MQP – DMS – 1201

A Cross-Platform Database Infrastructure Monitoring Dashboard for The Hanover Insurance Group

---

Major Qualifying Project
for Management Information Systems

By:

Stephen Berselli, Kun Peng and Hao Zhu

12/12/12

Sponsored by The Hanover Insurance Group
Submitted to the Faculty of
Worcester Polytechnic Institute
In Partial Fulfilment of the Requirements for
The Bachelor of Science Degree

Abstract

This MIS MQP developed a cross-platform database infrastructure monitoring dashboard for The Hanover Insurance Group. Both technical details and executive level insight are comprehensively represented. Critical health and performance information from four database platforms that support over 2,000 databases is integrated with independent monitoring tools in one centralized dashboard. Increasing database monitoring efficiency and cross-organizational communication, the dashboard is a lean implementation tool designed to transform Hanover's reactive monitoring approach into a proactive one.

Table of Contents

Table of Figures

Authorship

This MIS MQP was conducted and written by Stephen Berselli, Kun Peng and Hao Zhu.

Stephen Berselli managed our external project communication and served as our WPI advisor/project sponsor liaison. He was responsible for writing the abstract, authorship, acknowledgements, executive summary, background of Hanover, methodology, and recommendations sections.

Kun Peng managed our internal project communication and led the evaluation of Hanover's current system. Further, he was in charge of designing our dashboard with the aid of Henry Farineau and managing its implementation process/SiteScope integration alongside Paul Coughlin. Kun was responsible for writing the project problem statement and literature reviews.

Hao Zhu managed the technical aspect of our project with a focus on system analysis. He was responsible for writing the project goals, literature reviews, requirements inclusive of their implementation, statistical model and survey feedback analysis.

Title	Author(s)
Executive Summary	Stephen
Introduction	Kun, Hao
Background and Literature	All
Current System	Kun
Methodology	Stephen
Design of the To-Be System	All
Implementation of Prototype	Kun, Hao
Recommendations	Stephen, Hao

Acknowledgements

This project has been both possible and successful due to the direct contributions of many individuals. We are grateful for the outstanding level of resourcefulness and mentorship provided by each of the following individuals:

- To Karin Winsky for her continuous efforts to advocate for WPI students and promote MQP project opportunities at Hanover.
- To Hanover's DBA team for extending to us their time, resources and knowledge in the design and implementation of our project. Specifically: John Aragi, Paul Coughlin, Henry Farineau, Don Postma and Keith Van Riper.
- To our Hanover project mentor, Brandon Willis, for his commitment to advising our project and providing any necessary resources to be successful.
- And finally to our WPI project advisor Prof. Diane Strong for setting high expectations of us, providing forward-thinking project insight and most importantly, remaining grounded to continuously drive our team success.

As a student team, we would like to directly recognize you and your efforts to help us excel in this project. We could not have achieved such results without you.

Executive Summary

Strength in technology is achieved by intelligent and centralized systems that enhance cross-organizational communication. The Hanover Insurance Groups IT infrastructure is composed of many individually intelligent tools designed to maximize channels of communication. This MIS MQP developed a cross-platform database infrastructure monitoring dashboard that centralized many of those tools and significantly enhanced Hanover's cross-organizational communication.

Representing both technical details and executive level insight, the dashboard is an intelligent and centralized system that has clearly strengthened Hanover's IT infrastructure. Utilizing trended database health and performance information, the dashboard is a catalyst for transforming Hanover's reactive database monitoring practices into proactive ones. Communicating forecasted system errors before they occur, the dashboard directly increases Hanover's agent-based business value proposition by enabling greater uptime.

Designed and developed using an incremental methodology, the dashboard integrates with the business seamlessly. In addition, its key features can be largely attributed to our partnership with Hanover Technology Group (HTG) employees. A clear sense of the dashboard's usefulness and ease of use can be drawn from its centralized critical operating information from the priority databases of four IT platforms and the feedback data from our user survey. Dashboard usage by both Hanover general employees (i.e., DBAs) and Executives make this IT implementation both successful now and capable of evolving with Hanover's future business needs.

Chapter 1: Introduction

Problem Statement

The Hanover Insurance Group (Hanover) utilizes in excess of 2,000 separate databases that operate over four different platforms. A preliminary analysis of the Hanover Technology Groups' (HTG) current database monitoring efforts surfaced the following problems:

- Organization of database monitoring tools: Hanover's Database Administration (DBA) Team has separate database monitoring tools for each platform.
- Monitoring metrics: These different monitoring tools lack standardized measuring metrics. While some provide high-level performance overviews, others present detailed transactional metrics.
- Monitoring reports: The DBA Team receives automated email, pager and phone call performance alerts based on the severity of database issues that arise. The present automation standards for these alerts cause a mass overload of (at least) email alerts that do not receive proper or timely attention from administrators.

The accurate and efficient monitoring of this network is necessary to enable a proactive approach to uninterrupted business operations and system intelligence. Such monitoring efforts are strategic to enhancing end-user response time, a core competitive competency of Hanover and an insurance industry business value.

A centralized cross-platform database monitoring solution is needed to streamline current monitoring efforts, increase the HTG's DBA Teams' effectiveness and enhance the end-user response time. Brandon Willis, Director of HTG Operations, has sponsored

this MQP to evaluate the current monitoring efforts, conduct a systematic gap analysis and initiate the implementation of an industry best-practice based strategy (Figure 1).

Project Sponsor: Brandon Willis, Director of HTG Operations

Business Need: This project has been initiated to improve the effectiveness of the monitoring techniques used for thousands of databases across multiple platforms.

Business Requirements: We need to evaluate the current system, perform a gap analysis and develop optimized strategies to adequately monitor four different database platforms.

Business Value: Using the improved monitoring strategies, the DBA team will be able to track database availability and other relevant metrics from a high level more effectively and efficiently.

Special Issues or Constraints: We need to present our proposal by mid-October and present our prototype by December.

Figure 1: System Request

Request for Proposal

Figure 1 shows the system request for this project. The project sponsor is Brandon Willis and the business needs are to centralize cross-platform database monitoring and improve the overall effectiveness of the current monitoring techniques. As stated in the system request, the value of the solution is improved database monitoring through a tool that DBA teams can use to track database metrics more effectively and efficiently.

Project Goals

This MQP will focus on creating and implementing a centralized cross-platform database infrastructure monitoring solution to address the challenges associated with managing multiple database systems separately. We anticipate that this solution will encompass a monitoring dashboard featuring a graphical user interface (GUI) design similar to the current home grown Infrastructure Dashboard in Hanover's Business Service Management (BSM). This dashboard will present a single and consolidated view of database health and performance, thus strengthening or replacing the respective platform monitoring tools. The view coverage must transcend several layers of the application, database and operation system stack to ensure an accurate and comprehensive presentation of information. The solution should also facilitate the company-specific user workflows driving issue prevention and resolution in addition to performance optimization.

The following items will serve as project goals:

1. Gap analysis: Our group will compare the current database monitoring processes of Hanover with industry best practices and conduct a gap analysis.
2. Cost-benefit analysis: Our group will identify the financial risks associated with each change option.
3. System proposal: Our group will propose and demonstrate a systematic solution to the problem encountered by Hanover.
4. Migration plan: Our group will specify what actions should be taken when and by whom during the migration process.

5. Prototyping and/or demonstration: At the end of this project, we will present a prototype and/or product demonstration to Hanover depending on the type of solution recommended (home-grown or third-party).

Chapter 2: Background and Literature

The objective of this chapter is to describe Hanover as a company within its respective industry, provide background on each of the four database platforms and introduce the cross-platform infrastructure database monitoring concept.

The Hanover Insurance Group

Headquartered in Worcester, MA, Hanover is a leading holding company that offers a distinctive range of property and casualty products and services through its select team of global independent agents. Founded in 1852 as The Hanover Fire Insurance Company in Manhattan, NY, the company is now recognized as an industry top 25 leader. As a global company, Hanover is known best today for its devoted commitment to its customer base on a local level. Further, Hanover's historical track record proves that it is both strong-willed and persevering. With a presence exceeding 5,000 professional and dedicated team members, Hanover is well positioned for continued growth and market leadership.

As a growing enterprise, the company faces a challenge presented by its organizational data storage and monitoring strategy. The company's database platforms, comprised of Oracle, DB2 for LUW, DB2 for z/OS and Microsoft SQL Server, currently support in excess of 2,000 databases.

Single Platform Database Monitoring

The goal of database monitoring is to examine how well database servers are performing. Effective monitoring methods typically include taking periodic snapshots of current performance in different databases and gathering data continuously to help detect processes that are causing problems and track performance trends. In the following section, we describe the results of our literature review on single platform database

monitoring on four different database platforms: Oracle, SQL Server, DB2 for Z/OS and DB2 for LUW (Microsoft).

Oracle Database

Oracle database is an object-relational database management system produced and marketed by the Oracle Corporation. In 2011, the market share for Oracle database is 48.8% based on total software revenue which is more than the next four competitors combined (Morningstar, 2012). The Oracle database is typically used to store large data in tabular form since it provides the best performance for database, data security, good database administration, storage cost and redundant service for high availability (Hubpages).

The Oracle database has built-in tools which can monitor the health and performance of the databases proactively. Oracle Enterprise Manager is one of these tools. It is designed to be a single management console that can manage all the level of databases. "It can monitor the vital metrics related to database health, analyze the workload running against the database, and automatically identify any issues that need attention as an database administrator" (Oracle). Identified issues are presented in the Oracle Enterprise Manager as alerts and performance metrics. The issues and relevant metrics can be sent out as emails to database administrators. Oracle Enterprise Manager also has a GUI home page that allows DBAs to monitor the health of Oracle databases visually. "The general section provides a quick view of the database, such as whether the database is up or down, the time the database was last started, instance name and host name" (Oracle).

Moreover, the Oracle database monitoring tool allows DBAs to set critical and warning metric threshold values. "These values are boundary values to indicate that the

system is in an undesirable state when crossed" (Oracle). Alerts will be sent out to notify DBAs when particular metric thresholds are crossed or when a certain event occurs in the database.

Apart from alerts and diagnosis, Oracle database includes a self-diagnostic engine which is known as the Automatic Database Diagnostic Monitor (ADDM). This tool can help DBAs to diagnose database performance and determine if there are any potential issues in the database. Using this tool, resource bottlenecks, poor connection issues and lock contention can all be identified in advance of end-user interference. Oracle database also collects periodic snapshots of the database state and workload in order to facilitate the diagnosis. These periodic snapshots are then stored in a database as historical information (Oracle).

In addition, ADDM has an Automatic Workload Repository (AWR) report system which is Oracle's mechanism for gathering and preserving statistics useful for performance analysis. DBAs can generate an AWR report to review the statistics captured across a period of time for an instance or take snapshots and statistics of different instances and generate a report for all instances within the cluster. There are many useful metrics involved in one AWR report that can help DBAs to better assess the current health and performance status of their databases (Gopalakrishnan, 2011).

SQL Server Database

Microsoft SQL Server is a relational database management system developed by Microsoft from Sybase SQL Server code base. According to the Gartner group, Microsoft SQL Server has 20% in the market share. The main target of this database platform is the Windows market. Microsoft SQL Server includes professional, enterprise level database

management software and is easier to use and has more features than other database platforms. In addition, this database platform also has other advantages such as excellent data recovery support and its close integration with .NET platform (eHow).

Microsoft SQL Server and the Windows operating system both provide a variety of utilities that can be used to monitor Microsoft SQL Server databases (Microsoft). All these tools can help DBAs to determine whether performance can be improved, monitor user activity, troubleshoot problems and test applications (Microsoft).

"Activity Monitor is a built-in monitoring tool that provides information about SQL Server processes and how these processes affect the current instance of SQL Server" (Microsoft). The tool has different display panes and monitors performance metrics such as data file I/O and recent expensive queries in terms of CPU usage and read and write time (Microsoft).

Data Collector is a monitoring tool used to obtain and save data gathered from different sources. Data collection can either run constantly in the background or be configured to run at defined times. All the data collected can be analyzed later for performance trends and appropriate performance threshold values.

Other utilities, such as error logs and SQL Trace can also be used to assist DBAs in better monitoring databases and forecasting potential issues. Error logs provide an overall picture of events occurring in Windows and SQL Server agents. It contains information about events in SQL Server that is not available elsewhere (Microsoft). SQL Trace gathers events information, which then can be filtered out of the trace for their destination (Microsoft).

DB2 for LUW

DB2 is a leading relational database system developed by IBM. DB2 for LUW database software is designed for deployment on various Linux distributions, leading Unix systems, such as AIX, HP-UX, and Solaris, and also MS Windows platforms. Several editions of DB2 for LUW are available to satisfy specific business needs (Neagu, 2012).

IBM offers a free tool, Data Studio, which allows DBAs to easily monitor DB2 for LUW with a graphical user interface. The tool provides a global view of the system health with visual warnings and alerts by connecting to and monitoring multiple databases across different platforms from a single console. DBAs also have the ability to configure alert thresholds for health indicators, such as data server status and storage space utilization. Email or SNMP alert notification with information such as alert type, severity, and database can be set up as well. And when problems occur, DBAs can drill down into alerts to understand problems. In addition, browsing alert history is made available to help with analysis tasks (IBM, 2012).

The typical elements involved in DB2 for LUW monitoring fall into the following categories:

- Identification of the database manager, an application, or a database connection being monitored (IBM, 2012).
- Data primarily intended to help to configure the system.
- Database activity at various levels including database, application, table, or statement. This information can be used for activity monitoring, problem determination, and performance analysis. It can also be used for configuration.
- Information on DB2 Connect applications, including information on DCS

applications running at the gateway, SQL statements being executed, and database connections.

- Information on Federated Database Systems. This includes information about the total access to a data source by applications running in a DB2 federated system and information about access to a data source by a given application running in a federated server instance.

DB2 for z/OS

The DB2 9 for z/OS relational database management system is the largest of the DB2 family, often serving as an enterprise server handling many transactional systems including e-business, content management, enterprise resource management, business intelligence, and mission-critical systems. The DB2 for z/OS runs on the mainframe and is most often used to support the very largest databases and the highest transaction rates (Lawson, 2008).

Various tools and facilities enable monitoring of DB2 for Z/OS activity and performance, including facilities within the DB2 product as well as tools that are available outside of DB2, as shown in Figure 2. These tools include:

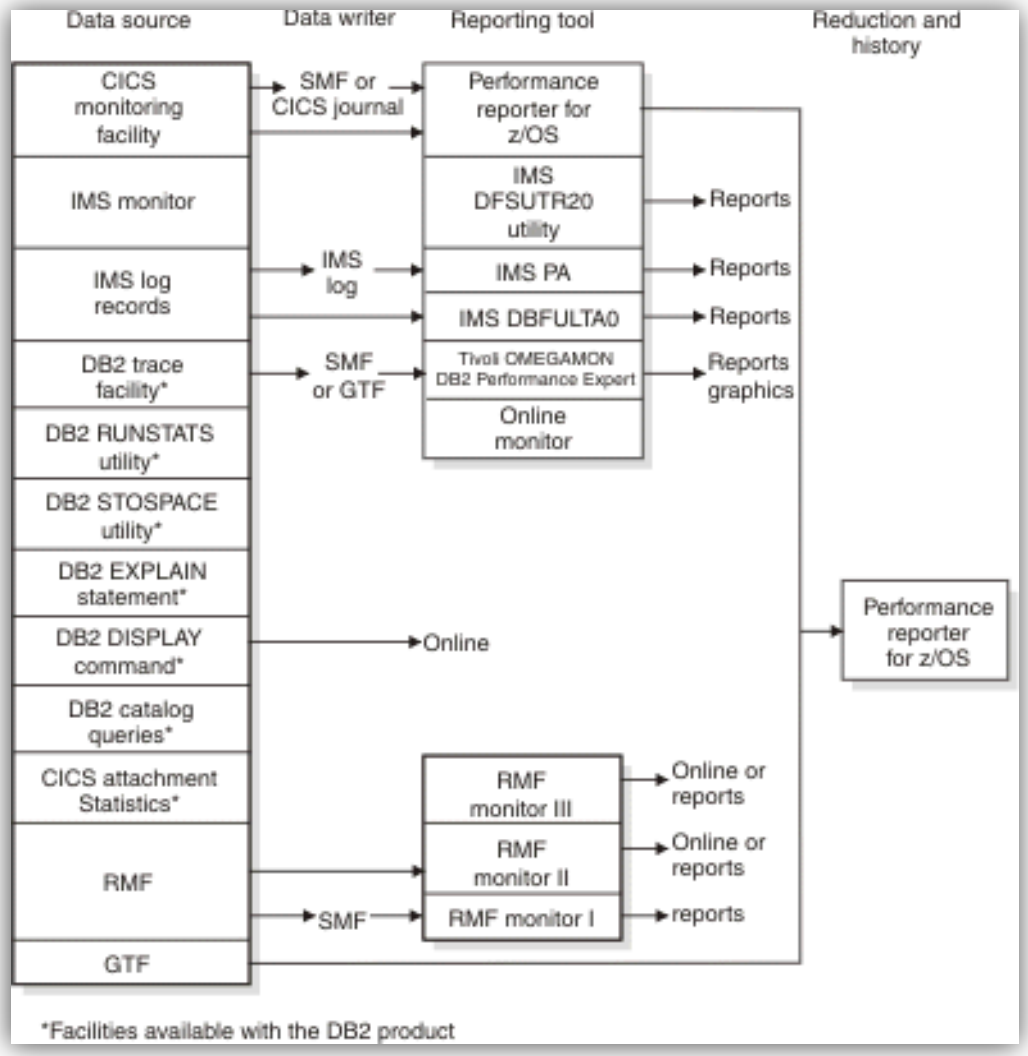

Figure 2: Monitoring tools in a DB2 environment **(IBM, 2012)**

- CICS Attachment Facility statistics provide information about the use of CICS threads. This information can be displayed on a terminal or printed in a report.
- OMEGAMON CICS Monitoring Facility (CMF) provides performance information about each CICS transaction executed. It can be used to investigate the resources used and the time spent processing transactions.

