Intelligent Information Management, **2013, 5, 35-41**
http://dx.doi.org/10.4236/iim.2013.52005 Published Online March 2013 (http://www.scirp.org/journal/iim)

A Comparison of Several Performance Dashboards Architectures

Mohamed Abdelfattah
Information System Department, Faculty of Computers and Information, Benha University, Benha, Egypt
Email: drmohabdo@yahoo.com

Received February 8, 2013; revised March 10, 2013; accepted March 19, 2013

ABSTRACT

A performance dashboard is a full-fledged business information system that is built on a business-intelligence and data-integration infrastructure. It has been one of the most hot research topics. Now many corporations have involved in the performance dashboard Architectures related techniques and many performance dashboard Architectures have been put forward. This is a favorable situation to study and application of performance dashboard related techniques. Though interesting, there are also some problems for so many Architectures forms. For to a novice or user with little knowledge about performance dashboard Architectures, it is still very hard to make a reasonable choice. What differences are there for different performance dashboard Architectures and what characteristics and advantages each has? To answer these problems, the characteristics, architectures and applications of several popular performance dashboard Architectures are analyzed and discussed in detail. From the comparison of these Architectures, users can better understand the different performance dashboard Architectures and more reasonably choose what they want.

Keywords: Dashboard; Dashboard Architecture; Data Federation

1. Introduction

A performance dashboard is a layered information delivery system that parcels out information, insights, and alerts to users on demand so they can measure, monitor, and manage business performance more effectively.

Although dashboards seem to have caught on as a management tool, the scientific literature has failed to keep pace with the developments. While textbooks [1,2] and articles in business press [3,4] on dashboards abound, only a handful of studies can be found in academic journals, providing little guidance for practitioners [5] and researchers. The few scientific studies on dashboards have looked at the motivation [6], implementation stages of dashboards [6], and selection of metrics [5], which left the critical issue regarding their design essentially unaddressed. However, there does not seem to be a simple solution to dashboard design. For example, the literature on information presentation format, which dashboards draw on, is in disagreement over the techniques of visual representations that should be used to improve decision making [6]. It is against this backdrop that we believe a review of the current literature that dashboards draw on as well as the development of an agenda for dashboard research could offer value to both practitioners and researchers. The dashboard architecture consists of the components that comprise the performance dashboard. The components in each layer represent a superset of functionality. Developers select one or more components (or buy a dashboard with the requisite combination of functionality) that best serves the needs of the stakeholders [7].

More papers [7] focused on dashboards from a design perspective in this paper; we will focus on the technical challenges and difficulties involved in integrating dashboards with legacy systems and applications that feed data into them. Also, if performance is an issue, an optimum data refresh rate would need to be established, which would depend on the needs and roles of the users.

The architecture of performance dashboards have followed the trajectory of software architectures in general, from mainframe computing to client/server computing to Web-based architectures. Today, rich Internet applications (RIAs) are gaining in popularity because they support dazzling multimedia and visual effects, boosting the appeal and usability of performance dashboards. RIAs, such as Adobe Flash, enable Web-based applications to exhibit the richness and interactivity of desktop applications.

The architecture of any software product involves weaving together three distinct sets or layers of functionality: 1) user interface; 2) application logic; and 3) data processing.

These functions typically are processed on one or more tiers of computers. Knowing where and how these layers are processed in your performance dashboard architecture is the key to understanding whether the system will meet the unique requirements of your organization. In the early days of computing, all three layers were processed by a single machine, the mainframe computer. By the 1990s, the layers were distributed across multiple machines in various types of client/server configurations.

Typically, the user interface was rendered by a desktop machine; the application logic was distributed between the desktop machine and one or more application servers; and data was processed by a relational database management system [8].

There are many the performance dashboards architecttures, each has its own characteristics and advantages, how to make a reasonable choice is a big issue. To this problem, a detailed introduction and comparison of several popular performance dashboards architectures is presented in this paper. From the analysis and comparison, users will be clearer to make their decisions. This paper focuses on a few key architectural issues that consider when deploying a performance dashboard. Specifically, it examines the performance implications of various user interface technologies and illustrates a half-dozen data architectures that are most commonly used to support dashboard implementations and products.

