Eckerson Group

Which Embedded Analytics Product Is Right For You?

An Evaluation Framework

Wayne W. Eckerson
April 2016

About the Author

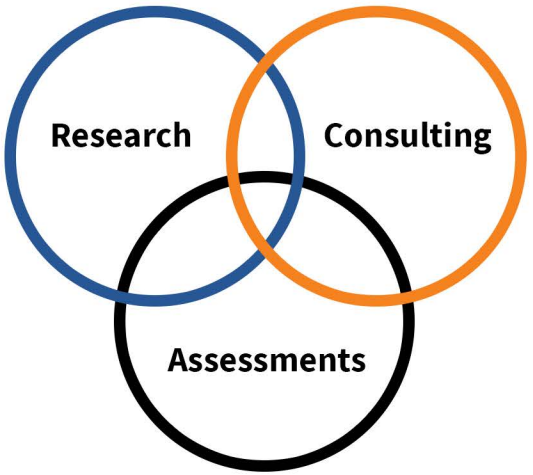
Wayne W. Eckerson has been a thought leader in the business intelligence and analytics field since the early 1990s. He is a sought-after consultant, noted speaker, and prolific author who thinks critically, writes clearly, and presents persuasively about complex topics. Eckerson has conducted many groundbreaking research studies, chaired numerous conferences, and written two widely read books on performance dashboards and analytics. Eckerson is the founder and principal consultant of Eckerson Group, a research and consulting firm that helps business and analytics leaders use data and technology to drive better insights and actions.

About Eckerson Group

Eckerson Group is a research and consulting firm that helps business and analytics leaders use data and technology to drive better insights and actions. Through its reports and advisory services, the firm helps organizations maximize their investments in data and analytics. Its researchers and consultants each have more than 20 years of experience in the field and are uniquely qualified to help business and technical leaders succeed with business intelligence, analytics, data management, data governance, performance management, and data science.

Report Sponsors

The research for this report and its companion report, **"Embedded Analytics: The Future of Business Intelligence,"** is sponsored by the following solution providers: Actuate (now OpenText), Dundas Data Visualization, GoodData, Izenda, Logi Analytics, Looker, Qlik, Sisense, and Zoomdata.



Executive Summary

As data proliferates, organizations are eager to monetize their data assets by injecting outward-facing applications with reports, dashboards, and self-service analytics. Today, these data-driven applications can increase customer satisfaction and create new revenue streams. In the near future, they will be a requirement for doing business.

This report outlines key criteria for evaluating embedded analytics solutions. It discusses three types of analytics that can be embedded (i.e., components, tools, and platforms) and five approaches to embedding analytics, from bundling and coexisting (which were popular in the 1990s) to modular, inline, and infusion methods, which are prevalent today with Web computing. The report then describes 12 criteria for evaluating embedded analytics solutions, ranging from business considerations to product functionality and technical architectures.

The Growth of Embedded Analytics

Organizations that want to put critical data at the fingertips of employees, customers, and suppliers are turning to embedded analytics. The idea is to insert analytic output (e.g., charts and dashboards) and self-service analytic functionality directly inside the applications that business people use every day. Rather than requiring workers to log into a separate product to view and analyze data, embedded analytics puts intelligence inside core applications where business people do most of their work.

Although embedded analytics is not new, it is a hot topic these days. In many respects, it represents the culmination of business intelligence (BI), which consists of the tools, techniques, and technologies that turn data into insights and action. Embedded analytics helps close the last mile of BI by operationalizing insights. It puts insights in the context of a business process and makes it easier for people to take action, such as order parts, send invoices, recommend products, or change prices.

At the same time, embedded analytics has become pervasive. Consumer applications such as TurboTax, Fitbit, and Amazon.com have transformed how people use and interact with data. Many new business applications, such as Concur, QuickBooks Online, and NetSuite, use analytical charts and dashboards to guide user behavior and provide a consolidated view of user activity. As analytics proliferates, people begin to expect and depend on embedded analytics functionality. What is now a competitive differentiator will soon become a requirement for doing business.

There is now an analytics gold rush …. Both software vendors and commercial providers are eager to monetize data assets using embedded analytics.

Monetizing Data. There is now an analytics gold rush, especially among independent software vendors who want to boost revenues by integrating data and analytics into their Web-based products. But it's also relevant for commercial organizations—such as retailers, insurance companies, communications providers, and even governments and non-profit organizations—that build outward-facing applications using in-house teams of developers. Both software vendors and commercial providers are eager to monetize data assets or enhance customer service using embedded analytics.

