A Service Oriented Architecture Supporting Data Interoperability for Payments Card Processing Systems

Joseph M. Bugajski[1], Robert L. Grossman[2], and Steve Vejcik[2],

[1] Visa International, P.O. Box 8999, Foster City, CA 94128
[2]Open Data Group, 400 Lathrop Ave, Suite 90, River Forest IL 60305
JBugajsk@visa.com, rlg1@opendatagroup.com, vejcik@opendatagroup.com

Abstract. As the size of an organization grows, so does the tension between a centralized system for the management of data, metadata, derived data, and business intelligence and a distributed system. With a centralized system, it is easier to maintain the consistency, accuracy, and timeliness of data. On the other hand with a distributed system, different units and divisions can more easily customize systems and more quickly introduce new products and services. By data interoperability, we mean the ability of a distributed organization to work with distributed data consistently, accurately and in a timely fashion. In this paper, we introduce a service oriented approach to analytics and describe how this is used to measure and to monitor data interoperability.

Keywords: data quality, data interoperability, service oriented architectures for analytics, payments card processing systems.

1 Introduction

VisaNet is the largest private payment data network in the world. The annual total payment volume on VisaNet recently exceeds USD $4 trillion. The network can handle about 10,000 payment messages per second. It supports every payment card brand including approximately 1.0 billion Visa cards. It permits card holders to make secure payments in 120 countries and most currencies through over 20 million businesses, who operate over 100 million point-of-sales acceptance devices. Above all, member banks that own Visa require that all the transactions be completed among all the member banks error free and with extremely high system reliability.

The systems that comprise VisaNet were, until recently, controlled by one information technology division. That division assured data interoperability. They enforced a rule that permitted only one source for defining and modifying payment transaction record format and semantic definitions. The computing and network infrastructure ran on IBM TPF and IBM MVS mainframe computers. Each mainframe ran an identical code base in one of several processing centers strategically situated around the world. Hence, data interoperability was assured because all data in VisaNet, and all processes that operated on data, were always the same.

The current guaranteed "uniformity" system environment for data interoperability comes at a not insignificant price with respect to time to market capability and cost effectiveness for faster growing economies; e.g., the Asia Pacific and Latin American regions. Visa determined that the old model for interoperability had to change to meet the demands of its member banks for quicker market entry for product modification and entirely new types of payment products.

In this paper, we describe an approach to measuring and monitoring data interoperability using large numbers of baseline models that are each individually estimated. These are examples of what, in the context of data interoperability, we call Key Interoperability Indicators or KIIs. The first contribution of this paper is the introduction of baseline models as an effective procedure for measuring and monitoring data interoperability.

The second contribution of this paper is the introduction of a service oriented architecture for computing the analytics required for estimating, updating, and scoring using these baseline models.

Section 2 contains background and related work. Section 3 discusses data interoperability and baseline models. Section 4 describes a service oriented architecture supporting baseline models. Section 5 describes a deployed application based upon a service oriented architecture for analytical models and which is scalable enough to support millions of baseline models.

2 Background and Related Work

Loosely speaking, the interoperability of data and services means that data and services can be defined and used independently of the application, programming language, operating system, or computing platform which implements them. Interoperability has a long history in computing. Major phases include: Electronic Data Interchange, object models, virtual machines, and web services, which we now describe briefly.

Electronic Data Interchange (EDI). EDI introduced a standard syntax and a standard set of messages for various common business transactions. EDI began by introducing standard formats for purchase orders, invoices, and bills of lading so that these could be processed without human intervention. In other words, EDI approached interoperability by requiring all applications using purchase orders to use a standard structured format. Each different business document had its own structured format.

Object Models. With the introduction of object models, interfaces for both data and methods became formalized. A wide variety of object models have been introduced, including Microsoft's COM and DCOM, Sun Microsystems Java Beans and Enterprise Java Beans, and the Object Management Group's (OMG) Object and Component Models. OMG's Common Object Request Broker Architecture or CORBA is an architecture providing interoperability for objects across different languages, operating systems, and platforms. For more details, see [4].

Virtual Machines. Java popularized the idea of supporting interoperability by mapping a language to an intermediate format (byte code) which could be executed

on any machine which had an interpreter for byte code (virtual machine). In this way, the same code could be executed on different operating systems and platforms. Note though that the same language (Java) must be used.

