor, it requires a new style of management. Meetings regularly involve scheduling conference calls across several time zones, company cultures often vary widely from nation to nation, and businesses must become more innovative to adapt to dynamic global economies. However, with an effective global management team, international businesses can gain a competitive advantage by operating at such a scale.

Deutsche Bank is a prime example of a company operating on an international scale. As a global investment bank, the success of the company is dependent upon its performance in many economies. This project, focused in the Global Exchange Services (GES) Information Technology division, applies to the concept of global business directly. The job scheduling systems analyzed in this report are used by branches all over the world, including Germany, England, China, and the United States [Personal Communication, GES Manager, Ian Ramsay, October 30, 2006]. A major part of this project included communicating with the different departments throughout the world to fully comprehend the implications of the job scheduling systems.

2.3 The Futures and Options Markets

Futures and options contracts are similar to purchasing an insurance policy for a house or car, "by paying small insurance premiums today, one can avoid the risk of paying large sums in the future" ["Market Overview", 2005]. For example, the tremendous uncertainty involved with the farming occupation resulted in a great need for protection against risk. When farmers plant their crop, two important factors are unknown, including "the price at which the output will be sold, and the size of the harvest" [Moschini and Lapan 1025-1049] In order to minimize the risk they face every year, farmers take advantage of futures and options contracts.

This project involves the futures and options market of Deutsche Bank; like the farmers who are trying to minimize the risk of falling crop prices or a bad harvest, investors are looking to minimize risk in their trades. In order to track the performance of these future and option investments, it is imperative to receive accurate data in a timely

manner. This project provides suggestions on how to supply this information in such a way.

2.3.1 Futures

Futures trading in America can be traced back to the 19th century, where the first exchange opened in Chicago. Trades were made on the three main agricultural products: wheat, soy beans and corn. As stated before, farmers were looking to hedge risk, and a futures contract provided them with the opportunity to do so. Years later, the Chicago Futures Market, the Chicago Board of Trade and the Chicago Mercantile Exchange are among the largest in the world; it is because of these exchanges that the world-wide futures industry has expanded into so many areas [Teweles and Jones, 4].

Buyers and sellers (traders) enter a futures contract to hedge risk against market position ["Futures Fundamentals Tutorial", 2006]. A futures contract is a legal agreement between a buyer and trader, the buyer agrees to receive a commodity and a trader agrees to deliver a commodity. The trader is usually involved with a futures commission merchant who carries out the dealing through an exchange. [Teweles and Jones 28-29]. The "price" of a contract refers to the agreed upon price of the delivered good (or financial instrument) at a specified future date ["Futures Fundamentals Tutorial", 2006]. The following image depicts the risk investors take when becoming involved in futures and options trading. There is potential of making a large return if one is able to survive the journey across the futures and options tightrope.

Figure 3: Risky Business ["The Basics of Futures & Options", 2006]

2.3.2 Options

The buyer of an options contract is given the right, but not the commitment, of buying or selling a particular asset at a strike price (a price the contract can be purchased for) on or before an established deadline. [Harwood et al. 36-39] Option contracts are binding securities that have strictly defined terms and properties. Two types of options are calls and puts. Calls give the owner the right to buy a commodity at a certain price within a given timeframe. The buyer of a call expects that their investment will increase before the contract is up. A put gives the owner the right to sell at a certain price within that timeframe and the holder of a put wants the stock price to fall before the contract expires.

Listed options are options traded on a national exchange, and have a fixed strike price and expiration date. Each listed option represents 100 shares of a contract ["Options Basics Tutorial", 2002]. Options represent a versatile investment tool, enabling investors to adjust or adapt their position in the market according to the situation they are in ["OBT", 2002] Participants are able to protect their position from a decline in their investment or to bet on a movement of the market. However, because options are so versatile, there is a large risk when investing.

2.4 Global Exchange Services

Global Exchange Services (GES) is a subdivision of Global Markets and is Deutsche Bank's global futures and options server. GES is responsible for carrying out and clearing exchange-traded derivatives for their clients. The electronic trading platform used by GES allows clients to trade on any exchange worldwide, and can be customized to fit the needs of the customer. ["Global Exchange Services", 2006]

Figure 4: Technology in Global Markets [“DB at a Glance”, 2006]

In order for GES to continue providing high quality service, resources must be designated to support the system. The information technology division within Global Exchange Services (GES IT) designs and supports many applications, and specifically for this project, provides maintenance and support for the job scheduling systems. The current systems in operation – TaskMan, RANtask, and the Sydney Scheduling Perl Scripts system – are soon to be replaced by a single scheduling system now in production, known as the Job Execution Framework (JEF). ["Deutsche Bank appoints Peter McLady as Head of Global Exchange Services", 2006]

The three main reporting areas within the futures and options division of Deutsche Bank are trade, position and balance reports. Trade reports contain data about trades made that day, while position tables summarize the number of separate trades that have been completed and remain open from day to day. A balance report provides monetary information often subdivided to track commissions received for the bank and profits for clients.

2.5 The Global Incident Management System (GIMS)

The Global Incident Management System, or GIMS, is the current application used by Deutsche Bank to report all support issues regarding application problems. Users log into GIMS through a web-based interface, open a ticket that applies to a specific

application, and customize their request to address the problem. While a wide range of applications are supported by GIMS, we will focus specifically on how the system applies to TaskMan. In 2002, when TaskMan query requests reached a level too difficult to manage in an organized manner through individual emails, GIMS became the standard tool for reporting job submissions to TaskMan [Personal Communication, Ian Ramsay, October 30, 2006]. Currently, both support issues and job submission requests are handled simultaneously on the same interface in GIMS. Since its installation, nearly 8,000 issues have been logged pertaining to TaskMan, ranging from requests to input new queries, to modify existing tasks, or to raise attention to problems. The current GIMS submission process does not offer adequate information to the Operations Excellence team (OE) who completes the requests. Much of the information required to link tasks back to the original users cannot be stored in TaskMan, making solving problems very difficult for support personnel. Also, due to the customized level of input users can provide on GIMS ticket, the information provided is commonly plagued by inconsistency, thus limiting the effects of troubleshooting efforts. Below is a visual of the GIMS interface used to submit TaskMan issues.

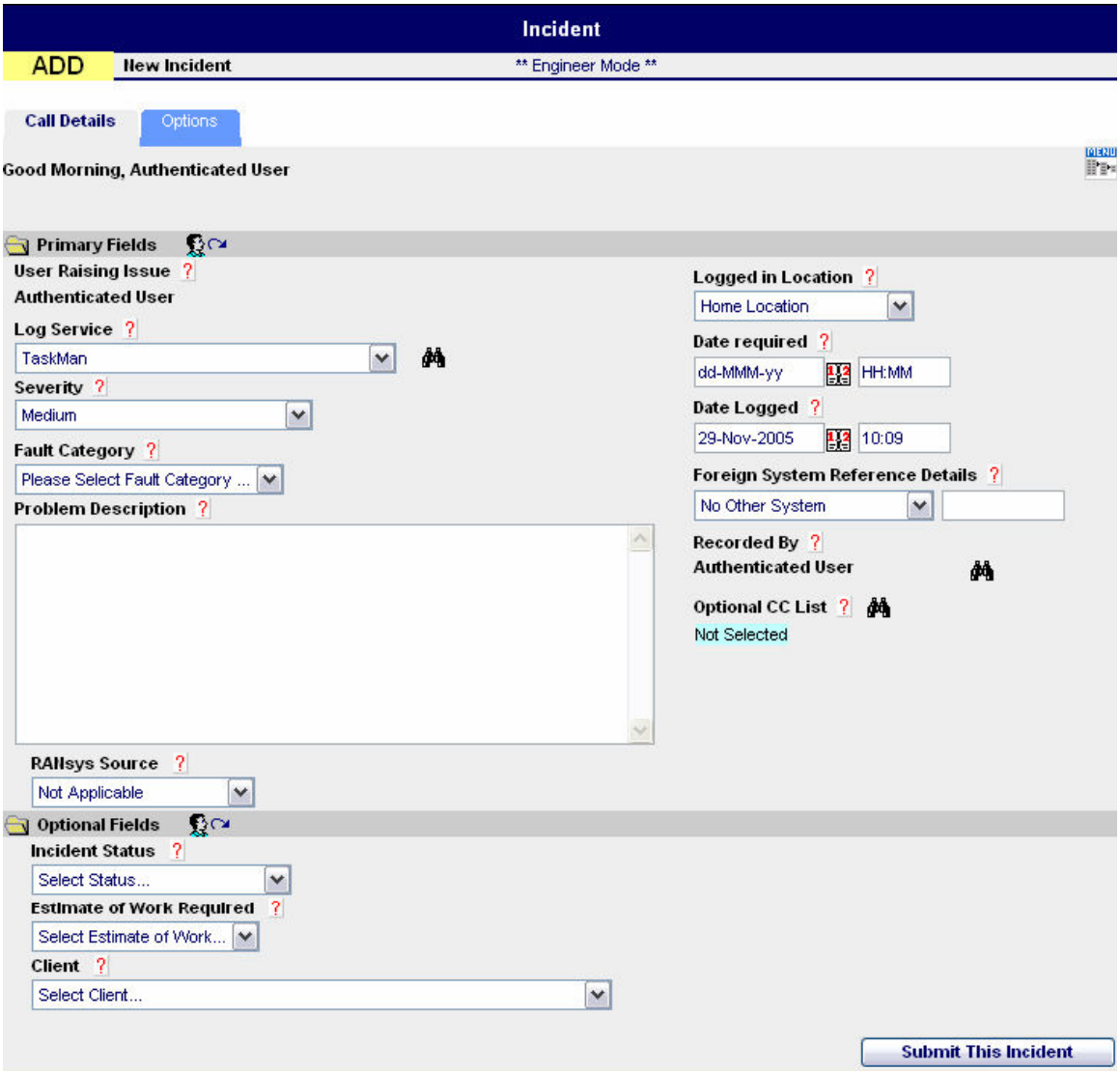


Figure 5: GIMS Ticket Interface [“OBI IT GIMS”, 2006]

2.6 TaskMan Job Scheduling System

2.6.1 History

TaskMan is currently the primary job scheduling system used in production at Deutsche Bank. Built in-house in 1998, developers expected TaskMan to run eight to ten jobs, mainly Structured Query Language (SQL) scripts, in order. The software was coded using Microsoft Visual C++ and designed to run in a Windows NT/2000 Server environment.

Shortly after the original version of TaskMan was released, word spread throughout the company about the capabilities of TaskMan – which TaskMan could help save time and effort. As a result everyone wanted to add their tasks into the system. Very quickly new tasks were added to TaskMan that could not always be handled such as working with the archival system, COOL. Because of this rapid growth, TaskMan was modified in a makeshift manner to ensure each job operations needed could be executed. Currently there are around 2,500 tasks within the system, 1,900 that are believed to still be running on a daily basis. Since TaskMan was not developed with expansion in mind, this ad hoc style of maintenance persisted until 2004. TaskMan's last update was to permit an environment change to Windows 2003 Server and to authorize tasks to run updated Java scripts. Since then, all TaskMan development has been halted and the focus has been put on developing the new scheduling system, Job Execution Framework (JEF). New jobs and amendments are still requested through TaskMan every week, however, new types of jobs that TaskMan cannot handle are sent to JEF.

2.6.2 The Job Submission Process

The following steps demonstrate the typical process currently in place for submitting a task to TaskMan (other job scheduling systems follow a similar process):

1) A User, generally GES Operations, requests information and builds a SQL script to retrieve the data from the RANbase tables. Also, Operations very often requests information on behalf of external clients
2) The User completes a TaskMan GIMS ticket specifying their name, division, a brief explanation of the information being requested, and the form in which the information will be received (email report, FTP drop box). The query is submitted as an attachment for OE to receive and implement.
3) Operations Excellence begins the creation of the job into TaskMan using the submitted SQL query. Furthermore, OE inputs the specific timing and repeatability information unique to each task as requested on the GIMS ticket. There are User Acceptance Tests (UAT) instances to check the query before it is

accepted by TaskMan, but they are not being utilized at the moment [Personal Communication, Green PSG Manager, Pete Lindsay, November 1, 2006].

4) The job is submitted into TaskMan as a configuration file to be run.
5) TaskMan executes the job based on the configuration file and the task dependencies.
6) Meanwhile, OE monitors the performance of the job in production and reconciles any problems that are brought to their attention. As of now, techniques to monitor issues vary. When small or time sensitive issues arise, the formal tracking process used through GIMS is often bypassed.
7) If any upstream delays (when a particular job can't start at its denoted time because its dependencies are not completed) or errors occur, a new instance of TaskMan is manually opened to run the overdue job. Once the postponed job is run successfully, the new instance of TaskMan is shut down.
8) Alerts are triggered if a job fails to run within a designated time period – typically one hour. If an alert is triggered, Support Level 1 (OE – outsourced) is contacted, followed by SL2 (OE – internal), and SL3 (GES IT) if necessary. Finally, the developers of TaskMan are contacted to settle any remaining complex issues.
9) The TaskMan output is distributed to the respective owners in the form of a report.

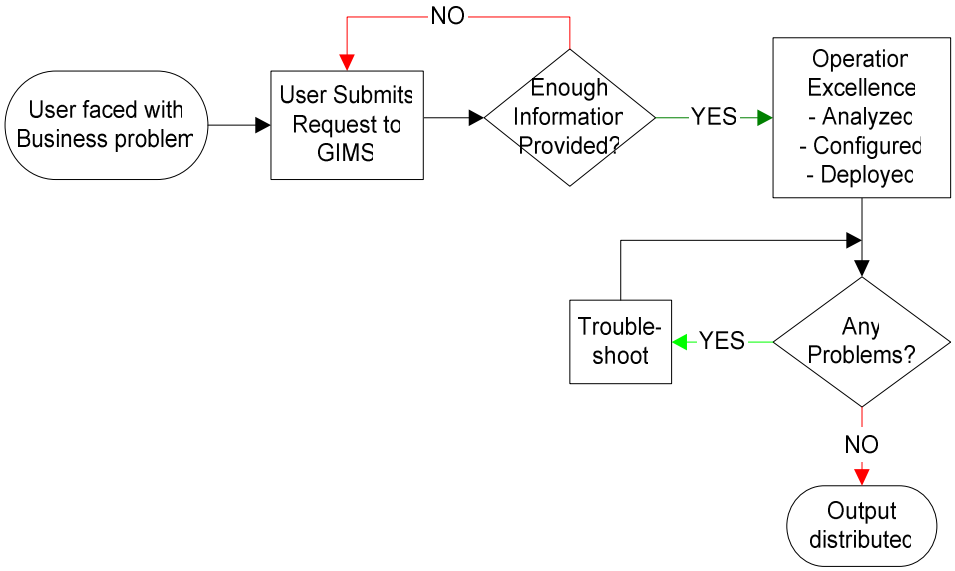


Figure 6: Current Job Submission Process

Refer to Appendix A for explanation of shapes

Refer to Appendix B for detailed chart

2.6.3 Architecture

TaskMan is a single, server-side application designed for data transfer and minor data manipulation. Therefore, it depends on external programs to execute most of the tasks requested. TaskMan uses an Oracle database to store and read successful tasks to verify that tasks are executed in the correct order. Two other important databases used by TaskMan are Rolfe & Nolan's RANbase and RANsys, which contain the financial data for Deutsche Bank's Futures and Options division. We focused on the requests from Operations that use RANbase, as this is more widely used. In order to receive the output of queries or other tasks, TaskMan is compatible with Microsoft Excel and formats the results as *.csv files. These results can then be emailed utilizing an external program called WinMail or posted on a shared-drive or drop box using an internal FTP client.

One important design factor in TaskMan is that it is single-threaded. This means that tasks can only be executed in order, one at time, based off the time-driven system that it uses. This is somewhat overcome by the TaskMan server running 23 different instances of the system at any given time.

2.6.4 Problems

The 'front-end' task submission process and 'back-end' code of TaskMan – utilized by Operations & OE and GES IT respectively – suffer from many problems. The main reason for these problems is that TaskMan was developed in a piecewise manner to address short-term problems with minimal attention to established programming practices. For example, early developments to TaskMan lacked a standard form of documentation and were not coded with expandability in mind. In turn, modifications to TaskMan were made to support new functions without considering the ramifications each had on the overall system design. Due to nearly 20% employee turnover (per year) within the GES IT department since TaskMan development, none of the original developers remain within the company [Personal Communication, Ian Ramsay, November 3, 2006]. Because of this, very few programmers have a comprehensive knowledge of the inner workings of TaskMan. These complications have forced Deutsche Bank to expand OE's responsibilities to include creating new jobs and modifying existing jobs within the TaskMan system.