Technology and Approaches

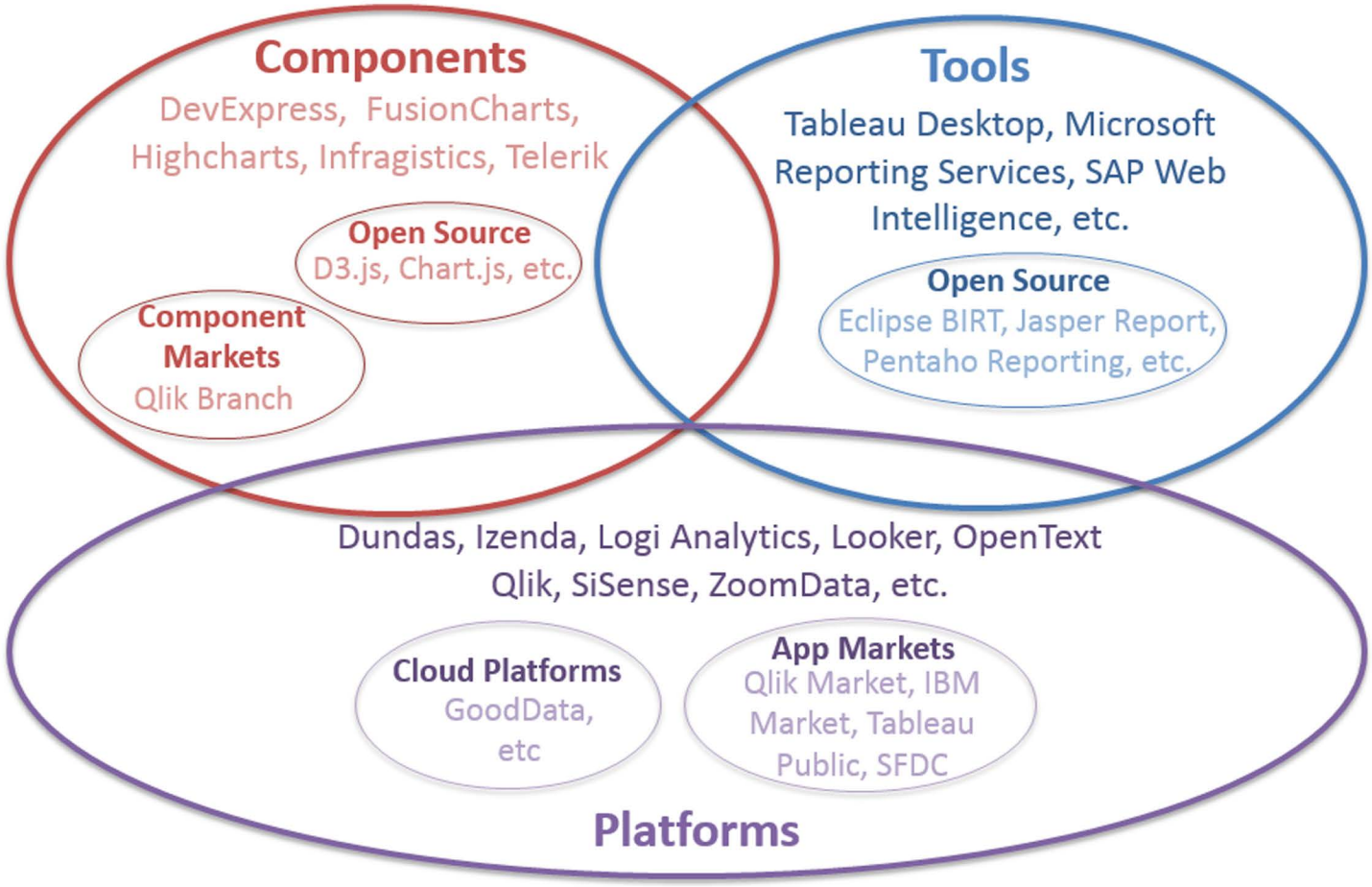
Given the rise of embedded analytics, it's not surprising that many BI and analytics vendors are now aggressively pursuing the market. Some have offered embedded solutions for years, but others are refocusing technical and marketing resources to exploit the new interest in embedded analytics.

Build Option. Although this report focuses on integrating commercial analytics tools within applications, there is another option. Organizations can create reports and dashboards from scratch and insert them into a host application. Or more likely, they can use a combination of BI components (see below), custom code, custom data connectors, and possibly open source reporting software to create custom analytic applications. Organizations generally take a "build" approach if they have a small deployment, need highly specialized features, or only need to access one or two data sources. (See our companion report **Embedded Analytics: The Future of Business Intelligence** for a more detailed discussion of the build versus buy debate.)

Selecting an embedded analytics vendor is a daunting task. There are many products on the market, and some key differentiators are visible only to developers. Since every organization has different business goals, target users, and technical environments, there is no one tool that fits every need. However, the best way to sort through the options is to understand the general categories of embedded products.

Three Types. Analytic products come in three basic forms: components, tools, and platforms. Each supports different styles of embedding and customization. (See figure 1.) To begin an embedded analytics, it's helps to first understand the basic differences between these technologies.

Figure 1. Technology for Embedding Analytics



Components are typically used to build analytics functionality from scratch or to extend an existing analytics implementation with custom visualizations and controls. Tools are purpose-built to support a specific mode of BI (e.g. reports or OLAP) and use white labeling and iFrames to support embedded solutions. Platforms provide an integrated analytic environment that supports the complete range of BI functionality and is optimized for embedded analytics. Platforms offer a rich sets of APIs and SDKs that make it possible for developers to customize and extend the platform and embed it in other applications.

Components

Description. Analytical components are prebuilt objects that can be embedded in a Web page and linked to a BI engine or directly to a data source. The most common analytical components are charts and visualizations such as bar charts, box plots, sparklines, gauges, maps, and so on. Other components include data selectors such as radio buttons and time sliders, layouts, forms, data connectors, site controls, and administrative utilities such as schedulers or engine wrappers.

Several years ago, most visualization components displayed static charts, but today's components are interactive once developers hook them to a back-end analytics engine. For instance, users can click on a chart or table embedded in a Web page or PDF to drill into detail, filter data, swap axes, sort or add columns, or apply a new calculation. Some even display real-time data so the chart twinkles as new events occur.

