Business Operation Intelligence Research at HP Labs

Fabio Casati, Malu Castellanos, Umeshwar Dayal, Ming Hao, Mehmet Sayal, Ming-Chien Shan
Hewlett-Packard Labs
1501 Page Mill road, Palo Alto, CA, USA, 94304
{casati,malu,dayal,mhao,sayal,shan}@hpl.hp.com

Abstract

In the last decade we have witnessed an increased trend towards automating all the interactions that a company has with its customers, suppliers, and employees. This trend is continuing even in the economic downturn, since automation not only allows for faster processing, but can also reduce costs and errors due to the diminished need for human involvement. Once a robust e-business infrastructure has been developed and deployed, companies quickly turn their attention to the problem of assessing and managing the quality of their e-business operations. This paper presents the concepts and architecture of a system that supports users in managing their infrastructure from a both a business (qualitative) and IT (quantitative) perspective. The platform also provides for the "intelligent" analysis of e-business operations as well as for making predictions over the quality of ongoing and future operations.

1 Introduction and motivations

To increase revenue, improve efficiency, and reduce costs, companies deploy and integrate different kinds of software systems and applications that can automate and manage the execution of mission-critical business processes, within and across organizations. The resulting software architectures are typically complex, and include a variety of technologies and tools. The left side of Fig. 1 shows a quite common infrastructure that supports both the execution of internal business processes and B2B and B2C interactions.

As the complexity of the system grows, the need for solutions that provide ubiquitous and automatic end-to-end management of business operations arises. In particular, business analysts/managers are faced with the need of quickly identifying problems affecting their operations, of assessing the impact of such problems in terms of quality of their business operations, and of understanding the root cause of the problems. However, identifying and resolving problems only covers one aspect of e-business operation management. Indeed, in an ideal scenario, the entire e-business infrastructure should be self-managing, by automatically reacting to problems by executing corrective actions [7].

To address these needs, at Hewlett-Packard Labs we have developed an architecture and a set of applications that enable the analysis, monitoring, and optimization of e-business systems. The work involves the definition and implementation of measurement models and tools combining and aggregating raw operational data from each component, as shown in Fig. **??**., as well as the development of a Business Intelligence (BI) solution for historical, real time and predictive analysis of the behavior of the entire e-business suite. As part of our work, we

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

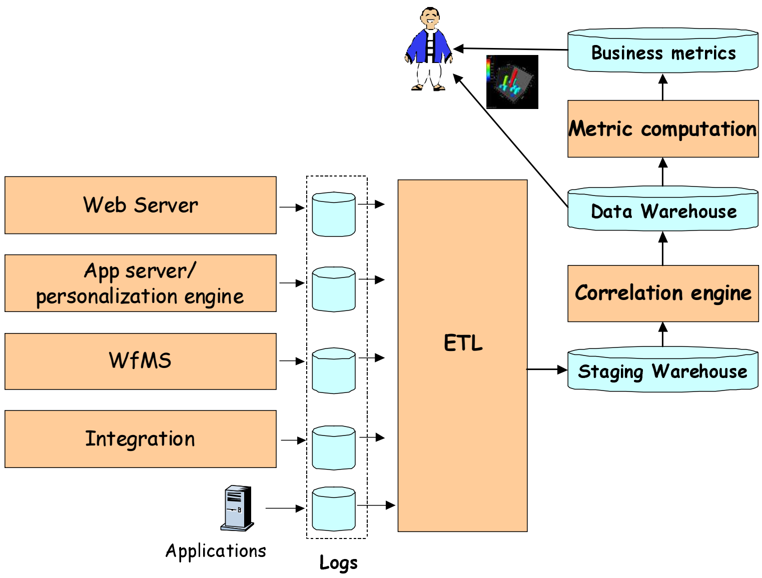

Figure 1: BOI architecture.

explore how to apply Data Mining techniques to analyze internal operations and external interactions, in order to understand issues regarding internal business processes, the operational behaviors of partners and customers, to improve the quality and efficiency of the business systems, and consequently to gain competitive advantage. We refer to this research and applicative area as Business Operation Intelligence (BOI), since it applies BI techniques to the management of business operations.

