A Case Study in Integrated Quality Assurance for Performance Management Systems

Liam Peyton, Bo Zhan and Bernard Stepien

School of Information Technology and Engineering,
University of Ottawa, 800 King Edward Avenue, Ottawa, ON, Canada

Abstract. On-line enterprise applications that are used by thousands of geographically dispersed users present special challenges for quality assurance. A case study of a hospital performance management system is used to investigate the specific issues of architectural complexity, dynamic change, and access to sensitive data in test environments. A quality assurance framework is proposed that integrates with and leverages the performance management system itself. As well, a data generation tool suitable for the requirements of testing performance management systems has been built that addresses limitations in similar commercially available tools.

1 Introduction

Increasingly, large organizations are deploying complex on-line enterprise applications that are used to run and manage critical business processes [5]. These applications often run on a network of centralized servers in a service oriented architecture (SOA) [12], and are accessed by thousands of geographically dispersed users via browser-based interfaces. In this case study, we examine some of the specific requirements and challenges faced in providing a quality assurance framework for a performance management and reporting system deployed at a large teaching hospital. This hospital has thousands of health care workers and maintains a data warehouse containing hundreds of millions of records related to patient treatment. The performance management system in the case study uses typical off the shelf components that are used in other hospitals, and which are used quite regularly in deployments for the biggest companies in the world (e.g. members of Forbes magazines "Global 2000").

Business performance management (BPM) enables an organization to understand the status of business processes across business and IT, in context against goals and trends, in order to improve business operations [6]. BPM is often used to enable specific management initiatives like balanced scorecard, Total Quality Management, Six Sigma, and Sarbanes-Oxley. Enterprise performance management systems that support BPM provide services to manage and report on data collected from across the organization's business operations into enterprise-wide data warehouses and data marts.

Peyton L., Zhan B. and Stepien B. (2008).
A Case Study in Integrated Quality Assurance for Performance Management Systems.
In *Proceedings of the 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 129-138
DOI: 10.5220/0001743701290138

A data warehouse is a subject-oriented, integrated, time-variant, non-volatile collection of information that is optimized for monitoring and analysis in support of management's decision-making processes [7]. Hospitals have been slower to adopt such technology than private enterprise, but it is now an active focus area [2]. Typically, the user interface for accessing the reporting, monitoring, and analysis functionality of the performance management system is a browser-based "portal". The "portal" view can be customized for each user to define their particular business view of enterprise data and highlight the individual reports and analyses that are most relevant to them. A hospital performance management system presents special challenges for quality assurance. In our case study we have considered the following:

- Architectural Complexity of Enterprise Performance Management

  Simplistic testing at the level of the user interface can detect quality issues, but it is not sufficient to determine root causes given the system's complex architecture. Moreover, each service in the architecture typically has detailed logs [10]. To leverage these, though, a mechanism is needed in order to correlate such a wide variety of data sources with quality issues experienced by users.
- Dynamic Change that affects Quality

  Even though it is assembled from off the shelf components, the quality of such a system (usefulness, performance, reliability, etc.) is affected by the volume and structure of data, and the specifications of both user interface "portals" and the reports viewed in them. Quality assurance must be managed in production the same as when testing.
- Sensitivity of Data (privacy)

  Often the data in the production environment is highly sensitive and access to it is strictly regulated by privacy law. To accurately assess quality in a test environment, there must be a mechanism to generate "test" data that is not sensitive but still exhibits similar characteristics to the "real" sensitive data when running reports.

In addressing these issues in our health performance management case study, we have prototyped a framework for collecting log files into a data warehouse in order to leverage the facilities of the performance management system itself to monitor, report and analyze quality. We have also developed a tool to generate "test" non-sensitive data that will exhibit the same performance characteristics as "real" sensitive data.

2 Case Study

The hospital in our case study is one of Canada's bigger teaching hospitals with 10,000 employees and over 1,000 physicians. In a typical year, there are over 100,000 emergency visits and 10 million lab tests. The data warehouse has over 100 million records, and adds tens **of thousands of new records a week [4].**

For privacy reasons, we are unable to have access to the "real" data in production. Instead, we generate "test" but similar data in comparable volumes. The data base schemas are the same. The off the shelf components used are either the same or are being considered for introduction into the production environment. We use a testing tool, OpenSTA [9], to automate test scripts and simulate up to 50 concurrent requests

over extended periods of time running reports to simulate the load that must be handled in production. Our focus has not been on testing the accuracy of the reports, but rather on testing the quality of service provided by the system in terms of:

- Usefulness - how much of the system is used by which users with what frequency.
- Performance - average response time under different levels of load.
- Reliability - monitoring system status and component failures over time under normal and extreme load.