Most visualization components today are available as JavaScript libraries. Some are free under an open source license, such as the highly popular D3 libraries, which many commercial BI vendors incorporate in their products. Others are licensed by commercial providers, such as Highcharts, DevExpress, Infragistics, and Telerik. Some analytic vendors have created marketplaces where developers can exchange source code for components that extend their underlying platforms. (See figure 1.)

Purpose. Developers generally use analytic components to build custom visualizations and embed them into Web pages and applications. For example, a media company might embed a charting component into an online news story to illustrate the main points of an article. An investment firm might embed charts inside a retirement application to help clients better understand the composition and activity of their financial holdings.

Developers also use components as the foundation for building custom analytics applications. They might use analytical components and an open source reporting framework such as Eclipse BI Reporting Technology (BIRT) to create, publish, run, and maintain a catalog of reports for a software solution. Similarly, they might use analytical components to extend a commercial BI tool with tailored charts and controls desired by target users. In fact, many BI vendors incorporate external JavaScript libraries to give their customers greater ability to customize the look and feel of their BI tools.

Technology. The problem with components is that you have to be a developer to use them. The components must be linked to a data source or analytic engine so they can be populated with data. These components generally don't allow users to export, print, annotate, zoom, selectively expand, or save the content, nor do they come with security, monitoring, or other administrative features, so developers must wrap the components inside a larger application framework. Finally, most components can't interact with other components unless a developer wires them together. This is important in the analytical world, where one chart often serves as a filter for another.

Fortunately, component technology has come a long way thanks to widely adopted Web standards, interfaces, and frameworks such as JavaScript, JSON, cascading style sheets, REST, Ajax, and jQuery. It's now much easier for developers to add analytic features to an application than it was a decade ago when language-dependent components such as COM, ActiveX, Java virtual machines, Flash, and Silverlight ruled the application development world.

Tools

The 1990s

Description. Starting in the 1990s, software vendors began shipping tools that made it much easier for developers and non-IT specialists to build reports and dashboards and disseminate them to a large population of business users inside an organization. These packaged BI tools replaced fourth-generation report programming languages such as FOCUS and RAMIS. Most were purpose-built to support a specific mode of BI, such as reporting, OLAP, dashboards, visual discovery, or predictive analytics.

BI tools have evolved significantly over the years. Many require developers to model data and create a semantic layer to insulate users from the complexities of back-end data sources. Over time, larger vendors consolidated purpose-built tools into a single portfolio or suite of tools that are integrated to varying degrees. Although most BI tools started as desktop or client/server programs with perpetual user licenses, most now run as Web programs and can be deployed in the cloud and purchased with a subscription, either annual, monthly, or, in some cases, hourly. (See "The Cloud BI Landscape: Products and Positioning.")

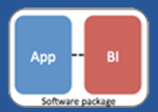
Bundled. Most BI tools weren't designed for embedding into other applications. In fact, in the early days, when Crystal Reports ruled the embedded BI world, software vendors simply bundled analytics software in the same box as the application. Customers had to install the analytics solution separately and then run a utility to link it to the application to run queries. Needless to say, the BI tool had its own graphical interface with a different look and feel from the host application and required users to switch contexts to analyze data and reports.

Coexisting. Soon, BI vendors inserted the code for their products inside the code base of the host application, creating a seamless install and common security and administrative

Six Approaches for Embedding Analytics

There are six ways to embed analytics into applications. The first two methods below were popular in the 1990s when desktop and client/server software dominated. The next two surfaced in the 2000s with the advent of the Web. The final two began appearing in the 2010s as BI vendors began morphing BI tools into analytic platforms. Inline embedding is the most dominant approach today, thanks to standard Web APIs such as JavaScript and REST.

The 1990s



1. Bundled. The BI tool is bundled into an application package, but it is installed and managed separately, and it maintains its own GUI.

2.Coexisting. The BI tool runs as libraries inside an application code base, but it is managed and upgraded separately, and it maintains its own look and feel.

functions. But combining code bases in a single package created interdependencies that made it difficult to upgrade an application with embedded code from a third party, creating sizable headaches for administrators.

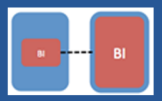
The 2000s

By the 2000s, most BI vendors deployed robust server environments to manage enterprise deployments. By "server-tizing" BI, the vendors created a loosely coupled relationship between the analytics software and host applications. This avoided many upgrade issues and dramatically improved scalability and performance.

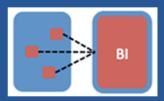
Modular. At the same time, BI vendors began letting customers "white label" their tools-essentially, change the tool's GUI to match their corporate branding and styles. White labeling blurred the visible distinction between analytic tools and host applications, making it feasible for companies to insert BI as a reporting module or dashboard page inside a host application or portal without end users knowing they were consuming a third-party application. This "modular embedding" gave BI functionality a home within an application, but still forced users to shift application context to analyze data and didn't infuse analytics in the guts of an application.

iFrames. With the advent of the Web, analytic vendors began using iFrames to embed charts and tables directly into an HTML page of a host application. This page-level integration was an improvement over modular embedding with its distinct tabs, modules, and pages reserved for analytics. But iFrames are brittle and the content inside them cannot interact with the rest of the page. For instance, hitting the browser's "back" button doesn't change the display inside iFrames. Most companies now use iFrames to display static menus.

The 2000s

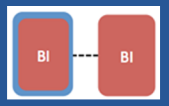


3. Modular. BI functionality (called from a BI server) runs in a separate tab or page in the host application and uses white labeling to conform to the look and feel of the application.



