Balanced Scorecard Report
THE STRATEGY EXECUTION SOURCE
Article Reprint No. B1003C

Using Dashboards to Revolutionize Your Performance Management System, Part 2: Implementation

By Mark J. Lorence

HARVARD BUSINESS PUBLISHING
Palladium
Executing Strategy

For a complete list of Harvard Business Publishing newsletters:
http://newsletters.harvardbusiness.org

For reprint and subscription information for Balanced Scorecard Report:
Call 800-988-0866 or 617-783-7500
http://bsr.harvardbusinessonline.org

For customized and quantity orders of reprints: Call 617-783-7626 Fax 617-783-7658

For permission to copy or republish:
Call 617-783-7587

MARCH–APRIL 2010 : VOL 12 NO 2

Using Dashboards to Revolutionize Your Performance Management System, Part 2: Implementation

By Mark J. Lorence, Director, Palladium Group, Inc.

*Besides clarifying the differences between dashboards and scorecards, Part 1 of this article (*BSR *January–February 2010) discussed two major approaches to dashboard design: bottom-up and top-down. Here, Mark Lorence examines three key considerations of dashboard implementation, offering valuable do's and don'ts from actual implementation projects.*

When you move from design to implementation—to building, testing, and deploying your dashboard application—the specific technology you choose will, of course, have significant impact. The amount of custom coding required to implement the dashboard screens, the ease of accessing multiple data sources, and the overall performance of the resulting application are all dependent on the particular software tools you choose. So are the maintenance, usability, and reliability of the application: adding new dashboard measures, modifying screen views, and updating the dashboard with new data. Every software platform has its own strengths, weaknesses, and idiosyncrasies, and examining the risks associated with various software platforms is beyond the scope of this article. However, regardless of the technology you choose, there are three general considerations you must address to implement your dashboards properly: ensuring data accuracy, allowing for iteration, and testing sufficiently.

Ensuring Data Accuracy

You've no doubt heard the real estate maxim, "There are only three things that matter: location, location, location." For dashboards, they're "data, data, data." The biggest challenge in implementing a robust dashboard solution is *ensuring the accuracy of the underlying data used to populate the dashboard measures.* Dashboards contain numerous key performance indicators (KPIs), which are often calculated by rolling up lower-level numbers—a process that can introduce errors into the final results. When planning your dashboard implementation, ensure adequate time in the project plan to address three specific data considerations.

1. How available or accessible is the data?

Simply put, is the data available in electronic form? Although financial data may be easily accessible from your general ledger system, operational data is often maintained in hard copy format, making it difficult to incorporate into the dashboard system. If so, additional input tools may need to be developed. For one large dashboard implementation project we carried out, only 30% of the data elements were available electronically; 50% were available from paper documentation and logbooks, and the remaining 20% weren't available at all. Most of the 30% were financial measures easily obtained from the client's back-end systems. For the 50%, we built several input templates for manually entering and validating the data before uploading it to the dashboard database. A planned enhancement will eventually replace the paper logbooks with automated systems, and a middleware application will transfer the data at the appropriate times.

Collecting the unavailable data required both technical and process changes. For example, the company wanted to track the reasons customers were switching to a competitor. The company knew which customers were defecting, when they were switching, and—in some cases—which competitor they had switched to. But the customer service representatives weren't documenting the reasons customers were switching, making it impossible to display an accurate measure. The "switch out" process metric was modified to include a reason code, reps were trained to record this reason code along with the other data, and the results were aggregated and displayed on the dashboard.

Such changes, of course, require additional testing time to ensure that the measures—especially the new ones—are being captured and summarized accurately. Examples like this confirm why a large dashboard implementation often represents a good opportunity to undertake an enterprisewide data validation initiative to ensure that data is clean and consistent throughout the organization. This can be a daunting task, given the number and diversity of legacy systems in most large organizations. But at a minimum, consider taking the same "go deep" approach we discussed in Part 1 and tackling a subset of your data—say, a specific function or group (e.g., revenues and expenses for finance, or head count and employee job level for HR).

2. Has your organization established common definitions? Is the definition of each measure clear to all users?

Dashboard projects would certainly be simpler if every measure were as clearly defined as "When the red light appears on your gas gauge, you have one gallon of gas left in the tank. Find a gas station—quickly." In reality, the aggregation of underlying data into dashboard measures and KPIs can introduce ambiguity, especially when time periods are being measured.