2.7 Other Scheduling Systems

Several other scheduling systems aside from TaskMan are used throughout Deutsche Bank. By fully understanding their specific features and drawbacks, possible recommendations and improvements can be made to JEF to fully address the company's needs. RANtask is the second most popular system – running about 300 jobs a day – but is limited to use at the Frankfurt office for job scheduling and the Sydney office for its reconciliation features**.** RANtask is a third-party application from Rolfe & Nolan, the same vendor as RANbase.

Instead of using an executable application, Sydney uses web-based Perl scripts to do their job scheduling. Even though these systems are not as prominent as TaskMan, they experience many of the same problems including:

- Error reporting issues
- Job execution delays
- A lack of controls regulating the job submission/creation process
- An inability to search jobs in the system
- Insufficient data stored linking the task to its owner.

An in-depth analysis into these secondary systems will not take place, however, the suggestions made for TaskMan and JEF will help solve problems and facilitate the migration of each system into JEF. After the implementation of JEF has been completed, the entire company will operate seamlessly as one entity.

2.8 JEF – Job Execution Framework

2.8.1 History

The Job Execution Framework, or JEF, is planned to replace the job scheduling systems currently used within Deutsche Bank. JEF is an in-house developed system that began development in April 2005 and is currently in the production phase. However, the new system has very limited capabilities at this point and can only execute selected jobs. GES IT plans to finish migrating necessary tasks from TaskMan and the other job scheduling systems used internally in Deutsche Bank to JEF full-time by late 2007, or early 2008.

2.8.2 Architecture

JEF is constructed as a server-side package of applications called "services" that work together or independently to perform different tasks. This is known in the programming community as Service-Oriented Architecture (SOA). This also follows closely with the Agile method of application development, in which applications are built function by function and put into production after testing each iteration. In contrast, in the traditional method of application development known as the waterfall method, the application is built, tested, and then maintained as problems occur. Along with using different services to perform each function of JEF, the entire code is multi-threaded to permit many jobs to run in parallel. JEF is utilizing Java as a development platform to cut down production cost with the extra abstraction layer that Java provides.

2.8.3 Service-Oriented Architecture

JEF is designed to take advantage of a common structure called Service-Oriented Architecture (SOA). SOA is built on the idea that each business function or process is broken down into a "service" that can communicate and interact with other services. The services used in an SOA environment can either be on different servers/machines within a network or on the same local machine. This evolution in architecture allows for a multitude of protocols, standards, and programming languages to be used in designing each service. SOA takes advantage of a common interface so the programmer only needs to know which service he wants to use, not how it works ["Service-oriented architecture"]. This allows the business to adapt and change its IT structure and services as the business itself changes. As a result, companies often save money because of reduced maintenance costs. SOA was chosen by Deutsche Banks' GES IT group because it fits the business model of futures and options very well. While the conceptual business functions are slow to change, the methods to perform each function are always evolving to adapt to new variables and procedures. Using these services "encapsulates at the application level" allowing users to understand exactly which service they need [Personal Communication, JEF Architect, John Hawkins, November 5, 2006].

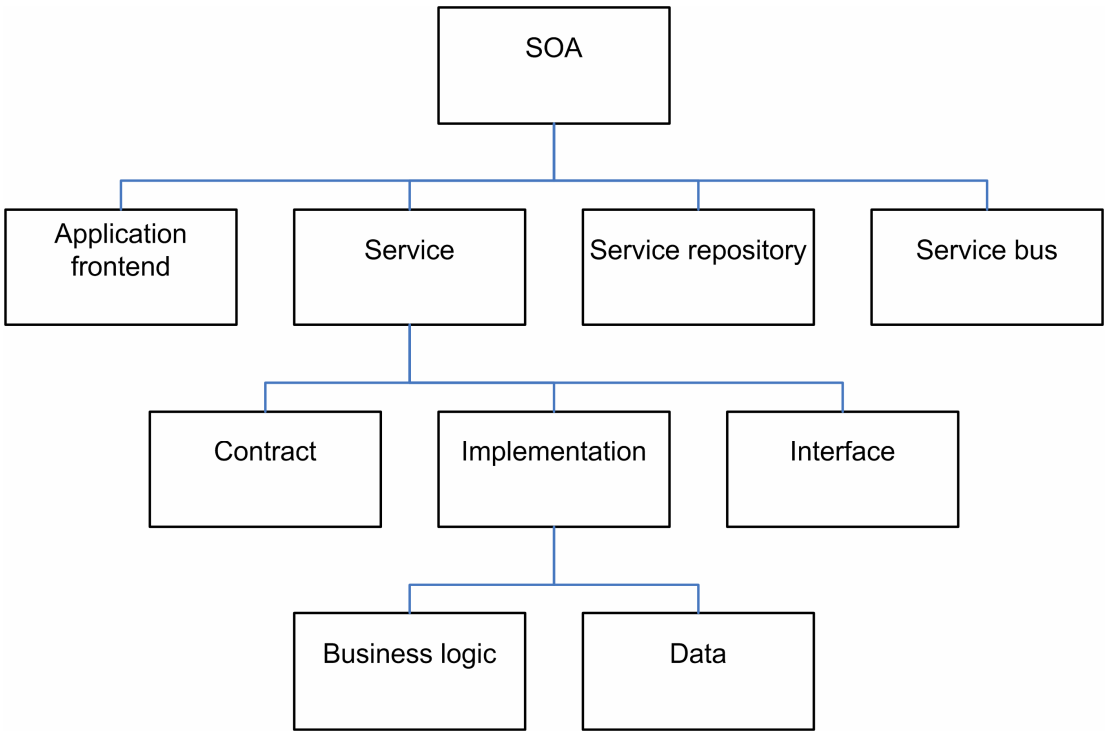


Figure 7: Elements of SOA ["SOA"]

SOA should be considered a basic model rather than a framework designed mainly for businesses, and has the potential to cut costs and save development time in highly complex environments. Utilizing SOA in a business promotes reusability at the functionality (macro/service) level, instead of at the class (micro/object) level ["SOA"]. Although SOA is an event-driven system, it is classified as client-server driven; this is because the client must initiate direct call to the service, enabling the service to interact with the client and vice versa. This concept has been compared to many other programming paradigms that have been introduced in the past decades including modular programming and event-oriented programming ["SOA"]. The difference is that SOA tries to separate the consumer from the implementation to maximize reusability and allow access that is platform independent.

The problem with adopting SOA into a program design is keeping track of all the messages being relayed. The environment must be able to handle this large volume of messages and manage how the information interacts with each service. Extensible Markup Language (XML), a robust internet standard that makes it easy for users to read and design their own data structures, is a commonly used protocol in SOA. However,

XML requires a large amount of processing power to parse or combine relatively small files. Finally, security is a major issue to consider for a SOA system. At times, services are made available outside the business' intranet exposing them to more threats than ordinary colossal proprietary applications ["SOA"].

2.8.4 Enterprise Service Bus

Enterprise Service Bus (ESB) is a major component which utilizes the power of SOA. ESB is rapidly becoming one of the top tools for businesses "to incorporate SOA principles and increase their sales" ["Enterprise service bus"]. ESB is not a framework to build SOA on, but is instead an Abstraction Layer above the underlying messaging system ["ESB"]. This abstract layer frees the architects and programmers from worrying about how multiple systems communicate, but rather on how to take advantage of this ability. It provides the developers with a wealth of predefined functional services such as the support of web-service standards (XML specifically), standards-based adaptors to support legacy systems and standardized security model. It is based on multiple standards by integrating the most useful principles already in place within the industry and building the adaptors to allow the use of each standard concurrently ["ESB"].

Sonic ESB was chosen to be the platform for GES IT because it can be added to existing systems as Sonic products are already in use as the underlying messaging system within most of Deutsche Bank, which reduces cost and complexity. This allows for increased elasticity to change requirements without down-time and can provide the ability for small solutions or enterprise-wide solutions to be deployed [Personal Communication, John Hawkins, November 5, 2006]. This abstraction layer makes IT's job configuration of the system rather than coding which saves time ["ESB"].

2.8.5 Control-M

Control-M is a commercial enterprise-level job scheduling system with features such as failover recovery and clustering, and includes simple functionality to execute DOS batch and UNIX shell scripts. This system could have been used as a replacement for TaskMan instead of JEF as it performs simple functionality more efficiently than

TaskMan; however there are many shortcomings that would eventually cause similar problems to those TaskMan is experiencing, including:

- A time-driven system
- The system could only handle DOS batch and UNIX shell scripts, usually leaving no information as to what had been accomplished within the scripts
- No built in logging
- System is robust but not scalable
- Hard to maintain and keep running
- Each client needed to have special client software installed

GES IT wants to move away from these problems and separate job scheduling from the job execution. However, many people are already familiar with Control-M, as it has been used elsewhere in the company, and opposed the creation of an entirely new software package. Finally, it was decided that it was possible "to adopt a compromise," and GES IT established Control-M as a ground work, and start building JEF as a complement to Control-M. [Personal Communication, John Hawkins, November 5, 2006]. Control-M is being utilized to handle all time-driven events for JEF. By taking advantage of its enterprise-level messaging system, Control-M can inform JEF of a time-driven event to trigger the corresponding job.

2.8.6 Similarities & Differences

Despite the many differences between JEF and TaskMan, there are some similarities that exist. They both are back-end office applications designed to aid users with the extraction and delivery of data. TaskMan and JEF both rely heavily on external programs such as RANbase and WinMail to perform most of the generic tasks they perform.

The most significant difference between JEF and TaskMan is that JEF separates job scheduling from job execution. This is accomplished in part because JEF is multi-threaded while TaskMan is single-threaded. This means TaskMan has to execute in-order and one job at a time while JEF can execute out-of-order and can simultaneously run however many jobs need to be run at that time. TaskMan is a completely in-house built system; JEF uses Control-M to schedule time-driven jobs as well as an internal system. JEF is event-driven; TaskMan is time-driven (time can be an event). JEF is a collection of configurable services; TaskMan is a single executable program. All

employees knowledgeable of the inner workings of TaskMan have left the company. To avoid repeating this problem, JEF is being built with documentation standards to allow future employees to have the same proficiency about the package as the original coder.

3.0 Methodology

The main goal of this project was to help Deutsche Bank's GES IT department identify problems with the existing process of submitting jobs, and provide quality resolutions for the future job scheduling system. To understand the various aspects of these problems, we interviewed two end users and three support personnel, and explored the many software systems used within GES IT. This included each job scheduling system (TaskMan, JEF, RANtask, Asia-Pacific Perl scripts), the most common databases used (RANbase, Business Objects), the system change-control software (RANsys), and the job submission and problem management system (GIMS). GIMS tickets were analyzed to identify the necessary information to be included with each request.

To make the best use of our eight weeks at Deutsche Bank, we broke up our project into three divisions. While the divisions were closely related, each required individual attention to focus on the expectations of our sponsors and advisors. We began by identifying all currently known and unknown problems within the TaskMan system. From there we defined the current process of submitting jobs to any scheduling system and established a set of controls to regulate problem areas. Lastly, we applied a modification of this set of procedures to aid in moving actively running jobs to the JEF system. A Gantt chart detailing the progression of our project is available in Appendix F.

3.1 Identified Limitations of TaskMan and Other Systems

As previously mentioned, the current task scheduling systems used throughout Deutsche Bank have many visible problem areas that must be addressed before they become an issue in the new JEF system. A portion of these problems have been identified by GES IT and addressed in the early developmental stages of JEF. However, a clear documented understanding of all the existing problems in TaskMan, RANtask, and other systems is not available. In order to identify a comprehensive list of the limitations of the current job scheduling systems, we conducted a series of interviews with qualified GES IT, OE, and Operations personnel and closely examined the current TaskMan system. By identifying all of the procedural and technical problems that exist

in TaskMan and other systems, we provided more suitable control recommendations to avoid similar issues from occurring while JEF is transitioned into operation.

3.1.1 Interviewed Pete Lindsay, Vilas Hirani, and Jens Balkmann

To gain an understanding of the issues surrounding TaskMan from the Operations Excellence perspective, we spoke with Pete Lindsay, the Green PSG Manager and Vilas Hirani, an OE Vice President. Green PSG is an outsourced support group located in China and provides SL1 and SL2 level assistance for TaskMan and other applications. Specifically, we focused on understanding the current job submission process that OE follows and what problems are frequently encountered. As a manager of a support team, Pete Lindsay offered a valuable first-hand perspective on the submission process, information regarding the techniques used to solve individual issues, and an input on the areas most in need of improvement. Vilas Hirani was utilized to further understand TaskMan's capabilities and discover exactly what information is provided for support issues. The information presented to us through these interviews was used to establish a basis for recognizing TaskMan problems and limitations from a support point of view. A transcript outlining the questions raised is available in Appendix E.

To understand TaskMan limitations from a user's perspective, we interviewed Jens Balkmann, a GES IT Operations manager. Through this process we determined how a user approached submitting a task, and the problems TaskMan presents along the way. Balkmann was also a valuable contact to discuss the use of RANtask, its differences from TaskMan, and other limitations. The specific points covered are included in this report in Appendix E.

3.1.2 Examined the Current TaskMan System

TaskMan's current logging system is constructed to record the start and end times for each task run each day in separate log files. These logs are the only source of information available to check how long tasks are taking to run each day, and possibly provide evidence toward a tasks performance over time. Through a careful analysis of the TaskMan log files, we were able to determine exactly what and how TaskMan records data. By studying the available data and identifying what information lacked, we

were able to make recommendations for new logging requirements for JEF. An analysis of the organization of the data stored provided a basis for a proposal to reconstruct the layout of the log files – in hopes to allow for a more easily accessible, organized, and workable database of logs. Particularly, we analyzed the data to be tailored for future statistical analysis that can quickly identify developing problem jobs.

3.2 Designed New Process Controls & Guidelines

Another major goal of our project was to design a set of controls and guidelines to regulate and standardize the job submission process. Before we could design appropriate regulatory procedures we had to first identify the current submission process. After confirming this process with Ian Ramsay, we defined problem areas and determined feasible solutions. Finally, using our solutions and previous knowledge, we developed recommendations for new controls and guidelines that both Ops and OE should implement to solve the problems inherent with the current submission process.

3.2.1 Deciphered Current Procedures

Through our interviews with members of GES-Ops and OE support, we learned the current steps taken throughout the entire job submission process. To gain an appreciation of the actions taken by the Ops team we interviewed Jens Balkmann. To better understand the support point of view, we talked with Pete Lindsay and Vilas Hirani. After these discussions, and considering the viewpoints of both groups involved, we were able to determine the entire process. Our interviews with Ian Ramsay and John Hawkins confirmed that the step-by-step process we uncovered is an accurate portrayal of what they have experienced in the company. The current process used by the employees of OE and Ops can be seen as a flow chart in Appendix B.

3.2.2 Discovered Problematic Steps & Trouble Areas

After determining the set of procedures currently followed by each participant in the job submission process we had to analyze the 'building blocks' of each step to find areas to be improved upon. The first resource used was our interviews. By asking each group the problems they have encountered in submitting a job, we were able to get a

fresh perspective, unattainable through a dry analysis. We also delved into the vast knowledge that both Ian and John provided from their past experiences and work with developing JEF. It was very important for us to take our own look into how each of the supporting systems mentioned at the beginning of this chapter work and interact with each other. Utilizing resources such as last year's MQP, the list of problems we came up with earlier, and our own past experience with similar problems played a major part in identifying defects with the current array of procedures.

3.2.3 Identified Solutions to Problems and Designed New Process

After uncovering the significant pitfalls of the submission process, we analyzed the process flow to identify possible solutions toward improvements. Again, we used interviews and follow up questions to determine what recommendations would be useable to provide the greatest benefits at a particular stage in the process. We also interviewed Iliana Dimitrova, a control specialist, to gain an understanding of other less common SQL query submission methods – in particular how Business Objects is being used to assist the process. Throughout this entire part of our MQP, we came up with multiple resolutions to present to Deutsche Bank, GES IT in particular, so they have options to choose from that best fit their company and department.

Finally, after determining multiple options to solve each problem area analyzed, we presented the most effective set of procedures to eliminate the problems with the current job submission process. To determine the priority and practicality of each recommendation, we again relied heavily on our analysis of the information gathered through our interview process. Also, we created a wizard application to help tailor the current job submission process to gather all of the recommended information.

3.3 Established Controls for Job Migration to JEF

With some components already in production, the new job scheduling system – the Job Execution Framework – is scheduled to replace the existing scheduling servers by early 2008. An important goal of this project was to provide Deutsche Bank with a formalized process and list of controls to ease the migration of jobs now in TaskMan and the other job scheduling systems to JEF. A large number of aborted and duplicate jobs

exist in the current job scheduling systems, slowing system performance. To ensure a smooth transition to JEF, we developed a formal set of guidelines for importing jobs into JEF that minimized the amount of unnecessary jobs from entering the system, only capturing the active jobs still needed. These guidelines included methods for evaluating whether a job is still required in order to avoid duplicate jobs, unused jobs, and other problem jobs from entering JEF.

