

EVALUATION CRITERIA FOR AN ABM SOLUTION

The Five Key Areas of Focus

CONTENTS

- 2 Technical Architecture
- 3 Usability
- 5 Activity-Based Costing
- 7 Reporting and Analysis
- 8 Activity-Based Budgeting

INTRODUCTION

There are five primary areas of focus that should comprise any comprehensive evaluation of an activity-based management (ABM) solution. These are central to the structure and functionality of the ABM software.

- Technical architecture
- Usability
- Activity-based costing
- Reporting and analysis
- Activity-based budgeting

Even if you are looking for a single-user solution to begin with, be sure to select a provider who has both the vision and the product functionality to support you in growing and expanding value-based management across your enterprise in the future.

The following information may be used as a guide in evaluating an ABM solution, or it may be used freely, in part or in its entirety, in the preparation of a request for proposal (RFP) or similar document.

Author: Richard Barrett

Contributors: Chris Grundy, MaryLouise Meckler, Jing Zhao

TECHNICAL ARCHITECTURE

Multi-User Client/Server Architecture

The application should allow and control the secured update of information by multiple concurrent users, not just allow multiple users to simultaneously view data. This requires complete functional and data (record) level security for individual users and groups.

Full Web-Based Functionality

If use of the application is to go beyond that of a small group of localized model builders, then the full functionality of the application should be deployed over the web, providing the ability to access and update model information through a browser front-end. This requires complete functional and data (record) level security for individual users and groups.

Scalability

The application should be scalable as the use of the product grows within an organization, supporting every stage, from small pilot projects to a full enterprise-wide implementation. This means the product must be able to handle extremely large, cross-functional mega-models as effectively as small departmental models. If your requirement is for an enterprise-wide solution that spans a number of business units, this may be easier to implement and manage with an application that offers the ability to link individual models with rules.

Efficient Data Architecture

The application should allow separate objects for accounts and cost-centers, allowing the cost-center to be defined once, and then associated with all valid accounts. It is NOT acceptable for the application to use a single data field to identify both the account and cost-center (i.e., duplication of data fields: "salaries-costcenter 1," "rent-costcenter 1," "supplies-costcenter1," etc.), as this approach constitutes a design limitation that creates an exponential increase in the number of entities required in the model, and causes a corresponding increase in the amount of maintenance work that is needed.

Enterprise Integration

The architecture should support integration with other applications as required.

USABILITY

Customizable Desktop

The user should be able to customize the user interface to create a workspace that provides shortcuts to the screens and features used most frequently and to allow them to work they way they wish. On specific screens, the user should have the ability to easily manage the presentation of information by controlling column size, sorting fields, etc.

Customizable Terminology

Terminology used in the application should be customizable to support either Consortium for Advanced Manufacturing - International (CAM-I) or organization-specific terminology.

Automated Data Import

The application should offer flexible data import capabilities, with support for a variety of file formats and no limit on the size of input source files. In addition, the data input parameters should only require definition once, after which the application should automate the process whenever the user chooses to update the model with new data from the same source.

Data Export

Flexible data export capabilities should be available, with support for ODBC or OLE DB for OLAP for dynamically linking to business intelligence and reporting tools.

User-Defined Periods

The application should provide the ability to define the number of periods/years of history/budget versions to be used/maintained by the application. The application should also provide the ability to maintain, view, and compare these different periods within a single model, on a single screen.

Large Name Fields

The application should support large field sizes to support descriptive naming of activities, products/services, customers, etc.

One-Time Maintenance

Model maintenance should be very easy and should not require the user to define a single object multiple times in the application. For example, if there are changes to department or cost-center names, the application should allow the change to be made in one place, and have that change be automatically reflected throughout the model in all references to the renamed department or cost center.

Inheritance of Business Rules

The application should streamline maintenance by automatically inheriting rules when new objects are added to the model. For example, when a new product or customer cost object is added, it should inherit the activity assignments and costs drivers defined for the group it is in. This can be overridden if required, but in most cases, the inherited values will apply.