3 Quality Assurance Framework

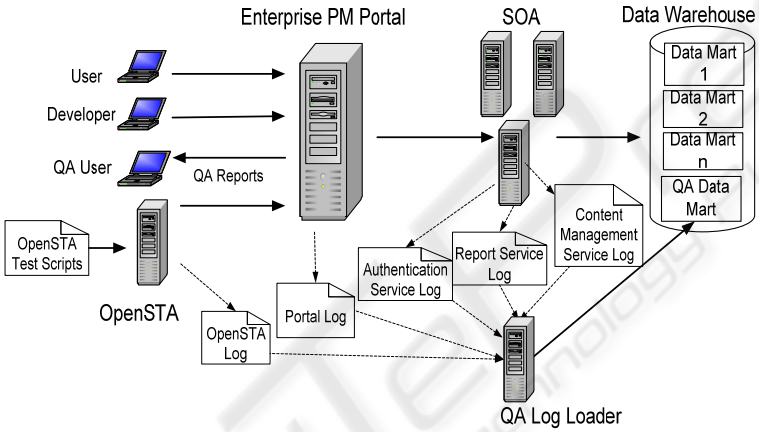

Fig. 1. Quality Assurance Framework.

Figure 1 shows the integration of our quality assurance framework with the SOA of the performance management system. The OpenSTA client simulates user actions in the Enterprise PM Portal. There are log files that contain detailed information from each component of the system. The OpenSTA log records the results and execution time of each user action, the Enterprise PM Portal and each service in the SOA has a log of actions performed. In addition, there is a database log and a server log for each physical box that monitors memory usage, file handles etc. The QA Log Loader processes these files into a QA Data Mart that can be used to report and analyze the results of any test run. Associated with the QA Data Mart is a QA Portal (Figure 2) to access the catalogue of QA reports, along with tools to analyze and monitor what is taking place, making it very easy to notice a quality issue and investigate the cause of it [8].

The result is that members of the QA team can do a deeper analysis of quality. For example, in one test run, the OpenSTA log showed slower than normal response times, but there were other reports to indicate if the database or the report service was slow, and yet another report that showed a particular server were running out of

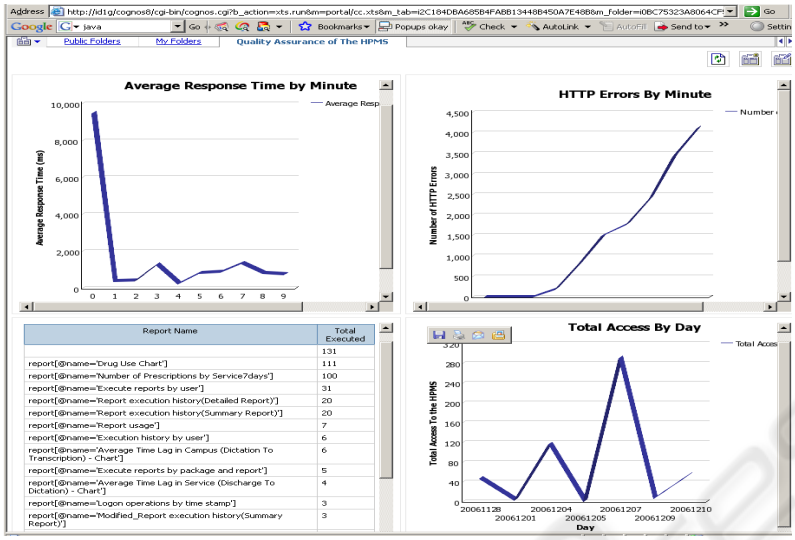

Fig. 2. Quality Assurance Portal.

memory. In this manner, the performance management system itself becomes a tool for processing log data and communicating the results of test runs. Another advantage of this approach is that the quality assurance framework is embedded into the performance management system and can remain there in production. This is important because quality characteristics can change as data and user volumes increase, or as new reports or portal configurations are added to the system.