3.3.1 Interviewed Ian Ramsay and John Hawkins

We began gathering information on the JEF migration process by speaking with the architect of JEF, John Hawkins, and Ian Ramsay, one of the main players in determining how and when the migration will take place. John provided invaluable information on the technical and architectural characteristics of JEF and identified the important concepts to carry over and new problems he is trying to avoid. Further, he described the current methods used to transition jobs into JEF, which provided a basis for our suggestions. Ian Ramsay gave us insight from a managerial perspective as to where current problems exist, where improvement areas may be, and areas we should explore. Understanding the capabilities and limitations of JEF and the current systems helped to define our path.

The interviews conducted to identify job scheduling limitations and new process controls provided background information to compile suggestions for establishing controls during job migration. Without understanding the processes and problems of various departments, it would have been difficult to provide effective recommendations.

3.3.2 Utilized Previous Research on TaskMan and JEF

The previous group to complete their Major Qualifying Project at Deutsche Bank conducted similar analysis, but the goals of the projects differ greatly. However, the report provided us valuable references such as background information and terminology. Also, the Task Event Calendar System (TECS) developed as a part of their project aided us to identify the activity status of a job in TaskMan. By examining their research and analysis, we were able to supplement our own findings to further maximize the use of our time and support our claims.

4.0 Results and Analysis

TaskMan, while a powerful application, has grown much larger than ever intended. The lack of controls surrounding the system has allowed it to grow to incorporate an overwhelming 2,500 jobs. Results gathered in this section of the report provide a strong basis for this project's conclusions for improvements to include during the transition to JEF. Utilizing our methodology, we gathered specific information about the problems surrounding the current job submission applications and process to understand their significance. Included are explanations of the limitations of the current applications, a detailed analysis of the current job submission process, and further results from our interviews. The analysis of each problem focused our view to provide the most valuable future recommendations.

4.1 TaskMan Limitations

While TaskMan has proven to be a useful tool, the limitations of the system will prove detrimental to the progress of business if they are not resolved. Through interviews and research into the TaskMan application, we uncovered the main problems that need to be addressed, including the following:

1) TaskMan's single-threaded architecture
2) A high volume of duplicate jobs running, or jobs based closely on the parameters of an existing job, as well as unneeded jobs that have never been closed
3) No link established in TaskMan to trace a job back to its source, notably its GIMS ticket. Furthermore, there is no place in TaskMan to view an overview of a job and its intended recipients
4) An absence of comprehensive UAT (user acceptance tests) used as checks of configuration environments
5) Insufficient logging data generally only includes the start and end times for each job making support issues difficult
6) Cannot view the progress of a job and its dependencies
7) Non-Optimal queries exist
8) Changes to a job can only be made now by logging into the production server itself which threatens the system stability
9) Memory leaks exist due to poor programming, requiring a scheduled manual restart

These problems, among others, are the main source of failure for the TaskMan system. TaskMan's single-threaded design, while not an issue in the improved JEF system, is still

the underlying cause of upstream delays in job execution. When an individual job fails and misses its anticipated end time, the dependent jobs to follow are pushed back until the failed job is manually restarted and completed. To add to this problem, TaskMan does not offer a comprehensive set of User Acceptance Tests. The current UAT checks in place are not widely implemented because most of the jobs being run access such a wide variety of applications that the tests are not sufficient [Personal Communication, Pete Lindsay, November 1, 2006]. Rather, jobs are manually inspected through a "four-eye" principle and a "sanity" check, a process that allows for inconsistency in the quality of jobs that enter the job scheduling systems.

A major issue with TaskMan and the other job scheduling systems is that there is no tool or method to view and analyze what jobs are currently running. As a result, many unnecessary and duplicate jobs are run with daily batches, placing stress on the server. Of the nearly 2,500 jobs in the TaskMan system, about 1,800 are still executing, of which an estimated 1,000 jobs are actually being used. With so many superfluous jobs in TaskMan, small delays cause occasional system backups. In addition, many of the queries being entered as jobs in TaskMan are not optimized, and could be designed to run more efficiently with some preliminary testing prior to submission [Personal Communication, Ian Ramsay, October 25, 2006]. This situation severely affects business, and causes ripple effects reaching as far as missing important company accounting deadlines [Personal Communication, Gregory Friel, Director of Group Technology and Operations, October 25, 2006]. However, TaskMan does not maintain adequate information about the jobs to rectify this issue. Job ownership, a job description, and other vital information are not stored anywhere in TaskMan. There is no link – such as a stored GIMS ticket number – to allow a support agent to contact the job's creator when a problem occurs. Also, TaskMan's logging system is designed to continuously update the start and end times of jobs. The system, intended for use by support personnel, does not log enough useful information to help evaluate and solve problems, making job progress impossible to trace. To view a complete list and visualization of problems with TaskMan and the other systems, please refer to Appendix D.

4.2 Problems with GIMS

While the current job scheduling systems have many faults, the supporting applications are not properly utilized as well. The Global Incident Management System (GIMS) is the tool for Operations users to submit new SQL queries and request changes to existing jobs. However, this web-based management system was never designed as a "job submission" or "configuration change" tool for TaskMan. By unofficially adopting GIMS as the required step to submit requests to TaskMan, users are forced to customize their requests on a case by case basis. This level of customization creates inconsistency in the level of detail provided with a ticket – as it is based on the user's personal judgment. Currently, the ticket consists of the following fields:

- Contact Information including name/phone number – automatically recorded (Owner, Group, Engineer, User).
- Request Severity (High, Medium, or Low).
- Fault Category (Application Problem, Configuration, Development Request, User Acceptance Test, and User Administration).
- "Free text" description.
- Date/Time entry including due date, SLA, and date rose, logged, and last progressed – automatically recorded.
- Optional Attachment feature to submit proposed SQL query.

After examining GIMS tickets submitted during October and November 2006, the most apparent category needing attention was the "free text" description. Descriptions differed widely between GIMS tickets, as users provided a varying level of detail. The current GIMS submission process does not include any set of controls to attempt to regulate the information provided by users. In the vast majority of the GIMS tickets that proposed a new SQL script, users did not provide a description of the job to be run. Without a description of what the new SQL query refers to, it is difficult for OE to govern whether a new job needs to be created from scratch, or rather modified from an existing job. In addition, the contact information recorded on the GIMS ticket is not sufficient to track future problems. Simply recording the user at the time of submission does not guarantee the user can be contacted if a future problem occurs – especially in a business with a high turnover rate.

While GIMS has many characteristics that require attention to better suit the job submission process, it is a rather inflexible tool. GIMS is currently used in conjunction

with many other applications, and redesigning the system would require extensive resources.

4.3 Job Submission Faults, Drawbacks and Bottlenecks

Many problems in the job submission process have been previously identified, but a further analysis as to how it affects the whole process will be detailed further. Please refer to section 2.6.2 for a detailed explanation of the current process for job submission, or Appendix B, for a flowchart of the process. We will start with the Operations team as they start the progression of events. After that we will state issues with GIMS as a job request tool, followed by the systematic transactions that take place with OE to create the job. Finally, we will examine the faults with the job submission process in regards to TaskMan.

4.3.1 Operations, End Users

When faced with a business problem – often brought up by external clients – Operations personnel use TaskMan to access, organize, and deliver valuable data to their respective sources. This often consists of writing an SQL procedure against the most commonly used database, RANbase. At times, the Operations team requests either inaccessible data or requests data from the incorrect table in the database [Personal Communication, Ian Ramsay, November 3, 2006]. This problem results from a lack of education and documentation on the RANbase database – little to no descriptive information is provided to Operations to explain what each table contains. Improper or inefficient citation of data in the SQL queries can cause a job to stall and eventually leads to many of the upstream delays discussed previously.

The Operations team is referred to as the "end user" as they are the dominate users of TaskMan. As of now, there are no governing rules in place to limit who and what department can present a new job to the scheduling system. This invites potential future problems because anyone, despite their qualifications, can access TaskMan's features. Determining these restrictive qualifications for TaskMan is an essential part of our recommendations. However, it is very hard to determine which jobs are from other departments as the Operations team will often submit for a colleague. The important

question is whether these employees have a practical business reason to be utilizing TaskMan.

As of now, there is no documented list or searchable collection of jobs that end users can access to determine if certain jobs already exist. This contributes to the numerous duplicate jobs that have overwhelmed TaskMan since it became a company-wide program. In many situations, an end user could simply amend their email to the final submission task to receive a copy of the output instead of creating an entirely new job. Instead, time and resources are wasted that could be used to complete other priority issues.

4.3.2 GIMS

GIMS is the current tool used to request jobs to be added to TaskMan. It is also the same system that is used to report errors and complications from Ops and other end users to OE. A main problem with GIMS is that it is not designed for what the employees of Deutsche Bank are using it for. GIMS stands for Global Incident Management System, which means it is used to track and handle problems that arise in the work place; however GIMS is creating more problems because it is being used improperly. Because this system is not designed for job submission, there are many faults that can't be overcome. GIMS is an inflexible system due to the various types of information entered into it. This inflexible system creates even more problems as many users refer TaskMan issues to a different application in their request, resulting in output delay, and wasted time by OE trying to figure out where the request should go, and who the request belongs to. This mainly results because most users are unaware there is a separate section for TaskMan within GIMS or they don't even know TaskMan exists [Personal Communication, Ian Ramsay, November 8, 2006].

Using GIMS as a job request tool for TaskMan also makes it difficult to restrict users from submitting jobs into the system. Every so often staff from OE will submit a job on behalf of an external client but because the system automatically takes in the submitter's information. The actual requester of the information is lost unless that OE staff member can later on remember who asked for their help. Similar problems occurred when jobs were submitted to TaskMan before GIMS was in place. "Tracking" and

"logging" of a job was through email [Personal Communication, Ian Ramsay, November 8, 2006], and hence there is no original GIMS ticket to refer back to.

4.3.3 Operational Excellence

OE is responsible for receiving job requests, converting them into TaskMan jobs, sending jobs to production, and then monitoring their progress to ensure there are no problems. They also resolve problems that are brought to their attention by the Operations team or other recipients of the final reports. One issue with this situation is that OE must deal with High Priority items first, and because GIMS has only three settings (High, Medium, and Low) most users place their jobs and issue request as High [Personal Communication, Ian Ramsay, October 30, 2006].

Other problem areas with the job submission process from a support point of view include:

- OE's role in the process has been reduced to more of a "middleman" position without much security to regulate the flow of jobs into their perspective systems.
- Not all information supplied on GIMS tickets is stored for TaskMan jobs**.**
- Difficult to trace down owners of jobs, particularly older jobs in the system.
- No formal testing of jobs being entered.
- To distribute a job into TaskMan, support must log into the production server [Personal Communication, Pete Lindsay, November 1, 2006]. This is very risky as the server environment is read/write, not just read-only and makes the system very unstable to local changes

4.4 JEF- Results and Analysis

4.4.1 Comparison to TaskMan and Other Job Scheduling Servers

As TaskMan approaches its final phase at Deutsche Bank, GES must begin to develop a job migration process to JEF. Although each system focuses on the same goal, their internal processing to complete tasks differs widely due to their architecture and capabilities. Because of the systems' complex architectures, differing limitations, and the dependencies of the jobs within TaskMan, a successful migration to JEF requires detailed planning and patience.

We will begin our analysis of JEF by explaining the similarities with TaskMan and the other job scheduling systems. Despite the improvements to JEF compared to the

current job scheduling systems, there are many areas that need to be addressed. First off, controls regulating who has the ability to submit jobs, how jobs are tested and submitted, and what information is required before permitting a job to enter the system have not been established. Limited, if any, improvements have been created in this area to prevent similar problems from occurring in the future. One of the main concerns is determining who should be able to submit jobs. Should this decision be made based on the department, job, or experience? Presently, any member of Operations and GES are able to submit jobs, and often submit them on behalf of external clients. What information should the support team require before allowing a job to run? In the current process, minimal contact and job-descriptive information is provided. In the conclusions and recommendations section, we will further discuss and recommend ways to improve JEF in these areas.

Although additional features need to be implemented into JEF, there are many differences from TaskMan (and other systems) that drastically improve the job scheduling performance. The major difference is that JEF is an event-driven, multi-threaded system. A multi-threaded system allows for many jobs to run simultaneously versus only one job running at a time. Despite the fact that TaskMan executes many instances simultaneously, limitations exist that are not present in JEF. For example, TaskMan is much more susceptible to upstream delays caused by individual job errors. JEF, in comparison, is able to continue working despite an error in the system, omitting such bottlenecks.

JEF's event driven architecture makes it possible to view if a particular event has ever been fired – a useful feature to help prevent storing dead or aborted jobs in the system for extended periods of time. If an event has never run and a GIMS ticket has not been raised reporting problems, this job can be assumed to be unneeded, and therefore disabled. However, some jobs may be needed only once a year or under disaster situations, so this job could be wrongly disabled unless notice is provided initially. Therefore, it is important to create a feature that notes how often a job should be run or under what circumstances it should be executed when the job is submitted. Again, we will discuss this further in the recommendations section.

The monolithic architecture of TaskMan resulted in a system freeze in 2002, preventing any improvements to the system. As a result, expandability became a priority in JEF to keep up with technological advancements and the ever-increasing volume of jobs being submitted.

4.4.2 How TaskMan Affects the Migration to JEF

Referring again to TaskMan and how it affects JEF, it is important to understand TaskMan's capabilities and features. When a GIMS ticket is raised, limited information is supplied including the severity of the incident (low, medium, high), user and engineer name, a problem description and date required. Despite this information being given, TaskMan does not have the necessary features to carry all this information along with the job once it has been submitted. It is nearly impossible to tell which job belongs to what user, the purposes of a job as well as what jobs are active. The only information available is located in the TaskMan logs, which provides the date the job began and the start and end time of a job, but no information on the events happening in-between. As a result, the migration of jobs to JEF may involve submitting each job on a case by case basis. [Personal Communication, John Hawkins, November 15, 2006]

Preventative measures have been taken to avoid these problems in JEF; particularly, "in JEF, there is no distinction between information on requests and information jobs run on" [Personal Communication, John Hawkins, November 5, 2006]. JEF uses an XML database that allows critical information to be stored, and promotes expandability to store additional information (discussed in the recommendation section).

4.4.3 JEF- Job Migration

As JEF begins to replace the current job scheduling systems, existing jobs will need to be transferred smoothly out of their old environment and into JEF. As of now, if a request is submitted through GIMS that is beyond the capabilities of TaskMan, the job will be deflected to JEF. These job types currently running in JEF mainly include Secure File Transfer Protocol (SFTP) and Transformation jobs [Personal Communication, John Hawkins, November 15, 2006]. The process of job submission by the raising of a GIMS ticket will still be used unless replaced by a new server.

The analysis conducted by the previous Major Qualifying Project provides valuable procedures for all migrations to help GES IT determine which projects to migrate and when. As stated in their report, it is important not to rush the migration process because there will be an increased "…risk of error during the early stages of implementation" if not done properly (MQP). Since jobs will have to be transitioned over on a case-by-case basis, it is recommended to keep the job running on the old system while it is migrated to JEF. Because both systems can run in parallel, this is a good check to ensure the job is properly running on JEF, and if not, can continue running on TaskMan.

The four job classifications listed in the previous MQP report are:

1. Independent Jobs
2. Leaf Jobs
3. Complex Jobs
4. Root Jobs

"Independent jobs" are autonomous of all other jobs, and their migration will have no effect on the completion of a task. "Leaf jobs" are the last job in the dependency chain and therefore, are affected by any change made to the jobs that run before it. "Complex jobs" have job dependencies before and after it runs, and therefore are can have a potentially great affect on the chain running properly. The most complex job is known as a "root job" because it is the kickoff job that triggers the chain. If this job is for some reason transferred too soon or disrupted, the entire task will not run. To successfully migrate from the current job scheduling systems to JEF, there are many important guidelines and processes that must be followed. The following figure helps to visualize where each job is located within a dependency chain.