4. iFrames. BI functionality runs inside a Web page using iFrames, but cannot interact with the rest of the page or program.

The 2010s



5. Inline. The host application or Web page calls analytic functions via JavaScript or REST APIs, creating a custom analytic GUI.

6. Fusion. Analytic functionality is fused into the workflow of an application, allowing users to update records and execute transactions without leaving an analytical view.

Platforms

The 2010s

Description. Today, many analytic vendors have taken the "loosely coupled" approach described above to its logical conclusion, morphing BI tools and suites into analytic platforms.

Rather than deliver a packaged set of monolithic tools, analytic platforms support a full stack of integrated analytic functions-from reporting and dashboards to self-service analytics, alerts, and collaboration to data preparation and predictive functions-all of which run on a unified, scalable architecture with common administrative and management functions. (See **Ten Characteristics of a Modern Enterprise BI Tool**) Also, they are designed from the ground up for the Web, the cloud, and mobile delivery. All the sponsors of this report-Dundas, GoodData, Izenda, Logi Analytics, OpenText, Sisense, Qlik, and Zoomdata-ship analytic platforms.

Additionally, analytic platforms provide a development environment for building custom analytic solutions that integrate tightly with other applications. They expose a rich set of application programming interfaces (APIs) that enable developers to integrate analytical output and functionality within Web pages and applications. And most offer software development kits (SDKs) that allow developers to extend existing analytic functions or create new ones, such as new data connectors, custom calculations, and novel visualizations. In short, analytic platforms make it much easier for developers to create highly customized analytic applications.

Analytic platforms are a developer's ideal playground. They tie the freedom of component-based development with a rich set of analytics services provided by a commercial analytic vendor.

Inline. Analytic platforms are a developer's ideal playground. They tie the freedom of component-based development with a rich set of analytics services provided by a commercial analytic vendor. Developers can use an analytic platform's APIs to create a completely custom front end that bears no resemblance to the GUI of the original tool. Because of this, some say that analytic platforms can be "decapitated"-that is, they allow developers to replace a packaged GUI with a custom one that is highly tailored to the needs of target users. This "inline" approach is the most popular form of embedding today. (See sidebar "Six Approaches for Embedding Analytics.")

APIs. Most analytic platforms support a rich set of JavaScript and REST APIs. The JavaScript APIs enable developers to create new visualizations from scratch or customize ones that come out of the box. Others let developers import third-party libraries, which they can hook into the platform's analytic engine. Some platforms provide built-in programming editors, while others let developers use the editor of their choice.

REST and other APIs provide access to back-end administrative and execution functions. They enable users working in the administration console of the host application to add, edit, and delete users and groups, set up and manage data connections, and schedule tasks, among other things. These APIs enable customers to manage a analytic platform from within the host application's administrative console, making it easy for administrators to add, delete, and update users and permissions. This is especially critical in a multi-tenant environment with many hundreds of customers and thousands of end users.

Fusion. Like BI tools, analytic platforms support modular and inline embedding, but they also go deeper and integrate data and analytics into the fabric of the application. This so-called "fusion" analytics allows users to take immediate action inside an application without shifting context. Many fusion applications apply predictive models to an application and let business users execute transactions from within a report or view. Some are used in real-time operational environments, such as transportation, stock trading, and call centers, where business users monitor data and take quick action.

For instance, a transportation analyst might reroute trains while viewing real-time traffic data displayed on an oversize wall display. A call center application might display a loyalty score inside a customer record so customer service representatives can determine the appropriate services or level of treatment to provide a customer in real time. A sales manager might view sales performance across regions and then reorganize sales territories without leaving the sales report. An inventory manage might be able to execute purchase orders to replenish stock or send an alert to a purchasing manager while analyzing inventory levels.

To support the fusion approach, an analytic platform must be able to communicate with the host application via a Web service or API call. In some cases, the analytic platform might query the host database directly via a SQL query to perform a lookup or execute a transaction or it might activate a stored procedure or trigger to execute an action.

Cloud Platforms

Cloud platforms are a subset of analytic platforms. These are platform-as-a-service analytic applications that can host multiple tenants or customers. The tenants share the analytic application, underlying data platform and model, and computing infrastructure, but their reports and dashboards are populated with different data. You could say that cloud platforms enable organizations to embed or cascade analytic deployments, one within another, creating economies of scale.

For example, a large company could deploy multiple, virtual analytic instances within a single enterprise cloud BI environment. The company can create an instance for every department, division, customer, or supplier. All tenants run on the same enterprise analytic infrastructure with the same corporate data model and reports, but can extend the environment to support unique local data and reporting needs. So, rather than deploy physically distinct analytic environments and replicate data and reports, a company can use this virtualized cloud-based approach to preserve corporate data standards and branding, while giving groups the flexibility to create views that meet their local requirements.

App Markets

Finally, a key indicator of the value of an analytic platform (or any application platform for that matter) is the number of applications that run on it. The more popular a platform, the more third-party developers are motivated to write, share, and sell applications, utilities, and extensions that run on it. This increases the value of the platform and makes it possible for software vendors to offer a range of functionality and options that they would never have time or resources to build themselves. Like the open source movement, app markets create a virtual development team that dwarfs the number of in-house developers.