- DB2 catalog queries help DBAs determine when to reorganize table spaces and indexes.
- DB2 Connect monitors and reports DB2 server-elapsed time for client applications that access DB2 data.
- DB2 DISPLAY command gives information about the status of threads, databases, buffer pools, traces, allied subsystems, applications, and the allocation of tape units for the archive read process.
- DB2 EXPLAIN statement provides information about the access paths used by DB2.
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS is a licensed program that integrates the function of DB2 Buffer Pool Analyzer and DB2 Performance Monitor (DB2 PM). OMEGAMON provides performance monitoring, reporting, buffer pool analysis, and a performance warehouse, all in one tool. OMEGAMON monitors all subsystem instances across many different platforms in a consistent way. DBAs can use OMEGAMON to analyze DB2 trace records and optimize buffer pool usage.
- IBM Tivoli OMEGAMON XE for DB2 Performance Monitor (DB2 PM), included in OMEGAMON, is an orderable feature of DB2 that helps with analyzing DB2 trace records.
- DB2 RUNSTATS utility reports space use and access path statistics in the DB2 catalog.
- DB2 STOSPACE utility provides information about the actual space allocated for storage groups, table spaces, table space partitions, index spaces, and index space partitions.

- DB2 trace facility provides DB2 performance and accounting information.
- Generalized Trace Facility (GTF) is a z/OS service aid that collects information to analyze particular situations. For example, GTF can be used to analyze seek times and Supervisor Call instruction (SVC) usage, as well as other services.
- IMS DFSUTR20 utility is a print utility for IMS Monitor reports.
- IMS Fast Path Log Analysis utility (DBFULTA0) is an IMS utility that provides performance reports for IMS Fast Path transactions.
- OMEGAMON IMS Performance Analyzer (IMS PA) is a separately licensed program that can be used to produce transit time information based on the IMS log data set. It can also be used to investigate response-time problems of IMS DB2 transactions.
- Resource Measurement Facility (RMF) is an optional feature of z/OS that provides system-wide information on processor utilization, I/O activity, storage, and paging.
- System Management Facility (SMF) is a z/OS service aid used to collect information from various z/OS subsystems. This information is dumped and reported periodically, such as once a day.
- Tivoli Decision Support for z/OS is a licensed program that collects SMF data into a DB2 database and allows DBAs to create reports on the data.

Summary

Table 1: Database Platform Comparison Table

	Oracle Database Monitoring Tool(s)	SQL Server Monitoring Tool(s)	DB2 for LUW Monitoring Tool(s)	DB2 for Z/OS Monitoring Tool(s)
Trend analysis	Yes	Yes	Partial	Yes
Transaction capture	No	No	No	Yes
Generating alerts	Yes	Yes	Yes	Yes
Graphical interface	Yes	Yes	Yes	No
Using within custom application	Yes	Yes	No	Yes

Cross-Platform Database Monitoring

Cross-platform Database Monitoring is the management of multiple database systems across different platforms at the same time. With the proliferation of information and the deployment of many distinct types and brands of databases, it has become a common practice for modern enterprises and a significant function within DBA work.

An Industry Survey

According to the 2011 SURVEY ON CROSS-PLATFORM DATABASE ADMINISTRATION (2011) conducted by Unisphere Research, DBA teams generally have a relatively small number of people to manage multiple database platforms, as shown by Figures 3 and 4. The situation at Hanover is similar, where 10 DBAs take care of over 2,000 databases across several platforms, a 1:200 ratio. Among the 289 managers taking the survey, a variety of database brands are represented across their sites and the 4 most popular databases are Microsoft SQL Server (76%), Oracle (67%), MySQL (38%) and DB2 (34%) (Figure 5), three of which are also the major database platforms at Hanover. Therefore, the topics in this survey should also be applicable and valuable to our current project.

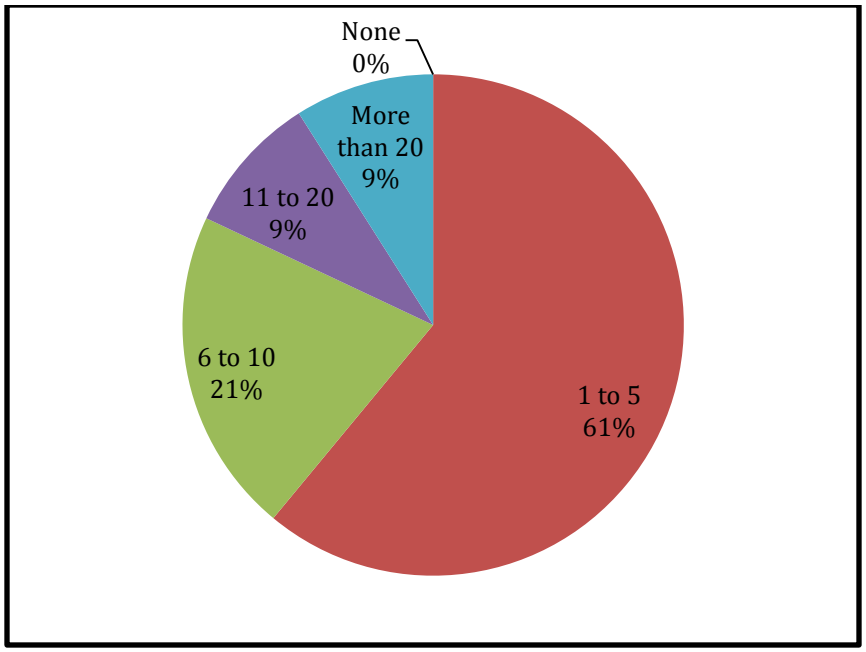

Figure 3: Number of People in Respondents' Teams

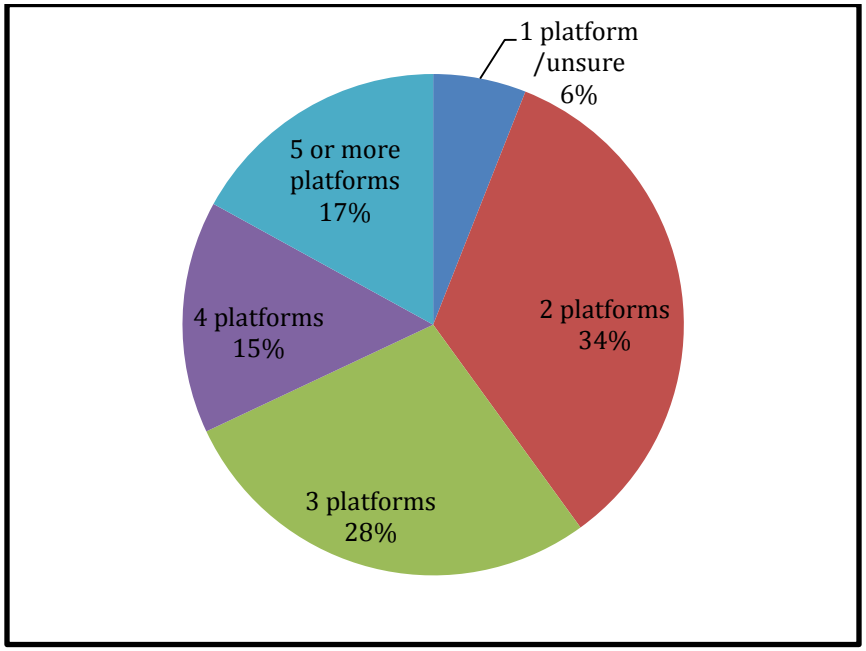

Figure 4: Number of Database Platforms at Respondents' Sites

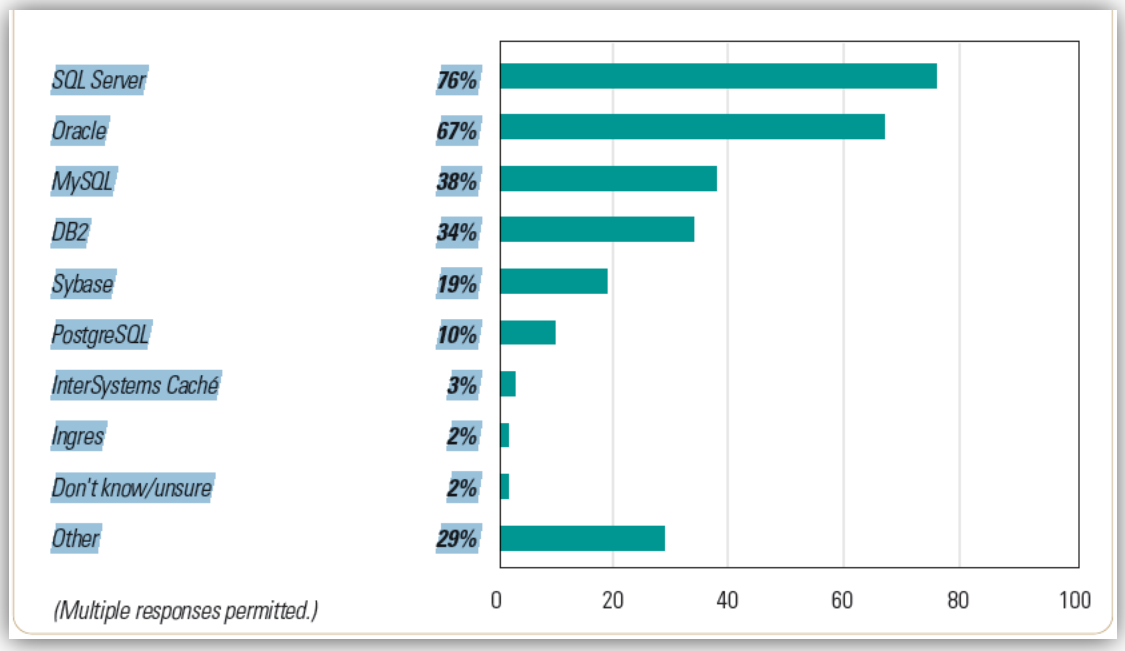

Figure 5: Database Platforms Managed at Respondents Sites

With regard to big challenges within multi-database management system environments, the largest segment of respondents, 41%, say they encounter issues with coordinating inter-database data integration. Another 37% report that they are struggling with the costs involved in managing many different DBMS tool licenses (Figure 6). To meet such challenges, most respondents (70%) have built their own home-grown tools or engage in their own troubleshooting methodologies (Figure 7). Moreover, a sizable segment of participants, 43%, also employs third-party cross-platform management tools to handle the environments. The above findings indicate that we would most likely choose between a self-developed tool and a third-party product as our proposed solution. No matter what the choice is, we should try to make sure that it addresses data integration issues and high costs of platform-specific tools.

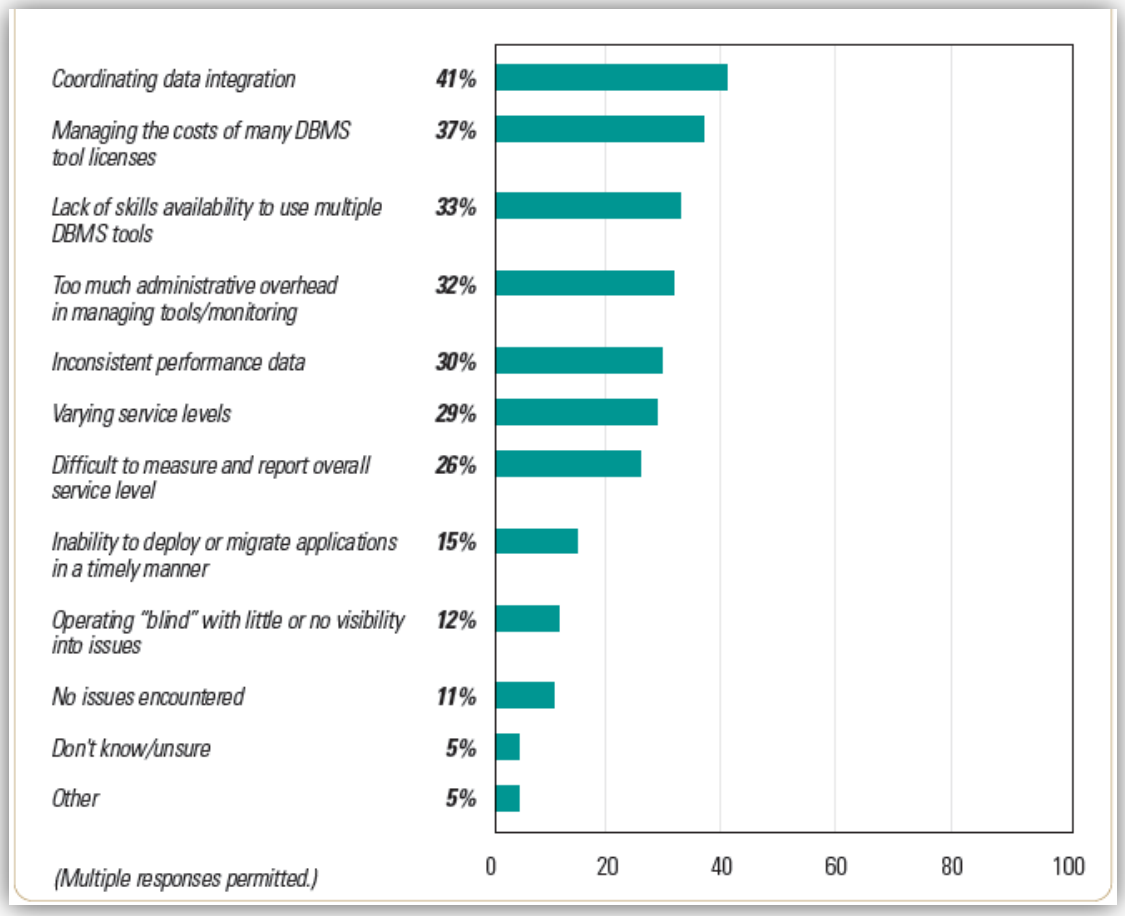

Figure 6: Multiple DBMS Challenges

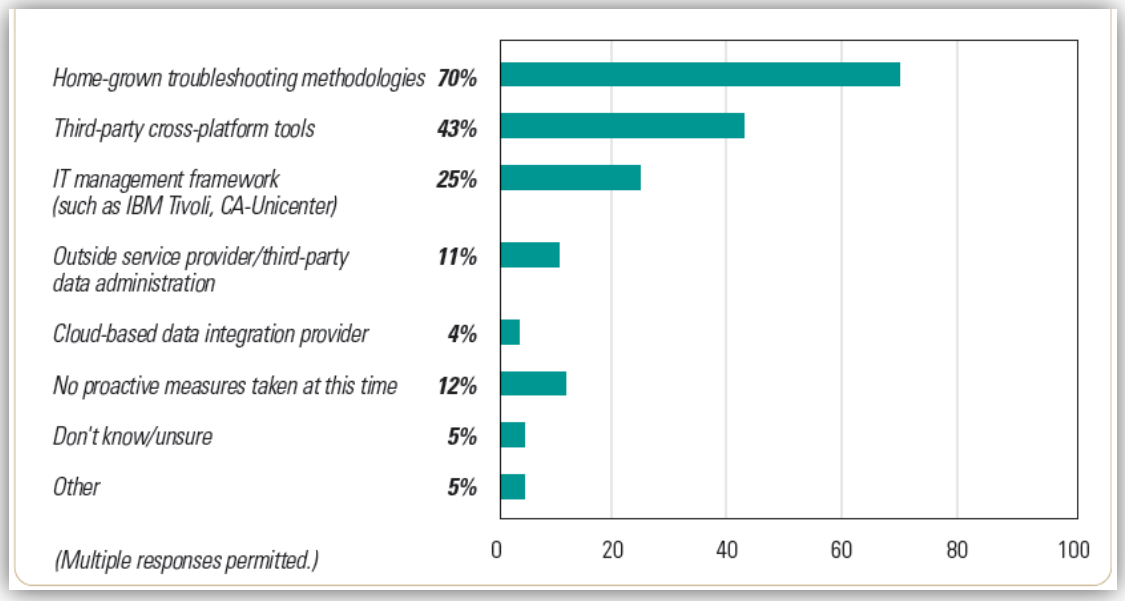

Figure 7: How Multiple DBMS Platforms Challenges are Addressed

The survey results also show that more than 75% of the sites use different tools for each platform (Figure 8). The use of such differing tools, however, seems to be only marginally effective in managing critical databases. In fact, only 16% would consider this practice to be "extremely" effective, versus a majority of respondents, 53%, that consider it to be only "somewhat" effective. Another 16% say the practice of using different tools for different DBMSs is ineffective (Figure 9). Thus, the adoption of a cross-platform solution seems to be what the technology tide is requesting.