Service Oriented Architectures. More recently, web services and service oriented architectures have popularized the use of XML to define data and message formats. In particular the Web Services Description Language or WSDL provides a way to define services that can be bound to different programming languages (e.g. Java, Perl, C/C++) and protocols (http, smtp, etc.)

Our approach to data interoperability is based upon a service oriented architecture that is specifically designed to support statistical and other analytical models, as well as various services employing them.

3. Data Interoperability

Not all data in a complex, distributed system needs to be interoperable. In this section, we distinguish between data that must be interoperable, which we call global data, and other data, which we call local data.

Our approach is based upon introducing two new primitive concepts: a new class of data called *global data* and a new class of services called *global services*. Data that is not global and services that are not global are called *local*. Not all data and not all services are expected to be global; rather, in general data and services are local and only when they need to interoperate across the enterprise are then required to be global. Global data and global services are designed so that different applications storing, accessing, querying or updating global data using global services can easily interoperate.

In our approach, the interoperability of global data is measured using statistical models called Key Interoperability Indicators or KIIs. In the remainder of this section, we define global data and global services, as is usual for service based architectures.

3.1 Global Data

Global data is any data with the following properties:

- The schema is defined using UML® or XML.
- Data is accompanied by XML based metadata.
- Associated metadata must be accessible through a metadata repository and discoverable through discovery services.
- Associated metadata must also specify one or more access mechanisms. Access mechanisms include service-based mechanisms, message-oriented mechanisms, or data protocol based mechanisms.
- Permitted values for individual field elements are also stored using XML and corresponding discovery services

3.2 Global Services

Global services are any services or functions with the following properties:

- Global services operate on data through interfaces. By inspecting the interface, a service can specify the operation it wants to perform and marshal the arguments it needs to send. Interfaces are independent of programming languages; on the other hand, to be useful, interfaces, such as J2EE or the Web Service's Web Service Description Language (WSDL), have mappings or bindings to common programming languages, such as C, C++, Java, Python, Perl, etc.
- Global services transport data using an agreed upon wire protocol. For example, web service based global services use the protocol specified in the WSDL corresponding to the service.
- Global services are registered. For example, web services may be discovered through Universal Description, Discovery & Integration (UDDI) services.
- Global services follow from UML descriptions of requisite functionality to remove details concerning implementation in the matter of the Model Driven Architecture (MDA®) and Meta-Object Facility (MOF®), both defined by the Object Management Group (OMG)[1].

Interoperability is achieved in several ways. First, separating interfaces from implementations facilitates interoperability. Services can inspect interfaces but not implementations. In this way, multiple languages can be bound to the same interface and implementations can be moved to a different platform. Second, by supporting multiple wire protocols, global services can improve their interoperability. Third, registration provides a mechanism for a client application to obtain sufficient metadata to ensure that it is bound to the correct global service.

3.3 Key Interoperability Indicators (KIIs)

Our approach to measuring the interoperability of global data is to measure and monitor a number of indicators of interoperability using statistical baseline models.

- We measure whether the values of fields and tuples of field values change unexpectedly as transactions move from one system to another.
- We measure whether the values of fields and tuples of field values change unexpectedly after changes and updates to the processing system and components of the processing system.
- We measure whether the values of fields and tuples of field values are correlated to various exception conditions in the system's processing of data.

These measurements are done by establishing baseline behavior with a statistical model, and then each day measuring changes and deviations from the expected baseline. Here is a simple example. A table of counts characterizing certain behavior in a specified reference baseline period can be measured. Counts during an

[1] UML, MDA, MOF and OMG are registered trademarks of the Object Management Group, Inc.

observation period can then be computed and a test used to determine whether a difference in counts is statistically significant. If so, an alert can be generated, and used as a basis by subject matter experts for further investigation.

Value	%
00	76.94
01	21.60
02	0.99
03	0.27
04	0.20
Total	100.00

Value	%
00	76.94
01	20.67
02	0.90
03	0.25
04	1.24
Total	100.00

Table 1. The distribution on the left is the baseline distribution. The distribution on the right is the observed distribution. In this example, the value 04 is over 6x more likely in the observed distribution, although the two dominant values 00 and 01 still account for over 97% of the mass of the distribution. This example is from [1].