2. Dashboards: Data Architectures

Different types of performance dashboards employ different architectures. Operational dashboards tend to query source systems directly and apply minimal transformations, while tactical dashboards query both historical and current data from a data warehouse. Strategic dashboards often create a local data mart to cache time-series data for specific metrics. This section provides a high-level examination of various data architectures that support performance dashboards, including benefits and drawbacks.

2.1. Direct Query

These tools are ideal for creating operational dashboards where users want to view current data from multiple systems in one place and don ’ t need to perform a lot of drill-downs or analysis, or create reports. Direct query tools generally support interfaces to a variety of SQL and online analytical processing (OLAP) databases as well as various file types and packaged applications. Since these tools usually don’t have a semantic layer, dashboard designers typically hand-code the SQL, embed it in dashboard objects (e.g., charts or tables), and define when the queries execute (e.g., on demand, at predefined intervals, according to a schedule) Data Transformations typically are done on the fly using SQL in memory on the server using Excel or a lightweight transformation engine **Figure 1**. The tools generally don’t store data locally, although many have some sort of snapshot capability that can create a time-series data set, if needed [9].

The benefits of a direct query architecture is that it is fast to deploy and fairly inexpensive and offers flexible data access. There is no semantic layer or elaborate security schemes to implement, no need to define facts or attributes, or to create hierarchies. The tools make it easy for developers to pull information from multiple systems and display it in one place so users don’t have to hunt for the data on their own.

On the downside, the tools aren’t dimensionally aware, so to create hierarchies, drill paths, or summaries, need to write scripting. The queries are generally hard-wired so that if a source system changes, it will break the queries, which will have to be rewritten. Also, the dashboard issues queries to a remote database to populate metrics with data or perform a drill-down. If the queries are complex or require lots of transformation, developers will need to rerun queries and cache the results or create aggregate tables in a data mart or data warehouse (DW) to ensure adequate performance. As a result, the tools generally require an IT developer to configure and run.

2.2. BI Tools

A BI tools architecture see **Figure 2**. issues queries via a semantic layer that simplifies data access by converting database schema into business-oriented dashboard objects (e.g., metrics, attributes, dimensions) that users can drag and drop onto a dashboard canvas to build dashboard pages BI tools are great for building an enterprise BI environment on a common set of data. BI report developers generally build dashboards using the BI tool’s reporting module or a specialized dashboard module. The reporting module enables developers to create reports in a dashboard format. Each chart in a report can link to more detailed data in the report or, if supported, link to an OLAP module that enables users to analyze the data dimensionally. Typically, BI tools run against a DW or data mart designed with a star or snowflake schema. However, in recent years, BI vendors have expanded the data access capabilities of their tools so that their semantic layers can query and join data from multiple sources. In some cases, the data are joined on the BI server via a transformation function while in other cases the tools support a federated query capability [9].

The benefits of BI tools are generally dimensionally aware, enabling full drill-down/across and slice/dice capabilities, as long as the toolset seamlessly integrates reporting and OLAP navigation capabilities. Also, if organization has already deployed a BI platform and a DW and most of the data that dashboard users want to see

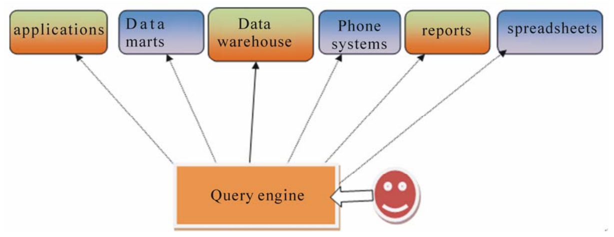

Figure 1. Direct query architecture.

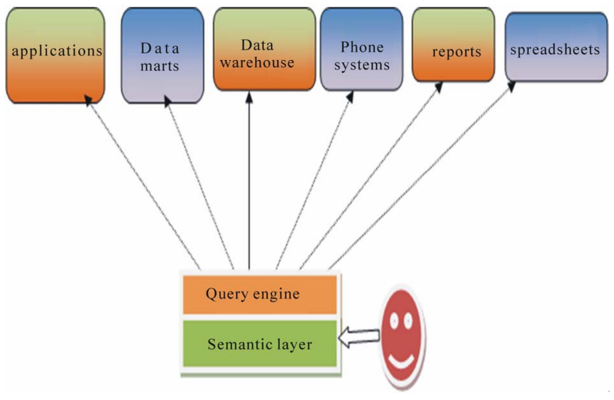

Figure 2. BI tools architecture.

already exist in the DW, it is very quick and inexpensive to build new dashboards. On the downside, BI tools issue queries to a remote database to populate metrics with data or perform a drill-down. If the queries are complex or require lots of transformation, developers will need to prerun queries and cache the results or create aggregate tables in a data mart or DW to ensure adequate performance.