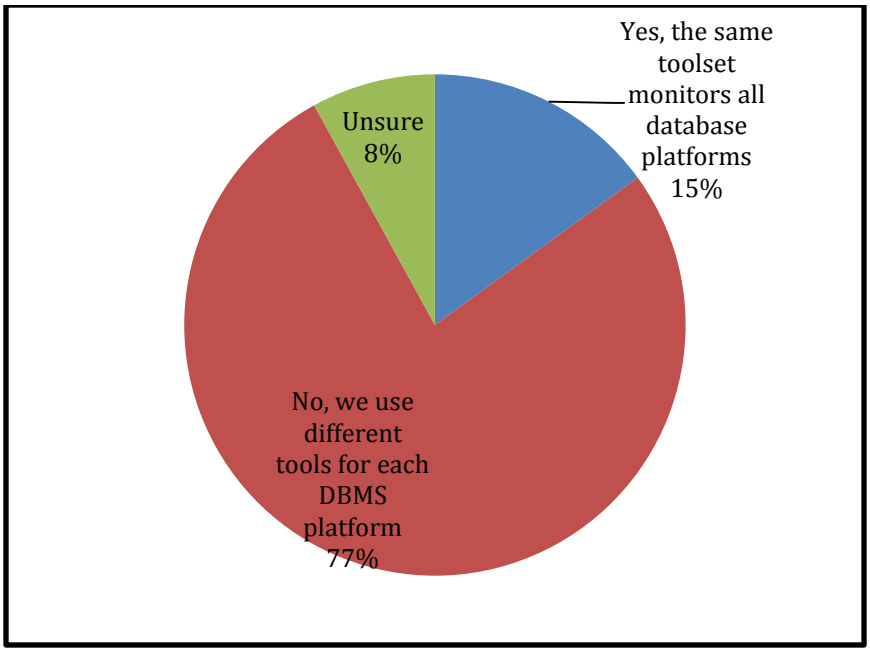

Figure 8: Employ Same Tools across Multiple Database Platforms

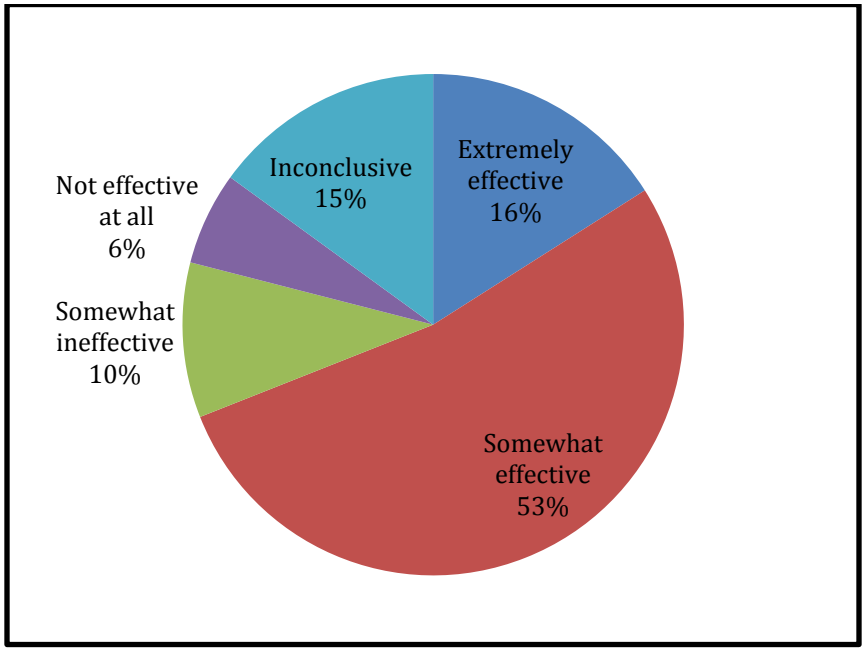

Figure 9: Effectiveness of Multi-tool Approaches for Cross-platform Monitoring

As noted earlier in this report, most respondents use homegrown tools and solutions to manage their complex database environments. Nonetheless, there is a split

here among respondents in terms of how frequently these solutions are put into action. A total of 38% respondents employ home-grown troubleshooting methodologies (e.g., scripts, command line, etc.) most or all of the time to address multi-platform administration and monitoring. Another 42% say they rarely or never use such resources, or simply do not know if they do (Figure 10). There is a lack of third-party tools employed to address multi-platform administration and monitoring as well. About 16% say they use third-party tools most or all of the time, versus 69% only using such tools some of the time, if at all (Figure 11). Such low levels of use create a great waste of resources. Consequently, we should pay special attention to the implementation use details after our proposal is approved to help Hanover successfully migrate to improved database monitoring practices.

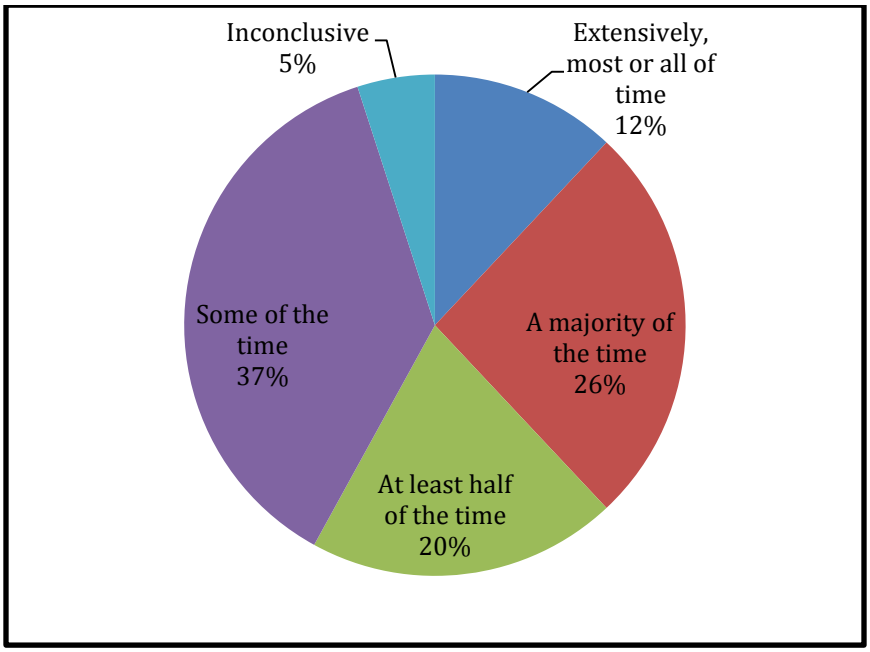

Figure 10: How often Home-grown Methodologies are Employed

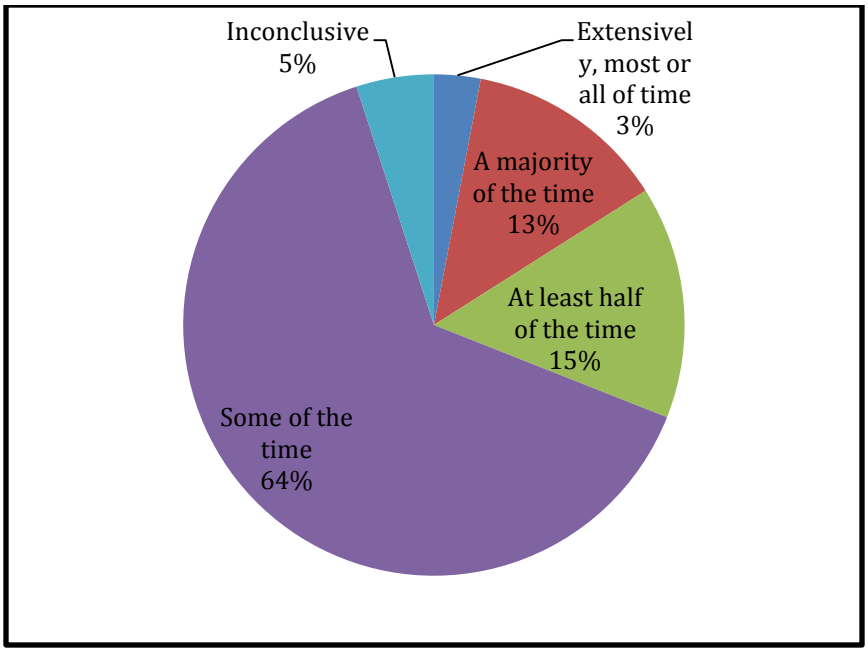

Figure 11: How often Third-party Cross-Platform Tools are Employed

General Requirements

To address the challenges associated with managing multiple database systems, a single and consolidated view of database health and performance is the key. The coverage of the view must extend across several layers of the application, database and operating system stack to ensure that the information presented to database administrators is both accurate and comprehensive. The solution should also facilitate the company-specific user workflows that drive problem resolution and prevention as well as performance optimization.

General requirements for a good cross-platform monitoring system include (Pearson, 2010):

- The ability to independently monitor all database performance related data in real time.
    - The system should be able to perform its monitoring job automatically after it is

configured correctly. No human inputs, such as scripts and commands, should be required so that DBAs can focus their time on problems and thus be more productive.

- Insight into the execution and performance of every component or process that could potentially disrupt database operations, including both the operating system and virtual machines.
    - The database is not isolated. It is dependent on the operating system as well as virtual machines and their storage subsystems for resources. Coverage of these components is necessary to accurately identify the source of problems.
- An intuitive user interface that enables the management of cross-platform performance in one place, accommodating different levels of administration skill and catering to the preferences of DBAs with different technical backgrounds.
    - The central benefit of a self-explanatory user interface is the abstraction of platform-specific complexity, enabling a service-centric perspective on performance. As a result, DBAs are more efficient while training costs are greatly reduced.
- Visibility into the transaction workload that is driven into various databases by application users and developers, and other processes.
    - The database is linked to the application by the transactions that are directed into the database. Measuring the database transaction workload helps DBAs grasp the service quality of the database.
- Sufficient depth and readily available functionality to support detailed analysis and optimization activities by aggregating and correlating activity from multiple

heterogeneous Database Management Systems.

  - The complexities of modern databases, and the critical applications they support, require far more sophisticated analysis of performance than is provided by the raw metrics of the database engine. A mechanism is always needed to gather and prepare sufficient historical data so that DBAs can determine important trends, identify chronic conditions, and prevent emerging issues.

- An economic solution package with a relatively low total cost of ownership, in regard to both initial deployment and the operations that follow.
  - Total cost of ownership refers to all the direct and indirect costs associated with an asset or acquisition over its entire life cycle (WebFinance, Inc, 2012). Technology that is designed to reduce operational costs as a fundamental part of its value must demonstrate a low total cost of ownership.

This background information has provided a thorough understanding of the various platforms used by Hanover. The associated strengths and weaknesses and the respective industry best practice standards for each platform have been discussed. With this information the next chapter presents a complete analysis of the current system used by Hanover.

Chapter 3: Current System

Currently, Hanover monitors four database platforms: Oracle Database, SQL Server Database, DB2 for LUW and DB2 for z/OS via its own homegrown tools or third party monitoring tools. There are two types of metrics used to monitor databases: health and performance metrics. Health metrics measure how well the databases are functioning and performance metrics measure whether databases are functioning (Oracle, 2010). At Hanover, different monitoring tools are used per platform causing specific health and performance metrics across the platforms to differ as well. While health metrics are monitored across all four databases, performance metrics may not be. In addition, the various DBAs specialize in different platforms, accounting for root metric differences across the platforms. In the following section, we will discuss the database monitoring practices for each specific database platform.

Oracle Database

Hanover does not use many monitoring tools for its Oracle platform databases. Although built-in Oracle Enterprise Manager is often used without any base license fee, it does not contain any additional diagnostic or tuning features. For instance, it does not have the feature of generating AWR reports. In addition, the Oracle Enterprise Manager does not store any information in the central database. Therefore, no historical information is kept.

There are some health and performance metrics that are monitored using the Oracle Enterprise Manager. Database availability, database size, block sessions, central processing unit (CPU) usage and memory usage are all monitored under the current tool. Moreover, there are also Hanover developed scripts written to scan Oracle database. If

critical errors are detected, scripts will send out emails to notify the primary database contacts and forward to secondary contacts.

Overall, the current monitoring tools for Oracle database are reactive rather than proactive to the possible problems. In addition, these tools are not very effective since database administrators can only get notified once tickets are issued. There are also some additional health metrics, such as storage warning thresholds, that need to be added to make the DBA team work more efficiently and productively.

SQL Server Database

Hanover has some database monitoring tools for health and performance metrics in Microsoft SQL Server. Some of these monitoring tools are homegrown SQL Server utilities. There are also statistics tools that are built into SQL Server and Microsoft Windows, such as SQL Server Trace. DBAs use activity monitoring to check what is currently happening in the system, but not to collect statistics. In addition, the SQL 2008 monitoring tool is very resource intensive.

For health metrics, there are three reports that are run at 6:30 AM daily. The first is a report for the backup taken for each database. It includes the time schedule to be taken to make sure the backup can run. In addition, final backup check is made at 11:30 AM daily. The second is a real time health report which searches for unreachable, unavailable and read-only databases. This health report can be run multiple times. DBAs will only be notified when the database comes back up but not when it goes down due to system setup. The last report is a space monitoring report which can check which boxes have less space than the average, all the volume and how much space is left.

There are many performance metrics that are monitored. Performance buffer cash ratio, number of physical reads and writes, flush, memory usage, processor, disk storage,

percentage of storage, dead lock and lock time are all monitored. Storage and processor usage are monitored externally. Also, SQL statements taking longer than five seconds to execute are monitored to speed up the process. There is no alerting system based on these performance metrics because there is no threshold used for performance monitoring. DBAs will receive alerts from the application team only when applications are not running.

DB2 for LUW

The usage of DB2 for LUW in Hanover is fairly small compared to other database platforms. Only 1% or 2% of all databases are running DB2 for LUW, or 20-40 of the 2,000. There are not many monitoring tools available. One tool is the built-in activity monitor which can be turned on when needed. The metrics that are currently monitored are database response time, database availability and manual storage management. Overall, these tools are reactive tools and there is no forecasting involved.

DB2 for z/OS

Although DB2 for z/OS adopts a reactive monitoring approach, this database platform is more proactively monitored than DB2 for LUW. OMEGAMON and APPTUNE are the two main monitoring tools currently used for DB2 for z/OS.

IBM OMEGAMON has GUI components and does much system monitoring. Currently, it is used to monitor the CPU utilization. The tool can monitor many details including transactions in the databases. In addition, the tool has several other features that are unknown by Hanover's DBAs, resulting in underuse of the tool.

APPTUNE is software developed by BMC Software Company to monitor DB2 for z/OS. The tool captures the health of one instance in the database and summarizes the snapshots of different instances. It can summarize statistics by user, by program and by

other categories. It monitors the CPU usage, aisle count, deadlock count, number of transactions, duration and other performance metrics. Although APPTUNE can also store historical information, it loses details of the stored information because it does not monitor statistics on individual transactions. Therefore, the tool is only used for high level monitoring. In addition, only 25% of the databases are monitored to lower the sampling overhead expenses.

The current monitoring tools do not notify DBAs immediately if a database is down. Therefore, the response time to such problems is relatively long.

DBA Team Responsibilities

The respective responsibilities of the HTGs DBA team are outlined below to present a sense of organizational structure. Each DBA leads an individual platform and a support team to ensure its continuous and high quality service.

DB2

Leader: Peter Wallace

Main responsibilities: Manage the scheme and test environment. Coordinate team support activities for DB2 DBAs.

Respective Team Support responsibilities:

Ed Schuster: Charged with monitoring performance.

Eunkyung Han: Citizen's office DB2 liaison (the sister company of Hanover)

Oracle

Leader: John Aragi

Main responsibilities: Manage the DB2 Linux and Oracle team members. Charged with maintaining PeopleSoft within Oracle.

Respective Team Support responsibilities:

Don Postma: Charged with leading the Oracle team and overseeing the MIS.

Brian Vigneaux: Charged with monitoring the Windshield application databases.

SQL Server

Leader: Keith Van Riper:

Main responsibilities: Charged to manage team and monitor HCS application databases. Respective Team Support responsibilities:

Laura Bridi: Charged with monitoring the Progress database.

There are also two offshore DBAs that support SQL Server and Oracle.

BSM Dashboard

Currently, the Business Service Management (BSM) division has dashboards to support both infrastructure and applications but not databases. These dashboards can serve as a functional and visual management model for a set of database dashboards. The figure below is a screen shot that shows the dashboard system currently used by Hanover to monitor the application infrastructure.