2 BOI Architecture

This section presents the overall approach and architecture of the BOI tool suite. In our architecture, we assume that either the individual components of the BOI platform have facilities for logging execution (meta)data to a file or a relational database, or that instrumentation has been put in place for this purpose, as described in [***]. Once logged, data are periodically collected by an ETL application and inserted into a staging data warehouse. Data in the staging warehouse is then processed by the correlation engine, that inserts correlation information, to denote records logged by different systems that logically belong to the same business transaction. Once correlation information has been computed, then the data is loaded into the warehouse. The warehouse is designed to support high-performance OLAP and multidimensional analysis of execution data coming from heterogeneous sources. Hence, the warehouse is a very useful component in itself, providing a wide range of reporting functionalities that are still missing in commercial e-business platforms, and even in more traditional middleware platforms. Moreover, BOI aims not only at enabling a low-level, IT view of the collected information, but the main focus is at a higher-level, itbusiness view. While the warehouse can provide information about number of executions or average duration of each execution, business users are likely to need information at higher abstraction levels. For example, they may be interested in analyzing the cost or the quality of a business transaction, where cost or quality are user-defined criteria, whose computation depends on the specific business domain and on the nature of the business transactions.

To enable and simplify the business-level analysis of low-level execution data collected by system logs, BOI provides users with a metric computation framework. This framework allows users to define business metrics, i.e., measurable properties that can be associated to data elements stored in the warehouse. The cost for a business transaction is an example of a business metric. Metrics are characterized by a definition and an implementation part. The metric definition includes the metric name and a data type, that can be Numeric, Boolean, or Enumeration (taxonomy). The quality metric defined above is an example of a taxonomical metric, while cost or profit are examples of numeric metrics. Finally, the metric definition includes the specification of the target entity, i.e., the type of elements to which the metric is applied to (e.g., events, process instances, business transactions, web service operations). The metric implementation consists of a set of references to mapping functions. A mapping function is parametric code (typically SQL or Java) that can be executed on top of the warehouse and returns a numeric or Boolean value. For example, mapping functions could compute the overall duration of a business transaction, or the number of nodes executed in a process instance, or whether the execution of a web service operation has been faster than the average. Within the metric implementation, it is possible to specify that a certain mapping function should be used to compute the metric for a given context. For example, users can specify that the cost for a process execution is dependent on the process duration, but only for instances of process ”Fulfill order”, while the cost for instances of process ”approve proposal” is instead dependent on the number of nodes executed within the process instance. Hence, while the metric definition remains the same, its implementation actually depends on the specific element being analyzed. This enables users to get, at metric analysis time, a uniform view of otherwise heterogeneous properties.

Once metrics and mappings have been defined, the metric computation engine executes the mapping functions to obtain the metric data (measurements), also (what else goes into this db??) stored in a relational database (not the datawarehouse?? further you say that metric framework is used to compute correlation info and before you said that once correlation info is computed data goes into DW). Such data can then be analyzed through either commercial reporting tools or though a built-in console, provided with the BOI solution, which among other features makes use of powerful visualization techniques. A detailed discussion of the metric framework is outside the scope of this paper. We refer the reader to [***] for details. We can now look back at the **correlation engine/problem of data correlation**, mentioned earlier. The metric framework basically tells the BOI system how the different data elements (e.g., process instances) should be stamped with a qualitative or quantitative measure. This suggests that the very same metric framework used for mapping low level data into business metrics can also be used for ”tagging” execution data with correlation identifiers, that define whether two or more data records logically belong to the same business transaction. It is up to the users to define what correlation means, and what are the characteristics of the records that imply a correlation. They can do so by defining special ”correlation” metrics, and by using mappings to define how the correlation should be determined. For example, for a RosettaNet transaction the correlation can be determined by looking at User IDs, product IDs, or other information embedded in the data records. BOI will take care of executing the mappings and therefore of generating the correlation identifiers.

3 Adding Intelligence

The BOI metric framework, along with the data warehouse, can provide analysts with information about the business metric they consider relevant for their business domain. While this is a very important aspect of the solution, it also naturally leads users to desire more advanced functionalities. In particular, besides metric analysis, business users typically demand more ”intelligent” information processing that can provide among others:

- Identifications of the causes of high and low quality executions.
- Prediction of process and service execution quality, and in general prediction of user-defined business metrics specified through the metric framework.
- Identification of patterns in the invocation of business transactions, to restructure and compose service interfaces into value-added services and simplify user interaction.