There are a number of challenges with this approach though. It is dependent on the existence and accuracy of the logs provided by the different components of the system. It can also be very challenging to correlate log entries to individual user actions at the interface level. At this time, we are only analyzing aggregate results of overall system quality. The approach also requires skills in data warehouse tools. Finally, there is a performance and behavioral overhead directly proportional to the amount of logging that is enabled. Log too much data and the quality of the system is impacted.

4 Generation of Test Data

To provide an accurate assessment of quality, we had to ensure that "test" data used was similar to "real" data in terms of both volume and actual values so that reports run in our test environment would have similar processing characteristics as in production. For privacy reasons, "real" data could not be used for testing. As well, data generated for a data warehouse has a very specialized structure called a star schema [7], in which data is organized into dimensions (Doctor, Diagnosis, Date, Prescription Drug, Lab Test) and quantifiable facts (Prescription Amount, Cost, Length of Stay). In particular, for a performance management system, data and reports are very much time-oriented.

In our approach, we first create by hand a series of very small data marts with fabricated but representative facts using the exact same dimensions as the production data warehouse. Then, we generate a full volume "test" data warehouse by obfuscating the identity information in the dimensions, and using a data generation tool to populate "test" facts from the very small data marts. For example, we built a very small data mart for antibiotic prescriptions during one week in January 2002 and cloned the data to fully populate a data warehouse from January 2002 to December 2006 specifying for each month in that time period what the total number of prescriptions should be and the average amount of each prescription. Rather than specific names for drugs, doctors or diagnoses, "test" names were generated like Drug1, Doctor2, and Diagnosis.

We evaluated some popular test data generation tools [1, 3, 11]. All of them could generate large volumes of data, one table at a time, using random data or predefined data sets. Some had adequate support for obfuscating data.

Unfortunately, none had specific support for linking generation of fact table entries to dimensions in a star schema. Nor did any have specific support for time based generation to fill out data on a month by month basis. Most importantly there was no support for ensuring that generated data matched aggregate targets on a monthly basis, or match a specified distribution model. As a result, we have built our own tool that we have successfully used to generate data for 3 different data marts. It has support for large volumes of data, obfuscation, star schema and aggregate targets on a time period basis. The support for aggregate targets on a time period basis is a bit simplistic, but work is progressing on a general expression engine that can be used to specify distributions.

5 Results

We evaluated our quality assurance framework in comparison with traditional black box testing (using a tool such as OpenSTA). There are two aspects to consider in evaluating a quality assurance framework. One aspect is the capabilities provided in terms of what aspects of quality assurance can be measured and the quality issues the framework address. We focus on the scalability, reliability, usage and privacy of the performance management system, since those qualities have the biggest impact on whether or not the system would be successful. The other aspect is how much effort is required to implement and maintain the quality assurance framework, and how much effort is needed to notice or investigate quality issues, as well as who can do it and how much skill and knowledge do they need. We also evaluated tools used in generating "test" data when sensitive "real" data is not available in a test environment.

5.1 Quality Assurance Capabilities

The criteria used for evaluating capabilities of a quality assurance framework for performance management systems are as follows:

- Support for assessment of non-functional qualities like scalability, reliability, usage and privacy of the performance management system.
- The ability to analyze detail level information of data collected.
- Support for aggregations on quality data with drill up/down in different dimensions such as time, services, etc.
- Support for alerts to automatically notify when data collected indicated a quality issue
- The ability to perform quality assurance across the entire architecture of the performance management system on a component by component basis.
- The ability to support ongoing quality assurance of the performance management system in production.

The following table shows the summary of the evaluation of black box testing with a professional tool and the integrated quality assurance framework in doing quality assurance for the hospital performance management system in our case study.

Table. 1. Quality Assurance Criteria.

Criteria		**Black Box Testing**	**Integrated Framework**
Ensure non-functional qualities	Scalability	yes	yes
	Reliability	yes	yes
	Usage	no	yes
	Privacy	no	yes
Detail Results		yes	yes
Aggregate Results with Drill up/down		no	yes
Alert system quality exceptions		no	yes
QA across individual SOA components		no	yes
Quality assurance in production		no	yes

Black box testing with OpenSTA can be used to simulate real user behaviors to the performance management system and report HTTP feedbacks returned by web servers with graphs and tables showing detail information of data collected. Useful reports are provided to show scalability and reliability of the performance management system on the whole system level.