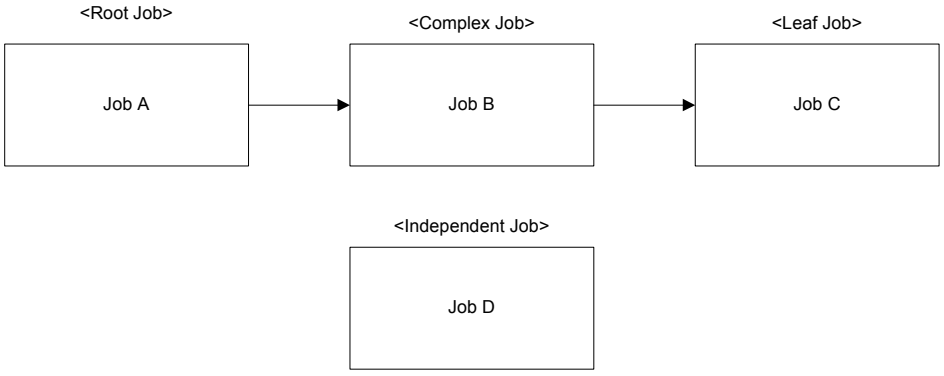


Figure 8: Job Types by Predecessor Status [Piette and Zipkin, 27-28]

As mentioned before, it is very difficult to see which jobs perform what tasks, but it is possible to note the dependencies. This can be done by studying the control files located in the TaskMan server as well as looking through the TaskMan Event Calendaring System (TECS). TECS is used on an as-needed basis and provides an overview of the TaskMan environment. It can be used to see which jobs need changes, and what jobs will be affected by that change. A Gantt chart shows the dependencies, but each job is given the same start and stop times so the time for a set of jobs to run is not accurately depicted. With this information, it will be possible to note the category each job falls into.

5.0 Conclusions and Recommendations

After analyzing our results, we were able to form many recommendations to prevent problems from occurring as Deutsche Bank transitions to the JEF system. We focused on providing recommendations on the following areas:

1. Job Submission Requirements
2. Role of Operations and Information Technology
3. Business Objects as an Alternative
4. Access Controls
5. Job Migration to the Job Execution Framework (JEF)
6. Standardized Reporting Application
7. Wizard
8. Searchable Job Library
9. Job Execution Frameworks Improvements

These recommendations are prioritized to provide Deutsche Bank with a perspective of which ones will have the most dramatic impact on improving JEF. As explained throughout our discussion, these events should, however, flow in a different chronological order.

5.1 Job Submission Requirements

After the necessary jobs are transferred from the old job scheduling systems to JEF and the essential users and IT staff are assigned updated access rights (as discussed in the following sections), new requirements for the job submission process must be enforced to ensure the success of JEF. Currently, there are limited controls to regulate the information provided when a job is submitted into the system. Through GIMS, users supply information such as the owner name, group, engineer, and a description of the desired action, but this information is not stored internally in TaskMan.

In order to improve data submission and collection, we recommend:

- Establish "quality gates"
    - Determine standard information needed with each job submission
    - Once a request has been made, OE must have determine what additional information the user should provide
    - Submit each job into the "Test Phase" only after all information is provided
    - Only after successful testing is performed can a job be submitted into production

Establishing a system of "quality gates" as a new set of controls will ensure that the necessary information /data collection and testing procedures are carried out. These gates should utilize a new application tool (explained in the following pages) suited to the job submission process aside from GIMS. After discussing information that would be helpful when submitting a new job, the following fields have been established as the first gate:

- Job title
- Type of Job (from a standard selection)
- Purpose of job
- Owner of job
    - Business group, team
- Who to contact if problem with job
- Form of output data
- Where to send data

Once these areas have been properly filled out, OE will then review the information and determine if there are enough details to implement the job in the testing phase of JEF. If not, OE will email the owner with more specific questions. In many instances there will be back and forth contact between the user and OE to properly capture the need and use of a job. Prior to entering JEF as a production job, the user must supply the following information at the second gate if requested by OE:

- Rectification date
- Scheduling details
- Implementation details

Once OE attains the proper information through communication with the user and the job has been successfully tested, it can be submitted to run in production.

The *job title* should be a brief yet descriptive heading to provide technical support and users with a reference to search for jobs and a high level understanding of the job. Although this information cannot be utilized in the current job scheduling systems, it should be enforced with any new job submission. The *purpose of the job* is an extension of the job title and should compile the objective and essential details of the job. As discussed in the following section, the more precise a job purpose and job title are, the more effective a job library will be to users.

The *owner of a job* is the person responsible for creating the job and knowing why the job is running and what the job accomplishes. As the main contact reference for

a job, it is recommended that the owner provide their specific business division and individual group to maintain a contact source if the owner leaves the company. One of the major problems that GES IT is facing when evaluating older TaskMan jobs is that the original owner has left the company and no information is recorded to contact a current employee to see if the job is still needed. When speaking with Dennis Klocke of CIB operations, he stated that his supervisor, Jens Balkmann, is frequently contacted directly with problems relating to Mr. Klocke's job. Jens must then forward the issue to Dennis, reducing his role to merely a middleman, as he is not familiar with everyone's individual jobs. Distinguishing the user so the correct person is contacted immediately would help to avoid confusion and make the error reporting system more efficient.

To avoid running a job longer than needed, it is suggested that the owner's of each task provide a corresponding end-date with submission. While many jobs are needed for years, for good house keeping practices, a default maximum should be placed on the run time unless it is renewed. For example, if the owner estimates a task should run for the next two years, an automatically generated email will several weeks prior to the prescribed end date to present the owner with the option to renew the job or abort it. If OE receives no response then the job should be assumed unneeded and able to be disabled.

Although JEF is a multi-threaded system, the job configuration should include a feature to input scheduling information to determine how often a job should run (daily, weekly, monthly, disaster relief, etc.) so GES IT and OE are able to note if a job becomes inactive. As described earlier, JEF's event-driven architecture executes jobs based on the triggers caused by specified events, including time. Therefore, it is possible that a job never fires in its entire existence if it is never triggered. This feature should be harnessed by GES IT to help identify and disable jobs that have never been executed [Personal Communication, John Hawkins, November 15, 2006]. The detailed scheduling information recommended is needed to allow technical support to differentiate between a job that has not run because it is inactive versus a job that is for special situations that have not yet occurred. An optional text box feature that allows the user to note any additional implementation details should still be provided to add any other job specific information.

We recommend that Deutsche Bank use these categories as the basic requirements when a job is submitted. Depending on the final architecture of JEF, these requirements should be expanded upon if the information can help to improve the overall process. These recommendations are designed to add value and standardization to the job submission process and prevent users from providing unimportant information.

5.2 Roles of Operations and IT

In addition, as mentioned by Group Technology and Operations Director Gregory Friel, the current job submission process requires the Operations team to undertake technical responsibilities that should involve IT professionals. Specifically, sophisticated users in Operations have undertaken the responsibilities of writing their own SQL queries, a task that should ideally rest in the hands of IT. Identified as the "fundamental problem," the recommended designations are described in the diagram below.

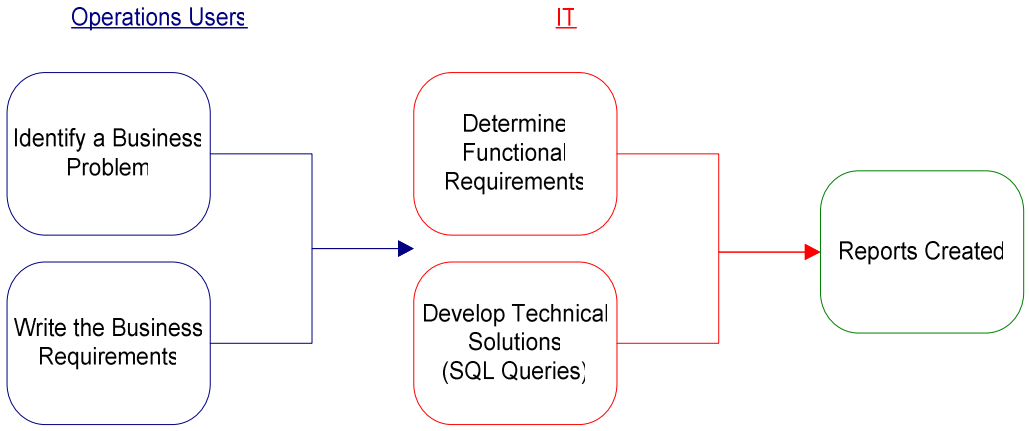


Figure 9: Operations vs. IT Responsibilities

The proposed wizard has been designed to address these concerns by limiting the input that Operations personnel can provide to solely business requirements, at which point IT personnel interpret the information to create technical solutions (in the form of an SQL query).

5.3 Business Objects as a Replacement Interface

Through our research and analysis of Business Objects as a possible option to replace the direct RANbase query submission process, we have formed several recommendations. While the application provides some benefits, the negative aspects have led us to not recommend implementing it company wide at this point. The overwhelming majority of users express unsatisfactory reviews of Business Objects citing that the program:

- Lacks technical support and updated manuals for personal instruction
- Has a slow turnaround time for job delivery in comparison to RANbase queries
- Experiences daily crashes interrupting work deadlines

Business Objects, however, is very user-friendly as individuals do not undertake the technical responsibilities of IT by writing SQL queries, but rather set up jobs by simply "dragging and dropping" tables to suit their business problem. The user also has regulated access to a variety of pre-made tables and templates that are not directly available through RANbase based on their individual qualifications and needs. Also, in the long term scope, Business Objects prevents against company wide restructuring if Deutsche Bank were to switch from the Rolfe and Nolan databases – as the application is not database specific.

The main reason Business Objects has not proven to be a successful tool is the lack of technical support available. Without a resident expert in GES, IT does not currently have the skills or manpower to solve the problems that users face on a daily basis. As mentioned by Ian Ramsay, GES IT lacks the resources to bring this expertise to the business unit without outside support. Through a cooperative effort with other business units that incorporate Business Objects, this source of expertise to provide support and training would become more affordable. Once an adequate level of support, documentation, and formal training regarding Business Objects is available to GES, the program can be utilized effectively as the standard job reporting tool.

5.4 Access Controls

There is no governance system used by the current job submission systems to restrict unqualified users from submitting requests into the system. As discussed by Ian

Ramsay and Gregory Friel, it is too easy for someone to submit a job without knowing what they truly want as an output to most efficiently solve their business problem. To avoid processing unnecessary or poorly written requests in the future, Deutsche Bank must:

- Determine the expertise and need of each department (or individual) pertaining to the job scheduling system
- Assign various access levels
- Store user name, contact information, business division and purpose of job in a database

We recommend that Deutsche Bank study the overall function and experience of a department / team and depending on the results, allow the group access to the appropriate level. Because there are so many users who have taken advantage of the job scheduling features that may not work in these departments, access should be granted on a case by case basis as well. This can be done by the user contacting an SL3 manager via email or phone, explaining their need for access and how they plan to utilize the job scheduling features. An additional control is to have a database of all employees with access so that only the names in the database will be recognized by the system and given access. If an employee tries to submit a job without proper authorization, an error message will appear stating that they do not have authorization to enter this page and to contact the closest SL3 manager. The name, email, location and telephone numbers to numerous managers should be provided within this error message.

The various access levels determine who can go "deeper" into the job scheduling system while restricting others from accessing features they should not handle. To improve the overall security and stability of the system, we recommend categorizing the levels into these three categories:

- Update access
- Write access
- Read-only access

The first two levels – update and write access – allow for the modification and creation of queries and should be restricted to technical personnel with the most SQL experience and need for direct access to job submission. In the opinion of GES IT manager Ian Ramsay, Green PSG and GES IT SL3's write and update access should be allowed, but limited. [Personal Communication, Ian Ramsay, November 28, 2006]. The technical support that

GES IT and CSG provide requires them to have access to the technical details of a job scheduling system; therefore these departments must have full read access to view all information. On the most basic level, "there should be a simple read access that gives an overview of job status available to all – has it run, any errors or warnings, next schedule run time." [Personal Communication, Ian Ramsay, November 28, 2006]. TaskMan users have no indication of a job's status; all they are provided with is an email stating that there is an error.

5.5 JEF- Job Migration

As the migration of job scheduling systems near, there are many steps that must be followed to ensure a smooth transition to JEF. To begin, it is recommended that:

- Each job currently stored in the job scheduling system be studied on a case-by-case basis
- Locate the owner of each job
- Document owner name, business division, contact information and purpose of job. Mark as "TRANSFER"
- Categorize each "TRANSFER" job into one of four categories
- Disable all jobs without known owners

GES IT must study each job on a case by case basis to ensure that only active, necessary and non-duplicate jobs are migrated. Due to the lack of information documenting the purpose and status of jobs in TaskMan and other systems, JEF engineers must locate the owner of each job or dependent job chain in the system to determine if the job is required (via the email contact provided). However, certain jobs (or chains of jobs) continue to run even though the owner has left the company or no longer needs the job but never notified OE. If no response is given within an appropriate time frame, the job should not be transferred to JEF, but rather disabled in TaskMan. *Only* if the owner confirms the necessity of a job, or researchers determine a job is needed by other means, should the job be marked as "Transfer." In addition, the updated owner's name and team must be recorded with the transfer data to prevent future contact problems.

Due to outdated contact information and the high turnover rate in Deutsche Bank, jobs exist where an owner is unknown or vital information is missing. In this case, there are limited options to decide if a job is needed. The most effective and feasible approach is to disable these jobs, and if someone contacts OE stating that they aren't receiving data

from that task and need it, re-enable and designate it to transfer to JEF. This person should be considered the owner and their name, contact information and team name should be listed with the transfer data.

Once the jobs to be transferred are identified, they should be categorized into one of four job classifications to determine when they can be reassigned to JEF. Due to the nature of "independent jobs", it is best for these to be migrated first. They are not dependent on any other jobs, and do not contribute to the completion of any other jobs. "Leaf jobs" are dependent on other jobs, but have no dependencies on them. Once all the jobs a leaf job depends on are migrated to JEF, it should be safe to transfer as well. Another solution, as mentioned in the previous project, is to treat the leaf job as an independent job and transfer it to JEF initially. Only after all the preceding jobs are completed in TaskMan, is the remaining leaf job triggered in JEF. Although this may eliminate some risk in terms of a job not running properly in JEF, errors may occur when trying to link a dependency chain between the two systems. A "complex job" depends on one or more job and has other jobs that depend on it. Because many dependencies exist before and after this job can run, it should only be migrated once all independent and leaf jobs have been transferred. Finally, a "root job" is the initial event that has many jobs dependent on it, but is not dependent on anything itself. Because an error in this job affects everything in the dependency chain, it is extremely important to properly test and analyze this dependency chain before the migration is complete. It is highly suggested that a "root job" continues to run on TaskMan in parallel to JEF until a successful run (or runs) is completed [Piette and Zipkin, 27-28]. For a more in-depth analysis on the different job categories, please refer to the "Methodology of Transferring a TaskMan task to JEF" section of the previous MQP.

5.6 Reverting GIMS

One of the other main issues that we identified is that GIMS has drifted far from its original purpose as an incident management system for support issues. In 2002, GIMS became the main job submission tool for TaskMan. For the TaskMan application specifically, GIMS support tickets are cluttered with a variety of requests related to the

job submission process. According to the research conducted in last year's MQP, GIMS tickets opened regarding TaskMan could be grouped into the categories including:

1. Troubleshooting a job or report
2. Modifying a job's distribution
3. Modifying a job's function
4. Creating a new job
5. Disabling/Enabling a job
6. Deleting a job
7. Creating an FTP dropbox
8. Other

Of these groupings, less than half (205 out of 454) of the tickets were classified as a troubleshooting issue, the original goal of GIMS. The remaining tickets raised over this quarter were related to TaskMan's job submission process. In order to more clearly designate these issues from support issues, we recommend developing a separate job submission/configuration application and reverting GIMS back to solely a support mechanism. This will improve the organization and productivity within Operational Excellence as they receive these tickets each day. Also, with a tool tailored specifically to the job submission process for JEF in the future, more appropriate information will be provided to allow for faster and more effective support in the event of a problem.

5.7 Wizard

In order to govern the job submission process and control how changes are made to existing jobs most effectively, we recommend developing a separate application aside from GIMS. A step-by-step wizard would be the most effective tool to use for this process – as it can be customized to fit several types of requests, yet remain standardized within each step to capture all the information needed for more effective record-keeping and support. Designed to support an iterative process between Operations and OE, the wizard will continue to facilitate the communication needed between the two groups to solve more complex problems. As mentioned previously, the current GIMS tickets used for TaskMan issues do not provide all the necessary information describing a job, are not linked within the TaskMan space, and allow for a high level of customization that is inconsistent from user to user. The goal of the proposed wizard is to provide clear and usable information to support personnel for every job submission and configuration

modification while remaining user-friendly and practical. Included with this report is a prototype of the recommended wizard that should be used for the submission process when JEF replaces the current job scheduling systems.

As mentioned in the results section, there is currently no link connecting information stored in TaskMan about a job to the original GIMS ticket unless entered into the configuration file. In order for the wizard to be an effective tool for JEF, a link to the database of jobs (in the form of a job incident number or job title) must be stored internally in JEF as jobs are submitted. Thus, if further issues develop that need support through GIMS, or configuration changes through the wizard, the job and contact information would be easily referenced in the database.