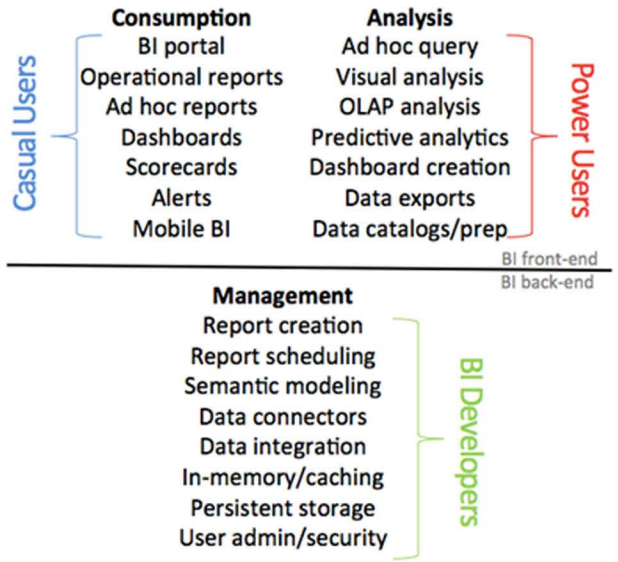
Evaluation Criteria

There are many factors to consider when selecting a analytic tool to embed. Below is a list of the most salient features and functions to consider when evaluating embedded analytic products. Each organization will need to discover the technologies that best map to their unique requirements.

1. Type of Product. As discussed in the previous section, it's important to understand the type of analytic product you want to embed. If you are taking a build approach, you will certainly look at analytic components. If you are buying analytics functionality, focus your research on analytic platforms that can be tightly integrated with other applications.

2. Functionality. The next step is to understand the analytics functionality required by business users, whether internal staff or external customers and suppliers. This requires creating user personas and mapping their information workflows to determine how and when each persona consumes information.

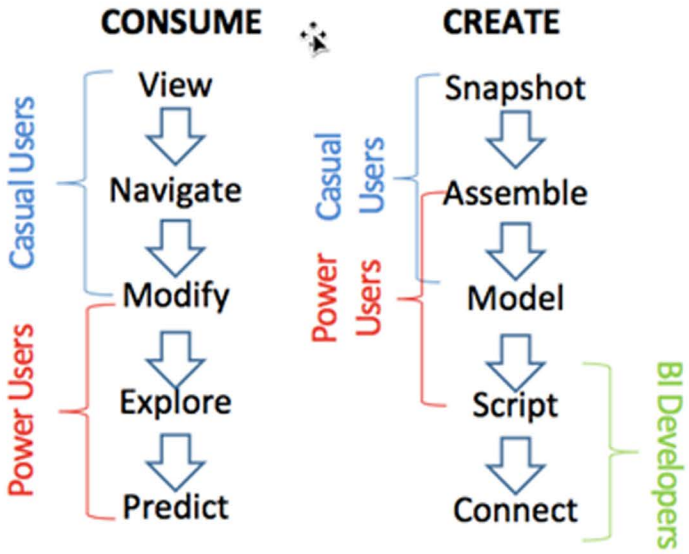
Figure 2. Common analytic Functions



Most business users can be divided into two groups, although an individual user might shift between groups based on their roles: (1) casual users who simply want to view a report or navigate the data in a dashboard, and (2) data-savvy business analysts who want to mash together data from different sources, visualize it, and publish the results in ad hoc reports and dashboards. There is also a third group comprised of BI developers who build complex reports and data models and manage the analytic platform. Each type uses different modules or sets of functionality in an analytic tool. (See figure 2.)

Once you determine analytic modules, then you need to examine how users want to interact with those modules, which raises the issue of self-service in an embedded analytic environment. What a casual user considers self-service is vastly different from what a power user considers self-service. Figure 3 maps the three user personas to self-service capabilities for both consuming and creating content.

Figure 3. Self-Service BI Mapping



Questions to ask:

- Does the analytic product have the requisite functionality to meet the current and future needs of your target users?
- If not, how easy is it to customize or build the functionality into the product?
- How easy is it to use the product to aggregate data and provide benchmarks to users so they can assess their performance against peers?
- How easy it is to inject analytic models into the product or application to support predictive capabilities?

3. Templates and Reusability. Some analytic tools are feature rich but are not easy to install and configure or use to create content for user consumption. Lightweight analytic tools purpose-built for a single mode of BI, such as reporting or dashboards, can be deployed within hours or days, especially cloud-based tools. But enterprise-caliber analytic tools may require more effort. Ideally, enterprise analytic tools come with out-of-the-box templates for specific vertical industries or functional solutions, such as marketing, sales, or operations, to accelerate deployment times. The templates generally consist of a data model and sample reports or dashboards with embedded metrics that hook into the model.

Questions to ask include:

- Does the tool offer solutions packages?
- Are the packages free or licensed separately?

4. APIs. Embedded analytic products live and die by the richness of their APIs. Unfortunately, most older analytic tools were not designed to expose internal functionality via APIs. It takes a major architectural redesign to rewrite application code in a modular fashion (i.e., microservices) and document methods and calls for external consumption. Many sponsors of this report designed their products from the ground up as platforms that can be easily embedded through a rich series of APIs.

Questions to ask:

- What types of APIs does the product support? What functions are available?
- Do the APIs cover all facets of the analytic product, from front end to back end?
- How well documented are the APIs? Are there code samples and videos that demonstrate how to use them?
- Can you import third-party libraries and link them to the analytic platform?
- Can the product call host application APIs to execute transactions or update a database?