But black box testing does not access the underlying architecture of the performance management system; hence qualities of components/service inside the system infrastructure can not be tested and deep analysis on the system qualities can not be done. It can not ensure usage and privacy of the system due to lacking of related data. It only provides simple aggregations on data. There are no mechanisms provided to alert users of system exceptions. Users have to run and view reports manually after a test run is completed. Finally, simple black box testing is inappropriate for managing quality assurance of the system in production.

In contrast, the integrated quality assurance framework is fully integrated into the performance management system and across the system architecture. It collects and consolidated logs that track system and component behavior into the data warehouse and models quality data in multi-dimensional schema that allow users to drill up/down through a detailed picture of quality assurance along different dimensions. It supports the management of non-functional qualities like scalability, reliability and usage, as well as the ability to monitor privacy of the performance management system. As well, the underlying alert mechanisms of the performance management system can be used to automatically generate alerts for quality assurance issues. For example, we can use email function of the system to notice a new report ready for users, or we can set up quality metrics and their ranges and make the system monitor those metrics against data and send email for data out of range. Finally, the integrated quality assurance framework can be used in production to continuously collect and monitor quality assurance data although this feature may need tuning and configuration as to how much log information is collected.

5.2 Implementation Effort

The criteria for evaluating implementation effort associated with a quality assurance framework for performance management systems are as follows:

- Efforts to implement the quality assurance framework
- Efforts to maintain the quality assurance framework
- Efforts to notice a quality issue
- Efforts to investigate and determine cause of an issue
- Additional efforts to perform quality assurance in production

Efforts we used in our experiment to perform quality assurance for the hospital performance management system are shown in Table 2.

Table 2. Effort Comparison.

Criteria		Black Box	Framework
implement		1 person day	1 person month
maintain		1 person day	1 person week
notice an issue	scalability	View reports	Alert to view reports
	reliability	View reports	Alert to view reports
	usage	Can not be done	Alert to view reports
	privacy	Can not be done	Analyze reports
determine cause	scalability	Inspect distributed logs	Analyze reports.
	reliability	Inspect distributed logs	Analyze reports.
	usage	Can not be done	Analyze reports
	privacy	Can not be done	Analyze reports
Production		Can not be done	Built In

In the black box testing framework, we do not need to be familiar with the infrastructure of the performance management system as we only target the web pages of the Enterprise PM Portal. There are efforts needed to analyze user scenarios and record user actions as test scripts. The test scripts were saved in OpenSTA project files and copied between client computers. Test scripts can be replayed by client computers to create required traffic on the system, and then reports are automatically created after test runs for analyzing collected data. This part was finished in one day.

The integrated QA framework, on the other hand required lots of effort to collect log data into the data warehouse. First, we needed to enable log functions for system components. Second, we needed to analyze logs and implement a log loader for them. The ETL process involves running the log loader and executing database SQL commands or functions, and double checking the data collected to ensure the quality of the data collected. Third, we used a metadata modeling tool to model collected data dimensionally in the data warehouse and publish a metadata model for reporting. Finally, the quality assurance reports and portal needed to be designed and created manually. It took one month to set it up.

In terms of maintenance for the black box testing framework, changes on web pages will cause OpenSTA to redo the work of recording and replying test scripts, but it is still quite simple and can be done in one day.

For the integrated quality assurance framework, significant maintenance efforts are needed when there are changes in the system infrastructure, for example:

- Changes in applications or services for the performance management system will bring different logs into the quality assurance framework. The log loader has to be compatible with those logs and the metadata model of the quality assurance data mart may need to be changed too.
- Changes on the granularity of logs of a component will cause configurations on the log loader and related changes between metadata objects in the quality assurance data mart.
- Changes on the goal of quality assurance may ask for new reports or modification on existing reports, or the quality assurance portal.

In general, all of this work can be done in one week. It can be faster if the integrated framework is designed to be compatible with different service and component configurations.