5.7.1 Documentation

The proposed wizard is a basic and instinctive web application that should be easy for Operations Users and OE to operate. The only client-side requirement is Internet Explorer 6.0 with support for JavaScript 1.3 enabled. The server that hosts the application can use any database program desired but the wizard is currently set up to take advantage of MySQL 5.0.15 along with Perl 5 scripts to build the web interface. All pages of the wizard are designed to fit a monitor with 1024*768 screen resolution. Each of our computers were by default set to this resolution and after searching the intranet we found that that company standards require a 992-pixel wide webpage [“Branding Our World”, 2006]. A 700-pixel height was chosen to utilize as much screen space as possible. Each page was constructed so that no scrolling was involved in order to prevent breaking up the information given by the user [Bollaert, 2006].

There are two different main pages designed for this program. The first one we will look at is the main page designed for the end users, as can be seen in Figure 10. There is a title bar located at the top with the DB logo for verification that the software being accessed is for internal company use. Below the title bar, one can see that a progress meter is present to inform users of how far along they are in the wizard process [Bollaert, 2006]. Figure 11 displays the different indicators on the progress meter. There is a the standard button for future steps, a depressed look for passed steps, a yellow highlighted image for the current step, and a crossed-out image for any steps that will not

be needed in the activity chosen by the user. Referring back to the main page, it is clearly explained to the user that they may cancel at anytime during the submission process, as well as save what they entered so far. The user is also informed that required information will be indicated by a red star [Bollaert, 2006]. The next object a user encounters is a selection of radio buttons which allow them to choose what activity they want to perform, as there are many tasks we want to move away from GIMS for. This will be explained in further detail in the next paragraph. Finally, the bottom of the page displays the navigation buttons which allows the user to move to the desired step. The "Cancel" button sends them back to the main page without saving any of the entered information. The user can also go backwards using the "Back" button after they progress past this first screen and the entered information will be kept. They can move forward with the "Next" button as they must be able to progress with all important information retained. There is also a "Finish" button that is not usable until the end. This button will ensure that all required information is entered and then add the job to a database for OE to review. Finally, the "Save" button will input all available information into a separate database for uncompleted jobs, so they may finish at a later time.

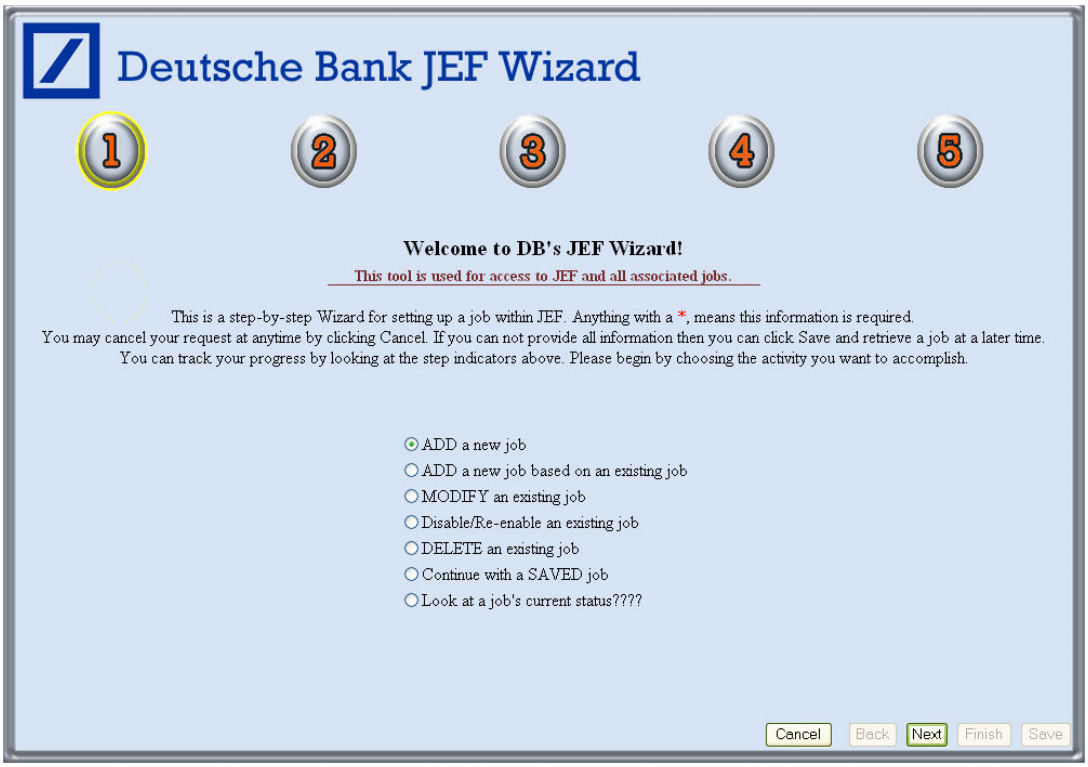


Figure 10: Main Page of the Job Submission Wizard

The user has many options of what they can accomplish by using the wizard. The first activity is to add a new job to the system. An example of Step 2 for adding a new job can be found on Figure 11. The second is to add a new job to JEF but based off of another job. If a user knows of a similar job but wants to change a few parameters, there is no reason to re-enter all the required information. Next, the user can modify a job; if they are the owner, changes can be made to any property. However, if they are not the owner, this person can add themselves to the email list or request to receive a file through FTP. The owner also has the ability to disable or re-enable a job at any time. Deleting a job is another option that should be granted to only the owner and will take the job out of JEF permanently. Since a user has is given the option of saving a job during this process, they are able to recover this job at a later date. Finally the user can view the job submission status. A flow chart for adding a new job and modifying/disabling/re-enabling/deleting an existing job can be found in Appendix G.

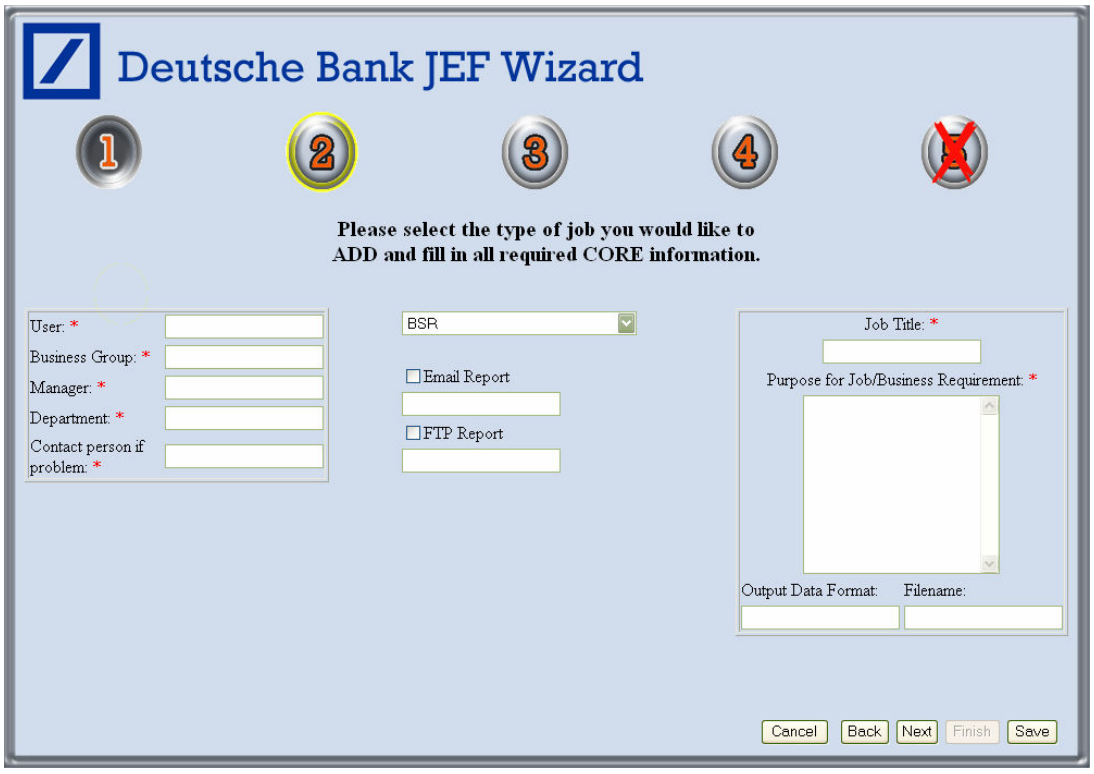


Figure 11: Example of Step 2 for Adding a New Job

Cookies are used to save all information from page to page, as the wizard could be used by multiple employees at the same time. Without the use of cookies, passing parameters from page to page is very insecure and unstable. Although there is a large build-up of cookies (one for every possible input), a more efficient way of storing all data until the end of the job does not exist. This would work for both the Save and the Finish functions.

The second main page developed, which can be viewed in Figure 12, is a job viewer that was built so OE can look over, build any required SQL statements, and approve the submitted job into JEF. As you can see, both main pages have a similar layout. The title bar is located at the top of the page to inform the user of what program they are using. Below that is a simple explanation of how to operate the application. A list of submitted and unapproved jobs that is refreshed every 60 seconds can be found underneath the operation explanation. This can also be constructed so updates to the list are made only when the database is updated (because of time constraints we chose the simple refresh). Once a user clicks on a job, the data fields are populated with the appropriate data. If the job as been already disapproved, a red pound-sign will placed next to the job owner's name. Some fields can be written to, such as the Engineer, while others such as Business Requirements cannot. The engineer can also input a specifically designed SQL script if required for a particular job. "Look Up," "Disapprove" and "Approve" buttons are located at the bottom of a page, where a user can search approve or disapprove a job. The "Look Up" button will open our suggestion of a Job Library, discussed in section 5.9, in a new browser window. OE staff member can double check to make sure there are no similar jobs already in the system. "Disapprove" will open an email to the job owner and will mark the job as disapproved in the database. "Approve" will add the job to JEF's XML database, and mark as approved in the wizard database.

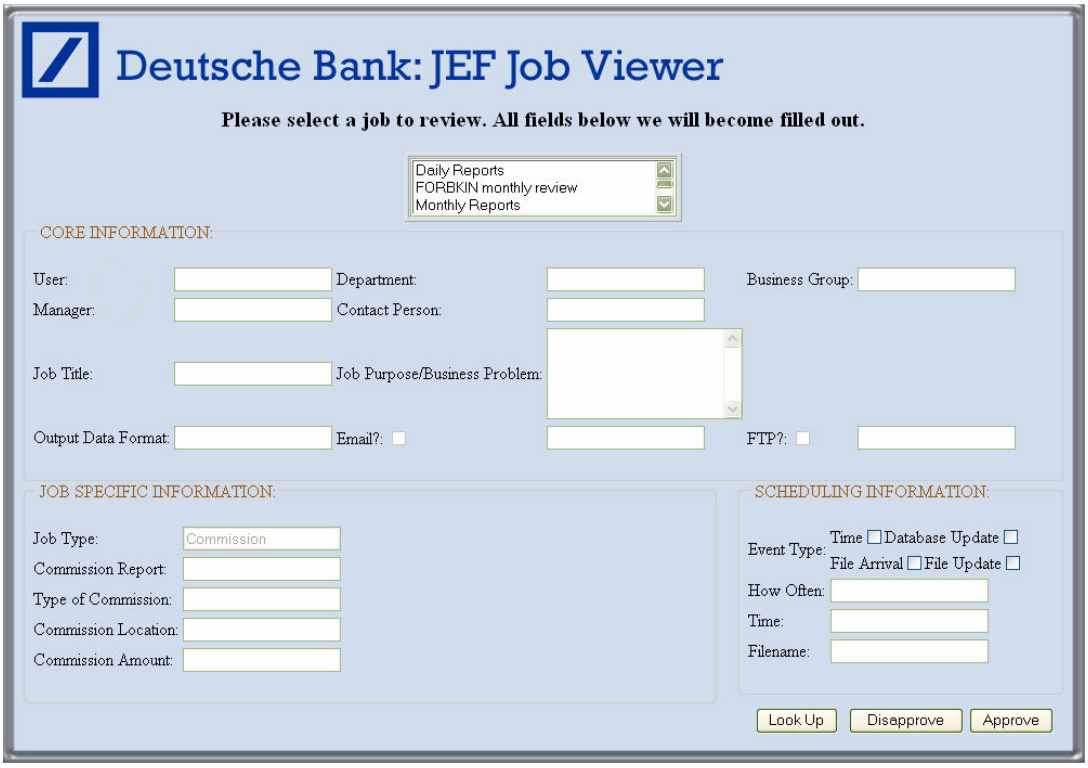


Figure 12: Example of the Job Viewer for OE

5.7.2 Database Layout

A standard set of databases had to be constructed in order for the wizard to perform all desired functions. Although a MySQL database is being used, the basic layout could be moved over to any system. We believe that all the information required on JEF's XML database should be the basis for Submitted_Jobs; the database all submitted jobs are placed in, and Unfinished_Jobs; the database for all the saved jobs. There would however be minor differences from JEF's database.

- Both databases would include a field for the priority of job submission, with levels ranging from Minor to Important to Business Critical with levels in-between.
- Another important field for both databases would be a Job ID. This ID would automatically increment based on the number of jobs entering the system, as well as make it easy to organize for later use.
- Submitted_Jobs should also have a column for Status, so users view the status on there job submission.

The last table we created is a User Access Rights table. This table would be similar to the database already used to access user information in GIMS and other web-based applications but we believe it should hold the following information:

1. Employee ID as a primary key
2. Username, desktop sign-in
3. Name of the user
4. Access rights (Write, Update, Read-Only)
5. Manager
6. Department
7. Department Team
8. Contact Email
9. Contact Phone

This database would allow the standard information of the user to be loaded automatically, saving the user valuable time.

5.8 New Job Submission Process

A major goal of this project was to establish a set of controls to standardize and prevent unnecessary information from entering the job scheduling systems. Applying the wizard application, the new job submission process should follow the general outline provided below.

1. Once again, the User (Operations or otherwise) is presented with a business problem and determines the basic information needed to acquire the solution
2. In order to prevent duplicate jobs from flooding the system, the user should search the proposed library/database of existing jobs to find if a similar problem exists. Jobs should be able to be organized and searched by any field provided (Job title, user, etc) and a brief detailed job description should be included.
3. If a similar job is found that the user can simply model the new job after, the user should access the wizard.
    - To request a copy of the report, user opens the wizard and chooses the "Modify Job Distribution" tab. Since the user is not the original owner of the job, he can only add his contact information and request an emailed report or include a FTP dropbox location. The user must then

click the “Submit” button to automatically compile and forward the information to OE for signoff.

- The recommended searchable database will allow users to use existing jobs as templates for other jobs. Rather then creating a new job from scratch, the user can use the existing job type as a template and make simple mortifications to suit individual business requirements. Once again, after these business requirements are noted in the wizard, the request will be sent to OE to make any changes to the technical requirements (particularly the tables used in the SQL query and job scheduling details).

4. If the desired job does not exist in the searchable database already, the user must describe the specific business problem and desired solution in the “Create New Job” tab of the wizard. Very basic required information must be provided to successfully transmit a new job to OE, such as up-to-date contact information including the user’s business group, the job title, and a detailed job description. Optional fields such as the exact configuration details can be provided initially, or through future communication with OE as the job is developed. With sufficient information provided, the user can submit the job to OE to enter a testing phase on real data in JEF. Also, if the user is submitting the job on behalf of an external client, such contact information should be provided in an optional field. While this information is not guaranteed to be provided, it must be stressed that it will protect against turnover within Deutsche Bank if a direct owner is available.
5. The wizard is designed to streamline the review/approval process for OE as well. By configuring it to directly link the data to the job scheduling system, the manually intensive task of retyping and entering the task configuration information can be avoided. Rather, this process should be automated so OE can focus their expertise and time on performing more thorough “four-eyed” reviews of each job in a consistent manner, and working with the user to solve any problems. It is also recommended to provide OE with the latest RANbase documentation to summarize and explain the tables being used. With a more

solid understanding of how data in RANbase is organized, OE will be able to analyze the business requirements provided to form the best technical solution (ex. SQL query). In addition, OE should access the searchable database of existing jobs to double check that a duplicate job does not already exist.

6. The enhanced role of OE will help limit the number of faulty and duplicate jobs that currently plague the TaskMan system. In addition, such universal controls are easily adaptable to new job scheduling systems, such as JEF, as technology inevitably continues to improve. Finally, after all required information is provided and the job has completed the testing phase in JEF, OE will enter the job into production. The configuration file should include job specific information to act as a reference to the searchable database.
7. A confirmation email with a summary of the job submitted should be automatically generated and sent to the recipients of the new task for recordkeeping. The email should include when the job was submitted, the title and description, contact information of the owner and recipients, and the job schedule (daily, weekly, annually, etc) for the recipients to review.

5.9 Job Library

As mentioned in the previous section, the current job scheduling systems have no way of recognizing which jobs are active. The limited information provided by the users is not stored or utilized, and it is nearly impossible to identify the purpose of the jobs within the system. As a result, duplicate jobs are being submitted into the system and valuable resources are being wasted by the users and support personnel.