5. Architecture. Some analytic tools require a separate .NET or Java application server, while others don't. A server gives analytic tools added heft to scale concurrency and data processing in enterprise deployments and a rich set of administrative and governance features. But other analytic tools have a much lighter footprint-typically, those with a direct query architecture that doesn't require an onboard database and data integration tools. (See next.) Often, these lightweight tools can be run on the same server as the host application, eliminating the need to purchase, install, and maintain a separate server machine. From an embedded perspective, a "server-less" analytic tool consists of a set of dynamic linked libraries (DLLs) or Java Archive (JAR) files that run alongside the code and libraries of the host application.

Questions to ask:

- Does the analytic tool require a separate server to run? If so, what type of server-Windows or Unix?
- If the analytic tool doesn't require a server, can it run on its own machine if required? If not, what type of application servers does it support?
- What programmatic architecture does the analytic tool support-.NET or Java or other?

6. Data Architecture. A big difference among analytic tools is the way they manage data. Some query back-end data sources directly and join and transform data on the fly, never persisting data locally. Others query data only once it's been extracted and loaded into a local in-memory or columnar database. Most do a mix of both to achieve an optimal balance between scalability and performance. (See figure 4

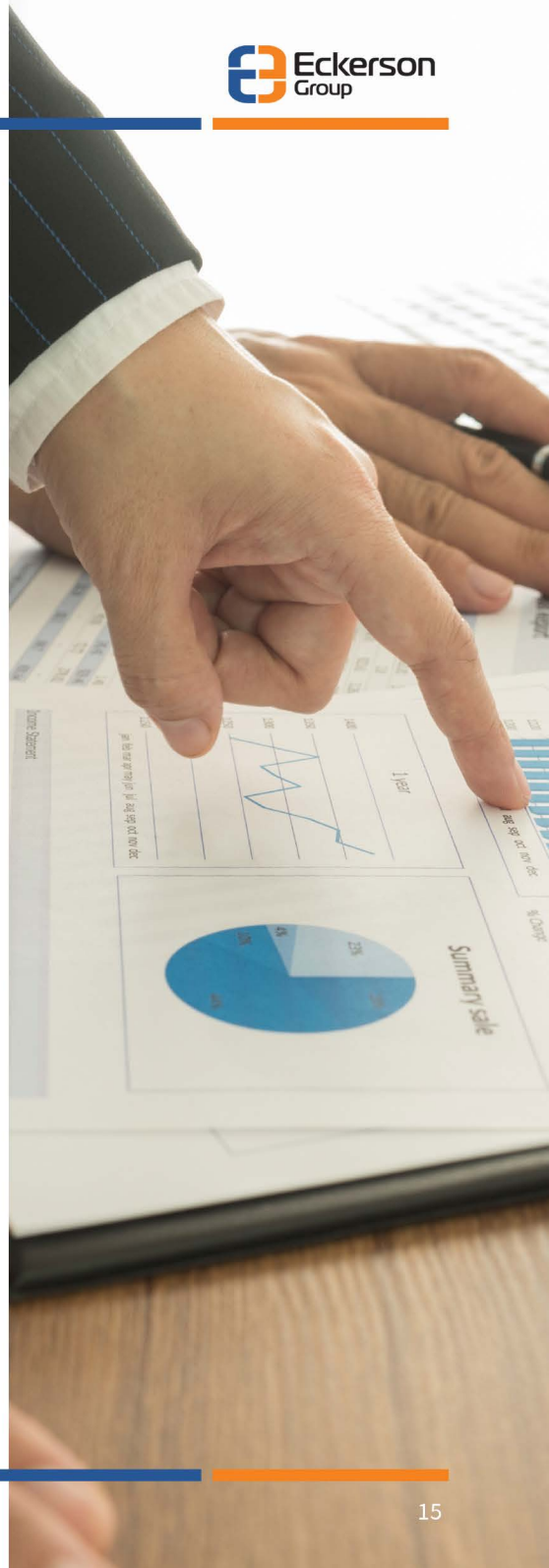
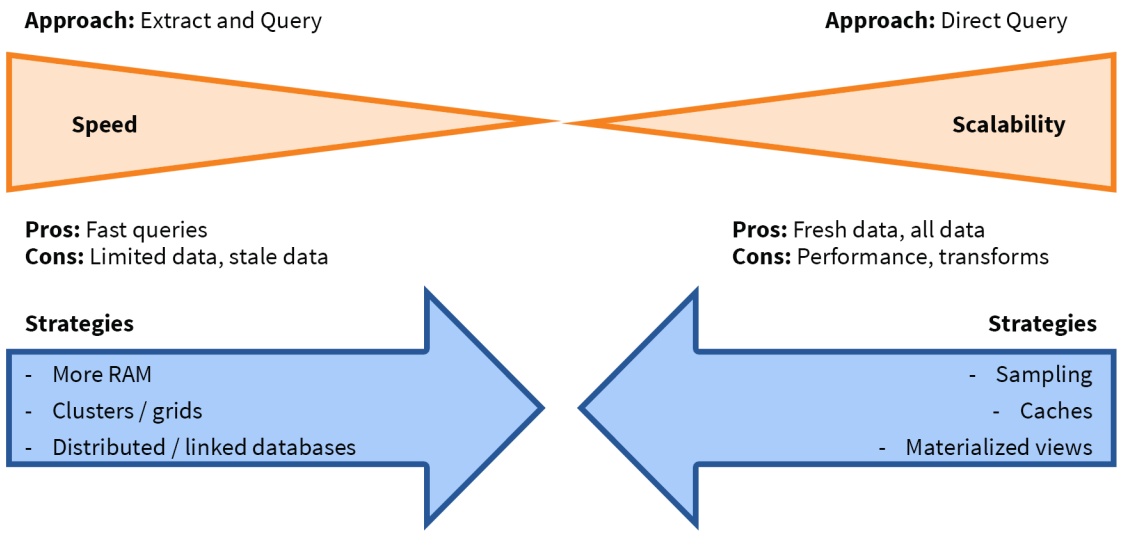