Alternate Hierarchies

The application should support multiple hierarchies so that data can be categorized in different ways to support various reporting and analysis requirements, without requiring duplication of the data.

ACTIVITY-BASED COSTING

Cost Allocations

The application should have the ability to allocate selected account/department costs internally within the same department or to other departments before assigning the account/department costs to the activities.

Activity Cost Reallocations

The application should also support the reallocation of activity cost data internally within the same department or to other departments.

Defaults

Having to manually make each assignment is both time-consuming and potentially confusing. Applications should offer default assignments that can be rapidly applied across an entire model together with powerful override features so that different paths can be defined in different responsibility centers, versions, and periods.

Reiterative (Simultaneous) Allocations

Reiterative (simultaneous) allocations should be handled automatically, continuing to reallocate until all costs have been reallocated to a specified level of accuracy. There should be no limitation on the number of departments involved in this process.

Direct Cost Assignment

The application should be able to optionally assign the account/department costs directly to the cost objects, bypassing the activity assignment if needed.

Derived Drivers

The application should support deriving cost/activity drivers from other existing drivers.

Time-Based Costing

In those responsibility centers with highly repetitive activities and a high level of controllable costs, it may be appropriate to allocate certain activities using a time-driven costing methodology rather than a driver-based methodology and the application should offer both methodologies as well as rules to develop hybrid allocations.

Relationship Analysis

The application should be designed to easily support multidimensional cost-object analysis without maintenance-intensive workarounds. This requires the ability to accurately assign costs to more than one cost object-view at a time. The application should not only show activity costs consumed by Product A and activity costs consumed by Customer X, but should be able to present the combination in a single, multidimensional view, from a choice of perspectives (i.e., a single view that reflects costs consumed by Product A sold to Customer X through Channel Y).

The application should also support profitability analysis with the same multidimensional presentation capability described above (for example: profitability by customer for a specific product through a specific channel), with the ability to view the information from any perspective.

Complete Traceback

A complete audit trail should be available, providing the ability to trace costs from the cost objects (products, customers, etc.) back to the activities, and then back to the originating general ledger account/department cost data, all in a single model. For example: for Product A costs, what portion is made up of salaries from Cost Center 1?

REPORTING AND ANALYSIS

Customizable Reports

The application should provide the ability to create customized reports to support the reporting requirements of different organizations, and varying reporting requirements within the same organization.

Automatic OLAP Generation

Charts and graphs should be easily and automatically generated from the application using OLAP technology. The OLAP cube should be automatically created by the application.

Detailed Dimensional Data

The cost objects such as customer, product and channel should be created as separate dimensions within the model in order to support multidimensional analysis.

Dynamic Spreadsheet Access

ABM data should be dynamically accessible using spreadsheets (Excel, Lotus 123).

ACTIVITY-BASED BUDGETING

What-If Analysis

The application should support user-defined scenarios for What-if analysis—supporting as many budget versions as required.

Integrated Resource Planning

The application should provide the ability to develop activity-based budgets using a "back calculation" approach to arrive at resource requirements, based on cost drivers and production volumes. The ability to add "back rules" that allow users to specify exactly how new levels of activities will be resourced adds considerably to the realism of "back calculation" and is an important requirement when a model is being used for detailed operational planning.

Support for Variable and Fixed Percentages

Calculated budget data should be based on the variable percentages defined for specified accounts.

Periodic Cost Spreading

The application should provide the ability to spread amounts across periods to support the development of yearly or quarterly budgets. The application should also allow summary amounts to be spread across detail periods, such as spreading quarterly costs to monthly periods.

Projections

The application should provide the ability to project amounts into the future, in order to support current year budgeting as well as long-range planning.

Budget Export

The application should include the ability to export activity-based budget data in an industry standard format, to be used as an input to other budgeting applications.

businessobjects.com

insight.businessobjects.com