To avoid repeating the current problems such as duplicate submission, we suggest:

- Construct a job library utilizing the stored information from job migration and new job submission
- User-friendly interface
- Constant updating of the database
- Allow users to search for a job based on key words

If new job details are required with every job submission, modification, deletion, etc, it would be much easier to track what the job does, who is running it, and who has submitted and used the job. It is recommended that a library of compiled jobs is created

by storing and updating all of the information pertaining to individual jobs submissions in a database. Users could log into the database and search for existing jobs to fit their needs. A searchable database would have many positive impacts on the job submission process including:

- Saving users time as they can simply reference existing jobs rather than creating new jobs.
- Reducing the risk of poorly written and duplicate queries from entering the system
- Improving support capabilities by maintaining a source where all information pertaining to a job is readily accessible.

We recommend that Deutsche Bank stress the importance of providing the needed details when submitting a job, so that the library can be as detailed and useful as possible. Clearer, more standardized submission requirements will save all users and GES IT members' time, and allow the new job scheduling system to run more efficiently.

5.10 Further Improvements for JEF

While much of our analysis and recommendations have focused on improving the controls around the job submission process, there are also some important features within the current job scheduling systems that must be improved upon in JEF, including:

- TaskMan's logging system, and
- TaskMan's error alert system.

Most notably, TaskMan's logging system does not provide adequate information for support to track problems as they occur. Only the start and end times of jobs are recorded in most instances – and the information is recorded in separate text files rather than a workable database. Also, batch files of jobs commonly overwrite previous logs. For these reasons, support has a very difficult time analyzing TaskMan's logs to identify and solve problems. It is nearly impossible to track a jobs performance over time to predict a developing problem or a deteriorating job [Personal Communication, Ian Ramsay, October 25, 2006]. With a more robust and practical logging system, support will be capable of monitoring job performance to prevent or rectify problems. As a workable database, the logging system will allow for many new opportunities to improve the capabilities of support. For example, the data recorded should be sorted each month to

identify the most consuming tasks (by time, CPU, or other variable). By examining the top tasks on this list, IT staff can determine if the performance can be improved. If a job is necessarily large and cannot be optimized further, the item should set aside as an approved job. This process will, over time, prevent poorly performing queries from hindering the system and creating additional problems.

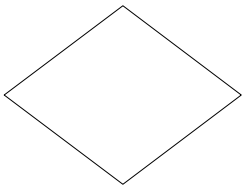
Other important features that should be reevaluated for JEF are the current methods that jobs are tested and alerts are raised. Presently, jobs require minimal, non-standardized testing procedures prior to their implementation in TaskMan. While some users do perform multiple tests prior to submission and general UAT tests or "four-eyed" reviews are conducted by OE, the tests remain inconsistent and inaccurate. User tests are generally on smaller sets of simulated data, and do not reflect accurate job performance. To combat this issue, we recommend developing a "testing phase" for jobs to run in JEF. Harnessing JEF's multi-threaded capabilities, jobs designated as tests should run up to 10-20 times on real data to monitor their performance and record an average completion time. Unexpected performance can be easily identified to prevent poor jobs from moving into production. Also, the average completion time recorded during this testing phase should be used as a basis for delivering alerts once in production. In TaskMan, alerts are sent to users if a job has not been completed after three hours, and then repeated each following hour. JEF, in comparison, is currently able to customize the alerts to each individual job. It is recommended that these alerts be customized according to the job's average runtime. By storing this average in an XML database, and choosing an acceptable variance, alerts can be triggered as a job's execution time drifts beyond a more suitable limit allowing for quicker and more efficient support.

Appendix A

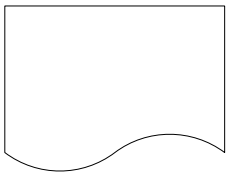
Flow Chart Shapes with Meanings

Starting Point
Terminating Point

Process



Decision Point

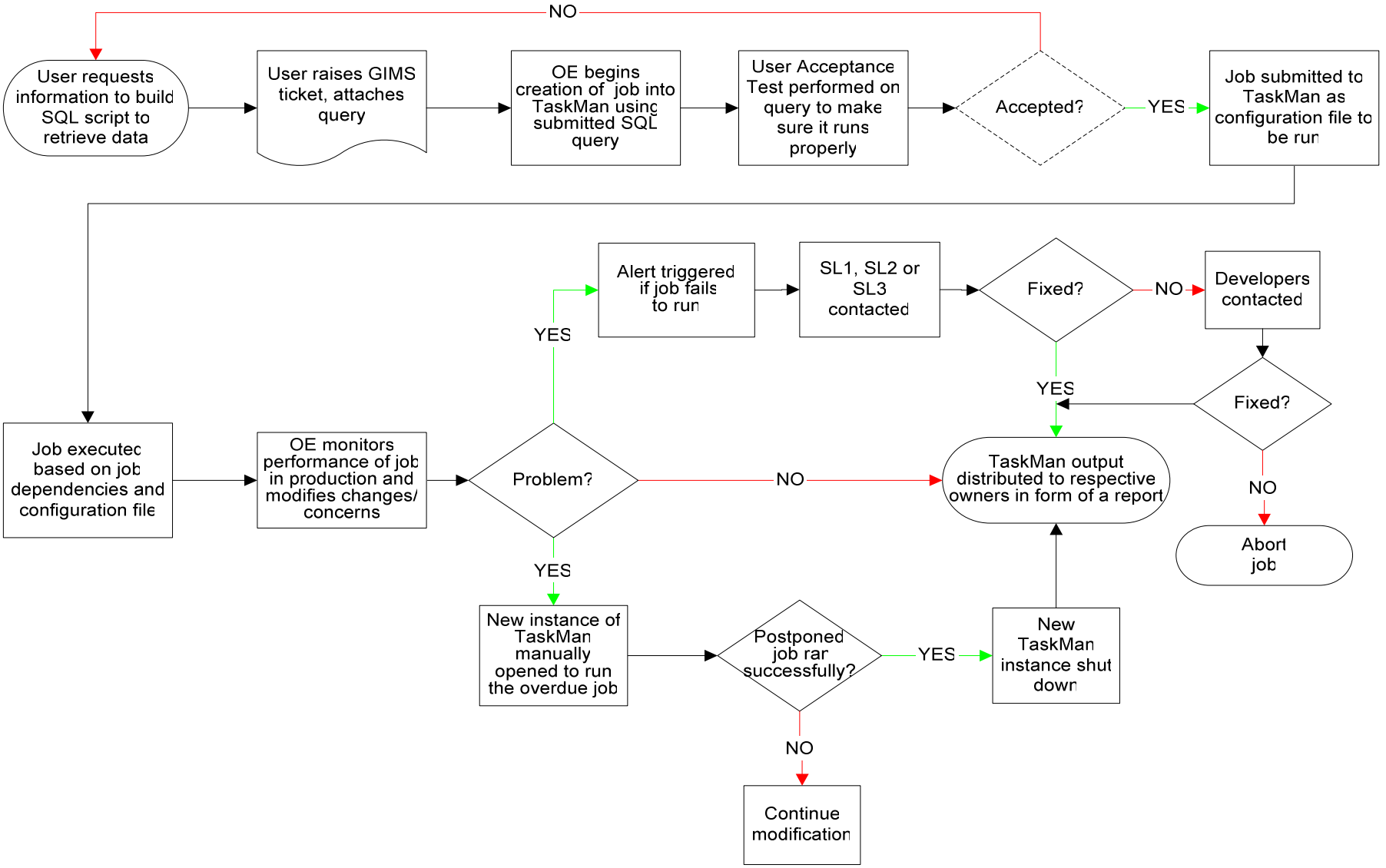


Document

Flow Connector

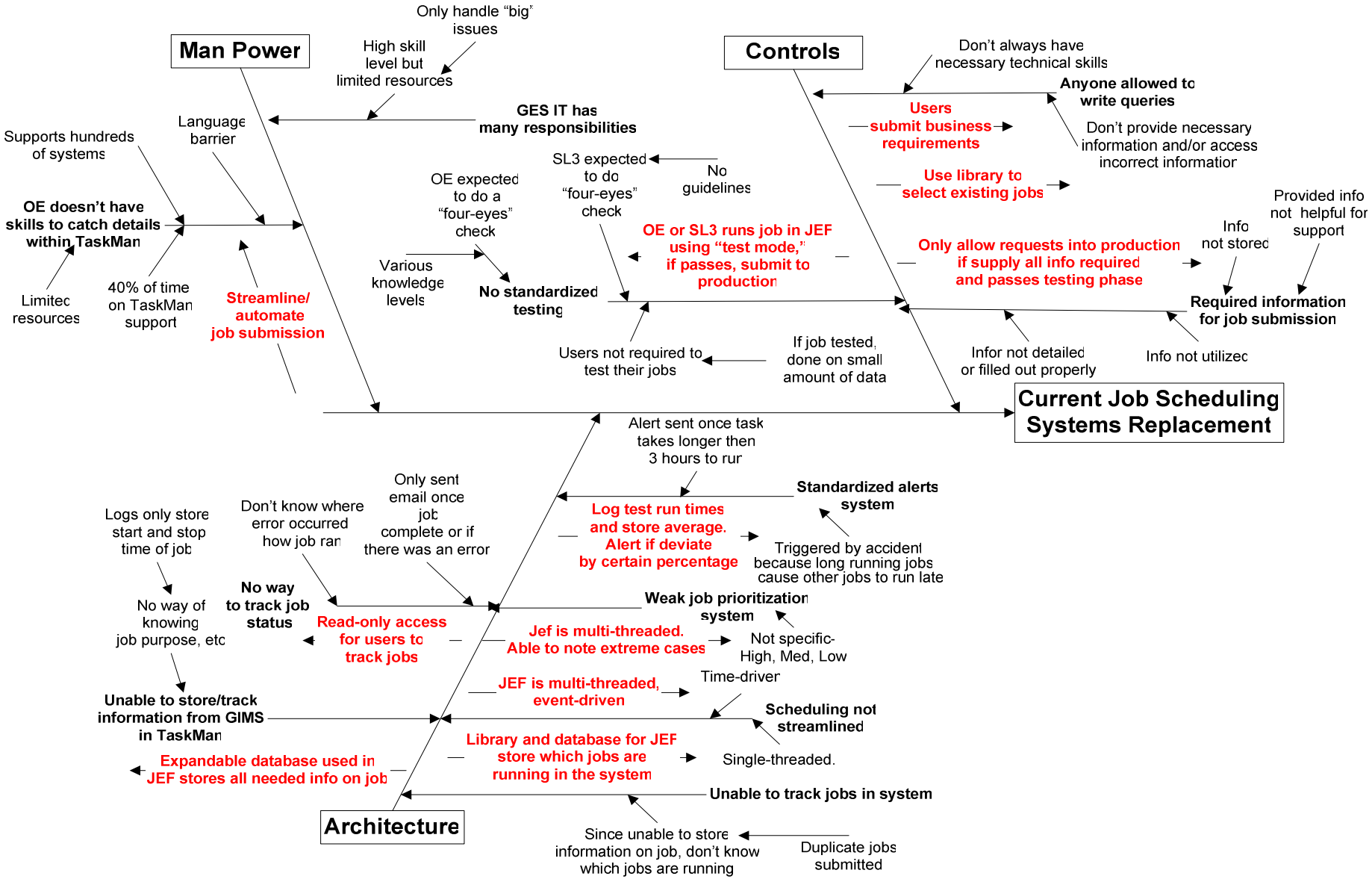
Appendix B

Detailed Job Submission Process Flow Chart



Appendix C

Cause and Effect Diagram of TaskMan Problem



Appendix D

Prioritized list of TaskMan, GIMS, RANtask, & RANbase Issues

The following issues, gathered by personal research and interviews with Operational Excellence, Operations, GES IT, and management are grouped and prioritized to give the reader a sense of all the problems that exist in the current job submission process.

TaskMan Issues

1. The lack of controls around the job submission process makes it too easy for operations/users to create new queries to solve their business problem. Ideally, this technical responsibility should rest in the hands of qualified GES IT members.
2. There are no established controls to:
    a. Regulate levels of access/determine who is qualified to submit jobs to TaskMan (within each department, individuals).
    b. Attain information about a job to understand if it is necessary.
    c. Provide uniform, effective testing of jobs prior to submission.
    d. Standardize information provided in GIMS requests.
    e. Limit duplicate and unnecessary jobs from being submitted.
3. TaskMan's job configuration files only hold limited information (job owner and two text fields). There is no direct link from TaskMan to the GIMS ticket. Therefore, it is difficult to track the progress of a job in TaskMan and understand its purpose and other details for support purposes.
4. The users cannot view or search what jobs are already running. As a result, users are forced to create many duplicate jobs, which strain the system. Also, this will make identifying the jobs to transfer to JEF difficult.
5. There is no standardized testing system established that OE must carry out. Prior to submission, teams are expected to conduct a "sanity-check" for all configuration requests, but this can be inconsistent from person to person.

6. There is no method for users to track the status of a job to see if it is working properly. A flow diagram of the progress and dependencies would be useful from a support point of view.
7. TaskMan's logging system only notes the start and stop time of a job each time it is run, but does not log any other statistics to track the progress of a job. For example, no information is logged about file transfer protocols (FTP).
8. The logging system is not in a workable database form; rather it is divided into separate text files everyday.
9. TaskMan uses a standardized alert system for tasks that have failed to run, or failed to complete within three hours. It would be more efficient to customize the alerts to individual jobs.
10. The architects of TaskMan are no longer at the company. This issue, along with the piecewise development of TaskMan, has made required maintenance much more difficult. Also, the original development and any modifications were not documented.
11. TaskMan has a time-driven architecture, yet jobs still depend on one another. This can lead to upstream delays and errors. Often jobs get lost in the system and are not delivered on time.
12. There is no prioritization of jobs within TaskMan.
13. Job dependencies in TaskMan will make transferring jobs to JEF difficult.
14. OE must log onto the production server of TaskMan to make any changes. This is bad for stability reasons vs. a web-access system.
15. The system requires a manual restart every week due to memory leaks.
16. When batch files are run, they often overwrite the logs already recorded. This makes identifying problems difficult for OE.
17. The TaskControl system is not robust and is not reliable.
18. TaskMan is used globally; however, it does not automatically adjust to time zone changes (daylight savings time), and different holidays between different countries. Requires the system to be rescheduled to keep everyone on track.
19. TaskMan support requires approximately 40-50% of OE resources.
20. Relies on external software, increasing the risk of a problem.

21. No chain of responsibilities for jobs running in system.
22. Older jobs have not been updated properly to note the current owner in Deutsche Bank. As a result, it is difficult to identify the respective owners of jobs to determine if they are needed due to the high employee turnover.
23. OE supports hundreds of applications, and therefore do not have specialized knowledge of TaskMan system. They do not always have the capabilities to determine if amendments are correct without the input of an SL3 manager.
24. TECS, while providing information on the dependencies of jobs, is limited.
25. From development perspective, there has been no major release of TaskMan for two to three years.

GIMS

1. Only limited information can be recorded in GIMS, lack of controls.
2. The GIMS system was not originally designed as a tool for job submission to TaskMan or other servers (rather for support and problem reporting). As a result, it allows for too much customization of tickets by the users.
3. The information provided on the GIMS ticket is not linked internally in TaskMan.
4. The prioritization feature within GIMS allows for the 'opinion' of individual user. Rather, requests should be prioritized based on specific guidelines.
5. GIMS fault categories (to sort tickets according to issues in TaskMan) are not referenced properly by users/do not reflect all possible problems.
6. Requests are often submitted on behalf of external clients, but the client is not listed as the owner in the ticket. This makes support difficult if more information is needed.
7. Some users are unaware what application certain issues should be reported to in GIMS. Some issues have been reported to other applications when in fact they were a TaskMan issue.
8. The jobs created in TaskMan before GIMS became the standard reporting tool have no source to determine information.

RANbase Issues

1. If queries continue to be written directly to RANbase, rather than through an interface such as Business Objects, if Deutsche Bank switches to a different database, all existing jobs will need to be rewritten.
2. Little documentation is provided by Rolfe & Nolan to explain the tables in RANbase. Likewise, there is no documentation on the tables that were created specifically for Deutsche Bank. Users must learn what each table contains through experience.
3. The RANbase direct queries are not user friendly. They require specialized knowledge of SQL script compared to a user interface as used in Business Objects.

RANtask Issues

1. RANtask was not designed as a job scheduling system.
2. RANtask has tendency to start a job (in a batch) and not finish it. The server interprets that the job has run successfully when in fact it has only begun a batch job.