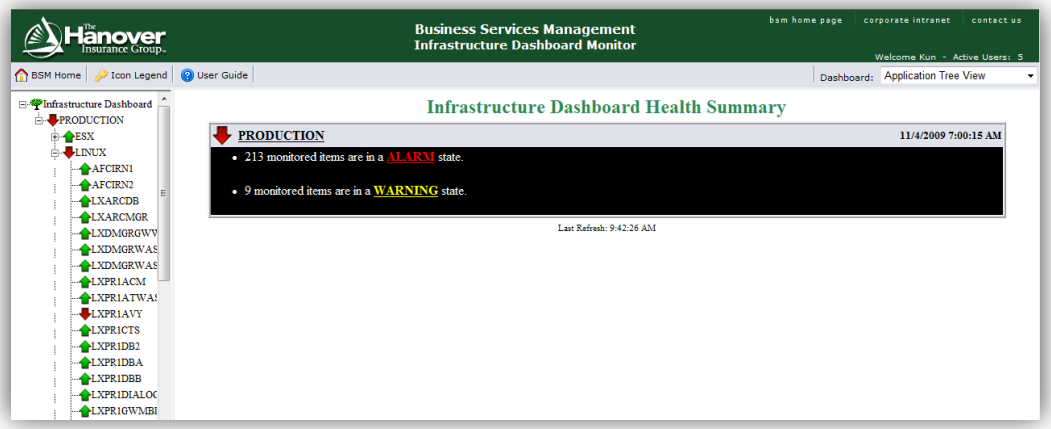

Figure 12: Hanover BSM Dashboard Screen Shot

There are several methods of gathering the appropriate metrics information and triggering events in the dashboard. For instance, applications can send a formatted string through FTP or call web services to push information to the dashboard. They can also use SNMP trap by designating destination IP addresses in the configuration file. Finally, email alerts can also be used to push information to the dashboard. Once events are triggered in the server, the internal process will make appropriate notifications such as email, pager or ticket to appropriate persons.

BSM dashboard is considered as a good solution to the cross-platform database monitoring problems that we have observed in Hanover. The dashboard is an internal platform which satisfies all the UI and safety standards in Hanover. Consequently, by using BSM dashboard, Hanover will incur less development and maintenance cost while minimizing the risks that our solution might not fit into the current system.

In addition, the dashboard also contains all the basic features that we need to have in our proposed solution. For instance, metrics data could be pulled from databases by running appropriate queries against the central repository in the back-end. Then, the dashboard could display the metrics data retrieved in predetermined format. Also, with the current dashboards, application developers are able to establish thresholds and set up alerts. Trending charts could also be drawn using historical data. Therefore, we can use the dashboard to set up appropriate thresholds for each database metrics and platform.

SiteScope

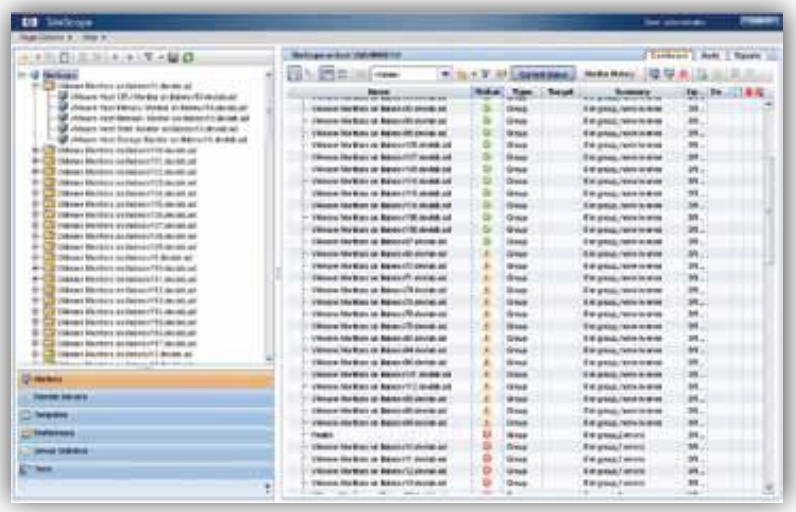

Figure 13: Hanover SiteScope Screen Shot

SiteScope (Figure 13) is the Hewlett-Packard based monitoring software used by Hanover IT group to monitor the availability and performance of different IT infrastructures such as servers, operating systems and applications. The tool provides a centralized and scalable architecture to support three key functions: data collection, alerting, and reporting. Data collection is performed through remote monitoring and does not require agents to be installed and maintained on monitored nodes. For alerts and reports, both standard methods, such as email and SNMP trap, and additional methods, such as HTTP post and database alerts, are supported. Alerts are sent to IT administrators based on preconfigured thresholds in defined intervals. In addition, SiteScope also enables generation of daily, weekly, and monthly summaries of single and multiple monitor readings.

Gap Analysis

Although many health and performance metrics are monitored currently, it is important to have common metrics for different database platforms for implementing cross-platform monitoring. In addition, no system can guarantee zero downtime and will not be able to support growth for the infrastructure in the future. Also, only critical databases (see list in Appendix A) are monitored leaving the other ones untouched.

All of Hanover's current monitoring tools for different database platforms use a reactive approach towards database issues. The industry standard for database monitoring is proactive where issues are predicted based on trending data and capacity planning.

Completely different metrics are currently used for Hanover's different database platforms. Common metrics need to be developed and appropriate metrics for each database platform need to be selected for the cross-platform monitoring.

In addition, the current DBA working environment is cubicle based therefore limiting shared team communication opportunities. Further, information is email based and a team information forum/portal does not exist. Hanover should have a centralized area for DBAs to view current monitoring results and solve potential issues proactively. A more agile environment will allow DBAs to work more efficiently and productively with lower database downtime. Operation command centers are a good reference for agile work environments. Nevertheless, these organizational approaches take time to implement and adapt to but should be considered for long-term business value.

Performance Metrics

SQL Overview

The SQL statements that we get from DBAs can be categorized into six types, which are historical DB backup information, historical DB storage information, DB portfolio information, realtime information, application information and server statistics.

Historical DB backup information

The historical data about database backup information is stored in DB_BACKUP_STAT table. An appropriate SELECT statement is given by DBAs to retrieve the latest backup information. In addition, we have the choice of retrieving all backups from either all actively used databases or just a single database.

Historical DB storage information

The historical database storage information is stored in PERFSTAT.DB_TS_STAT table. DBAs give us SQL statements to list the growth trends of databases.

Historical DB portfolio information

For database portfolio information, all the data is stored in the STORSTAT.DB_INFO table. A simple SQL statement is given to retrieve relevant information

Realtime information

Queries are given to run on each individual Oracle database. Top 25 SQL statements by run time and by execution time can also be retrieved.

Application info

This query will need to be executed in each individual Oracle database. There is also a table, PERFSTAT.ORACLE_LICENSE_HIST, that stores the daily connection high water mark.

Server Statistics

The Oracle DBAs do not currently store any historical server related statistics. Therefore, only server statistics in SQL Server is kept.

List of Metrics

Below is a list of expected performance metrics developed and issued by our sponsor Brandon Willis:

Historical DB backup information

- Backup type by DB (Full, Incremental, archive)
- Start time, end time including date/hour/min
- If the backup was successful or if it failed

Historical DB storage information

- Growth statistics by DB

Historical DB portfolio information

- DB name, server and DB version

Realtime information

- Validation that DB's are online
- File system/storage warning thresholds
- Top run time SQL statements
- Top executed SQL statement

Application info

- Number of Concurrent users per DB

Server Statistics

- CPU utilization

- Memory Utilization
- I/O stats

Mainframe Information for DB2 on z/OS

- LPAR statistics/utilization

With these SQL statements and metric-based information, we could retrieve the appropriate information from the databases and display them in the dashboard.

Chapter 4: Methodology

This project follows an incremental methodology based on the waterfall model with characteristics of prototyping for final system implementation into The Hanover Insurance Group (Figure 14).

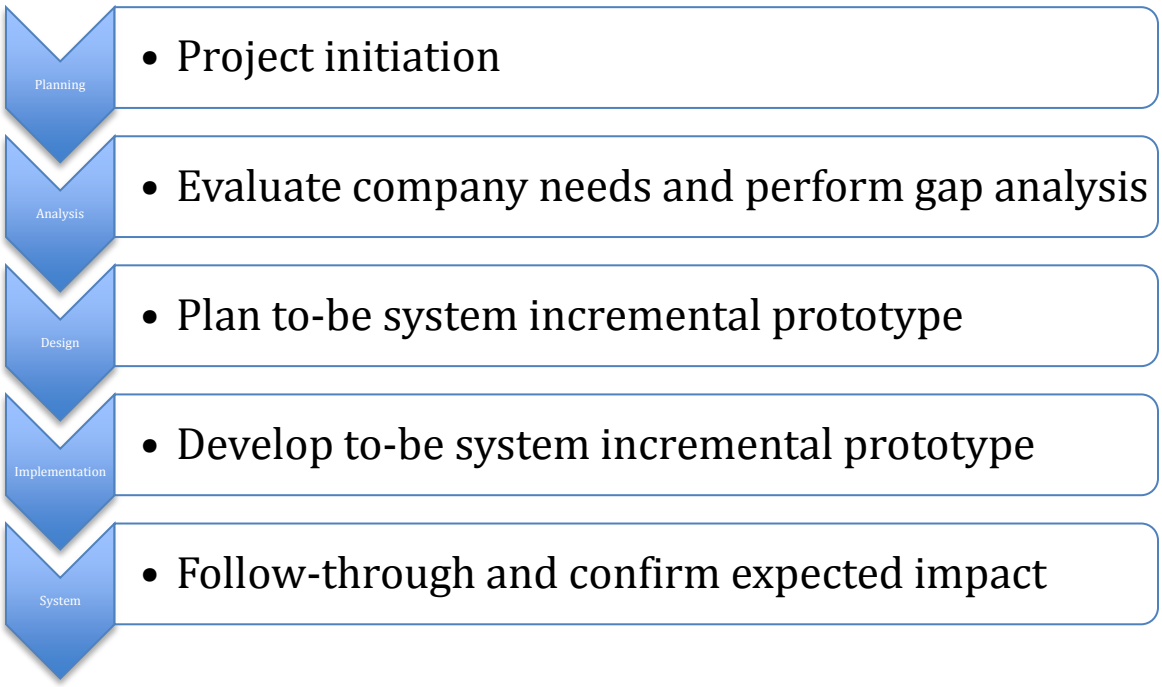

Figure 14: Project Incremental Methodology

Planning

During the planning stage, our group completed all tasks related to project startup, including administrative functions and workspace logistics. This stage was completed within the first week of the project.

Analysis

During the analysis stage, our group first determined the current system status within Hanover. This involved two types of interviews: overarching system interviews and deep-dive platform interviews. The overarching system interviews sought information regarding the overall database monitoring strategy used by Hanover, current

system capabilities and support/operational resources needed. Interviews were held with various HTG representatives to achieve an accurate and holistic perspective. The deep-dive platform interviews sought specific information regarding each of the four platforms used for database monitoring. Interviews were held with the respective DBAs for each platform. Appendix B lists our data collection interviews and activities.

With this information about the current system, we conducted a gap analysis. This involved analyzing the seemingly reactive database monitoring business mode, identifying common metrics, forming a centralized monitoring strategy and presenting information and recommendations for an agile workplace environment.

Design

During the design stage, our group took the gap analysis information we gained from the analysis stage and applied it to our to-be system model or implementation proof of concept. We consulted system architects and DBAs in this stage to develop logical and well-developed incremental dashboard concept versions.

Implementation

During the implementation stage, our group worked directly with HTG representatives to oversee the step-by-step integration of our incrementally developed prototype.

Prototype

Our group followed an incremental prototype development cycle. Partnering directly with each of the DBAs and affected end-users, we developed our prototype over time and in progressive increments. This development approach enabled us to continuously revisit exceeding the end objective, work directly with the DBA team

weekly to make proper adjustments/advancements, and learn from a true life cycle development.

System

During the system stage, our group worked with Hanover to confirm effective new system integration and performance. In addition, we evaluated what the future system needs could be.

Project Schedule Overview

The project task timeline can be viewed in Table 2 and seen in Gantt chart format in Figure 15.

Table 2: Project Task Timeline

Task Name	Duration	Start	Finish
MQP Initiation	6 days	Wed 8/22/12	Wed 8/29/12
System Evaluation	16 days	Wed 8/22/12	Wed 9/12/12
Literature Review	16 days	Wed 8/22/12	Wed 9/12/12
Gap Analysis	13 days	Wed 9/5/12	Fri 9/21/12
MQP Report Development	71 days	Wed 8/29/12	Wed 12/5/12
MQP Proposal Presentation	1 day	Wed 9/26/12	Wed 9/26/12
Incremental Prototype Design and Development	47 days	Wed 9/26/12	Thu 11/29/12
MQP Final Presentation and Completion	1 day	Wed 12/12/12	Wed 12/12/12

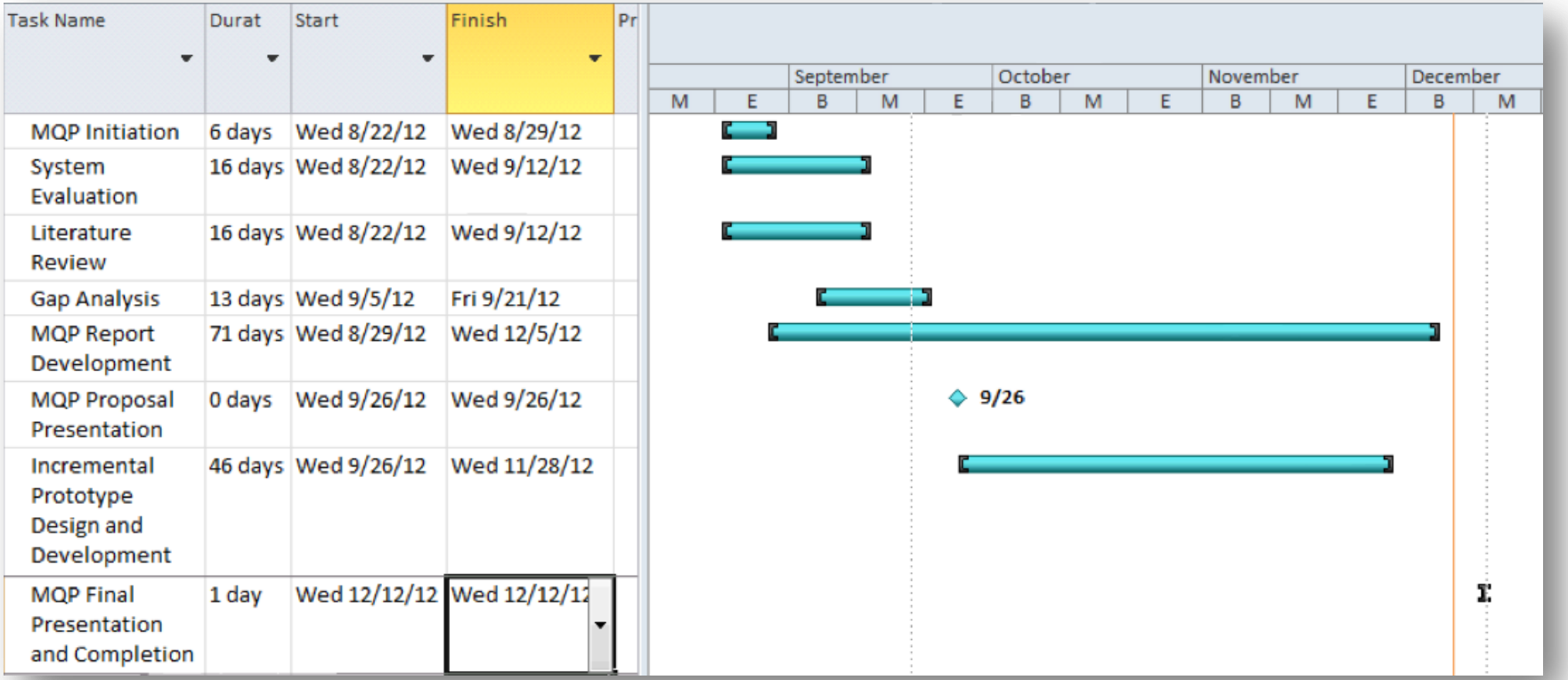

Figure 15: Project Task Gantt Chart

Chapter 5: Design of the To-Be System

This chapter begins with a summary of the features our proposed monitoring system should possess, based on our analysis of user requirements. It then describes the design consideration for the system prototype and concludes with a feasibility analysis of the proposed system.

To-be System Requirements

The proposed Database Monitoring Infrastructure Dashboard should have similar GUI design to the current Infrastructure Dashboard in the BSM used by Hanover. The dashboard provides a centralized integration place for database performance metrics across each platform. We envision the new cross-platform database monitoring solution to provide the following key capabilities:

1. Consolidated Cross Database Management: The system provides in-context integration that centralizes and simplifies the management of different database platforms and has a consistent look and feel that facilitates the expansion of DBAs' skills and capabilities in managing databases.
2. Real-time Database Monitoring: The system monitors database performance by automatically gathering statistics at predefined time intervals, with easy-to-understand graphical displays and in-time data update, which are expected to help reduce the reaction time of the DBA team.
3. Alert and Notification Management: The system reduces unexpected outages and minimizes firefighting by letting users configure notification and alert to meet specific requirements.

