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. 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 work ongoing at HP Labs, aimed at enabling a simple, qualitative, and "intelligent" analysis of e-business 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 their 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 as well as 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. In an ideal scenario, the entire e-business infrastructure should be self-managing, by automatically reacting to problems by executing corrective actions.

To address these needs, we are developing 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 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. We refer to this research and applicative area as Business Operation Intelligence (BOI), since it applies BI techniques to the management of business operations. The work presented here derives from earlier efforts focused in the workflow management domain, documented in [1, 2].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

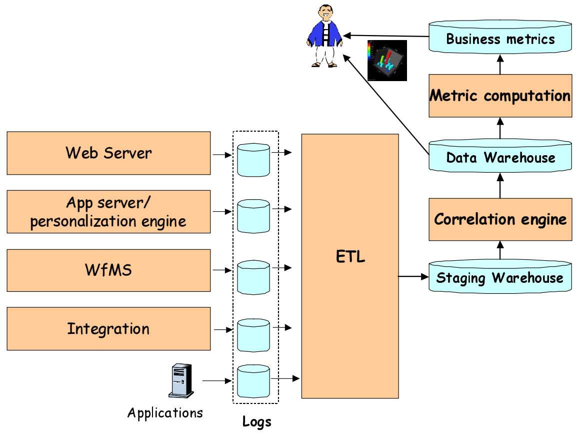

Figure 1: BOI architecture.

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. 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, the data is loaded into the warehouse. The warehouse is a very useful component in itself, providing a wide range of reporting functionalities that span across the whole e-business suite. Moreover, BOI aims not only at enabling a low-level, IT view of the collected information, but also at providing a higher-level, business view. In fact, 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.

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). "Quality" 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*, i.e., 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 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.

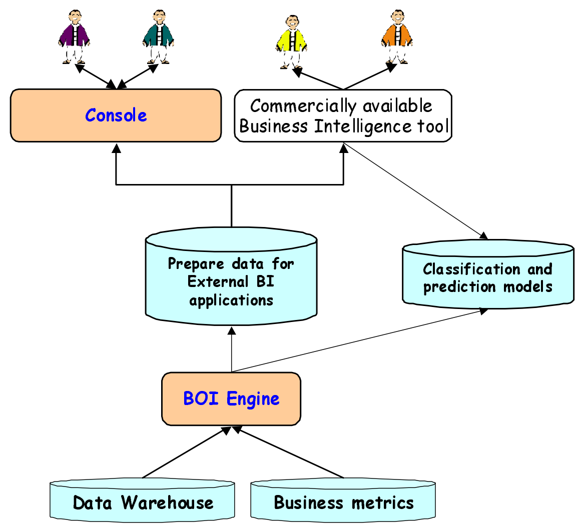

Figure 2: The BOI Engine prepares data for BI applications and generate classification and prediction models.

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). 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.

3 Adding Intelligence and Management

The BOI metric framework, along with the data warehouse, can provide analysts with information about the business metric they consider relevant. While this is a very important aspect of the solution, it also naturally leads users to desire more advanced functionalities. In particular, users typically demand more "intelligent" information processing that can provide for example root cause analysis and prediction of user-defined business metrics. 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. 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. We successfully applied this technique in several domains, and specifically in the context of workflow management [1].

In addition to providing information to users, 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.

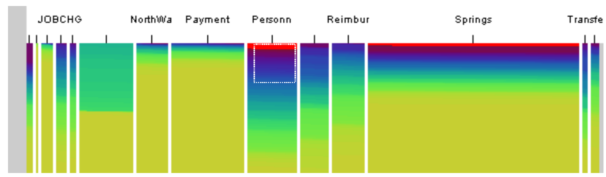

Figure 3: Analysis of process duration by process type.

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: send a message to a JMS compliant bus, send an event to a business process, invoke a Java application, trigger BOI optimization algorithms, or send an email or SMS to a user. The sample policy shown below defines 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. *IF predicted_completion_delay(I)> 03:00:00 THEN send_event(I,large_delay_predicted).* How the event is handled depends on the business logic implemented by the process. The management engine only takes care of notifying the process.

4 Visualization

A key component of a BOI solution is *visualization*. The challenge here is how to visualize high volumes of data in a way that is easy to consume. Basic bar charts have been useful, but only show highly aggregated data. Therefore, we have developed a new visualization technique - pixel bar charts [3]. Pixel Bar Chart integrates the idea of bar charts with an X-Y diagram. The approach is to show each metric data item (e.g. the cost for each process 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*. The pixel bar chart technique has been prototyped in several business service applications at Hewlett Packard Laboratories. Fig. 3 illustrates 63,544 process instances executed within HP. Each pixel is a process instance. The x-axis is classified by process name; the y-axis is ordered based on the duration (which is the metric we are representing in the figure). The colors (from light to dark) in the different bars represent different duration time values in days.

Pixel bar chart, best viewed with colors, deliver both aggregate and detail information. Aggregation is performed visually by the human eye. At the same time, users can view the detail, since they have instance-level information, represented by the pixels. For example, this is useful to observe situations where the average duration happens to be very high, but where the average value is influenced by a small number of transactions being particularly problematic, while most transactions are just fine. With pixel techniques, this phenomenon can be spotted immediately. On a pixel bar chart, users can also select an area of the chart (possibly spanning across multiple bars), and then drill down, the semantics being that the next analysis will only focus on the selected pixels (instances).

References

[1] D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. Procs. of VLDB'01. Rome, Italy, Sept. 01.

[2] M.Sayal, F. Casati, U. Dayal, and M.C. Shan. Business Process Cockpit. Procs. of VLDB'02. Hong Kong, China, Aug. 02.

[3] D. Keim, M.C. Hao, U. Dayal. Pixel Bar Charts: A Visualization Technique for Very Large Multi-Attribute Data Sets. Information Visualization Journal. 2001