Appendix E

Interview with Ian Ramsay, October 30, 2006

- RANbase
  - 2 terabytes
  - 600+ tables
- Don't spend too much time looking at architecture or systems and tables
  - Focus in on most important/ used

- Don't focus on changes to Oracle
  - "easy" changes already made

- Focus on developing controls*
  - Controls in submission so that good queries are let in while bad queries are addressed
    - Added value
    - Don't need to address why bad, just that they are

- UAT- User Acceptance Test

- At the moment there is nothing that describes the jobs that are running
  - Sentence or descriptive text would help so user and IT can tell what's running.
  - "Bad queries are not so much that the person doesn't write sequel properly, but more that they haven't formulated their business problem"
    - If stated what they want they would be able to search for that job before running it

- OE- frontline support staff
  - Amend jobs as specified on GIMS ticket
    - Don't have skills to know whether they are doing good or bad things
    - Support many business lines and applications
      - Support hundreds of systems, so hard to be specialized
    - In order for them to support new guidelines/controls, must give procedural instructions

- GES IT
  - More specialized
  - 60 servers
  - GES IT should control guidelines/ controls

Interview with Peter Lindsay, November 01, 2006

Initial Questions:

1) Could you describe your current position? What role do you play in regards to SQL query submission process for TaskMan?

Response: "I manage Green PSG, the team within OE that provides level 2/3 support for the GES apps (amongst other things). We would normally only be involved in the deployment of SQL scripts into TaskMan, though this often extends to setting up the job within TaskMan itself, including when it will run, whether it repeats, etc."

2) What is the current process for submitting jobs? What role does OE as a whole play?

Response: "The process for submitting jobs to us is through GIMS. As above, we would normally configure the TaskMan job that is going to run the supplied SQL. Green PSG's main function is to support the production environment. This means monitoring the overall health of the system and following up on specific job failures as well as identifying and chasing up repeat failures."

3) Are any checks/testing procedures for SQL queries followed to regulate what is added to TaskMan? If so, what is in place? If not, what types would be most beneficial?

Response: "Again, GES developers (SL3) would normally build SQL queries for the system to run. There is a UAT instances within TaskMan but I'm not sure that it is used. Our process when given SQL to run is to create the job in production and monitor. We use an internal (to Green PSG) Basic Change Control process to ensure that any changes in production are four-eyes'd and that for any new jobs or configuration changes a record is kept in the GIMS. My personal feeling is that a well defined process with a clear stages and a division of responsibilities between OE and GES would be beneficial."

4) How frequently are queries that are not running properly discovered? How are these identified? What process does OE take to rectify these issues?

Response: "TaskMan writes job failures and late alerts (where configured) to the Windows Event Log these are then alerted to us via Tivoli. Each Tivoli alert is then followed up by the support team. There were 1453 alerts raised resulting from 273 unique events during the second week of October (the last week for which I have stats). The process to rectify these is to investigate each error starting with the error message, late alerts and you go looking for what it is dependent on. Other common types would be authentication errors at the remote system or missing feed files."

5) Does OE log when new jobs are submitted to TaskMan? Does OE note when changes are made to any queries or when problems are encountered?

Response: "GIMS tickets should be created for any new job / configuration change. GIMS would also be raised for any problems if they are reported by an end user or another support group (including GES IT). The resolution for each Tivoli Alert is not automatically recorded. In addition the team keeps up to date our knowledgebase 'The Support Log' with any information gleaned about specific jobs / issues."

6) How often are the users unable to be identified or contacted to resolve the problem?

Response: "I'm not sure of actual numbers but it is regularly the case that we do not have the information to hand that would tell us where the data comes from or who is affected by a problem with a job."

7) What percent of your time do you (or other OE personnel) spend doing support, submitting jobs, etc for TaskMan?

*Response: "Green PSG have 12 heads for GES support who work shifts to cover 24 * 7, about 50% of this effort goes on TaskMan related support work Though the percentage of this time spent on submitting new jobs is low."*

8) Which RANbase tables do you find to be the most used?

Response: "I don't think I'm best placed to answer this question. We can probably do checking within TaskMan to see which tables are referenced by the most jobs (if you think it is important) but it would not be easy to work out number of rows within each table read or updated from the TaskMan side. Indiv_trans and indiv_post are the only ones I know off the top of my head."

9) Since TaskMan is designed with a single-threaded architecture only allowing one job to be run at a time, what procedure does OE follow when there are multiple jobs that need to be run at the same time? How does OE decide how to manage these situations?

Response: "Jobs are scheduled to run at a specific time or based on certain dependencies when they are created and this is not often changed so it tends to be a one time decision. TaskMan is single threaded within each instance but there are approximately 20 instances running concurrently. Tasks tend to be grouped into an instance where there is a relationship between the jobs anyway so the dependencies often dictate the schedule. The time when we most often see issues with this is if delays upstream cause a TaskMan job to run later than usual and it conflicts with a job that normally it would not. The solution is to manually run the job which involves creating another instance of TaskMan and manually triggering the job."

10) How often are tasks delayed or unable to run at the desired time due to other failing/inefficient tasks causing backups in the system?

Response: "This varies wildly and is thus difficult to quantify. It happens when upstream things are delayed usual an R&N batch."

11) Do you think it would be helpful for Managers, if possible, to sign off the queries before they are sent to OE? Do you think a governing process such as this would reduce the amount of repeated/poorly written queries entering the system?

Response: "A governing process could help as long as it promotes intelligent review of jobs and not just a 'rubber stamp'."

12) Is there any more information that you think should be included on TaskMan requests to improve the overall process? For example, a description of the task to be run so OE knows if a duplicate task already exists.

Response: "We attempt build up our knowledge of what a job does as we go. Yes, for new jobs getting this information up front would be helpful. Where does the data come from, what if anything is done to it, where does it go, who cares and when, would all be nice to know."

13) On average, how many queries for new jobs to be run by TaskMan are submitted each week?

Response: "I don't have this information without searching GIMS, as a very rough estimate I would say < 5 new tasks and 10 – 15 significant configuration changes. GIMS would be the place to look."

14) Are there any improvements that you feel will be beneficial to the entire job scheduling system or specifically the OE process of submitting tasks into TaskMan? TaskMan has many faults and few redeeming features

See Discussion

15) From the OE perspective, are there any other issues and problems you see with TaskMan? Are do you see any specific problems that the general user is facing?

See Discussion

Additional Discussion:

- CSG stands for Client Support Group (Level 1)
- Manages Green PSG (Production Service Group?); outsourced.
- Mentioned that OE struggles to maintain the 2,500 jobs that exist on the system, particularly the ones that aren't very well documented. It would help the troubleshooting process if the jobs included: where it is coming from? to who? etc.
- Alert process: After a designated cut off time is reached (ex. 1 hour), alerts are triggered. Then, after every block of time passes (ex. 15 minutes), subsequent alerts are triggered. All alerts are recorded in the Window event Log.
- Pete mentioned that he felt that as a team, OE is good at recording information regarding changes/updates to jobs in support notes.
- The problem now lies that many jobs are predated, and it is difficult to troubleshoot these problems without knowing what they are being used for, to whom, etc.
- Standardizing the way problems are recorded would be helpful.
- The UAT system is currently not used across the board for TaskMan. The reason is that there are so many different applications and systems being used. The "universal" characteristic of UAT cannot be applied to so many applications with different properties.
- He does not feel that there are certain responsibilities that GES IT and OE must hold specifically as long as there are clear divisions, and procedures established. Nevertheless, GES should generally focus on the development of jobs while OE should remain focused on the deployment aspect.
- In order to prevent the problems Ian identified (TaskMan job changes not being recorded through the GIMS process), Pete mentioned that GIMS could afford to be more formalized.
- With less emphasis on TaskMan related support, OE could focus more on critical issues coming in. Presently, he mentioned that OE struggles to stay on top of critical issues, and has very limited time for any medium priority jobs. Issues may be able to be addressed before they become of critical importance.
- Regarding the recommended process of managerial approval, he questioned the value of such a system to users specifically. TaskMan is often "behind the scenes" to users and many do not have much, if any, knowledge about it.
- Mentioned that RANtask was not designed as a job scheduling system.
- Faults and features of TaskMan/RANtask:
    1. Logging from a support point of view is not adequate. It often only includes time/date stamp for job start/end. (Occasionally more)
    2. Batch files can overwrite the log files, making the troubleshooting process difficult. (standard output would be useful)
    3. TaskMan is good at re-running tasks if needed.
    4. RANtask: sometimes does not finish running a batch
    5. With TaskMan, it is impossible to get an overview of if tasks are run. A flow diagram of which jobs are dependent on which, and what has been completed is not available. Possible to suggest a method for implementing this "overview" feature.

6. Right now, in order to make changes, must logon to production server for RANtask/TaskMan. This is bad for stability reasons.
7. Recommended possible solutions to stability problems. (Web-access?)
8. Single-threaded design in problematic as well.

- By having users understand the process involved somewhat, it would make the troubleshooting process easier.
- There are two situations in which an upstream delay occurs:
    1. Job in TaskMan is delayed by another job in TaskMan.
    2. Jobs delayed externally (ex. A file does not exist)
- TaskMan opens a new instance, and after that is run, the original instance won't be run.
- Testing tasks is sometimes not possible
- Estimated that a total of 800-1000 GIMS tickets are opened per month relating to all of GES IT. From this, estimated that about 30-50 deal specifically with TaskMan. However, very often GIMS tickets for the GES department are recorded to the incorrect system (and the underlying cause may relate to TaskMan).
- Limited involvement with RANBASE.
- Queries are very slow due to ever increasing volumes of data.

Interview with Vilas Hirani, November 06, 2006

Initial Questions:

1) Could you describe your current position? What role do you play in regards to SQL query submission process for TaskMan?

Response: "OE Service Manager / Take on Manager role for GES IT applications"

2) What is the current process for submitting jobs? What role does OE as a whole play?

Response: "All configuration jobs are submitted via GIMS by the user. The GIMS is vetted by CSG to ensure it has the correct information required, then passed to PSG, if necessary, for action. PSG create the jobs on their configuration environment, get a colleague to check (4-eyes principle), and then deploy to production. GIMS then closed."

3) Are any checks/testing procedures for SQL queries followed to regulate what is added to TaskMan? If so, what is in place? If not, what types would be most beneficial?

Response: "Would expect the support teams to carry out a sanity check for all configuration requests, but nothing specifically in place for the SQL deployment."

4) How frequently are queries that are not running properly discovered? How are these identified? What process does OE take to rectify these issues?

Response: "Long running query issues would be discovered either at point of creation with sanity check by the analyst, if 'possible freeze' alerts raised by tasks typically running longer than 3hrs, or by accident with long running jobs causing other jobs to run late."

5) Does OE log when new jobs are submitted to TaskMan? Does OE note when changes are made to any queries or when problems are encountered?

Response: "New jobs can only be submitted by Sourceforge, at which point either a GIMS number or an explanation is entered in the free text field before committing."

6) How often are the users unable to be identified or contacted to resolve the problem?

Response: "All new jobs should be created with a user name and the implementer's name, but this may not get added at the time of creation. Also we have a lot of historic jobs that may not have this information. But it shouldn't be impossible to trace a user or department, as the end report will need to be either FTP'd or emailed to somebody."

7) What percent of your time do you (or other OE personnel) spend doing support, submitting jobs, etc for TaskMan?

Response: "TaskMan support takes approximately 40% of the OE resource."

8) Which RANbase tables do you find to be the most used?

Response: "I would not be able to say offhand without further investigation."

9) Since TaskMan is designed with a single-threaded architecture only allowing one job to be run at a time, what procedure does OE follow when there are multiple jobs that need to be run at the same time? How does OE decide how to manage these situations?

Response: "We have 23 single threaded instances of TaskMan running at the same time. The instances are now split up on a regional basis for the most part."

10) How often are tasks delayed or unable to run at the desired time due to other failing/inefficient tasks causing backups in the system?

Response: "Not too much of problem generally, except when we have late running R&N batch times that cause knock on effects to downstream tasks."

11) Do you think it would be helpful for Managers, if possible, to sign off the queries before they are sent to OE? Do you think a governing process such as this would reduce the amount of repeated/poorly written queries entering the system?

Response: "This would definitely be useful; alternatively we could have an initial signoff by shift managers in OE beforehand and then pass to SL3 to vet before implementation."

12) Is there any more information that you think should be included on TaskMan requests to improve the overall process? For example, a description of the task to be run so OE knows if a duplicate task already exists.

Response: "Ideally we should have a standard suite of SQL reports in Business Objects rather than TaskMan. New requests should be catered for by the standard suite which may need to be extended depending on the request."

13) On average, how many queries for new jobs to be run by TaskMan are submitted each week?

Response: "Green PSG manager (Pete Lindsay) should be able to give you an approximate number, alternatively you should be able to run a TaskMan GIMS query to elicit this information (though it will require manual breakdown)."

14) Are there any improvements that you feel will be beneficial to the entire job scheduling system or specifically the OE process of submitting tasks into TaskMan?

Response: "Green PSG manager (Pete Lindsay) may have some thoughts around this."

15) From the OE perspective, are there any other issues and problems you see with TaskMan? Are do you see any specific problems that the general user is facing?

Response: "Green PSG manager (Pete Lindsay) may have some thoughts around this."

Further Discussion:

Checks currently done:

- 4-eyes principle:
  - Helps but it's not all encompassing
  - PSG may not be geared up for SQL an may not be qualified to check all details
- "Sanity check":
  - Basic check of query when submitted into the system
  - Hit or miss with skills
  - Better then nothing
    - ***could run query or monitor first run and record time and use for future comparisons

Restart:

- Manual reboot every Monday to clear out memory leaks
  - "good housekeeping rule"
- Every 24 hours, each running instance is restarted by another instance
  - Task Control monitors other instances if an instance is not working properly
    - Not robust and sometimes won't restart

Who should be allowed to use TaskMan?

- User should provide sequel so either the query is vetted through SL3 shift leaders or give a blanket statement with requirements
  - Established set of requirements
  - SL1 and SL2 have inconsistent skills to review

Inefficiencies:

- Check before submittal of query
- Similar queries are being run
  - Logically group queries
  - "Each query is treated as a new request"
    - Should have better knowledge of what is running

Business Objects:

- U.S. less familiar
- Issues with using
  - Takes a while for request to submit and turn around time longer then with TaskMan

Interview with John Hawkins, November 15, 2006

Initial Questions:

1. How do you identify active/inactive and needed jobs?

Response: Haven't done anything to identify active jobs- Unable to do this because no way to identify what is being run in TaskMan. Haven't put in any procedures in JEF because TaskMan doesn't include clients, etc so have to submit each job on a case-by-case basis

*It is possible to set up a job in JEF that is never fired because JEF is event-driven so it's possible that job will never fire in its entire existence. Able to note job is dead in JEF because able to see if it has ever run. *Should have some note if a job is supposed to run only under certain conditions (once a year or only under disaster situations)*

2. Is TECS being utilized and what information is it supplying?

Response: Used on an as-needed basis such as when an overview of the system is given. Don't need on a weekly basis and used for whole scale view of TaskMan environment: ie- there exists an issue with STP in back-end (insecure with transfer or passwords), use TECS to see which jobs are being used in conjunction so that can see if a change needs to be made to a job and if all other jobs are affected.

3. Is there a way to see job dependencies? How so?

Response: All information is in the control files. TECS gives Gantt chart view, however, no runtime dependencies

4. How are dependency issues dealt with?

Response: In JEF, jobs sleep until needed job is complete. Want to map events as they occur so there is a central mapping so able to look at event to see what other events should be triggered.

5. What is the process of transferring jobs currently in the system to JEF?

Response: Only jobs transferred to JEF are those that TaskMan cannot satisfy (SFTP and transformation jobs) However, same process of raising a GIMS ticket still holds.

6. What are the 8 types of jobs currently programmed into JEF?

Response: 8 different job AREAS. They are developed as requirements come in. Developers are finding that there is a refracturing exercise because there is only one type of job- "pick up, transform, and deliver." Will expand but not enumerated yet

Information needed:

1. *Core*
2. *Job specific*
3. *Scheduling*

All method data JEF runs on is stored in XML database and can be extended to keep who owns job, avg run time, number of failures, etc

Follow up Questions:

1) I don't know how well you know Business Objects but is there a way to interface with it so that maybe in the future instead of sending SQL directly RANbase we can use Business Objects to access the database incase DB changes the underlying database?

Response: Business Objects. I haven't look at the communication options on Bus. Obj. yet ... we've just taken delivery of the latest version (XI) and whenever we get time this will definitely be looked at. Business Objects isn't great, but it's far better than any system we could develop in house and so it makes sense to keep using it. Currently our communication with Business Objects from taskman is often simply though the use of "trigger files" written by TaskMan that BO polls for. I can see some benefit for using a BO query within JEF as opposed to direct SQL, but BO is very much dedicated at human readable reports ... I'm not sure the overhead that it introduces would be a good trade-off against the abstraction the universe would provide.

