1. Product Overview

Fair Isaac’s main BRM product is Blaze Advisor. However, the company also offers various products that can be used to supplement the development and functionality of rules-based applications built with Blaze Advisor. These include the Model Builder tool for creating predictive models that can be integrated with Blaze Advisor’s rule-based applications and the Decision Optimizer platform for optimizing Blaze Advisor applications. Additionally, Fair Isaac offers several pre-built BRM applications targeted at specific domains and industries—including credit card fraud prevention, risk management and marketing. This report focuses primarily on Blaze Advisor’s features for building and managing BRM applications.

<table>
<thead>
<tr>
<th>Table 1. Overview of Blaze Advisor, Key Modules and Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blaze Advisor Builder</td>
</tr>
<tr>
<td>Blaze Advisor Rule Server (ARS)</td>
</tr>
<tr>
<td>Blaze Advisor Repository</td>
</tr>
</tbody>
</table>
Table 1. Overview of Blaze Advisor, Key Modules and Functionality

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Maintenance Applications</td>
<td>Wizard-based facility for automated generation of complete rule management applications that operate over any Web browser based on predefined templates. RMAs allow non-technical business users to review, modify, delete, or create rules within the structured constraints without knowing a single word of programming or specialized syntax and without using a specialized third-party editing environment. Rule management screen text and cosmetics can be modified using any desired HTML authoring tools to take on the look and feel of familiar applications within the enterprise.</td>
</tr>
<tr>
<td>Quick Deployers</td>
<td>Wizard-based facilities that simplify deployment of Blaze Advisors applications via automated code generation to create configuration files for immediate deployment in the products of major vendors. Can deploy rule services for major J2EE application servers (e.g., WebLogic, WebSphere, Oracle iAS, Sun ONE, jBoss, etc.) and a .Net assembly. Also generates deployments for processing in Java, Microsoft Transaction Server, JSR-94 rule engine interface compliance, and Web Services.</td>
</tr>
<tr>
<td>Pre-built Templates and Patterns</td>
<td>In addition to rule templates for assisting with creating rules, rule sets and rule maintenance applications (for both IT and end users), Fair Isaac also offers pre-built templates for Insurance, Mortgage, and Healthcare that serve as “starting points” for customers developing applications in these areas.</td>
</tr>
<tr>
<td>Model Builder</td>
<td>Optional tool for building scoring, predictive and other types of analytic models; designed to ease development and deployment of analytic models by providing the ability to export models into Blaze Advisor to form the rules set for decision-processing. Allows organizations to more easily build and deploy advanced BRM applications that combine both predictive and rules models within the same environment by eliminating the need for developers to re-code analytic models. Also reduces difficulty associated with deploying analytic models because model functionality is literally embedded within Blaze Advisor rules.</td>
</tr>
<tr>
<td>Decision Optimizer</td>
<td>Optional platform for business users and IT to manage and fine tune decision models for use in Blaze Advisor and Model Builder environments. Provides facilities that allow business users to understand complex logic behind decision-processing models, and a simulation capability for determining the effects that changing model constraints and criteria (e.g., customer scoring values, etc.) will have on decision processing while also taking into account a company’s own constraints. Also provides ability to test and compare effects of different models on decision processing; and offers a feedback mechanism for gathering the results of tuning decisions, which administrators can analyze to improve future model design, deployment, and optimization efforts.</td>
</tr>
</tbody>
</table>

Table 1 provides an overview of Blaze Advisor and its main modules. It also summarizes additional tools that can be used to supplement rules-processing functionality. Blaze Advisor is a rule-based development and management environment for building Java or .NET applications. In addition to an inferencing-based rules engine, Blaze Advisor provides an integrated development environment (IDE)—complete with multi-user repository that supports team development—and testing and debugging tools. Integration capabilities include support for XML, Java, COM, CORBA and Web Services. A COBOL generator version is available for organizations wanting to execute rule-based applications in mainframe environments.
Blaze Advisor supports both IT developers and non-technical business end users through the extensive use of wizard-based development facilities and reusable templates. These deployment facilities allow for creating front ends and templates which both constrain how a business user can write a rule and allows for generating a point-and-click Web interface that can be edited to match end-user needs and expectations.

Blaze Advisor can also execute analytics—in the form of scorecards (i.e., scoring models), decision trees, segmentation trees and other advanced algorithms—directly within its rules processing environment.

Blaze Advisor and its add-on facilities provide a complete set of tools for developing, deploying, and maintaining business rule services as software components, making it well suited for building and managing large, complex, and frequently changing business rules applications.