3.1 Unattended Monitoring with Preset Indicators: Various predefined indicators are utilized to monitor critical components for problems and to collect workload statistics on an enterprise wide basis. This reduces the need for DBAs to write and maintain custom scripts for each database system separately.

3.2 Proactive Alarm and Bottleneck Notification: Alarms are integrated in-context for faster problem resolution. With threshold customization capabilities, DBAs can decide how far in advance they want to be notified of unexpected database conditions with proper thresholds. Alarms exist for a number of conditions, including locking, storage, bottlenecks, contention, inadequate memory, etc. Alarm notification can be in the form of screen highlight as well as email and mobile device.

4. Historical Data Reference and Analysis: The system collects and stores performance information in a repository or a flat file, which can be used to provide summarized and detailed performance trending and diagnostics information. The comparison results between real-time data and the stored information may also trigger notification and alarm.

5. Overview and Detailed Performance Information: The system supports the DBA team by showing performance anomalies, discrepancy from trend and out-of-bound conditions with grouping of key performance indicators and statistics, summarized and displayed in a context easy to identify issues. For greater depth, DBAs can drill down to details and perform diagnosis accordingly.

6. Customizable Browser Based Console: The system consists of a browser-based, platform independent interface that provides the information DBAs need to effectively and efficiently locate and resolve database performance issues, keeping underlying database complexity transparent. The browser-based interface is built using mature technology that Hanover has been using for a long time to ensure performance and security, create a consistent look and feel and offer user customization for personal preferences and needs.

Database metrics

After meeting with Oracle and SQL Server DBAs, we found that there are several categories of metrics: historical database backup information, historical database capacity information, database portfolio information, health checks information, application information and server statistics for distributed systems. Of all those metrics, backup information and server statistics are the two main categories that we will monitor and implement in the prototype. The reasons are that we need to have numerical and character data types to make sure that the dashboard can display them appropriately. Historical backup information and server statistics are the representatives of these two data types.

In addition, the metrics of only critical databases will be monitored in the dashboard. Therefore, after meeting with our customer, we came up with a list of critical Oracle and SQL Server databases (see Appendix A).

Prototype Design

Our proposed cross-platform database infrastructure monitoring dashboard will be composed of two subsystems which are Hewlett Packard SiteScope as the backend statistics gathering mechanism and Hanover's home grown Business Service Monitor Dashboard as the frontend graphical user interface. SiteScope periodically runs

predefined SQL queries and scripts gathered from the DBAs and generates log files. Dashboard then reads the logs and displays the information. At the same time, SiteScope sends DBAs warnings and/or alerts in the form of email and/or paging while Dashboard uses visual indicators such as yellow flag for warning and red flag for alert to point out issues. These indicators and alerts will be based on thresholds set by our users. In this way, our prototype satisfies the need of both real-time monitoring and issue notification.

Dashboard UI Design

Concerning the user interface design, Figure 16 shows the screen shot of the Foglight® database performance monitor. This is a good design format we can use as a guideline. As we can see from Figure 16, the colored line shows the overall performance. Below it is the summary for each database platform. Further down lies the detailed information for specific databases.

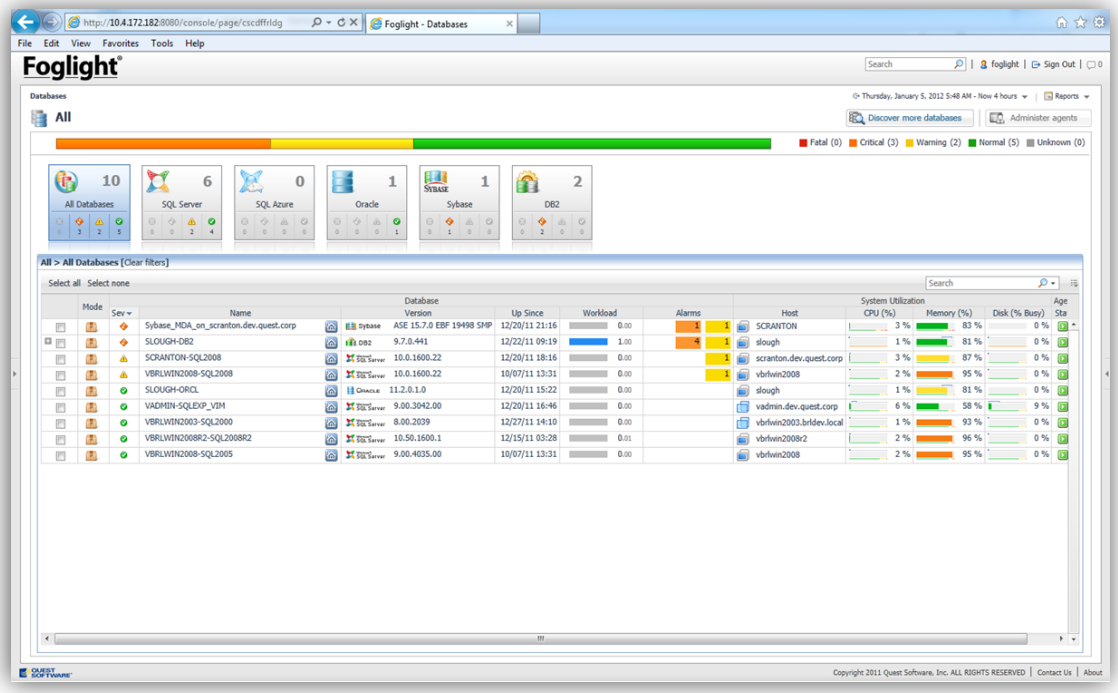

Figure 16: Quest Foglight® database performance monitor

We then used SiteScope to run all the SQL statements. As a result, text files are generated and sent to the dashboard team. The text files are parsed into six parts including SID, Type, Status, Start Time, End Time, Elapsed Time, in order to be incorporated into the newly designed dashboard prototype. The data from each part is then displayed in the dashboard.

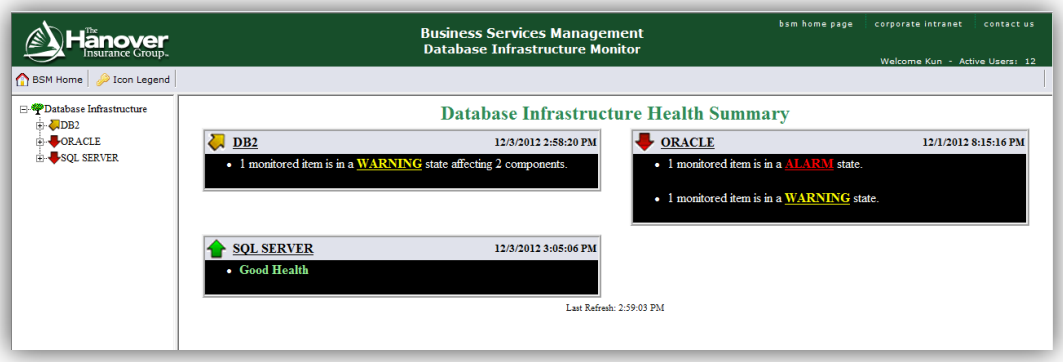

Figure 17: DB Monitoring Dashboard

Figure 17 is the actual user interface of the database dashboard that we designed. The design of the prototype follows the design standards set by Hanover. On the left, there are three main types of databases to be monitored which are DB2, Oracle and SQL Server. Under each database type, there are specific metrics types such as backup information and CPU usage. On the right side, a summary of the health statuses of all the database platforms are displayed.

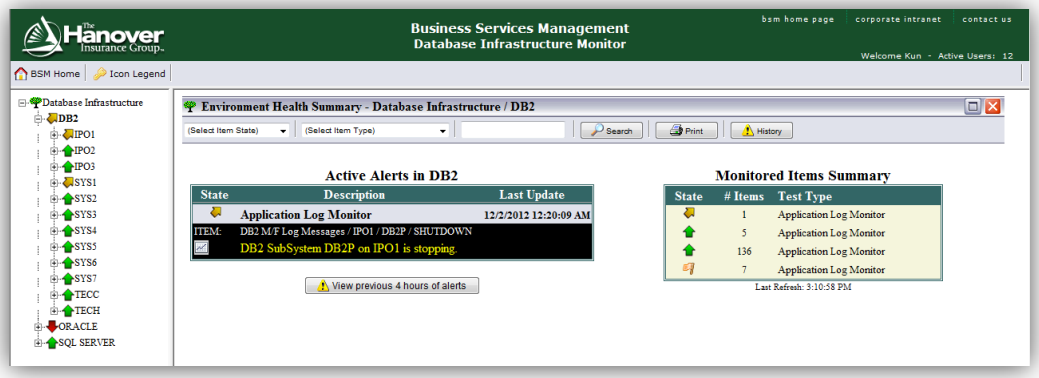

Figure 18: DB Monitoring Dashboard

Figure 18 shows the summary of each specific database platform. A summary of the active alerts is displayed in the center. On the right side, a summary of the monitored items is displayed.

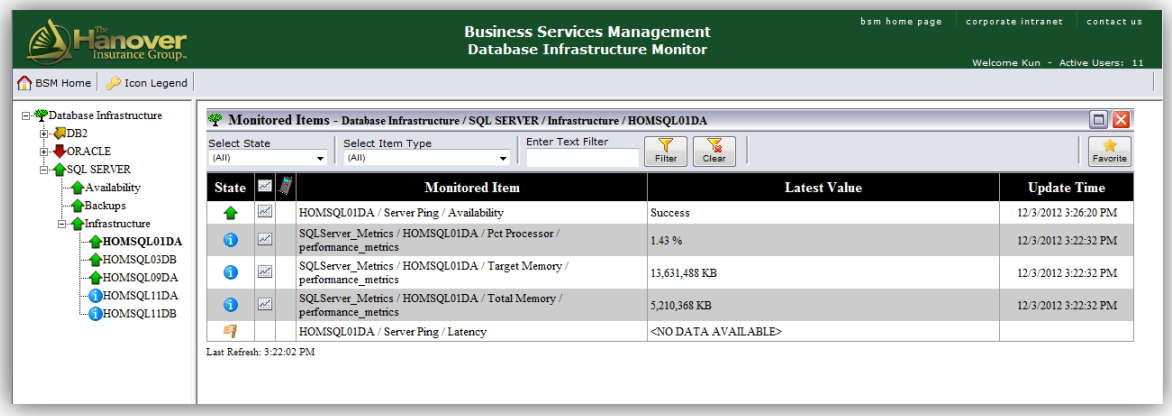

Figure 19: DB Monitoring Dashboard

Figure 19 displays a specific database's metrics. All the critical databases and their respective monitoring values and times are displayed in the center screen.

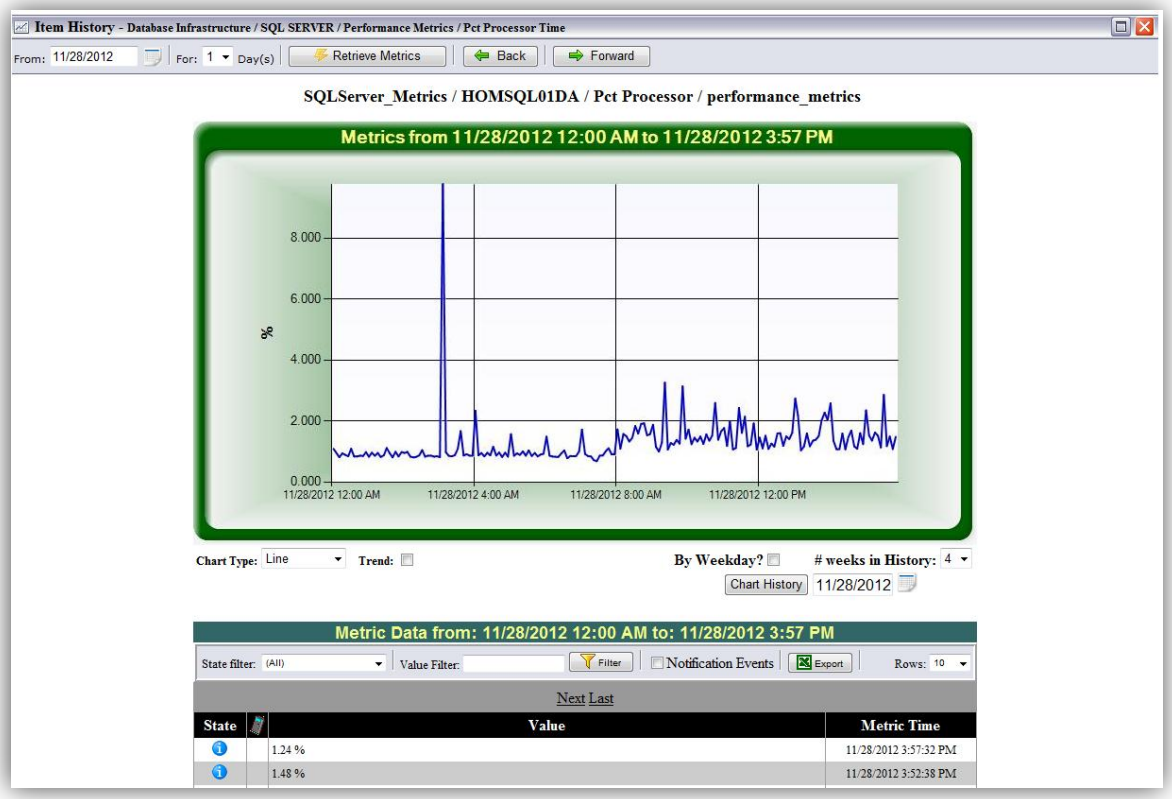

Figure 20: DB Monitoring Dashboard

Figure 20 displays the specific database metrics monitoring value in a graph format. In this way, DBAs can understand better the overall trends of the database metrics.

In the first version of the prototype, only Oracle backup information was monitored because these were the only metrics available for monitoring. We then incorporated other metrics such as memory usage and CPU usage into the dashboard in the second version of the prototype. In the third version, we added metrics for SQL Server databases to make the prototype a cross-platform monitoring tool. In the last version, we added DB2 database platform to the prototype and reorganized the way that the metrics are presented.

Feasibility Analysis

The feasibility analysis below demonstrates that the prototype design above is positioned well for continuous operations success and improvement at Hanover. Our project's low risk profile will deliver direct value without integration concerns to Hanover.

Technical Feasibility

The cross-platform Database Monitoring System is technologically feasible with a moderately low level of risk. Hanover's cross-platform system familiarity risk is low.

- The DBA Team has strong knowledge of the company's existing platform-specific database monitoring tools and has clear requirements for the new integrated system.
- Hanover's Dashboard Development Team has ample development experience with the company's existing BSM including very basic database monitoring capabilities, but it has not worked with cross-platform database monitoring.
- The new system will be developed upon the current BSM system.
- Various mature market products can act as a reference.

Hanover's risk regarding familiarity with the technology is moderately low.

- The Dashboard Development Team has some knowledge of SQL data retrieval.
- DBAs already have the required SQLs that are already imbedded in scripts used for monitoring individual database platforms.
- Our group will help coordinate the development process.

The project size is considered medium risk.

- The project team will likely consist of 10 or fewer people (not including our group).

- The layout and functionality of this system will be similar to those of the BSM.
- The project time frame is somewhat critical as our group only has 9 weeks to finish the prototype.

The compatibility with Hanover's existing technical infrastructure should be fine.

- Our solution uses health and performance information generated by existing individual database monitoring tools.
- SQL data retrieval works for all the database platforms.

Economic Feasibility

To perform a cost/benefit analysis, we made the following assumptions. We assume that the newly developed system will help the DBAs reduce monitoring time spent by 1 hour per day per database platform. We also assume that DBAs receive a salary rate of $40/hour (Salary.com, 2012). Therefore, we have $40/hour * 1 hour * 4 platforms * 5 days * 52 weeks = $41,600/year for saved cost in DBAs.

With the new system, database errors will be easier to detect and email alerts will be more efficient improving Hanover's overall operating efficiency. To assess the value of this efficiency improvement, we estimate that the system can further save DBAs half an hour per day per platform. Therefore, Hanover will be able to get intangible benefits of $40/hour * 1 hour * 4 platforms * 5 days * 52 weeks = $20,800/year resulting from this improvement.

There will be two types of costs involved: Development costs and operational costs. For development costs, the dashboard developer spent 10 hours developing the dashboard prototype. Therefore, we estimate that it will take 80 hours to develop the fully functional system. The salary rate for one individual developer is $40/hour (Salary.com,

2012). Therefore, we have the total development costs of $40/hour * 80 hours = $3,200. As an external consulting team from WPI, we charge $10,000 for providing advices and developing the prototypes. For operational costs, we assume that it takes one dashboard person half an hour/day to maintain all the relevant database dashboard metrics. The salary for the dashboard person is $40/hour (Salary.com, 2012). So, we have the total operational costs of $40/hour * 0.5 hour * 5 days * 52 weeks = $5,200.

In addition, the discount factor that we use (2.8%) is the current yield of 30-year U.S. government bonds. Table 3 shows the economic feasibility analysis that we performed. We started the project in September 2012. Therefore, only one quarter of 2012 is counted in the analysis. The cumulative net cash flow shows that Hanover will have -$13,200 cash flows in the last quarter of 2012. Hanover will receive benefits starting the year of 2013. Also, the break-even point indicates that Hanover will be able to breakeven at the end of the first quarter of the year 2013.