2.3. Mashboards

Mashboards enable power users to drag and drop predefined content from a BI tool and external Web pages onto a dashboard canvas and "mash" them together. Mashboards are ideal for creating ad hoc dashboards for themselves.

A mashboard is effectively a dashboard container for predefined BI content (e.g., charts and tables), various types of controls (e.g., radio buttons, tabs, selectors, pick lists), and external Web pages. Each mashboard object can pull data from a different source and be updated at different intervals **Figure 3**.

Mashboards don't require a DW or data mart, just a BI tool that can create reports and convert them into report "parts" or gadgets. Gadgets are mini-applications that run inside a container environment and have associated functions (e.g., a toolbar for sorting, filtering, charting, etc.) and services, such as the ability to communicate with other gadgets. For instance, gadgets may synchronize their displays automatically using a common filter without scripting.

The downside of mashboards architecture is first implement a BI environment and write reports using professional report writers, this can take time and cost a significant amount of money. Also, mashboards generally limit users to using existing BI content; finally, mashboards may not support all the functionality offered in the complete BI platform.

2.4. In-Memory Dashboards

In-memory dashboards have become quite popular lately because they are quick to deploy, affordable, and fast. The tools load all data into memory, providing speed-of-thought visual analysis. (See **Figure 4**) Also known as visual analysis tools, in-memory tools are ideal for departmental dashboards without stringent data freshness requirements. Originally, visual analysis tools were designed for business analysts who wanted to explore and manipulate small- to medium-size sets of data in a visual manner. The tools excel at rapidly filtering data with lots of variables and displaying results using a variety of chart and table types. The tools make it easy for analysts to explore data quickly, identify trends and outliers, create custom groups, and apply regression and other statistical functions. Although most data sets are loaded into memory in a batch job at night, most visual analysis tools have a query engine so users can, if they desire, add data to their data set [9].

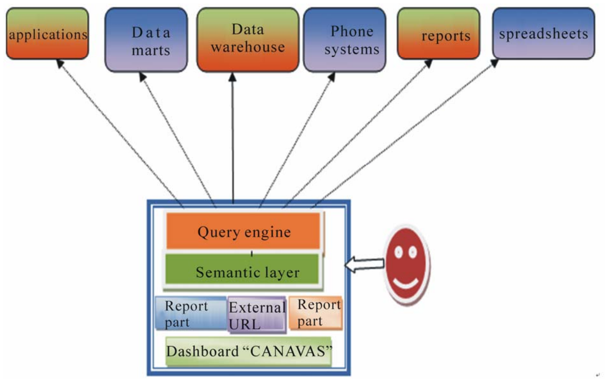

Figure 3. Mashboards architecture [9].

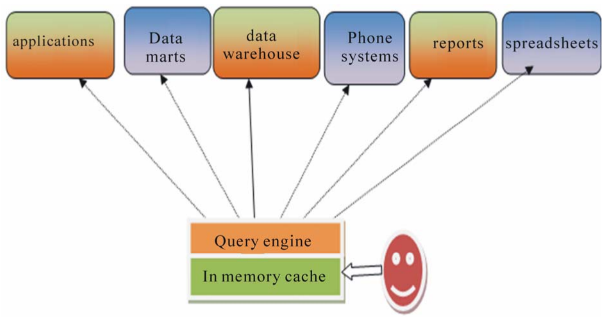

Figure 4. In-memory dashboards architecture.

Today, visual analysis tools have become popular with casual users because of their highly visual, interactive interface. Typically, a business analyst creates an analysis or “dashboard” using the desktop tools and then “publishes” the view to a server where others can access and interact with the live view. Analysts generally simplify the display and turn off more advanced functions to make the tools more accessible to casual users. And visual analysis vendors have added features that make it easier to create and publish analytical output as departmental dashboards. Because of their in-memory architecture, visual analysis tools can’t easily scale beyond the amount of data that the desktop or server machine can hold in memory. Users must be careful not to exceed the size of allowable in-memory storage, or they could lose data unless the tool has a mechanism to spill lightly accessed data to disk automatically. Visual analysis tools aren’t dimensionally aware and support lightweight transformations only. If you need to clean, transform, or integrate data derived from complex database schemas or create complex drill-downs or aggregations, you are going to do a lot of scripting.