2. Product Architecture

2.1. Architecture Overview

Figure 1 provides an overview of the Blaze Advisor architecture, which consists of three key modules that support the authoring, deployment, maintenance and administration of business rules.

Figure 1. Blaze Advisor Architecture

Rule Authoring Environment/Graphical IDE: Blaze Advisor Builder
Blaze Advisor Builder is a graphical IDE for developers to create an architecture for business rules applications as well as for authoring business rules. It is a stand-alone Java application. Deployment of rule projects is done through the Blaze Advisor Rule Server. Blaze Advisor Builder also features a Rule Maintenance Application (RMA) module—a deployment facility developers use to create structured front-ends for simplifying management and updating of rules by business users.
Blaze Advisor Rule Server (ARS) is a server-based execution engine for deploying rule-based applications built with Blaze Advisor Builder. ARS is a 100% Java object conforming to both J2EE (1.2) and Web Services standards. It is typically deployed as a J2EE Enterprise Java Bean (EJB), Model Driven Bean (MDB) or Web Service. Other configuration options include deploying ARS as a COM/MTS object, a .NET component with a C/C++ interface, or within EAI layers such as IBM MQ Series, JMS and RMI. ARS is installable on any application server or Web server, or it can run as a standalone server.

Rules Repository/Rules Database: Blaze Advisor Repository
Blaze Advisor Repository is an XML-based repository that is separate from the execution engine (i.e., ARS). It supports rule service deployment for multiple applications—ensuring consistent enforcement of business policies and procedures across an organization—and repository-based team development.

Blaze Repository is a set of XML entities (files, documents, etc.) which can be stored in any mechanism (JDBC/RDBMS, LDAP, SCCM). Configuration options include saving XML rule files in a LDAP system that supports JNDI, a database that supports JDBC, or saved as flat files. The repository can use CVS or Blaze Advisor’s built in versioning system to maintain history information and to provide check-in/check-out functionality.

All rule structures, rule services, allowed value controls, and audit trail data (e.g., “last modified date”, “last modified by”, and “reason changed”, etc.) are stored in the repository. Likewise, rules created or changed in rule maintenance applications are also stored in the repository. The rule server monitors changes in the repository and updates affected rule services without interrupting production operations. The deployment manager provided with the product handles the assembly process for runtime execution.

Blaze Advisor Repository also provides various management properties, which allow organizations to add their own meta data, in effect, extending the repository to hold any information pertinent to a rules project. (For more on management properties see Section 4.6.)

2.2. Business Rules Engine
Blaze Advisor uses a forward and backward chaining inference engine (to process rules and other decision logic) that is based on an optimized version of the original RETE algorithm. It can efficiently manage and execute rule bases with many thousands of conditions. This algorithm uses rapid pattern matching to relate the conditions that are true at any given moment to the appropriate rule or rules. The rule engine can quickly look through large sets of rules to find the proper ones to apply based on case-specific conditions. No code is required to specify which logic subroutines should be fired in which order.

We must also point out that Blaze Advisor version 6.1 provides the option of using the RETE III algorithm, which is orders of magnitude faster than the original RETE algorithm.

2.2.1. Rules Management Approach
Blaze Advisor uses a rules management approach that is based on a combined repository and OO programming language (i.e., Structured Rule Language) that is designed to make writing and reading business rules as English-like as possible.

The Blaze Advisor rules engine allows several options for specifying procedural execution, including ruleflow sequencing for ordering groups of rules, and within each group (or rule set) the designer can choose engine-determined processing or sequential processing of rules in a predefined order. Other methods for controlling order of execution include setting relative priorities, event-based conditions, and explicit calls to rules from other rules. This allows the engine to primarily work in a forward
chaining fashion, with the capability to do backward chaining as well. An option is also available that allows special performance improvements for situations where rules can be fired in sequence. In addition, a compiled-sequential option (which requires an additional license) for native Java speed is also available.

2.2.2. Enterprise Data Interoperability

Blaze Advisor features the ability to build and execute rules that reference data stored in databases, XML documents, business events (including IBM MQSeries), Java/J2EE objects, CORBA objects, COM objects, and customer-defined objects which might be found in third-party data models. This is done using wizards which examine the definition of external data models and automatically construct Blaze Advisor internal references to their elements so that no dictionary definition step is required before starting to write rules. Wizards also construct runtime linkages so that rules which reference the current value of a data element or change its value automatically invoke the proper underlying interface code to retrieve or update the value in the external data source. Rules can make simultaneous use of data elements from a variety of data sources, including multiple databases.