4. A Service Based Architecture for Analytics

In the last section, we gave relatively standard definitions, from a service oriented architecture point of view, of types of data and services (global) that are designed to be the basis of data interoperability across an enterprise. In this section, we introduce a service based architecture for analytics based upon four types of services: services for preparing data, services for producing analytical models, services for scoring, and services for producing OLAP reports.

4.2 Services for Preparing Data

In practice, extracting, aggregating, and transforming data in various ways represents a significant fraction of the cost of developing complex enterprise applications. In addition, small differences in the transformations can result in the lack of interoperability of an application. For these reasons, our model singles out services for preparing or transforming data from one format to another. Data preparation and transformations are generally comprise much of the work of developing analytical models.

More formally, *derived data* is the result of applying global services implementing one or more of a class of predefined transformations. For our applications, we used the transformations defined by the Data Mining Group [2]. These include:

- Normalization: A normalization transformation maps values to numbers. The input can be continuous or discrete.
- Discretization: A discretization transformation maps continuous values to discrete values.

- Value mapping: A value mapping transformation maps discrete values to discrete values.
- Aggregation: An aggregation transformation summarizes or collects groups of values, for example by computing counts, sums, averages, counts by category

The Data Mining Group's Predictive Model Markup Language or PMML allows these types of data transformations to be defined in XML. Although these four types of transformations may seem rather restrictive, in practice, a surprisingly large class of transformations can be defined from these.

4.3 Services for Producing Models

In our service-oriented approach to analytical processes, services operate on data and derived data to produce rules, statistical models and data mining models, which we refer to collectively as analytics. See Figures 1 and 2.

The DMG's PMML markup language captures the most common analytical models [2]. These include statistical models for regression, clustering, tree based regression and classification models, association rules, neural networks, and baseline models. So, to be very concrete, analytical services for producing models can be thought of as operating on data and derived data to produce PMML models. Because of this, these types of analytical services are sometimes called Model Producers.

4.4 Services for Scoring

Given an analytical model described in PMML, a scoring service processes a stream of data, computes the required inputs to the analytical models, and then applies the analytical model to compute scores.

Sometimes scoring services are called Model Consumers since they consume analytical models to produce scores, in contrast to Model Producers, which consume data sets, viewed as learning sets, to produce analytical models.

One of the main advantages of a service oriented architecture for analytics, is that it is very common for the model producers and consumers to be deployed on different systems, with quite different requirements. For example, in the case study described in Section 5 below, the model producer can be deployed on several systems, while the model consumer must run on an operational system with very stringent security requirements.

For many applications, including the application described in Section 5 below, these scores are then post-processed using various rules to produce reports and alerts.

4.4 Services for Reports

In the same way, although outside of the scope of this paper, one can define OLAP services as services that operate on global data and global derived data to produce OLAP reports.

5. Case Study – Key Interoperability Indicators for Monitoring Data Interoperability

5.1 Introduction

For approximately one year, the Visa KII Monitor, based upon a service oriented architecture, has been measuring and monitoring millions of separate baselines models in order to monitor data interoperability.

Separate PMML-based baseline models are estimated periodically and used to monitor each day behavior for twenty thousand member banks, millions of merchants and various other entities. Given the very large numbers of individual baseline models involved and the quantity of data processed each, the production of the models, scores, alerts, and reports are, by and large, automated by the KII Monitor and require little human intervention.

5.2 KII Monitor Architecture

The Visa KII Monitor consists of the following components:

- A service oriented application that transforms and prepares data, either for producing models or producing scores.
- A data mart containing data for the observation period, as well as historical and historical analytical models.
- A Baseline Producer, which takes a data set over the learning period and estimates parameters of baseline models.
- A Baseline Consumer / Scorer, which takes the current baseline model and the data for the current observation period and produces scores indicating the likelihood that the data differs in a statistically significant fashion from the baseline model.
- A service oriented application that produces alerts and reports by analyzing the scores produced by the Baseline Consumer and applying certain rules.

5.3 Baseline Models

The Visa KII Monitor measures a large number of different baselines, including the following.

- The KII Monitor measures whether the values of payment fields change in unexpected manners as transactions move from one system to another.
- The KII Monitor measures whether the values of payment fields change in unexpected manners after changes and updates to the processing system and components of the processing system.
- The KII Monitor measures whether the values of payment fields are correlated to declines and other exceptions to the normal processing of transactions.