Implementation and maintenance effort is more significant with the integrated quality assurance framework, but once in place the effort associated with assessing quality assurance is greatly reduced. With the black box testing framework, the effort needed to notice a quality issue is usually high. First, there is no alert provided to users for system exceptions, users need to discover them manually. Second, although a simple reporting capability is usually bundled into black box testing tools, there is typically no relationship set up between reports and there is no portal to help users easily go through reports. To find quality issues, users need go through all reports and figure out the data relation between reports; this is difficult even for quality assurance experts. As well due to the limitation on the type of data collected, there is no support for assessing quality issues such as the usage and privacy of the system.

Reports in the integrated quality assurance framework are created based on the quality assurance data mart model, a star schema, and reports content can be custo-

mized to allow users to easily find out data and graphs that they are interested. Users can drill up and down in dimensions to see the system qualities in different levels and drill though different reports to analyze related information. The quality assurance portal can also be customized by users to organize reports, which help users to focus on their goals. The integrated quantity assurance framework can notice issues on scalability, reliability, usage and privacy of the system.

The black box testing framework cannot perform quality assurance across the performance management system architecture; hence, it cannot report what happened inside the box. To investigate and determine causes of an issue, users need manually inspect system components' logs distributed in the system. Those logs often have complex structure, complex information and very long content, and users have to look through the content line by line. To do this, time and efforts tend to be huge. Without related data collected, it cannot investigate and determine cause of a usage or privacy issue of the system

The integrated quality assurance framework can perform quality assurance across the performance management system architecture. The framework can collect data from logs automatically by the ETL process, so data in the quality assurance data mart is well structured, cleaned and meaningful. The quality assurance portal allows users to view quality data in a single place and reports let users find the relevant data very quickly. The integrated quality assurance framework can report issues on scalability, reliability, and usage of the system. The integrated quality assurance framework does not provide an automatic way to flag privacy violations, but it does provide usage analysis reports from audit logs that can be used to discover and investigate potential privacy violations.

In general, while the integrated quality assurance framework requires greater effort and cost to implement, it does provide significant benefits and requires less cost and effort to use both in a test environment and in production.

6 Conclusions

The initial results of this case study are promising. We can generate "test" data in volume to assess the quality of a hospital performance management system in a test environment. Through careful analysis of system architecture and log files coupled with data engineering of a quality assurance data mart, we have been able to create an integrated quality assurance framework that leverages the performance management system to provide deep analysis of quality at an aggregate level which is available both in the test and production environment. In future work we hope to support deep analysis at a detail level, and provide even more sophisticated data generation features.

References

1. Advanced Data Generator 1.7.0, Upscene Productions, http://www.upscene.com/, last retrieved: 2008/04

2. P. Chountas, V. Kodogiannis, Development of a Clinical Data Warehouse. Medical Information Systems: The Digital Hospital (IDEAS-DH'04), pp 18-14, September 2004. ISBN: 0-7695-2289-0
3. DTM Data Generator 1.15.04, DTM Soft, http://www.sqledit.com, last retrieved: 2008/04
4. A.J. Forster, The Ottawa Hospital Data Warehouse: Obstacles and Opportunities. IT in Healthcare Series, June 15, 2005
5. Y.F. Jarrar, A. Al-Mudimigh, M. Zairi, "ERP implementation critical success factors -the role and impact of business process management", IEEE International Conference on Management of Innovation and Technology, 2000
6. J.J. Jeng, "Service-Oriented Business Performance Management for Real-Time Enterprise", E-Commerce Technology, 2006.
7. R. Kimball and M. Ross, “The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling”, Second Edition, Wiley, 2002.
8. L. Peyton, A. Rajwani, "A Generative Framework for Managed Services", International Conference on Generative Programming and Component Engineering, October, 2004.
9. OpenSTA (Open System Testing Architecture), http://www.opensta.org, last retrieved: 2008/04
10. C. Rankin, The Software Testing Automation framework, IBM Systems Journal, Software Testing and Verification, Vol. 41, No.1, 2002
11. TurboData 4.0.6, Canam Software, http://canamsoftware.com,last retrieved: 2008/04
12. W3C Working Group, Web Services Architecture, Note 11 February 2004, http://www.w3.org/TR/ws-arch, last retrieved: 2008/04