2.5. Data Federation

Data federation uses distributed query technology and a global semantic layer to query and join data from multiple sources on the fly and display the results in one or more objects on a dashboard screen. (See **Figure 5**) Data federation is ideal for creating dashboards when the data are spread across multiple systems yet users want a consolidated view of information. Many BI tools now embed data federation capabilities to provide more flexible data access. Data federation integrates data from multiple, disparate sources inside or outside the organization on the fly. It provides business users and application developers a single, easy-to-use interface to access heterogeneous sources, making remote data appear as if it resides in a single local database. When users submit a query, data federation software behind the scenes calculates the optimal way to fetch and join the remote data and return the result [9]. Often the tools ship data from one database to another to perform a join. The ability to shield users and application developers from the complexities of distributed SQL query calls and back-end data sources is why some vendors call this technology data virtualization software. Data federation is ideal to create a full-fledged data mart or DW. Some federation architectures leverage XML and Web. On the downside, data federation works only when source systems are available and have enough capacity to handle streams of ad hoc queries without bogging down transaction processing tasks. Generally, the tools work best with small volumes of data that are clean, consistent, and don’t require much transformation. To obtain the best performance, it’s best to use data federation to support short, tactical queries on current data rather than strategic queries against large volumes of historic data.

2.6. Data Marts

Data marts create a local store of data explicitly to support the performance dashboard [10]. Data marts are designed to ensure adequate performance for dashboard metrics and ancillary applications that go beyond basic metrics monitoring. These marts include strategy maps, time-series charts, multidimensional analysis, what-if modeling, collaboration, text-based annotation, and reporting. (See **Figure 6**) Typically, designers create a logical data model or schema in a relational or multidimensional database that supports the dashboard’s metrics and applications. Then developers create source-to-target transformation code that populates the data mart with the correct data, usually in a batch operation at night. The data mart then accumulates data over time to support time-series and other historical analyses and calculations. Some data marts may store only summarized data, but others especially those supporting operational and tactical dashboards [9] may contain detailed data. And some direct query and in-memory tools maintain a local store of transformed data to support hierarchical drill-

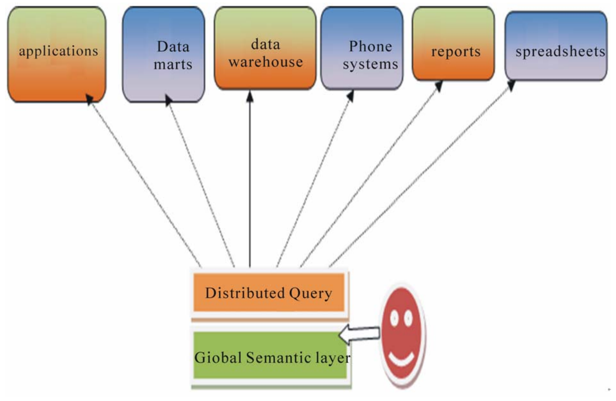

Figure 5. Data federation.

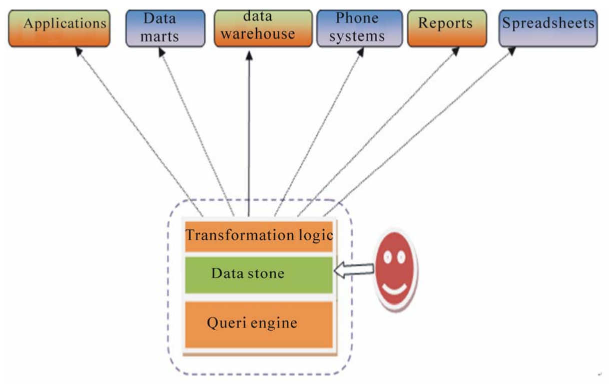

Figure 6. Data marts.