Separate baselines are computed for each member bank, and for merchant, and for various other entities. This results in millions of individual baselines being computed and monitored.

5.4 KII Alerts and Reporting

Here is an overview of the basic steps that the KII Monitor uses to generate alerts and reports.

First, parameters used for segmenting baseline models are selected. These include the time period, the region, the specific payment field values being monitored, and the various logical entities, such as banks, merchants, etc. For each separate segment, a baseline model is estimated and periodically updated.

Using the baseline model, during each observation period, usually a day or a week, the following steps are performed.

1. Scores that represent statistically significant deviations from baselines are generated by the Baseline Consumer / Scorer.
2. The approximate business value for the scenario associated with the baseline is estimated. If the business value is above a threshold, a report containing one or more alerts is generated.
3. The report is passed to a subject matter expert for investigation, and, if appropriate, for remediation.
4. The scenarios are monitored for future compliance.

5.5 Standards

The Visa KII Monitor described in this section was one of the motivating examples for a proposal for baseline models that is currently under consideration by the Predictive Model Markup Language Working Group of the Data Mining Group [3] for inclusion in PMML version 3.2.

6. Summary and Conclusion

In this paper, we have described how Visa used service oriented architecture to measure and monitor payment data interoperability. The interoperability of payment data has emerged as a key requirement as Visa introduces its distributed computing

environment where six operating regions have more local control of their computing environments.

Data interoperability was defined by introducing the concepts of global data and global services, which can be defined relatively easily using the standard concepts of a service oriented architecture.

In this paper, we used baseline models to measure data interoperability and a service oriented analytics architecture to produce baseline models, as well as alerts and reports based upon them.

Working with the KII Monitor during the past year has taught us several lessons concerning data interoperability and the implementation of a service oriented architecture for analytics:

1. Baseline models are effective for uncovering certain important data interoperability and data quality problems for complex, distributed systems, such as the payments card processing system described above.
2. An important practical consideration when using baseline models for monitoring data interoperability is to select a sufficient number of segments and appropriate threshold levels so that the number of alerts produced have business meaning and relevance, but not so many different segments that so many alerts are produced that they are not manageable.
3. An important design decision for the project was to use the Predictive Model Markup Language (PMML) to represent baseline models.
4. Once PMML was chosen, it was natural to design the system using services for preparing data (using PMML-defined transformations), services for producing PMML models, PMML-based services for scorings, and services for producing alerts and reports.

For the past year, this service oriented architecture for analytics has provided a solid foundation for the weekly alerts that are produced by the KII Monitor.

References

1. Joseph Bugajski, Robert Grossman, Eric Sumner, Tao Zhang, A Methodology for Establishing Information Quality Baselines for Complex, Distributed Systems, 10th International Conference on Information Quality (ICIQ), 2005
2. Data Mining Group, retrieved from http://www.dmg.org on June 10, 2006.
3. Data Mining Group, PMML 3.1 - Baseline Model, RFC Version 5.2.7, retrieved from www.dmg.org on June 10, 2006.
4. Aniruddha Gokhale, Bharat Kumar, Arnaud Sahuguet, Reinventing the Wheel? CORBA vs. Web Services, The Eleventh International World Wide Web Conference, retrieved from http://www2002.org/CDROM/alternate/395/ on June 20, 2003.
5. Web Services Interoperability Organization, retrieved from http://ws-i.org/ on June 20, 2003.