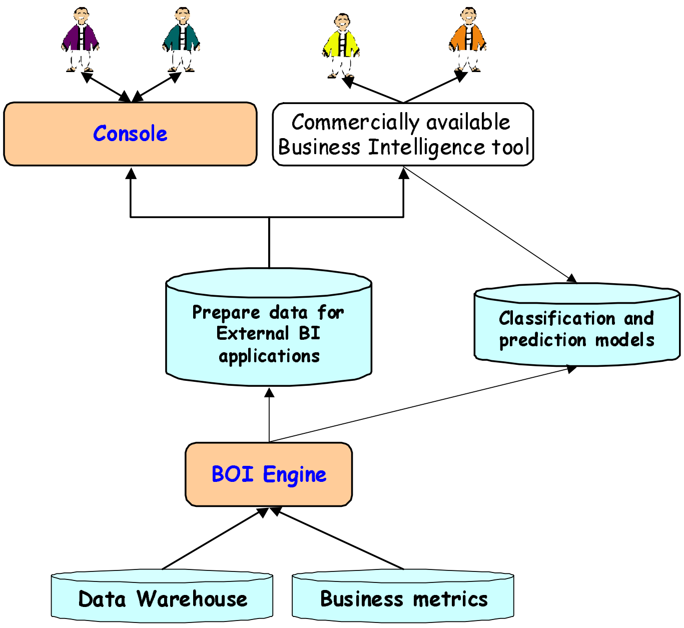

Figure 2: The BOI Engine prepares data for BI applications and generate classification and prediction models.

To provide these and other form of intelligent data analysis, BOI is equipped with a business intelligence engine that operates on top of both the warehouse and the metric data, and identifies explanations for the value of business metrics, prediction of the future value of such metrics, as well as service invocation patterns (see Fig. **??**). The engine includes simple algorithms that perform basic analytics. In addition, it can also prepare data so that it can be consumed by a commercial data mining tool, such as Oracle Darwin or SAS. The approach towards generating analysis and prediction models out of metric data consists in applying data mining techniques to measured elements (process instances, operations, events, etc). In particular, analyzing metric data can be mapped to a classification problem, where measured elements constitute the training (labeled) set, and the different metric values (e.g., good or bad) are the classes. For each class, a classifier model is learned from the training set along with a set of classification rules, i.e., mappings from a condition on the objects' attributes to the class, with the meaning that objects whose attributes satisfy the condition belong to that class. These conditions become the basis for root cause analysis providing a deeper insight into the causes of the metric values and allowing the user to address such causes.

A similar approach can be taken to perform predictions. To understand how classification can be used for predictions, consider the problem of trying to predict the quality of a web service conversation (see [***] for a discussion on web services conversations). A conversation between a client and a service provider can go through several stages. The BOI engine can process past execution data and build a separate classifier for each conversation stage, correlating service execution data available at that stage with the quality metric. At run time, BOI can examine live execution data, determine in which stage the conversation whose quality is to be predicted is, and apply the classifier to determine the (predicted) quality, along with a confidence level. We successfully applied this technique in several domains, and specifically in the context of workflow management [***].

4 Management

The above sections have presented methods and techniques for gathering information about the quality of e-business operations. We now briefly discuss how BOI can interact with the system in order to take corrective actions and, in general, to manage system operations in reaction to the computed values of business metrics.

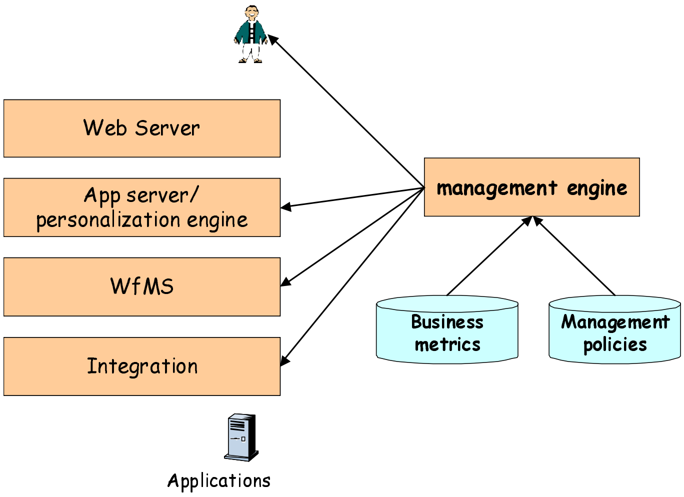

Figure 3: Management component of the BOI solution.