Figure 4. Data Architecture Trade-offs



A tool's architecture ultimately determines how the tool behaves and how much it costs to manage. Direct query architectures run directly against source systems and return fresh or real-time data. But when there are a lot of users or large volumes of data, they can struggle to meet performance requirements, requiring administrators to create real-time caches or materialized views. Conversely, extract-and-query architectures offer fast performance, but require considerable up-front work to ingest, transform, and model data for query processing. They swap scalability and fresh data for high performance.

Questions to ask:

- **Data sources.** What data sources and applications does the tool query? Besides relational databases and local files, the tool should support Hadoop, NoSQL, cloud-based file systems, streaming engines, and search indexes.
- **Data Access.** Does the tool access data via generic APIs (e.g., ODBC or JDBC), or does it use custom connectors designed to exploit the native features of the database and improve performance?
- **Data Transformation.** Does the tool transform data in the remote database, a local database, or in a target report? If the tool transforms data on the fly, what range of transformation functions are available? If it transforms data locally, does it require an IT developer or can business analysts prepare the data?
- **Data Federation.** Does the tool query and combine data from different sources to provide a complete view of data?
- **Data Security.** Can the analytic tool add predicates to queries to control access to database tables and columns?
- **Federation.** Can the tools query both cloud and on-premises data sources? Can they virtualize multiple back-end sources so they look like a single data source to business users?
- **Data Freshness.** Does the platform gives business users access to up-to-date data in real-time? Or does an IT administrator need to load the data into a database and model it?

7. Extensibility. APIs allow developers to call existing functions inside a analytic solution. But what if developers want to add or extend functionality that does not exist? The tool should make it relatively easy for developers to add new functionality to the analytic platform, whether via a wizard-driven interface, lightweight scripting, or by adding third-party plug-ins. This includes front-end extensions such as new charts, visualizations, calculations, localizations, or list selectors. It also includes back-end extensions such as data connectors and administrative utilities (schedulers, backup routines, and systems alerts).

Questions to ask:

- Does the tool offer a software developer's kit for adding or extending analytic functions?
- Does it offer an API that enables developers to hook third-party visualization objects into the platform or modify existing objects to meet business needs?
- Does it offer an API that enables developers to create connections to non-standard data sources?
- What types of third-party plug-ins does the tool support, and how easy is it to register the plug-ins with the platform and manage them?
- Does the product offer an online exchange

8. White Labeling

Most customers want to customize a analytic tool GUI to match their corporate branding. Tools should allow customers to change the basic look and feel of a GUI without coding. For example, they should be able to swap out a vendor's logo with their own, edit the application title, add custom headers and footers, modify icons and artwork, select background colors, fonts, and object styles, and customize the help menu with site-specific content. More detailed customizations will require developers to modify the analytic tool's cascading style sheet (CSS) with JavaScript.

Questions to ask:

- Which aspects of the GUI can be changed?
- Which changes can be configured using a point-and-click interface?
- Which features require modifying the CSS using JavaScript or another language?
- Can you upload your own CSS file for more granular control of the GUI?

9. Security Integration. Authentication and access control are critical issues for embedded applications. First, the analytic tool should be capable of working within an application's preferred security model, accepting single sign-on tokens via Kerberos, Lightweight Directory Access Protocol (LDAP), Microsoft Active Directory, Security Assertion Markup Language (SAML), or some other mechanism. Second, roles and rights established in the host application should be passed to the analytics tool to ensure end users are granted appropriate access. This control should span access to analytic features (such as charts, reports, dashboards, input controls, and user functions) as well as data (including data sources, tables, columns, and rows).

Questions to ask:

- What security models does the analytic tool support?
- Does the analytic tool have to maintain its own list of user profiles, or can it work from the list stored in the application or identity management system?
- How does the tool prevent users from seeing specific analytical functions or features, such as parameters in a list selector?
- How does the tool prevent users from accessing specific metadata, reports, and visualizations from a search bar?
- How does the tool ensure data security to the row and column level?
- Can a host application define and manage access rights within the analytic tool in an automated fashion?

10. Cloud Support

Analytics tools are increasingly moving to the cloud. Many run client software entirely from within Web browsers, including authoring and administrative modules. And some actually run as cloud services, either as a software-as-a-service or platform-as-a-service. Because cloud application providers are the biggest buyers of embedded analytic software, it's critical that analytics tools work well within a multi-tenant, cloud application. This means that a analytic tool's administrative functions should flow through to the host application so administrators don't have to use multiple tools to manage users and govern access.

Questions to ask:

- Can an administrator provision new analytic users from within a multi-tenant cloud application?
- Does the analytics tool integrate with the application's security framework? (See Security Integration, above.)
- Can the analytics tool securely access customer data, whether it's stored in a single database or in separate databases for each customer?
- Can the analytics tool append predicates to queries based on security privileges?