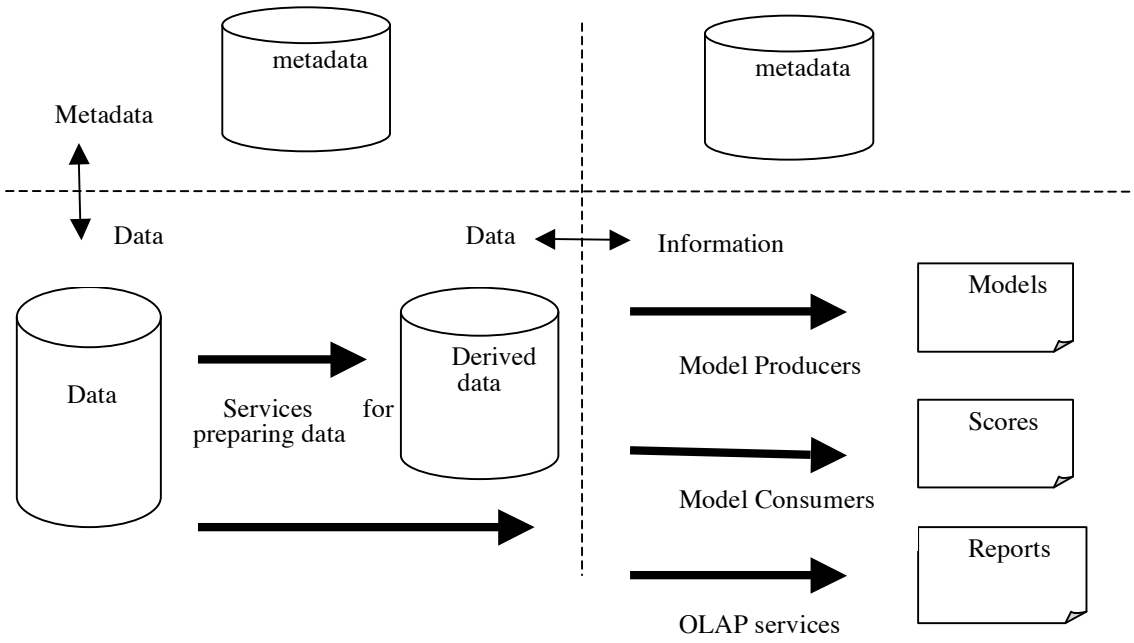

Fig. 1 Specialized global services applied to global data produces global derived data, which in turn produces global information such as analytical models, scores, and reports.

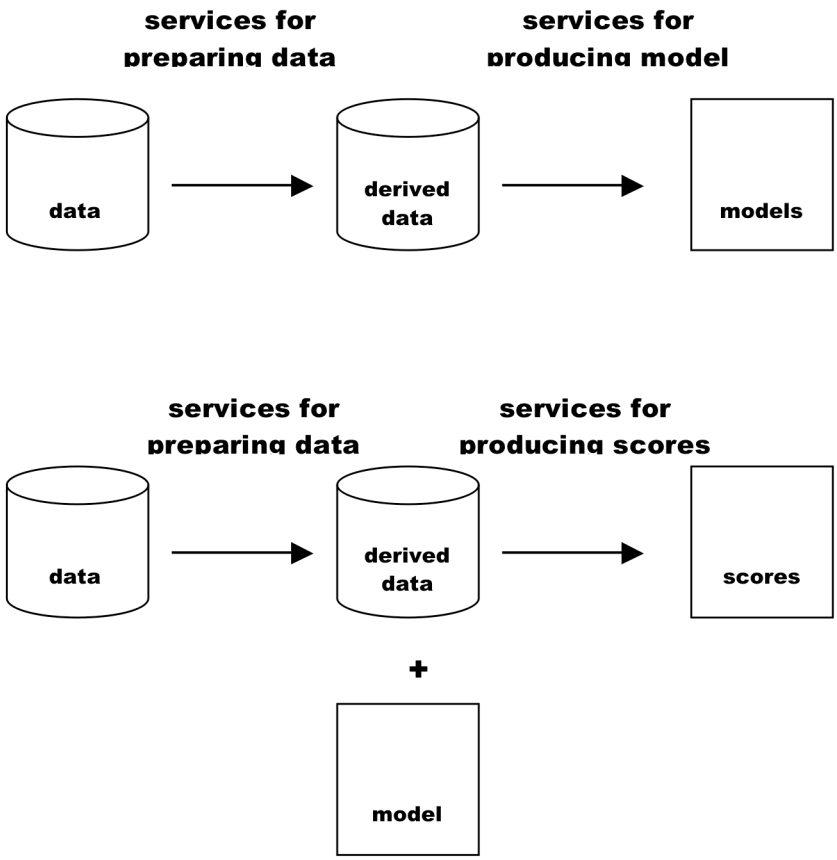

Fig 2. This diagram illustrates three types of services used in our service oriented architecture for analytics: services for preparing data, services for producing models (model producers), and services for producing scores (model consumers).