2.2.3. Maintaining Data Quality

As part of the explicit resolution of missing data, Blaze Advisor is able to request data from external data sources and end-users. Blaze Advisor employs several methods to interact with end-users including question sets, external program calls and messaging (XML based and otherwise). This enables the BRE to request data and information from the user as is required, hence driving the question process flow. This also allows the amount of data requested to be limited to that which is required and reduces the number of questions asked.

When a missing data value cannot be resolved, an appropriate exception is generated based on the missing information and the decision process halted. This process will ensure the application handles the missing data, communicates the problem effectively to the client application and exit with causing further execution problems.

Designers can also write rules specifying how the application should act when encountering dirty or missing data. In this case, the action taken by the BRE in response is entirely up to the designer (i.e., they are not required to throw exceptions and halt the process unless they chose to do so).

3. Interfaces and Ease of Use

Blaze Advisor features two environments for rule authoring and maintenance. One is Blaze Advisor Builder, a full function IDE primarily intended for IT developers. The other is the rule maintenance application(s) (generated by Blaze Advisor Builder) designed for business users.

3.1. Development Environment: Blaze Advisor Builder

Blaze Advisor Builder’s graphical IDE features specialized editing windows and interfaces for different types of rules, functions, question sets, ruleflows, class definitions, and other entities. It allows side-by-side viewing of entities with movable windows or tabs. It also contains separate test interface windows, output windows, and editing windows. It provides a sophisticated navigation tree for immediate access to any object in a rule service, and it supports hyperlink access to any rule referenced in a compilation error or warning message.

♦ Using Blaze Advisor Builder, a developer can:
  ♦ Import an object model
  ♦ Create rules, rulesets, functions, variables, and other entities associated with the rules
  ♦ Test, debug, explore, and report on rule projects
- Generate rule deployments
- Create the structure for non-technical users to create and maintain rules outside of builder
- English-like Rule Syntax

Rules can be written using English words and grammar such as “If customer’s average balance is more than ...”, or by using the mathematical symbols and object model “dot notation” familiar to programmers. An extensive selection of rule language keywords is provided, and rules can take advantage of regular expressions and powerful pattern constructs to dramatically reduce the number of rules required.

The Blaze Advisor IDE includes a full set of editors that simplify the creation of entities, and generate much of the syntax. Figure 2 shows the three different syntaxes available for referring to the data values, dot notation (applicant.isEmployed), English-like (the creditEstimate of applicant), and possessive syntax (applicant’s loanType). As shown, the different syntax types can be mixed within a rule. All syntaxes are available as soon as the data is mapped; there is no dictionary definition step required.

![Figure 2. The Blaze Advisor Builder IDE](image)

Blaze Advisor defines rules without specifying procedural code for ordering or controlling sequences. It executes appropriate rules based on relevance at time of execution controlled by ruleflow sequencing, rule events, time/date effectivity, and priority.
**Visual Ruleflow Editor**
This editor makes it fairly simple to break sophisticated conditional business processes into steps. Drag-and drop icons allow graphical construction of business process definitions as collections of events, steps, branches, actions, conditions, and loops. Tasks can be represented by decision tables, decision trees, drill-down ruleflows, functions or rulesets.

**Creation of Advanced Rulesets**
Rulesets enable logical groupings of rules to be shared and reused, and activated and deactivated as a single entity. Rulesets make it easy to associate blocks of rules with a step in a ruleflow and can be activated based on context (e.g., only to “qualified” customers) as well as being passed parameters and assigning return values.

**Execution Browser and Debugging Tools**
Blaze Advisor provides a graphical representation of rule execution within a ruleflow, and can highlight specific rules, functions etc. as they fire. Breakpoints and watch points can be set and rule execution can be traced showing a visual representation of rule-firing sequences during testing. Rulebases can be analyzed for potentially conflicting or otherwise invalid rules.

**Cross-Reference Browser**
A “cross-reference” browser tool provides a complete graphical representation of interrelationships between rules, functions, and object properties that can be viewed allowing rapid review of all uses of a given function, ruleset, rule, etc. Any object (rule, ruleset, class, enumeration, etc.) may be queried interactively to check various dependencies, such as what it incorporates, what it reads, what it modifies, what other objects refer to it, etc. A dependency report for the entire project may be generated at any time in a hyperlinked Web document or in a static table of contents format.