Table 3: Economic Feasibility

Benefits	2012	2013	2014	2015	2016	2017	5 YR Total
Saved cost in DBAs		40,467	39,365	38,293	37,250	36,235	191,609
Improved Operational Efficiency		20,233	19,682	19,146	18,625	18,117	95,804
Total Benefits		60,700	59,047	57,439	55,874	54,352	287,413
Costs							
Development Costs							
Development Labor	3,200						3,200
WPI Project Fee	10,000						10,000
Operational Costs							
Operation Labor		5,058	4,921	4,787	4,656	4,529	23,951
Total Costs	13,200	5,058	4,921	4,787	4,656	4,529	37,151
Total Benefits - Total Costs	(13,200)	55,642	54,126	52,652	51,218	49,823	**250,262**
Cumulative Net Cash Flow	(13,200)	42,442	96,569	149,221	200,439	250,262	
Return on Investment	673.63%						
Break-even Point	0.49						

Organizational Feasibility

Granted the positive technical and economic feasibility forecasts above, organizational feasibility is historically the most difficult to accomplish. Although new IT systems may integrate well with a company's operations and economic position, the most important question is whether or not users will accept it and adapt to it?

To measure the organizational feasibility of our project from a direct business value perspective, its strategic alignment can be analyzed. Our project's main goal is to make Hanover's current database monitoring approach increasingly proactive for enhanced end-user (system-user) response time. Hanover's core business competitive competency is its well-recognized end-user response time, making this project value-added and strategically aligned from a future business perspective.

A stakeholder analysis further supports the value-added position of our project's organizational feasibility. During and after implementation, this project will directly affect the HTG and its DBAs. Saving them more time daily (see detailed description in the economic feasibility analysis above), our project will allow the HTG DBA team to proactively monitor the mass database complex. In turn, this will allow them to initiate other tabled business initiatives and progress Hanover further. On the other hand, the new monitoring approach will require positive feedback and user comfort by the DBA team. For this reason, any team-based usage concerns may be a cause for system dismissal and therefore no value-added monitoring benefit gained.

A value-added strategic alignment and planned beneficial stakeholder analysis make our project organizationally well-fit and feasible. These factors further support our project's low to moderate risk feasibility analysis overall. The prototype implementation process can begin due to this analysis and proof of minimal anticipated business risk.

Chapter 6: Implementation of Prototype

Implementation Process

With the designs approved by our sponsor, Brandon Willis, we started to implement the prototype. We first collected all the necessary database metrics and SQL statements from DBAs. We then gave those SQL to Paul Coughlin to be run on SiteScope. The text documents generated by the SiteScope were sent to Henry Farineau who is in charge of the dashboard. Henry parsed all the text into the following six sections: SID, Type, Status, Start Time, End Time, and Elapsed Time. The data from each section is then displayed in the dashboard.

We finished our first version of the prototype for the database backup metrics for the Oracle database platform and reviewed it with DBAs to make sure that it meets their expectations and satisfies all the standards. We iterated to the second phase of the prototype development by adding new metrics to the first prototype. Then, we added metrics from SQL Server database platform to the dashboard, making it a cross-platform monitoring solution.

Finally, we performed statistical analysis on the historical data and came up with appropriate threshold for certain metrics. We discussed these threshold values with DBAs to make sure that they understand our approach and agree with the values. We also discussed with DBAs the type of alert that they would like to receive if a certain threshold value is achieved.

Implementation of Requirements

The prototype we have developed implements most of the requirements documented in Chapter 5: Design of the To-Be System. Below is the implementation summary corresponding to those requirements:

1. The prototype is built upon the existing Business Service Management system, which is used by DBAs currently. For test purposes, it consolidates only Oracle Database and Microsoft SQL Server into one centralized dashboard.
2. The prototype gets statistics feeds from the HP Sitescope system periodically. It is able to display the information both textually and graphically, if applicable. This feature makes it easier for non-database-experts to get information.
3. The prototype allows users to easily configure alert thresholds for different database metrics. DBAs will be notified on screen, by email or by paging depending on the severity of the event, based on criteria supplied in advance.
4. HP SiteScope stores retrieved statistics in a plain text file. DBAs can perform trend analysis and review problem diagnostics with this history log, even when the database management system is down.
5. The prototype displays system overview by default. To get more details, DBAs can drill down to categorized detailed information, such as backup types for database backup module, with simple clicks.
6. The prototype is completely browser based and is only internally accessible. The prototype currently does not support interface customization, which might be realized in future iterations.

In conclusion, our prototype proves the feasibility and usability of the proposed system and serves as a solid groundwork for further development.

Database Threshold Setting Statistical Model

EWMA Control Chart for Database Monitoring

Control chart is an on-line process-monitoring technique widely used to detect the occurrence of process shifts by using a graphical display of a quality characteristic that has been measured or computed from a sample versus the sample number or time. The chart contains a center line that represents the average value of the quality characteristic corresponding to the in-control state. Two other horizontal lines, called the upper control limit (UCL) and the lower control limit (LCL), are also shown on the chart. These control limits are chosen so that if the process is in control, nearly all of the sample points will fall between them (Montgomery, 2009). The control chart approach is applicable in this project because database monitoring is essentially the monitoring of a database operation process.

The control chart we choose to adopt, EWMA (or exponentially-weighted moving average), is a specific type of control chart used to monitor either variables or attributes-type data based on the monitored business or industrial process's entire history of output (6.3.2.4. EWMA Control Charts). While other control charts treat rational subgroups of samples individually, the EWMA chart tracks the exponentially-weighted moving average of all prior sample means. EWMA weights samples in geometrically decreasing order so that the most recent samples are weighted most highly while those distant samples contribute very little.

One of the most important and distinctive characteristics of the EWMA method is its relative robustness in the face of non-normally distributed quality data. As EWMA can be viewed as a weighted average of all past and current observations, it is very insensitive

to the normality assumption. Therefore, EWMA is almost a perfectly nonparametric (distribution-free) procedure (Montgomery, 2009).

Implementation of the Model

Currently, the DBAs in Hanover set database thresholds based on experience only. Our team thinks this approach might be too subjective to be optimal, so we would like to recommend a statistical method to help them make this kind of decisions.

The model our group comes up with is based on the concept of control chart. It helps DBAs analyze database threshold through statistical reference from history data rather than by solely replying on their past experience and subjective inference. Our model consists of two major components: the normality test process to check the authority of the model and the calculation of upper control limit to attain threshold for database metrics.

Normality test requires sorted history data as the input. We perform it with the help of Microsoft Excel. The test first calculates the cumulative proportion of each data point (CP_n) by using the formula:

CP_n=(1/(sample size+1) n=1

CP_n=(1/(sample size+1)+CP_{n-1} n>1

The inverse of the standard normal cumulative distribution for each cumulative proportion is then computed automatically by using a predefined function, such as the NORMSINV function in Excel. Finally, the process outputs a normal quantile plot with sample data as X values and NORMSINV results as Y values. DBAs should interpret this

graph as: the more the plot appears to be linear, the more normal the data are and thus the more confidence or predicting power our model would have.

Exponentially Weighted Moving Average (EWMA) is the statistic method we employ to calculate the upper control limit, beyond which the database system might be actually running in an abnormal state. This method starts with calculating the estimated variance of the EWMA statistic, which is $s^2_{ewma} = (\lambda/(2- \lambda)) s^2$ when the number of historical observations is not small, where λ determines the rate at which 'older' data enter into the calculation of the EWMA statistic and s is the standard deviation derived from the historical data. The value of λ is usually set between 0.2 and 0.3 (Hunter, 1986) and we use 0.3 in our model to give recent measurement more weight.

The whole model concludes by returning the upper control limit, $EWMA_0 + 3s_{ewma}$, where $EWMA_0$ is the average of the historical measurements.

Demonstration with CPU usage data

Firstly, we generate the normal quantile plot shown in Figure 21 using 2 weeks of historical data from Oracle database log. The plot's sample data excel worksheet is attached in Appendix D.

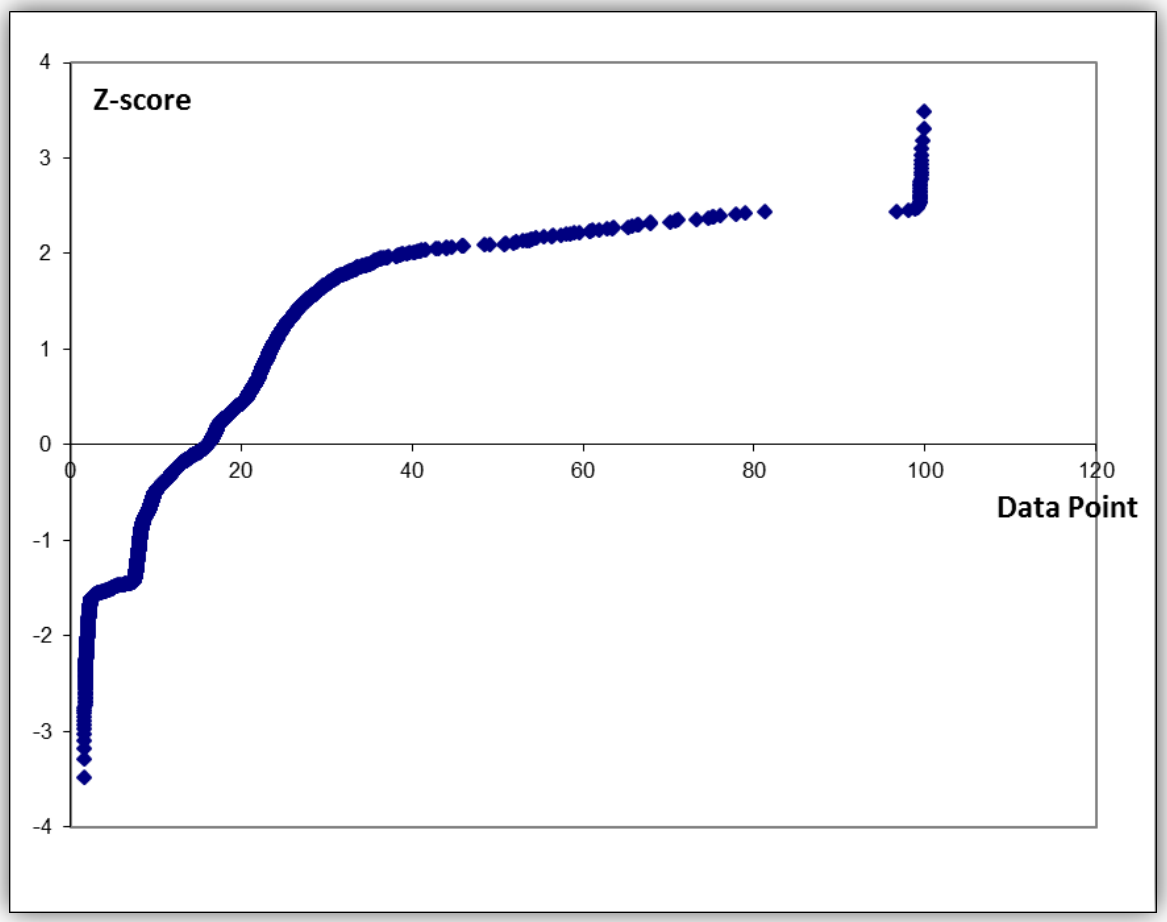

Figure 21: Normal Quantile Plot

We can tell from the above plot that the predicting power of our model will be limited by the non-normality of the observations. Therefore, we recommend DBAs take their experience into more consideration while looking at the model result.

Then we calculate the threshold as:

s2ewma = (λ/(2- λ)) s2=0.3*1.7*132.042=67.341

UCL= EWMA0 + 3sewma=16.747+3*11.491=41.365

In conclusion, our model recommends setting CPU usage alert threshold to be 41.365 in this case. A sample control chart with this threshold is shown is Figure 22.

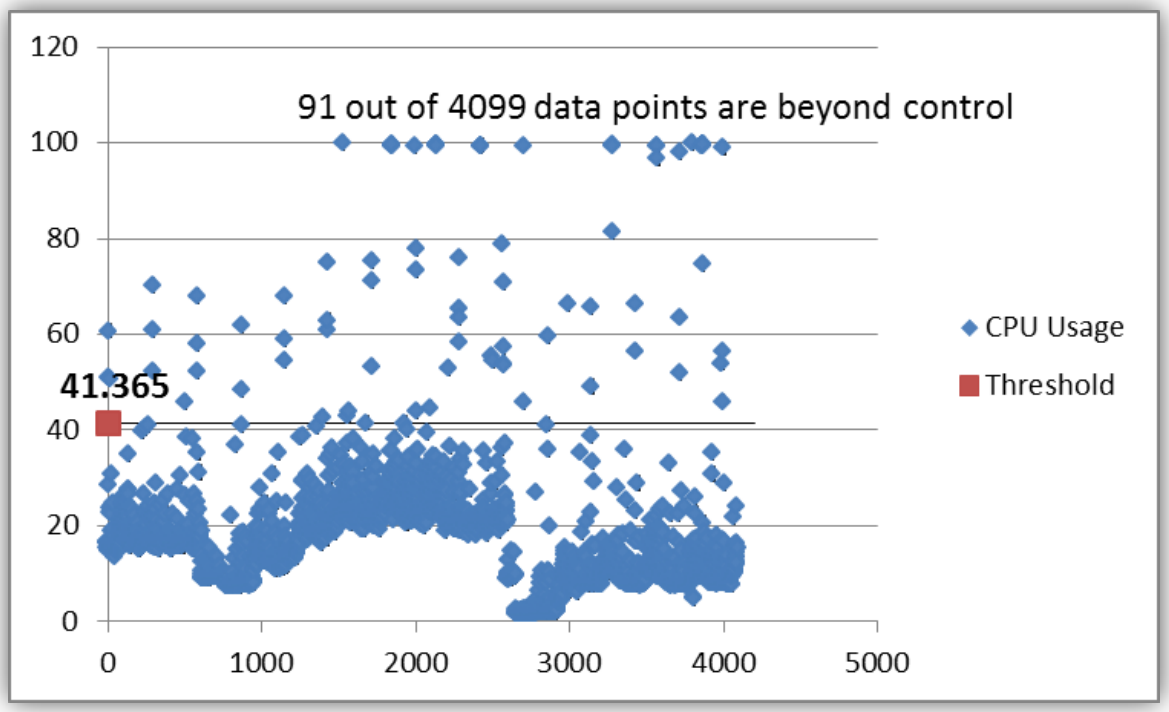

Figure 22: Sample Control Chart

Prototype Feedback

We created an evaluation form (attached in Appendix E) for the dashboard's usefulness and ease of use to gather direct feedback from the DBAs. We received four responses in total, three from individual DBAs and another from our sponsor Brandon Willis. The results are from a scale of 1-7, 1 being unlikely and 7 being likely. The analysis and summary of this feedback is presented in Table 4.

Table 4: Response Summary

Usefulness				Ease of Use		
	Average	StDevP.			Average	StDevP.
Help me complete tasks more quickly	5.25	0.83		Easy to learn how to use	7.00	0.00
Improve my job performance	4.75	0.83		Able to do what I need with ease	6.50	0.50
Increase my productivity	4.00	1.22		Clear and understandable to use	6.50	0.50
Enhance my workplace effectiveness	4.75	1.30		Flexible to interact with	6.25	0.83
Make my job easier	5.00	0.71		A tool easy to gain skill within	6.50	0.50
Be useful in my job	5.25	1.09		Easy to use	7.00	0.00

From the 48 total data points we received from this evaluation (4 responses * 12 data points/response), we more intuitively present the data again in Figure 23 to better represent usefulness and ease of use.

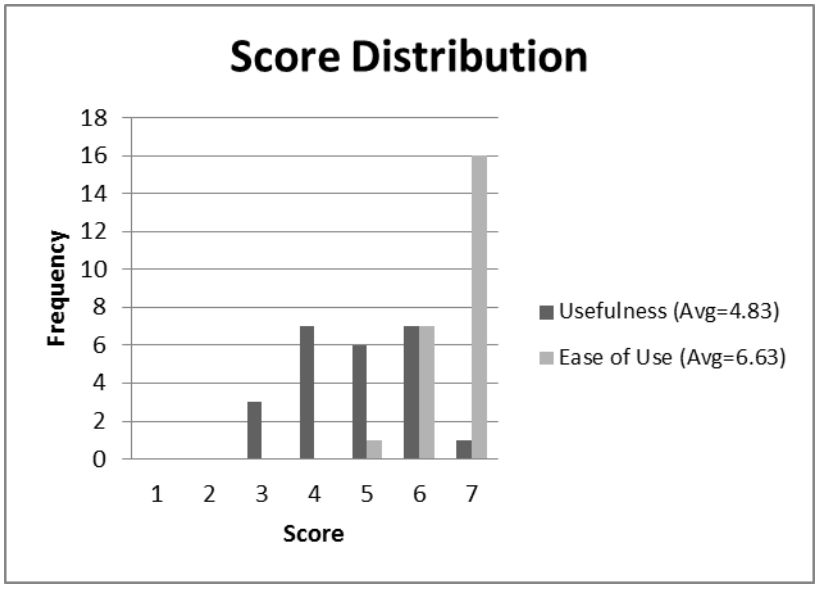

Figure 23: Response Summary Chart

Reliability Analysis

We further analyzed this data using Cronbach's alpha, a coefficient of internal consistency, to test for reliability. The Cronbach's alpha for usefulness is 0.95 and it is 0.87 for ease of use. According to P. Kline's scale (Kline, 1999), the internal consistency for usefulness (0.95) is "excellent" and that for ease of use (0.87) is "good". Therefore, we determined that our survey results are indeed reliable.