2) Do you think it would be a viable and a good choice to make it so the Wizard automatically builds the job by itself instead of having OE have to build it themselves, hopefully in effect saving time??

Response: I think in theory it sounds good, but in practice it would require the user to have too much knowledge. We would aim to provide a wizard construction facility for the support team to provide them a quick and constrained way for building jobs but the extra knowledge this would still require (sftp key types, directory names, etc) is not something that the typical ops user would be comfortable with. In addition, I do still like the idea of a 4-eyes approach to rationalise the requests ... we had an incident in the last week where an ops user requested an extra 22 reports at the end of one of our early morning batch runs in Ransys. Each report was incorrectly specified and took over an hour to complete ... this meant that our batch end (which should have finished early morning) continued throughout the day. Because neither IT or support were involved it was 2pm-3pm before we realized what had happened ... and by that time we were already in a major outage situation.

3) I wanted to double check how alerts were being implemented in JEF...is it user defined, does OE control that, is it set by job type, etc?

Response: There are certain alerts that are configured for all executing jobs ... late alerts, fatal failures, etc. There are others that are specific to the job type being

executed ... file not present, parsing error, transformation error. We try to ensure the alerts are specified at the job type (i.e. class) not the job instance level. This means we can document for support at a template level (i.e. pickup-transform-deliver) rather than for each job (e.g. Marshall-wace reports delivery). The are no alerts a user can specify - only notification of points within a process (i.e. email of a report, etc).

4) Can you take a look at my visio and tell me if this is the correct differences in data structures between TaskMan and JEF?

Response: Yes - that looks right. As above, we want the view that the user gets to closely resemble the business process they are requesting a solution for.

Interview with Dennis Klocke, November 21, 2006

1) How often do you submit queries?

   Submits GIMS tickets through TaskMan 1-3 times per month
   A lot of RANtask issues handled by Green PSG 2-4 times per month

2) Are you responsible for testing the query before you submit it?

   He tests queries on his own. Testing depends on what kind of ticket he is raising, have different specifications such as run a query for 5 days on a UAT and if it's successful, the job is sent to production

3) Have you ever had an issue were a job you have run been aborted?

   Yes. A big problem is that RANbase queries very slow especially if running multiple days of data.

   If try to query London data from Germany, very slow
   Possible reason is that RANbase tables are old and indexes don't work correctly

4) How are you notified when something goes wrong with the jobs you have run? How long does it take for the issue to be resolved?

   Email is sent by Green PSG, never call regarding performance issues

5) What training/experience do you have in regards to writing queries and working with databases?

   Has formal training because of his degree as a software engineer. Most of his colleagues must learn on the job/as they go because there is no formalized SQL training provided by DB. Many use Query Builder, but in the interest to save money, QB was taken off of computers and replaced by Business Objects.

   Colleagues have just started using B.O. and "hate it"

6) Are there any personal problems that you noticed while writing/submitting queries?

   Communication not good with Green PSG. A lot of miscommunication where emails are misunderstood and GPSG asks incorrect person for help. (incorrect user)
   Green PSG slower then in the past. Workflow process should be reviewed.
   People setting up jobs not always on the same level

7) What types of information do you normally supply while submitting a GIMS ticket?

Brief job description and user

8) Do you think that supplying a more detailed job description on the GIMS ticket would be beneficial? (So OE is able to inform you of problems, and help OE find duplicates)

Yes, for things such as the purpose of the job, etc, to compile information for a searchable database. Also, should star/ distinguish who the user is and who should be contacted if there is a problem because contacting the wrong people

9) Would it be useful to see what other jobs are running so you can request access to that information instead of having to create a new query?

Yes, very interested.

10) Are there any improvements that you feel will be beneficial to the entire system or for your process of submission?

Store who requested the GIMS ticket
Store name of the reconciliation file
Explain what the job is good for in a few sentences

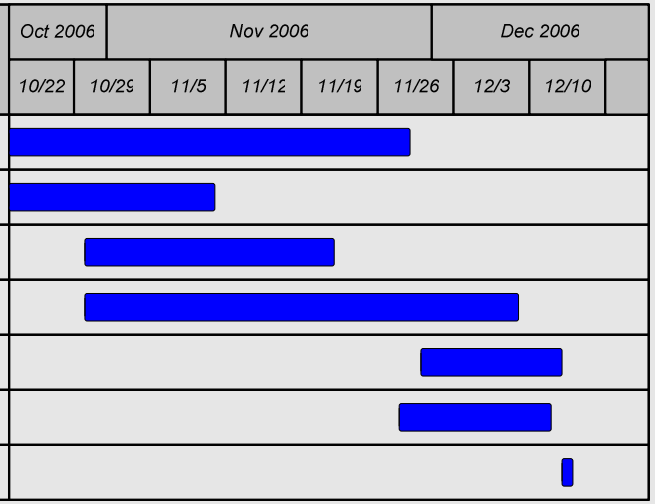
11) Who should be able to send jobs to TaskMan, etc?

Hinted that his team must be able to submit jobs, but there should be some governance throughout the company. Many people come to his team asking if a query is ok to submit.

Appendix F

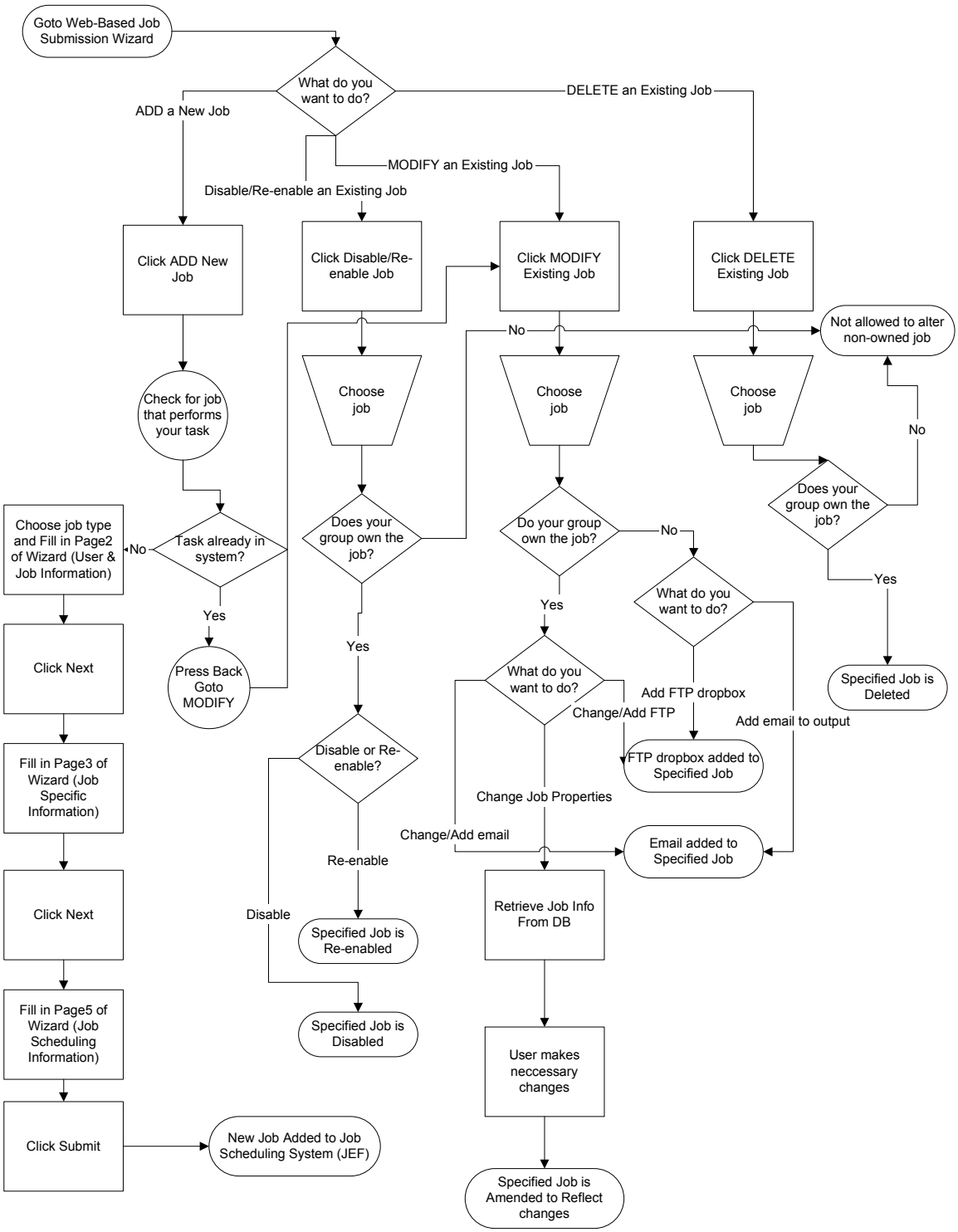
B-Term Deutsche Bank MQP Schedule

ID	Task Name	Start	Finish	Duration
1	Interview Users, Technical Support, Managers	10/23/2006	11/28/2006	27d
2	Study TaskMan, GIMS, JEF, RanBase	10/23/2006	11/10/2006	15d
3	Analyze findings from studying servers	10/30/2006	11/21/2006	17d
4	Write Major Qualifying Project Report	10/30/2006	12/8/2006	30d
5	Construct and Practice Presentation	11/30/2006	12/12/2006	9d
6	Review and Modify Report	11/28/2006	12/11/2006	10d
7	Final Presentation	12/13/2006	12/13/2006	1d



Appendix G

Detailed Job Submission Wizard Process Flow Chart



Appendix H

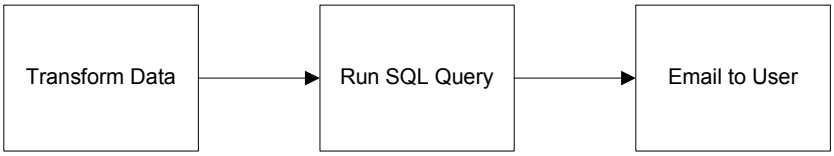
Proposed New Job Submission Process (With Wizard)

1. Users determines problem and basics for acquiring solution
2. User searches Library/Database of current jobs to find similar problem and solution
    a. If found & User only needs to request copy
        i. User opens Wizard
        ii. User choose Modify Job
        iii. Since User did not create job, user can only add email name or include FTP drop box and file name
        iv. Users clicks Submit
    b. If found & User needs to modify properties, etc
        i. User copies query or job properties
        ii. User modifies query/properties as needed
        iii. User performs simple TEST on sample data (real data if possible)
        iv. User opens Wizard
        v. User choose Create New Job
        vi. User follows Wizard steps, including all necessary information
        vii. User clicks Submit
    c. If not found
        i. User creates SQL/job to solve problem
        ii. User performs simple TEST on sample data (real data if possible)
        iii. User opens Wizard
        iv. User choose Create New Job
        v. User follows Wizard steps, including all necessary information
        vi. User clicks Submit
3. Job is presented to OE
4. OE does checks
    a. 4-eyed
        i. Use own knowledge of SQL, JEF/TaskMan
        ii. Use RANbase documentation (hopefully)
        iii. Use searchable Library/Database and double check for duplicate job
5. OE signs off within system
6. OE clicks Approve
7. Job is submitted to JEF automatically
    a. JEF performs 20 TESTS on RANbase tables to find average time of access
8. OE inputs job into TaskMan
9. After submission, User is emailed that job is submitted

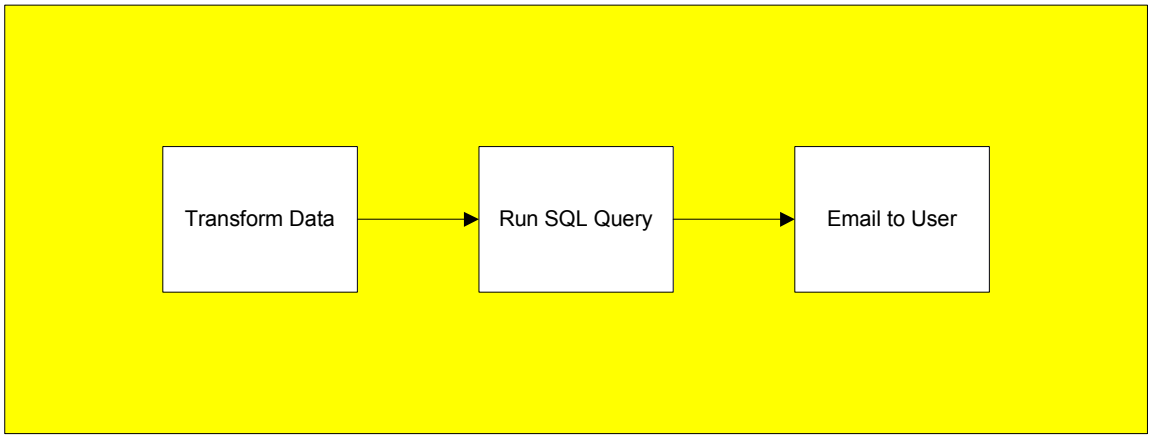
Appendix I

Comparison of TaskMan and JEF Job Data Structure

TaskMan Data Structure - 3 separate jobs



JEF Data Structure - 1 task with 3 separate jobs



References

Bollaert, Jodi. “Crafting a Wizard” IBM developerWorks. 1 Sept 2001. IBM. 29 Nov 2006 <http://www-128.ibm.com/developerworks/library/us-wizard/>

“Branding our World – Corporate Design Guidelines.” Deutsche Bank Intranet. Oct 2006. Deutsche Bank. 29 Nov 2006 <https://brandportal.intranet.db.com/en/download/E-Style_Guide_Intranet_final.pdf>

"CI Operating Committee." Deutsche Bank Intranet. 2006. Deutsche Bank. 20 Nov 2006 <http://cib.db.com/db/en/en_gb/dbNetwork/corporate_information/management_structure/operating_committees/CI.html>.

“DB at a Glance” Deutsche Bank Intranet. 2006. Deutsche Bank. April 2006 <http://www.db.com/en/downloads/company/DB_at_a_glance.pdf>

"Deutsche Bank appoints Peter McLady as Head of Global Exchange Services." Press. May 09, 2006. Deutsche Bank. 02 Nov 2006 <http://www.db.com/presse/en/content/presse_informationen_2005_2691.htm?month=8>.

“Enterprise service bus” Wikipedia: The Free Encyclopedia. 2 November 2006 <http://en.wikipedia.org/wiki/Enterprise_service_bus>

"Functional Committees." Deutsche Bank Intranet. 2006. Deutsche Bank. 5 Nov 2006 <http://cib.db.com/db/en/en_gb/dbNetwork/corporate_information/management_structure/functional_committees.html>.

"Futures Fundamentals Tutorial." Investopedia.com. 2006. 23 October 2006 <http://investopedia.com/university/futures/>.

“OBI IT GIMS” Deutsche Bank Intranet. 2006. <https://grtsupport.risk.db.com/mkdprods/index.asp>.

"Global Exchange Services." July 04, 2006. Deutsche Bank Global Exchange Services. 03 Nov 2006 <http://gm-gcf-lemg.cio.gto.intrane...6/US_GES%20Ops%20-%20Showcase.ppt >.

Harwood, Joy, Richard Heifner, Keith Coble, and Janet Perry. "Managing Risk in Farming: Concepts, Research and Analysis." Agricultural Economics Report. AER774(1999): 36-39.

"Market Overview." Hedge Street. 2005. Hedge Street. 1 Dec 2006 <http://www.hedgestreet.com/abouthedgestreet/marketsnapshot.html

Moschini, Giancarlo , and Harvey Lapan. "The Hedging Role of Options and Futures Under Joint Price, Basis, and Production Risk." International Economic Review 46(1995): 1025-1049.

"Options Basics Tutorial." Investopedia.com. 2002. 20 Nov 2006 <http://investopedia.com/university/options/>.

“Organizational Structure” Deutsche Bank. 7 Dec 2006. Deutsche Bank. 5 Nov 2006 <http://www.db.com/en/content/company/organizational_structure.htm>

"Our Company." Deutsche Bank Intranet. 2006. Deutsche Bank. 5 Nov 2006 <http://www.db.com/en/content/company/our_company.htm?ghpnavigation=ENG_Our_Company>.

Piette, Matthew, and Justin Zipkin. Deutche Bank's TaskMan System and the Job Execution Framework. MQP. 2005.

“Service-oriented architecture” Wikipedia: The Free Encyclopedia. 2 November 2006 <http://en.wikipedia.org/wiki/Service-oriented_architecture>.

Teweles, Richard, and Franks Jones. The Futures Game: Who Wins, Who Loses and Why. 3rd. McGraw-Hill, 1998.

"The Basics on Futures & Options." Kansas City Board of Trade. 2006. Kansas City Board of Trade. 28 Nov 2006 <http://www.kcbt.com/trading_basics.html