downs, dimensionalized queries, and what-if modeling. Data marts often are used to support strategic dashboards (*i.e.*, balanced scorecards) that require elaborate data models to support unique features, such as strategy maps, initiatives tracking, and text documents. A strategic dashboard model typically includes tables for objectives, metrics, people, organizational structures and hierarchies, and initiatives. The main benefit of data mart architecture is that it minimizes the risk of poor performance when data are scattered across multiple systems. Instead of trying to query remote databases on demand, as other architectures do, a data mart consolidates the data up front and stores them in a form that is conducive to fast queries. A data mart can prejoin and preaggregate data to support metrics and views in the dashboard so the query and transformation logic doesn't have to do this on the fly. A data mart is also needed to support application modules, such as strategy maps, initiative tracking, and metrics maps, among other things, that require complex joins among multiple tables. On the downside, a data mart, like its DW big brother, assumes that you know in advance what metrics, applications, and data you are going to use in the dashboard. Also, it is harder to support near-real-time data delivery in a data mart since data transformation and query execution are separate jobs, not part of a single query stream, as in other architectures.

2.7. Complex Event Processing

Complex event processing (CEP) captures and filters real-time events and triggers actions based on predefined rules. CEP is ideal for supporting operational dashboards that are used to monitor real-time processes, such as trading systems, sensor data from pipelines, global positioning systems, or radio-frequency identification chips, or traffic data from computer networks, transportation systems, and Web sites, among other things. Most CEP systems contain: a data acquisition engine that captures events streaming off a messaging backbone as well as historical data from a DW or external sources; a calculation engine that aggregates events over time and holds them in memory; and a rules engine that defines targets and actions to take when an event object exceeds a predefined threshold (See **Figure 7**) [9]. CEP is like an intelligent sensor that takes a continuous reading of activity generated by one or more interrelated business processes and detects patterns that trigger automated responses. The systems are designed so that business users can build the rules for creating objects and triggering responses [9].

The main advantage of CEP is that it provides built-in support for real-time monitoring so that you don't have to re-architect a DW to support trickle feeds or near-real-time refreshes. But CEP is not just for real-time data; it can extract historical data from a warehouse and use it to compare to current data when executing rules. Some small-and medium-size businesses have deployed their DWs using CEP, giving them real-time capabilities from the start. On the downside, the systems have a lot of moving parts and require technical expertise to set up and maintain. Plus, from an analytical perspective, because they cannot store large volumes of historical data, they are more appropriate for niche applications rather than a large DW. Also, users require significant training to set up and manage rules to ensure that the system doesn't spit out lots of irrelevant alerts or, conversely, fail to notify appropriate users about significant events.

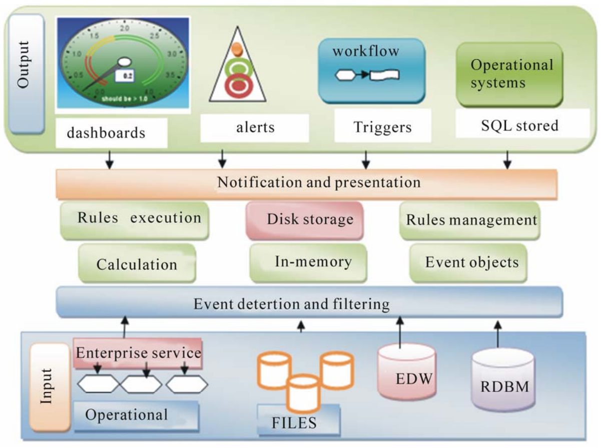

Figure 7. Complex event processing.

3. A Comparison the Architecture of Performance Dashboards

Currently, there are kinds of the architecture of performance dashboards; each has its own characteristics and advantages. To better understand these architectures, we analyze in detail and give a comparison from different implementation aspects. Direct Query is ideal for creating operational dashboards compared to other architectures but isn't data dimensionally. Mashboards generally limit users to using existing BI content compared to other dashboard architecture. Data Federation, Data Marts and Complex Event Processing are more strength in built in dashboard architecture that have many characteristics such as Data Federation built on distributed query technology and a global semantic layer to query and join data from multiple sources on the fly and display the results in one or more objects on a dashboard screen; Data Marts often are used to support strategic dashboards (*i.e.*, balanced scorecards) that require elaborate data models to support unique features; CEP is that it provides built-in support for real-time monitoring so that you don't have to re-architect a DW to support trickle feeds or near-real-time refreshes. The characteristics and implementation of these architectures are summarized as **Table 1** shows. From

Table 1. Comparison of dashboard data architecture.