Conclusion

The responses from our primary users indicate that our dashboard will help them complete tasks more quickly and efficiently and help make their jobs easier. However, our primary users also foresee a relatively low increase in their productivity as related to our dashboard. Regarding our dashboard's ease of use, each of our primary users gave fairly high scores, indicating that it is easy to learn how to use our dashboard and that it is flexible to interact with.

In addition, we consider the dashboard to deliver both detailed information for DBAs and high-level information for Managers/Executives. All DBAs indicated that they are likely to both use the new dashboard and continue to use the current monitoring methods. Brandon Willis, a HTG Director, indicated that he now plans to only use our dashboard.

From the feedback section, we also learned that some DBAs would like to see even more databases added to the monitoring efforts. Also, there was common feedback for a more iconized and visual alert-based user interface. All of these feedback points are valuable and able to be implemented in future iterations of the dashboard.

Chapter 7: Recommendations

An Agile Hanover

Our team recognized a clear need for organizational changes to support the intangible facets of our targeted business value. Our dashboard aims to increase client application uptime by improving database monitoring and error identification tactics. We further recommend Hanover adopt an agile workplace environment to enable holistic progression towards a proactive database monitoring business environment. This adoption will support Hanover's evident business growth by enabling its technology infrastructure to best integrate with the increasing amount of information and speed that it is processed. Ideally, an agile Hanover will enable streamlined communication and result in enhanced performance.

Current State

Hanover currently provides a typical office building environment of block-cube staging. The panoramic view in Figure 24, in addition to the views in Figures 25-27, show this layout within one of the HTG operation areas.

Figure 24: Panoramic View of HTG Operations Area

Figure 25: Diagonal View of HTG Operations Area

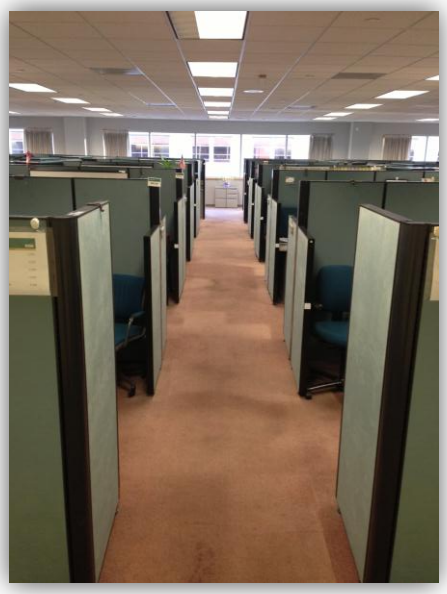

Figure 26: View of HTG Operation Area Row

Figure 27: View of HTG Operations Area Cubicle

With these visual aids as a reference, it is clear that open communication is limited by this block-cube layout thereby constraining continuous performance improvement.

Future State

Referencing the IT agile workplace renovation at Unilever Americas in Englewood Cliffs, NJ, different design and technology can "allow so much mobility," drive open communication initiatives and drastically improve performance metrics (Architizer, 2009). Intelligent design and utilization of any space can also increase total worker population capacity while decreasing individual worker space and enhancing morale. At Unilever, for example, the agile renovation brought worker capacity up to 120 from 68, reduced individual space by over 50% and when surveyed 6 months later, found

that workers "felt comfortable and connected to more people than before." The views in Figures 28-30 showcase the agile transformation at Unilever.

Figure 28: View of Agile Workspace Design at Unilever

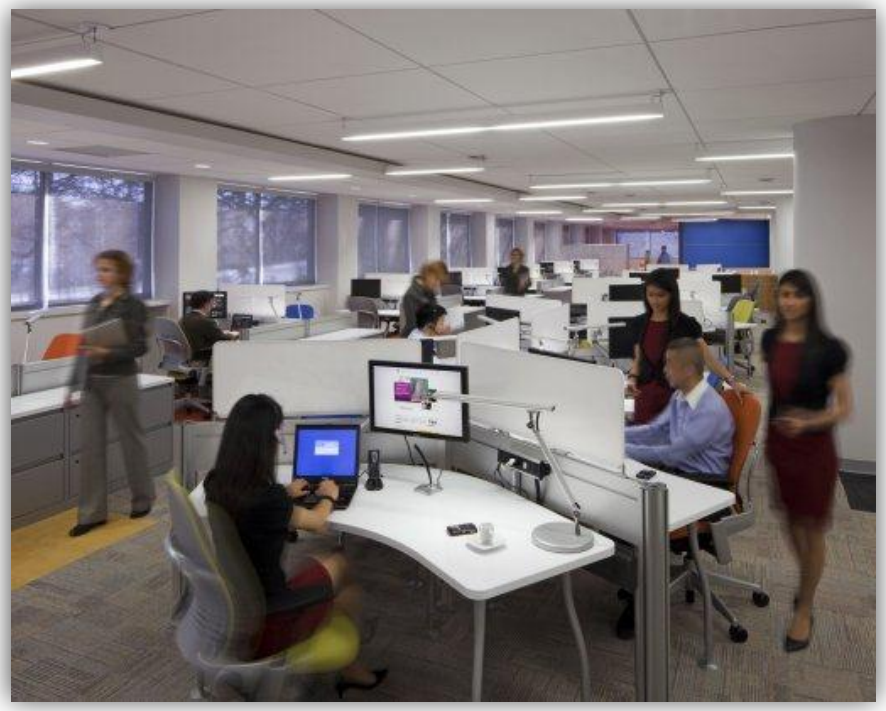

Figure 29: View of collaborative Agile desk space at Unilever

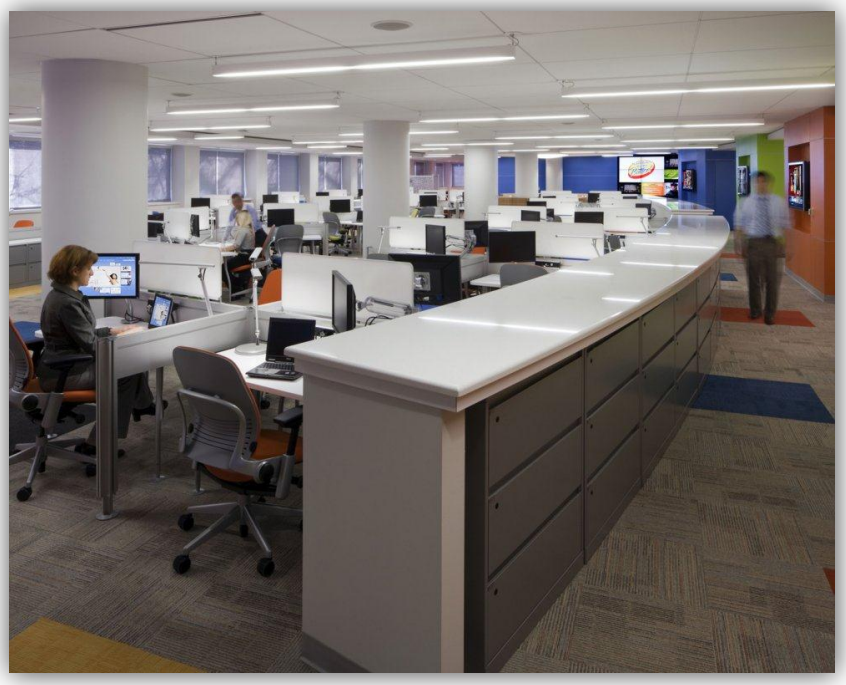

Figure 30: View of open Agile workspace at Unilever

An agile workplace transformation at Hanover would replicate the results found at Unilever and enable the HTG DBA team to share information in an easy and centralized fashion. This intangible business value would complement the technical dashboard implementation our team achieved and further progress Hanover towards a proactive database monitoring approach.

Conclusion

In this project, our team started with the general goal to design and develop a cross-platform database monitoring system. To begin, we researched the four relevant database platforms and related industry best practices for integrated database monitoring. We then analyzed Hanover's current database monitoring approach and discussed what a future state might require with both management and DBAs. Afterwards, we designed and developed a prototype accordingly by working together with the system monitoring and the dashboard team. We also devised a statistical model to help Hanover set database thresholds. Finally, we further recommended some organizational changes to help Hanover adopt the new centralized monitoring approach.

By doing this project, the three of us were able to gain significant growth in both technical and nontechnical areas. First, we gained significant knowledge from our deep-dive research into the four major database platforms currently used by Hanover. We understood the basic structure, the SQLs and the monitoring best practices for Oracle Database, Microsoft SQL Server, DB2 for LUW and DB2 for z/OS. We also had the chance to discover their differences ourselves by integrating them into one monitoring system. Second, we better understood the knowledge we had acquired at WPI, such as statistics, by applying it to solve real business technology value problems at Hanover. Such experience will help to prepare us for future careers. Third, we improved our time management and prioritization skills. We each had to multitask most of the time by taking part in both report writing and prototype development. As a countermeasure, we always tried to carefully plan and arrange our schedule and activities well in advance. Finally, we managed to successfully complete this project in the advanced 2-term timeframe and meet or exceed all expectations. Last but not least, this project provided us

with a great chance to practice our communication skills. Not only did we communicate and cooperate within our team, but also coordinated well with various stakeholders, such as the DBA team, the dashboard team and Hanover management and Executives.

In conclusion, our Hanover MQP project has been both successful and rewarding.

Appendix A: List of Critical Databases

Application	PL	BI/CL	Corp	CS Breakdown
ACT/AVENUES COMMERCIAL TECH		X		
AUSUM		X		
BILLING CENTER	X			
BOND DIRECT		X		
CAAMS		X		
CALL CENTER PORTAL	X			
CERIDIAN			X	HR
CITRIX		X	X	All Corporate
COMMISSIONS	X			
CSS			X	Claims
FACSYS		X		
HCS	X		X	Claims
HYPERION ESSBASE			X	Finance
IBANK		X		
INTRANET / INTERNET		X	X	Marketing
ISONET		X		
IVR	X	X		
LEGAL FILES			X	Claims
LYNX	X			
MY HANOVER POLICY (MHP)	X		X	
OUTLOOK		X	X	All Corporate
PAMS			X	Finance
PEOPLESOFT			X	Finance
PLEXUS / IPS		X		
PLUS		X		
PMS ALL LINES	X	X	X	Claims
POLICYBUILDER		X		
POLKVIN		X		
POS ALL LINES	X	X	X	Marketing
RAM (Campania)		X		
RTR - REAL TIME RATER	X			
SALESFORCE			X	Marketing
SAMS		X		
SERVICE CENTER		X		
THE AGENCY PLACE (TAP)	X	X	X	Marketing
UNDERWRITING LOSS RUNS		X		
USPS		X		
VICCTIR (HIS)		X		
Application Counts	11	24	15	

core applications
agent facing applications
other significant apps

Appendix B: Data Collection Schedule

August 29th, 2012

- 3-4 PM: John Aragi, Peter Wallace

September 5th, 2012

- 10-11 AM: Don Postma, Brandon Willis (Oracle Platform)
- 11-12 AM: Keith Van Riper, Brandon Willis (SQL Server Platform)
- 3-4 PM: John Aragi, Ed Schuster, Peter Wallace, Brandon Willis (DB2 Platform)

September 12th, 2012

- 10-11 AM: Henry Farineau, Brandon Willis (BSM/Monitoring)

September 19th, 2012

- 10-11 AM: Paul Coughlin (System Request/Monitoring)

October 3rd, 2012

- 1:30-2:30 PM: Henry Farineau (SQL for Dashboard)
- 2:30-3:00 PM: Paul Coughlin (SiteScope)

October 19th, 2012

- 10:00-11:00 AM: Henry Farineau (DB Dashboard Planning)

October 24th, 2012

- 9:00-10:00 AM: Operations Center Tour

November 5th, 2012

- 1:00-1:30 PM: Data Center Tour
- 1:30-2:00 PM: Henry Farineau (DB Dashboard Review)
- 2:00-3:00 PM: Keith Van Riper (Server Statistics Review)

November 7th, 2012

- 2:00-3:00 PM: Brandon Willis (1 month scope out project discussion)

November 21st, 2012

- 10:00-11:00 AM: Paul Coughlin (CPU data retrieval)

November 28th, 2012

- 1:30-2:00 PM: Henry Farineau (Dashboard technical review)
- 2:00-2:30 PM: Paul Coughlin (Dashboard SiteScope review)

December 3rd, 2012

- 1:00-2:00 PM: Brandon Willis and Henry Farineau (Final technical review)

December 5th, 2012

- 4:00-5:00 PM: Brandon Willis (Final report review)

December 12th, 2012

- 10:00-11:00 AM: Brandon Willis, Karin Winsky, DBA team, and Professor Strong (Final MQP presentation, formal)

Appendix C: SQL Statements to Retrieve Monitoring Statistics

Oracle

The historical data is stored in the ORASTAT1 database located on lxdv2mis. The logon/password is dbstatro/dbstatro. It has been granted access to the table accessed in the queries contained in this document. Unless noted in the queries, all of the tables are in the STORSTAT schema.

Historical DB backup information:

The data is stored in DB_BACKUP_STAT. This table has the history data for the Oracle backups for test and production. The types of backups include ONLINE, INCREMETAL Level 0 & 1, ARCHIVE LOGS, or Exports.

List latest backups:

select * from storstat.latest_backups;

List last 7 days backups:

select * from storstat. last_7_days_backups;

List last 30 days backups:

select * from storstat. Last_30_days_backups;

List all backups for all actively used databases:

```
select a.*,fnc_elapsed_time(a.start_time,a.end_time) elapsed_time from
storstat.db_backup_stat a, storstat.db_info b
where a.sid = b.sid
  and b.db_usage not in ('RETIRED','ACTIVE')
order by a.sid,a.start_time;
```

List all backups for a single database:

```
select a.*,fnc_elapsed_time(a.start_time,a.end_time) elapsed_time from db_backup_stat a
where sid = '&DB_NAME' order by start_time;
```

Historical DB Storage information:

This data is stored in the PERFSTAT.DB_TS_STAT table. The are Excel spreadsheets on the DBA Wiki (http://apc.allmerica.com/dba/KB/Oracle%20DBA%20KB%20Wiki/Home.aspx) that have Oracle database storage by month and year, i.e., 2012 Monthly DataBase Growth Stats.xls.

To list the last months database growth stats:

```
select * from last_months_db_growth;
```

To list the growth trends of a database execute this PL/SQL block:

```
prompt
ACCEPT sid CHAR prompt 'Which ORACLE_SID do you want monthly storage trend
for? '
prompt

declare
  v_collection_date   date;
  v_total_size        number;
  v_begin_year        number;
  v_end_year          number;
  v_growth            number := 0;
  v_previous_size     number := 0;

begin

  select min(to_char(collection_date,'YYYY')),max(to_char(collection_date,'YYYY'))
into v_begin_year,v_end_year
  from db_ts_stat
  where sid = upper('&&DB_NAME');

  dbms_output.put_line('  MONTH          BYTES               GROWTH');

  for y in v_begin_year..v_end_year loop

    for m in 1..12 loop

      select max(collection_date) into v_collection_date
      from db_ts_stat
      where sid = upper('&&DB_NAME')
       and collection_date between to_date(m || '/01/' || y,'MM/DD/YYYY')
                        and last_day(to_date(m || '/01/' || y,'MM/DD/YYYY'));
```

```
    select sum(total_size) into v_total_size
    from db_ts_stat
    where sid = upper('&&DB_NAME')
      and collection_date = v_collection_date;

    if v_previous_size = 0
    then
      dbms_output.put_line(to_char(v_collection_date,'MM/DD/YYYY') || ' ' ||
to_char(v_total_size,'999,999,999,999,999'));
      v_previous_size := v_total_size;
    else
      v_growth := v_total_size - v_previous_size;
      dbms_output.put_line(to_char(v_collection_date,'MM/DD/YYYY') || ' ' ||
                    to_char(v_total_size,'999,999,999,999,999') || ' ' ||
                    to_char(v_growth,'999,999,999,999,999'));
      v_previous_size := v_total_size;
    end if;

   end loop;

  end loop;

end;
/
```

DB Portfolio Information:

The data is stored in the STORSTAT.DB_INFO table:

```
select * from db_info where db_usage != 'RETIRED' order by db_type desc,
sid,db_usage;
```

Realtime Information:

These queries will have to be run in each individual Oracle database.