Take "On-time delivery." Does the value of your metric represent an average for

last quarter, or yesterday's result? That must be clear so that every user is working from the same assumptions. When low-level values are aggregated into summary values, the dashboard display must indicate the time period. You can accomplish this through naming ("Daily on-time delivery results for Thursday, July 8, 2010") or by grouping measures with similar time periods on the same screen and including an "as of" date at the top of the screen. For consistency, standardize these dates; if you're displaying weekly values, make them all "as of Monday" or "as of week ending MM/DD/YYYY." Even financial ratios, like return on invested capital (ROIC), can be calculated differently. The textbook definition is straightforward: ROIC = (operating earnings × (1–tax rate)) ÷ invested capital. Yet some companies include goodwill and intangible assets in the calculation of invested capital. Make sure you're calculating such values consistently across all the dashboard screens in your organization.

Don't forget to clarify measures that have similar names, especially if users will be expecting certain data to reconcile between dashboard screens. We were implementing a dashboard for a health insurance company that had a summary measure of "patient days" on one screen and detailed measures for individual departments on other screens. The values, however, didn't add up. The detailed screens showed 100, 200, and 300 patient-days, but the summary screen showed 620. All four values (100, 200, 300, and 620) were being calculated from detailed transactional data; the total was not being derived by adding the figures from the three departments. The reason for the discrepancy? The word "day" was defined differently throughout the company. For some departments, a "day" started at midnight, but for others, it began when the patient was admitted to the hospital. So before we could display any meaningful dashboard metrics, we had to establish common definitions for the specific data elements, calculate the data correctly, and aggregate it accordingly—ensuring we were comparing apples to apples when we displayed the metrics on the screens. In some cases—especially when using data feeds from external sources—this may require additional manipulation of the input data stream before it is loaded and calculated in your dashboard database. For the health insurance company, the resulting common definitions were documented in a data dictionary that was accessible from the dashboard and used for online help as well as for all subsequent modifications to the dashboard screens.

3. How well do you manage different data sources?

Does the data come from multiple sources? The most actionable dashboards are those that include not only *internal* data from the organization's financial or operational systems but also *external* data from websites, commercial databases, and other sources. External data would include sales or market share information from third-party providers, website analytics from Internet service providers, and compliance statistics from government agencies.

External sources introduce another layer of complexity into your implementation, because each source must be identified and the data brought into your dashboard in a standardized, consistent fashion. Robust dashboards require a significant technical infrastructure, including an extract, transform, and load (ETL) layer. There are several tools that can perform ETL, but all require extensive coding to ensure the data is being loaded correctly.

Data accuracy and common definitions must also be addressed, along with the actual format of the data elements: for example, is the data in a comma-delimited file? Are leading zeroes suppressed? How are negative numbers represented? Most providers publish specifications for their data files, which must be incorporated into your ETL layer and updated if the provider's format changes. This is especially important if the provider is sending low-level information (e.g., "market share by ZIP code") that needs to be aggregated and displayed in a different format (e.g., "market share by county"). Don't assume that every provider looks at data the same way your organization does.

Allowing for Iteration

The second major consideration is the need for iteration. Unlike transactional systems, in which you're automating a known process and can design 90% of the functionality in the first pass, it takes several iterations to get a dashboard right—that is, before the final dashboard application is usable. This is largely because when users see what a dashboard can do, they invariably identify additional functionality or analytical support that they want incorporated into the application. One company's dashboard application included a screen displaying sales and orders by business unit. Executives used this screen during management meetings to review their sales pipeline and identify factors affecting projected revenue. While testing the first iteration, we noticed that much of their discussion focused on orders for which the revenue could not be booked because the shipments were being detained by customs officials in foreign ports. So we added indicators to the screen that would flag new orders as potentially troublesome and calculated a "customs adjusted" revenue number for planning purposes. In other situations, users have asked to change tabular displays to graphical ones, or vice versa. Sometimes these kinds of preferences can be identified in the design phase through the use of paper dashboard screen mock-ups and use cases, as we described in Part 1. But more often, new requirements arise when users interact with a functioning application.

One way to manage the iteration process is to build a prototype—a functioning dashboard application with limited scope that gives users a chance to interact with the application before the final version is built. This allows you to test the screen layouts, the types of measure displays, and the overall usability of the dashboard. It also allows you to tailor the implementation to specific user groups—adding extra detail screens for those who require drill-down capability; augmenting the summary

screens with additional, action-based measures; or streamlining the flow between the variance, trend, and forecasting analytical paths common in most implementations.