	Direct Query	BI Tools	Mashboards	In Memory Dashboards	Data Federation	Complex Event Processing	Data Marts
Mechanisms	Screen elements linked directly to individual queries	Query objects that represent a database in business terms for users.	Drag and drop predefined content from a BI tool and external Web pages onto a dashboard canvas and "mash" them together.	Queries populate a queryable cache	An EII tool dynamically queries data from multiple sources to populate screen elements	captures and filters real-time events and triggers actions based on predefined rules	Dashboard queries its own persistent data mart loaded in batch.
	Deploy quickly	Abstract query objects	Ad hoc dashboards.	Deploy quickly	Multiple sources	built-in support for real-time monitoring	Multiple sources
Pros	Low cost	Dimensional views	Don' t require a DW or data mart, just a BI tool that can create reports and convert them into report "parts" or gadgets.	Fast response	Semantic layer abstraction	extract historical data from a warehouse	Dimensional model
				Rapid navigation	Quick to deploy		Historical context
					Prototype before you persist		Fast complex queries
	No depth, limited drill down	Generic ODBC connections	Take time and cost a significant amount of money	Static data sets	No history	cannot store large volumes of historical data	No right time data
Cons	No dimensions	Primarily historical data in DW	Not support all the functionality offered in the complete BI platform	visual analysis tools aren't dimensionally	Data quality issues	fail to notify appropriate users about significant events	Non-integrated
	Hard-wired queries			a lot of scripting	Complexity		

Table 1, it can figure out that the implementations of this architecture of performance dashboards are quite different, there are much common between them.

4. Conclusion

A dashboard is a new technology widely studied in recent years. Now there are numerous ways to architect a performance dashboard. How to understand and use these architectures is a big issue. Each has its trade-offs, and many companies use multiple approaches to support their performance dashboards. The key to selecting the right architecture understands user requirements and the complexity of the metrics and applications the performance dashboard needs to support. Focused on the aspects such as the architectures, characteristics, application and so on, a detailed comparison has been presented in this paper. From the analysis and summarization, users can better understand the characteristics and better choose of dashboard architecture. In future, we will measure the performance of data-dashboard architectures and will develop the optimal dashboard-data architecture.

REFERENCES

[1] S. Few, "Information Dashboard Design, the Effective Visual Communication of Data," O'Reilly Media, Inc., Sebastopol, 2006.

[2] N. Rasmussen, C. Y. Chen and M. Bansal, "Business Dashboards, a Visual Catalogue for Design and Deployment," John Wiley & Sons Inc., Hoboke, 2009.

[3] A. Miller and J. Cioffi, "Measuring Marketing Effectiveness and Value: The Unisys Marketing Dashboard," *Journal of Advertising Research*, Vol. 44, No. 3, 2004, pp. 237-243. doi:10.1017/S0021849904040334

[4] T. Kawamoto and B. Mathers, "Key Success Factors for a Perfomance Dashboard," DM Rev, 2007, pp. 20-21. http://www.information-management.com/bnews/2600366-1.html

[5] K. Pauwels, T. Ambler, H. C. Bruce, P. LaPointe, D. Reibstein, B. Skiera, *et al*., "Dashboards as a Service: Why, What, How, and What Research Is Needed?" *Journal of Service Research*, Vol. 12, No. 2, 2009, pp. 175-189. doi:10.1177/1094670509344213

[6] E. O'Donnell and J. S. David, "How Information Systems Influence User Decisions: A Research Framework and Literature Review," *International Journal of Accounting Information Systems*, Vol. 1, No. 3, 2000, pp. 178-203. doi:10.1016/S1467-0895(00)00009-9

[7] O. M. Yigitbasioglu and O. Velcu, "A Review of Dashboards in Perfomance Management: Implications for Design and Research," *International Journal of Accounting Information Systems*, Vol. 13, No. 1, 2012, pp. 41-59. doi:10.1016/j.accinf.2011.08.002

[8] L. Zeng, H. Lei, M. Dikun, H. Chang, K. Bhaskaran and J.

Frank, “Model-Driven Business Performance Management,” *IEEE International Conference on E-Business Engineering ICEBE*05, Beijing, 2005, pp. 295-304.

[9] W. W. Eckerson, “Performance Dashboards Measuring, Monitoring, and Managing Your Business,” John Wiley & Sons, Inc., Hoboken, 2011

[10] C. Vercellis, “Business Intelligence: Data Mining and Optimization for Decision Making,” John Wiley & Sons Ltd., Hoboken, 2009.