Database status:

```
select database_status || ' and ' || status from v$instance;
```

Top 25 SQL by run time:

```
select *
```

```
from (select substr(sql_text,1,80) sql_text,elapsed_time/1000
elapsed_seconds,rows_processed,rows_processed/(elapsed_time/1000)
"ROWS/SECOND",hash_value,address
     from v$sqlarea
     where elapsed_time > 1000
     order by elapsed_time desc)
where rownum <= 25;
```

Top 25 SQL by executions:

```
select *
from (select substr(sql_text,1,80)
sql_text,executions,rows_processed,rows_processed/executions
"Rows/Exec",hash_value,address
     from v$sqlarea
     where executions > 100
     order by executions desc)
where rownum <= 25;
```

Application info:

This query will need to be executed in each individual Oracle database.

```
select count(*) concurrent_users
from v$session
   ,v$process
where v$session.paddr = v$process.addr
  and v$process.background is null;
```

There is also a table, PERFSTAT.ORACLE_LICENSE_HIST, that stores the daily connection high water mark.

*Select * from PERFTAT.ORACLE_LICENSE_HIST order by HIST_DATE;*

Server Statistics:

We, the Oracle DBAs, do not currently store any historical server related statistics.

SQL Server

```
-- Historical DB backup information.
-- For those that don't use LiteSpeed, the query is different for different
-- Versions of SQL.
-- Has to be run on each SQL Server instance.
-- For those backed up with LiteSpeed (the majority of DB's) the query is:

select D.DatabaseName, AT.ActivityTypeName,
     A.StartTime,A.FinishTime,
```

```
     Case A.BackupSize when 0 then 'Failure' else 'Success' end as Status
from LiteSpeedLocal..LiteSpeedActivity A
 inner join LiteSpeedLocal..LiteSpeedDatabase D
  on A.DatabaseID = D.DatabaseID
 inner join LiteSpeedLocal..LiteSpeedActivityType AT
  on A.ActivityTypeID = AT.ActivityTypeID
order by DatabaseName, StartTime

-- For native SQL Server backups, there no record of failed backups.
-- For native backups, the query is:

select database_name, backup_start_date, backup_finish_date,
Case type when 'D' then 'Full' when 'I' then 'Diff'
         when 'L' then 'Log' end as Backup_Type
from msdb..backupset
order by database_name, backup_start_date

-- Historical Capacity information - Growth statistics by DB.
-- Run against HOMSQL10DA DBA database.
-- BackupStatsDly table two years of historical data at one week intervals.

select Stat_dt, Instance_nm, DB_nm, DB_size, Log_size
from DBA..Summary_space_stats
order by Instance_nm, DB_nm, Stat_dt

-- DB Portfolio - DB name, server and DB version.
-- Run against HOMSQL10DA DBA database.
-- BackupStatsDly table does not contain historical data.

select distinct BS.Instance_nm, BS.db_nm, substring(sql_vers,1,4) SQL_Vers
from DBA..BackupStatsDly BS
inner join DBA..Instance_info II
on BS.instance_nm = II.Instance_nm
where II.sql_vers in('6.5','2000','2000-LS',
  '2005','2005-LS','2008','2008-LS','2012','2012-LS')
order by BS.Instance_nm, BS.db_nm

-- Validation that DB's are online.
-- Has to be run on each SQL Server instance.

-- For SQL 2000 and prior versions:

select name DBNm ,
   State = case
      when (status & 2) = 2 then 'IN_TRANSITION'
      when (status & 64) = 64 then 'PRE_RECOVERY'
```

```
      when (status & 128) = 128 then 'RECOVERING'
      when (status & 256) = 256 then 'NOT_RECOVERED'
      when (status & 512) = 512 then 'OFFLINE'
      when (status & 32768) = 32768 then 'EMERGENCY_MODE'
      else 'ONLINE' end
 from master..sysdatabases
order by DBNm

-- For SQL 2005 and later versions:

select name DBNm, state_desc State
 from master.sys.databases
 order by DBNm

-- File system/Storage warning thresholds.
-- Run against HOMSQL10DA DBA database.
-- The default warning threshold is 80% used.  Exceptions can be seen as follows:

select Instance_nm, Drv_letter, max_use_pct
from DBA..drive_max_use

-- Historical, daily(not real time) drive usage can be seen as follows:

select Stat_dt, Instance_nm, Drv_letter, Total_space, Available_space
from DBA..volumestats
order by Instance_nm, Drv_letter, Stat_dt

-- Top run time SQL statements.
-- Has to be run on each SQL Server instance.

select top 50
 qs.total_elapsed_time/1000 "total_elapsed_time(ms)",
 (qs.total_elapsed_time/1000)/qs.execution_count "avg_elapsed_time(ms)",
 qs.execution_count,
 (qs.total_logical_reads + qs.total_logical_writes) / qs.execution_count "Avg I/O",
 substring(st.text, (qs.statement_start_offset/2)+1
    , ((case qs.statement_end_offset when -1 then datalength(st.text)
     else qs.statement_end_offset end - qs.statement_start_offset)/2) + 1) as statement_text,
 db_name(st.dbid) as DB_nm
from sys.dm_exec_query_stats as qs
 cross apply sys.dm_exec_sql_text(qs.sql_handle) as st
order by
 qs.total_elapsed_time desc

-- Top executed SQL statements.
-- Has to be run on each SQL Server instance.
```

```
select top 50
 qs.total_elapsed_time/1000 "total_elapsed_time(ms)",
 (qs.total_elapsed_time/1000)/qs.execution_count "avg_elapsed_time(ms)",
 qs.execution_count,
 (qs.total_logical_reads + qs.total_logical_writes) / qs.execution_count "Avg I/O",
 substring(st.text, (qs.statement_start_offset/2)+1
    , ((case qs.statement_end_offset when -1 then datalength(st.text)
     else qs.statement_end_offset end - qs.statement_start_offset)/2) + 1) as statement_text,
 db_name(st.dbid) as DB_nm
from sys.dm_exec_query_stats as qs
 cross apply sys.dm_exec_sql_text(qs.sql_handle) as st
order by
 qs.execution_count desc

-- Number of Concurrent users per DB.
-- I assume this is the number of users currently using a database?
-- If so, we don't currently have that information.

-- Server Statistics for Distributed Systems.
-- Run against the DBAmon database on each instance.
-- PerfStats table contains two weeks of historical data in five minute intervals.
-- PerfStats_hist table contains a year of historical data in one hour intervals.

select TS, Pct_Processor_Time,
       Target_Server_Memory_KB, Total_Server_Memory_KB,
       Page_lookups_sec, Page_reads_sec, Page_writes_sec,
       Pct_Disk_Time, Avg_Disk_Queue_Length
  from PerfStats
order by TS
```

Appendix D: CPU Usage Data

The first 100 of 4,099 total records are shown to demonstrate enough data and maintain a reasonable space allotment.

Record No.	Enter SORTED Sample Data Below	Cumu-lative Proport.	*z*-score
1	1.663	0	-3.487365135
2	1.668	0	-3.297467468
3	1.679	0	-3.181842329
4	1.68	0	-3.097558393
5	1.682	1/820	-3.030805337
6	1.689	1/683	-2.975320139
7	1.7	1/586	-2.927714518
8	1.716	1/512	-2.885942132
9	1.739	2/911	-2.848669656
10	1.742	1/410	-2.814978562
11	1.76	1/373	-2.784208453
12	1.761	2/683	-2.755868166
13	1.778	3/946	-2.729582217
14	1.779	1/293	-2.705056937
15	1.789	3/820	-2.682058187
16	1.795	1/256	-2.6603962
17	1.799	1/241	-2.639914975
18	1.801	4/911	-2.620484669
19	1.804	4/863	-2.601996024
20	1.806	1/205	-2.584356209
21	1.807	4/781	-2.567485667
22	1.811	3/559	-2.551315678
23	1.815	4/713	-2.53578647
24	1.816	5/854	-2.520845717
25	1.817	1/164	-2.506447346
26	1.824	3/473	-2.492550571
27	1.825	6/911	-2.479119106
28	1.827	5/732	-2.466120529
29	1.829	5/707	-2.45352574
30	1.829	3/410	-2.441308527
31	1.83	4/529	-2.429445189
32	1.836	1/128	-2.417914222
33	1.839	4/497	-2.406696057
34	1.851	5/603	-2.395772831
35	1.852	7/820	-2.385128195
36	1.861	8/911	-2.374747142
37	1.861	5/554	-2.364615869
38	1.871	9/971	-2.354721644
39	1.872	8/841	-2.345052703

40	1.873	2/205	-2.33559815
41	1.877	1/100	-2.326347874
42	1.887	8/781	-2.317292476
43	1.888	3/286	-2.308423202
44	1.892	6/559	-2.299731883
45	1.893	9/820	-2.291210889
46	1.902	8/713	-2.282853075
47	1.903	4/349	-2.274651747
48	1.909	5/427	-2.266600621
49	1.912	3/251	-2.25869379
50	1.912	1/82	-2.250925697
51	1.913	5/402	-2.243291102
52	1.92	6/473	-2.235785066
53	1.921	11/851	-2.22840292
54	1.922	13/987	-2.221140253
55	1.927	11/820	-2.213992886
56	1.929	5/366	-2.206956863
57	1.941	1/72	-2.200028429
58	1.955	13/919	-2.193204022
59	1.956	2/139	-2.186480254
60	1.957	3/205	-2.179853905
61	1.966	14/941	-2.17332191
62	1.971	8/529	-2.166881349
63	1.975	13/846	-2.160529437
64	1.978	1/64	-2.154263517
65	1.986	13/820	-2.148081053
66	1.989	8/497	-2.141979621
67	1.99	5/306	-2.135956902
68	1.994	7/422	-2.130010679
69	1.995	12/713	-2.124138828
70	1.997	7/410	-2.118339313
71	1.998	4/231	-2.112610184
72	2.001	17/968	-2.106949569
73	2.009	6/337	-2.101355672
74	2.011	5/277	-2.095826767
75	2.015	3/164	-2.090361196
76	2.015	18/971	-2.084957365
77	2.017	4/213	-2.079613741
78	2.018	16/841	-2.074328847
79	2.032	10/519	-2.069101262
80	2.032	4/205	-2.063929616
81	2.037	13/658	-2.05881259
82	2.044	1/50	-2.053748911
83	2.045	5/247	-2.04873735
84	2.045	5/244	-2.043776723
85	2.046	17/820	-2.038865884
86	2.047	3/143	-2.034003729
87	2.048	8/377	-2.029189188
88	2.062	17/792	-2.024421229
89	2.063	15/691	-2.019698853

90	2.08	9/410	-2.015021093
91	2.083	18/811	-2.010387013
92	2.095	16/713	-2.005795708
93	2.096	12/529	-2.0012463
94	2.098	13/567	-1.996737938
95	2.1	19/820	-1.9922698
96	2.104	17/726	-1.987841085
97	2.113	15/634	-1.983451018
98	2.113	6/251	-1.979098849
99	2.113	12/497	-1.974783847
100	2.115	1/41	-1.970505303

Appendix E: Dashboard Evaluation Form

HTG Database Infrastructure Monitoring Dashboard Evaluation

Name and Title: _____ Date: _____

Objective: The answers you provide below will be used to impartially judge the **usefulness and ease of use** of the new database infrastructure monitoring dashboard.

To access the dashboard, please open your internet browser and type in web URL "MSYS". Select "Database Infrastructure" in the list box and click "Select Application" to launch the dashboard.

1) Please circle each of the platforms that you support, either directly or indirectly:

DB2 DB2 for LUW Oracle SQL Server

2) Please rate the perceived **usefulness** of the new dashboard using a 1-7 scale where 1 denotes unlikely and 7 denotes likely.

   Using the dashboard will:

a) Help me complete tasks more quickly	1	2	3	4	5	6	7
b) Improve my job performance	1	2	3	4	5	6	7
c) Increase my productivity	1	2	3	4	5	6	7
d) Enhance my workplace effectiveness	1	2	3	4	5	6	7
e) Make my job easier	1	2	3	4	5	6	7
f) Be useful in my job	1	2	3	4	5	6	7

3) Please rate the perceived **ease of use** of the new dashboard using a 1-7 scale where 1 denotes unlikely and 7 denotes likely.

   The dashboard is:

a) Easy to learn how to use	1	2	3	4	5	6	7
b) Able to do what I need with ease	1	2	3	4	5	6	7
c) Clear and understandable to use	1	2	3	4	5	6	7
d) Flexible to interact with	1	2	3	4	5	6	7
e) A tool easy to gain skill within	1	2	3	4	5	6	7
f) Easy to use	1	2	3	4	5	6	7

4) Will you use the new dashboard? Please circle “YES” or “NO”.

5) Please circle the statement below that best represents your monitoring tendency.

   a) I am likely to use the new dashboard and will discontinue my use of current monitoring methods.

   b) I am likely to use both the new dashboard and continue to use the current monitoring methods.

   c) I am likely to use the old monitoring methods only.

6) Please provide your feedback or opinion of the new dashboard below as related to its usefulness and ease of use.

_____
_____
_____
_____
_____

Bibliography

(2009). Retrieved December 3, 2012, from Architizer: http://www.architizer.com/en_us/projects/view/unilever-americas-it-agile-workplace-renovation/29915/

6.3.2.4. EWMA Control Charts . (n.d.). Retrieved 11 17, 2012, from NIST/Sematech Engineering Statistics Handbook: http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm

eHow. (n.d.). *Advantages & Disadvantages of Microsoft SQL*. Retrieved 09 18, 2012, from eHow: http://www.ehow.com/list_7228389_advantages-disadvantages-microsoft-sql.html

Gopalakrishnan, K. (2011). *Oracle Database 11g Oracle Real Application Clusters Handbook.* Oracle Press.

Hubpages. (n.d.). *Why Oracle Database is best choice*. Retrieved 09 18, 2012, from Hubpages: http://giteshtrivedi.hubpages.com/hub/Why-Oracle-Database-Administration-is-Primary-Requirement

Hunter, J. S. (1986). The Exponentially Weighted Moving Average. *Journal of Quality Technology*, 203-210.

IBM. (2012). *DB2 Version 9.1 for z/OS*. Retrieved 2012, from IBM Information Center: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z9.doc.perf/src/tpc/db2z_usetools2monitorperformance.htm

IBM. (2012). *DB2 Version 9.7 for Linux, UNIX, and Windows*. Retrieved 2012, from DB2 solution Information Center home: http://pic.dhe.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.mon.doc%2Fdoc%2Fc0021593.html

IBM. (2012). *IBM Data Studio Version 3.1.1*. Retrieved 2012, from IBM InfoCenter: http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp

Kline, P. (1999). *The handbook of psychological testing (2nd ed.).* London: Routledge.

Lawson, S. (2008). *DB2 9 for Z/OS Database Administration: Certification Study Guide.* MC Press.

Microsoft. (n.d.). *Activity Monitor*. Retrieved 09 18, 2012, from Microsoft MSDN Library: http://msdn.microsoft.com/en-us/library/cc879320(SQL.105).aspx

Microsoft. (n.d.). *Monitoring SQL Server Performance*. Retrieved 09 18, 2012, from Microsoft MSDN Library: http://msdn.microsoft.com/en-us/library/ee377023(v=bts.10).aspx

Microsoft. (n.d.). *Performance Monitoring and Tuning Tools*. Retrieved 09 18, 2012, from Microsoft MSDN Library: http://msdn.microsoft.com/en-us/library/ms179428.aspx

Microsoft. (n.d.). *Server Performance and Activity Monitoring How-To Topics*. Retrieved 09 18, 2012, from Microsoft MSDN Library: http://msdn.microsoft.com/en-us/library/ms191511(v=sql.105).aspx

Microsoft. (n.d.). *SQL Trace*. Retrieved 09 18, 2012, from Microsoft MSDN Library: http://msdn.microsoft.com/en-us/library/hh245121.aspx

Montgomery, D. C. (2009). *Introduction to Statistical Quality Control, Sixth Edition.* Hoboken, NJ: John Wiley & Sons, Inc.

Morningstar. (2012, 04 09). *Oracle Is #1 in the RDBMS Market Segment for 2011* . Retrieved 09 18, 2012, from Morningstar:

http://www.morningstar.com/invest/articles/36190-oracle-is-1-the-rdbms-market-segment-2011.html
Neagu, A. (2012). *IBM DB2 9. 7 Advanced Administration Cookbook.* Packt Publishing.
Oracle. (2010). *IBM DB2 Database Metrics*. Retrieved 09 26, 2012, from Oracle: http://docs.oracle.com/cd/E11857_01/em.111/b28748/database_db2.htm
Oracle. (n.d.). *Proactive Database Monitoring*. Retrieved 09 18, 2012, from Oracle: http://docs.oracle.com/cd/B16276_01/doc/server.102/b14196/montune001.htm
Pearson, D. (2010). *Key Methods for Managing Complex Database Environments.* Quest Software, Inc.
Salary.com. (2012). *Database Administrator Salary*. Retrieved 11 19, 2012, from Salary.com: http://swz.salary.com/SalaryWizard/Database-Administrator-Salary-Details.aspx
Unisphere Research. (2011). *DATA CROSS-CURRENTS: 2011 SURVEY ON CROSS-PLATFORM DATABASE ADMINISTRATION.*
WebFinance, Inc. (2012). *What is total cost of ownership (TCO)? definition and meaning*. Retrieved 2012, from BusinessDictionary.com: http://www.businessdictionary.com/definition/total-cost-of-ownership-TCO.html#ixzz26C3crc1S