Note that, while this is an important aspect of a BOI solution, it is highly system-specific, since the kinds of reactions that can be performed depend on the specific system that BOI manages. Still, some aspects of system management can be factorized into a generic solution. More specifically, system management in BOI is defined through simple policies, comprising a condition and an action. The condition is a Boolean expression over the value of metric data. The action can be one of the following (see Fig. 1): (***this could be a comma separated text instead of bullets to save space***)

- Send a message to a JMS compliant bus
- Send an event to a business process
- Invoke a Java application
- Trigger BOI optimization algorithms
- Send an email or SMS to a user

A sample policy, defining that once the completion of a process instance I is predicted to be late more than 3 hours, an event should be notified to the process instance. How the event is handled depends on the business logic implemented by the process. The management engine only takes care of notifying the process.

IF predicted_completion_delay(I)> 03:00:00 THEN send_event(I,large_delay_predicted)

Conditions of policies are continuously monitored by BOI. As soon as the condition becomes true, the action is performed by invoking the proper system. The configuration of which action corresponds to which system occurs at deployment time.

5 Visualization

The rapid growth of business and services on the Internet has led to the availability of large volumes of data. Business research efforts have been focused on how to turn raw data into actionable knowledge. It is crucial for companies to analyze their business contracts and to reduce the risk of violation. Basic bar charts have been useful, but only show highly aggregated data. In this paper, we describe a new visualization technique - pixel bar charts [1,2]. Pixel Bar Chart integrates the idea of bar charts with an X-Y diagram. The approach is to show

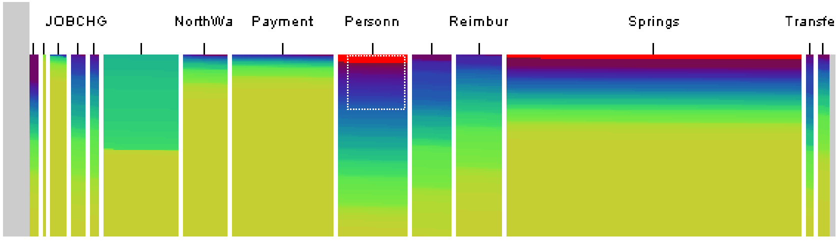

Figure 4: Analysis of Business Activity Elapse Time by Business Activity Name.

each data item (e.g. a business instance) by a single pixel in the bar chart. The detail information of one attribute of each data item is encoded into the pixel color and can be accessed and displayed as needed. The pixel bar chart technique has been prototyped in several business service applications at Hewlett Packard Laboratories.

Fig. 4 illustrates 63,544 business instances at HP web sites. Each pixel is a business instance. The x-axis is classified by activity name; the y-axis is the duration time ordered from bottom to top. The colors (from lemon to dark burgundy) in the different bars represent different duration time values in days. The red area presents the instances which duration time exceeds 60 days. Many important facts may be discovered in Fig. 4 as follows:

- There are 14 different activity names. The ”Springs” business activity has the most number of instances with widest bar width.
- Most activity’s elapse time is low (¡1 day, color lemon) except Personnel (¿47 days, colored blue and red)
- Large number of business activities achieves good response time (color lemon, and green).

In Fig. **??**, the pixel bar chart shows the business activity in one day (24 hours). Each pixel represents a business instance. The x-axis is time slot (hour); the y-axis is the business elapse time. The color represents different elapse time values in milliseconds. The analysts can obtain the following information about the business activity:

- The busiest time is during the hours between 9 and 14. (with wider width bars). Evenings have long running activities.
- Amount of instances achieving good response time (colored green).

It is further interesting that bar chart indicates the 9th hour has the worst elapse time shown in the bar chart. Actually, there are large area colored with lemon and green, which means that there are many instances with good elapse time. Only a very small number of instances colored dark burgundy with very long elapsed time.