Sometimes the results of the prototype are surprising. We built a prototype dashboard for an energy company, taking advantage of the new graphical capabilities in the latest version of one software tool; we had speedometers, gauges, dials, even a map of the country with push-pin icons showing the location of their power plants. When we presented the prototype to executives, they said, "That looks great, but could you give us something that looks a little more like the Excel spreadsheets we're used to? We prefer rows and columns of numbers rather than those maps." We modified the design to include a mix of graphical and tabular data.

Testing Sufficiently

The third major consideration is testing. Given the amount of data from multiple sources, testing time for a dashboard implementation needs to be double—or even triple—that required for a typical transactional system. Nothing will undermine your dashboard effort—or, worse still, your decision making—more than a dashboard that displays incorrect information. It's in the testing phase that organizations often uncover data disparities or bugs.

Dashboard testing should be both automated and manual. Scripts can be written to load data, update the database, and populate the dashboard screens. Typically, these scripts will set the dashboard to a "known" state, such as the first day of the month or the last day of the quarter. Dashboard values are then set to known quantities, which provides a clean starting point for testing. Scripts can also be used to simulate the passage of time; from the clean starting point, data can be loaded to the dashboard to represent the next day, next week, next month, or next quarter. These scripts test how easily the data loads into the dashboard database as well as indicate whether additional manipulation or calculations are required before the resulting metrics are displayed. Manual testing can be done to improve navigation and information display. Users walk through the dashboard's capabilities and validate numbers, thinking out loud as they interact with the dashboard—saying what they'd like to see, where they expect the information to be, and what they will do with the result. This testing is similar to the usability testing done for large websites.

Keep in mind that the latest dashboard designs are incorporating capabilities for mobile devices such as iPhones and BlackBerrys. These devices extend the functionality of your dashboard by providing key information to users even when they're not at their desks, but they also add to the testing time because there's another delivery vehicle that must be validated.

The Value of Doing It Right

Implementing a dashboard application is a challenging task. You have all of the development issues inherent in a transactional system combined with the usability demands of an interactive website, complicated by the need to satisfy a diverse group of users, each of whom has different information needs and preferences. The leading software vendors are making dashboard implementation easier, but the process still requires disciplined project management, a thorough knowledge of the software tools, and attention to the three major considerations described here. Dashboards are an integral part of your performance management system and, when designed and developed correctly, will help you achieve revolutionary results. ■

Mark J. Lorence is a member of the Strategy practice at Palladium Group. His recent work focuses on linking strategic objectives to operational measures via dashboards to improve performance management systems.

Continue the dialogue

Palladium Execution Premium Community (XPC) members can continue the discussion at: www.thepalladiumgroup.com/LorenceBSR

Not a member? Join at: ***www.thepalladiumgroup.com.***

Implementation Gotchas

Local software variations. *While using a leading spreadsheet's add-on to input and validate weekly sales numbers, we discovered that there were differences between the German and U.S. versions of the underlying Visual Basic macros. If your dashboard will have international users, make sure you test the input as well as the output tools.*

Page-display capacity and response time. *Most dashboard development tools limit the amount of data that can be displayed before performance is impaired. For one leading tool, if the number of cells (rows × columns) exceeds 1,300, performance will degrade significantly. Be aware of the technical limitations of any tool before you design screens with voluminous data.*

Security. *Different tools apply security controls at different levels. Some tools limit you to the screen level, others to the field level. If you have different users with different levels of access, make sure you design the screens to display information at the appropriate level. Displaying "Current Head Count" is probably acceptable for all users; displaying "Current Average Salary of Level 13 and Above" probably is not.*

Traffic lighting. *All tools allow you to color-code values with red, yellow, or green traffic-light icons. However, the threshold values may be different based on the metric; for some values, +/- 5% may be green, whereas for others, the tolerance may be only +/- 0.5%. In a given performance area (such as "monthly warehouse operations"), try to group metrics with similar tolerances together to avoid the visual chaos that can result from too much traffic lighting.*

Browser compatibility. *Although your corporate standard may be Internet Explorer, remember that users may be accessing the dashboard from home machines, older PCs, or even mobile devices. Test a variety of browsers.*

Layout. *Typical analysis paths through a dashboard environment include variance views, trend views, and forecast views. Combining these views on one screen is easier from an implementation standpoint—and enables you to see numbers and trends at the top. The downside? Space limitations will reduce the amount of data you can show on a given screen and make visibility harder. Separating these views across different screens may increase the usability of the application but requires more development steps and time.*