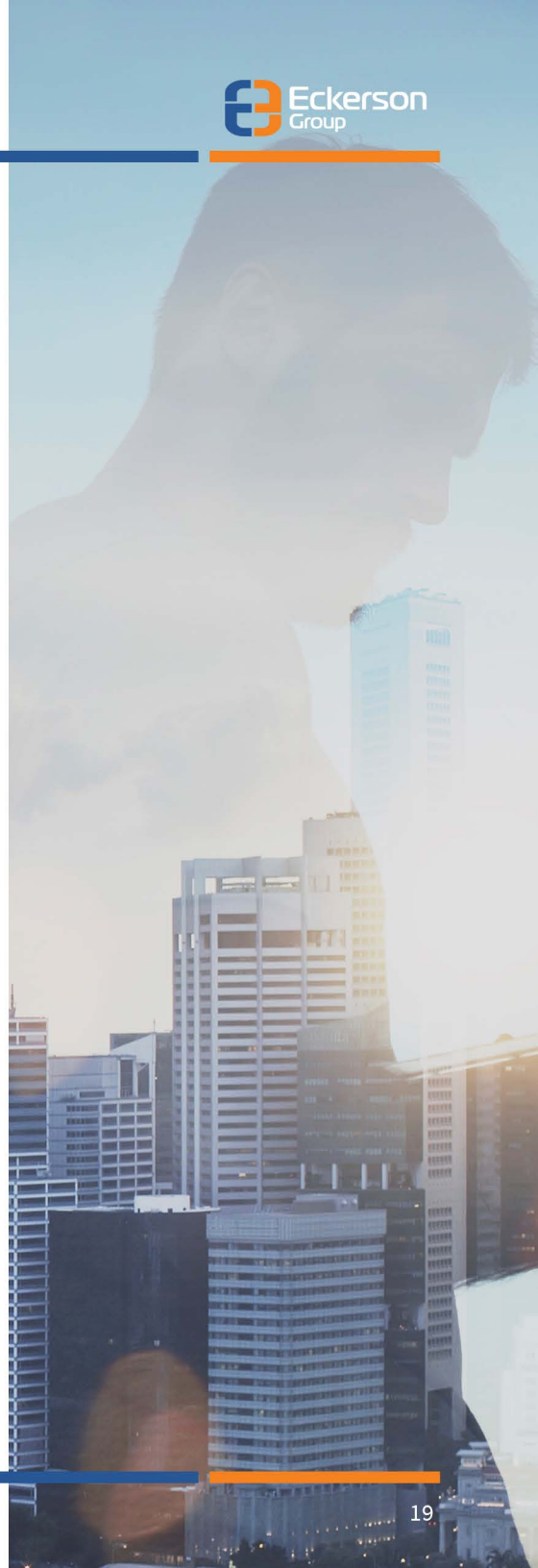
11. Pricing Model

Standard perpetual licenses based on numbers of users don't work well with an embedded model. That's largely because providers have difficulty estimating the number of users in advance and often can't track usage. Also, commercial software providers don't like to pay a large up-front license fee for software that might take them years to monetize.

Consequently, most analytic platform vendors offer flexible pricing based on the long-term value of the embedded solution. For some, this means pricing by server or core, while others prefer "value-based" pricing that uses revenues or the number of employees or some other value as a proxy for usage. Another option is a royalty structure where customers pay a small up-front fee and a percentage of every sale of the enriched software.

Questions to ask:

- What is the pricing model? Is there any flexibility to change the model?
- What are the upgrade fees for adding new cores?
- Is there an annual maintenance fee? If so, what does it include?
- Is support, training, and consulting bundled into the license or sold separately?
- What is the average sales price?
- What level of discounting does the vendor offer? (Ask current customers.)



12. Vendor Services. The biggest factor in the success of an embedded analytics solution is the commitment of the analytic vendor to the project. An embedded analytics project requires significant up-front planning, especially for commercial software vendors who need to price, market, sell, and support the new analytics functionality. A project team needs to define goals, gather requirements, evaluate architectures, design customizations, train developers and users, and commercialize the offering. This can take several weeks to several months, depending on the complexity of the project and the skill and experience of the vendor consultants.

Questions to ask:

- What consulting services does the vendor offer? At what price?
- What experience does the vendor have helping customers scope and plan an embedded analytic project?
- What training services does the vendor offer at what price?
- Is the vendor willing to handle customer support calls?

Does the vendor have an active community where developers can find answers and interact with experts?

Time to Embed Analytics?

In this era of big data, organizations of all sizes and shapes are eager to boost the value of their applications by injecting them with data and analytics. Software vendors are leading the charge in delivering data-driven applications, but they are followed closely by organizations in many industries that want to monetize their data assets and get closer to customers and suppliers.

Embedding analytics into applications has never been easier. Most analytic vendors recognize the opportunity and have revamped their products to make them easier to embed. Analytic vendors now compete on the breadth, depth, and documentation of the public APIs and SDKs that enable developers to customize tools and embed them into other applications.

Before selecting an analytics tool to embed in your applications, there are many factors to consider, ranging from technical features (such as software and data architectures) to business capabilities (such as pricing, support, and services). Selecting the right software is not easy. It takes considerable time to find a product that will meet your requirements, but the payoff is considerable, especially if you plan to charge clients for new analytics features.

Need help with your business analytics or data management and governance strategy?

Want to learn about the latest business analytics and big data tools and trends?

Check out **Eckerson Group** research and consulting services