**Quick Deployers**
Quick Deployers are wizard-based facilities designed to simplify deployment of Blaze Advisor applications. They automate code generation to create configuration files for immediate deployment of Blaze Advisor rules applications in the products of major vendors. Through a question and answer session, wizards collect deployment information and generate appropriate installation files for any supported application server, ant build files, Web Services, and configuration files. (For more on Quick Deployers see Section 6.)

### 3.2. End-User Environment: Rule Maintenance Applications (RMA)
The rule maintenance applications (RMAs) generated by Blaze Advisor allow business users to safely and easily create and maintain rules in a graphical environment without technical assistance. Users can review, modify, delete, or create rules within the structured constraints without knowing any programming or specialized syntax and without using a specialized editing environment.

System controlled pick lists, radio buttons, and other controls make RMAs easy to use and safe, as the values that can be selected come from enterprise data sources. This give users the ability to review and manage rules using everyday terminology without being forced to learn rule syntax. In addition, RMAs can feature decision tables, scorecards and decision trees to make it easy for business users to maintain rules using graphical techniques. (For more on the use of these techniques see Section 4.2.)

RMAs are generated by using the Rule Maintenance Application (RMA) wizard. This wizard allows you to:
- Define the RMA’s style and functional characteristics
- Define the application scope and decide whether to include banners, a table of contents, list filters, a login screen, etc.
- Define labels, page text, data displays, fonts, and colors
Select the style of data entry and display controls

Once the application is generated, you can use any Web HTML design tool to make final stylistic customizations (i.e., incorporate the look and feel of other applications within the organization.) Figure 3 provides a screen shot of the Blaze Advisor Rule Maintenance tool for business users.

![Blaze Advisor Rule Maintenance Center](image)

**Figure 3. Blaze Advisor Rule Maintenance Center (for business end users)**

A Rule Maintenance Authorization and Tracking facility allows you to control which users have access to which rule management screens through standard Web security controls like LDAP lookups and ID/Password logons. It also automatically records rule modification documentation in the rule repository associated with each change that is made. User name, date, time, version number, HTML links to external reference sources, and arbitrary comment text can all be associated with any rule modification.

4. Business Rules Expression

4.1. Defining Rules and Rule Sets

The Blaze Advisor Structured Rule Language (SRL) is an OO language designed to make writing and reading business rules as English-like as possible. It shares many features of common programming languages, and is intended for use by programmers to create the entities, control the execution flow, and perform the operations required by the (rule) decision making service. However, it also provides syntax for authoring individual rules and rule sets that are easily understandable by people with little or no programming background. You can use common English words such as “is less than” and
“exceeds” as well as common English representations of objects and properties such as “customer’s age” and “the age of the customer.” (For more on Blaze Advisor Builder see Section 3.1.)

Blaze Advisor rule sets separate rules into logical groups for ease of management and control, for reuse between rule services or business applications, for order dependent operations, and for functional enhancements to existing applications. Blaze Advisor Ruleflows are built in a simple graphical editor. The flow is intuitive to business people and does not rely on technical concepts such as UML. Data can be passed from task to task using ruleflow variables.

4.2. Creating and Maintaining Rules
Blaze Advisor supports three different graphical methods for creating and maintaining rules:

♦ Decision tables—allow vast numbers of conditions and corresponding actions to be shown in a grid. Designers can create tables in columnar, row, or combined formats with multi-level condition groups. A table cell can link to another table for nesting or can contain any number of actions, including anything that can be expressed in rule syntax. A Decision Table editor supports Cut, Copy and Paste functionality for cells, rows and columns. Extra rules can be added before or after the decision table is evaluated.

♦ Decision Trees—show navigation paths displaying condition assumptions leading to the branch in focus. Actions can be changed via a click-and-type interface. New branches can be added to the tree dynamically. A Decision Tree editor supports easy modification of split points, focusing on a portion of the tree and allows non-exclusive branches leading to multiple actions when appropriate. Extra rules can be added before or after the decision table is evaluated. Blaze Advisor’s Decision Trees work well for situations where the conditions are not symmetrical, which would not work well as a table.

♦ Scorecards—allow weight factors to be assigned and maintained by business experts. Users or system architects can create and maintain reason codes used to track and report on most significant factors in a calculated score. Significance can be determined from pre-set priority levels or from a variety of calculated importance algorithms. For companies in sectors such as finance, insurance, government, or other highly regulated industries, Blaze Advisor offers ways to set, capture, and record significant factors used in calculating a score. By defining customized Reason Codes, organizations can make use of scorecards as an auditable component of their decision processes. And contributing factors to a score can be tracked and reported based on relative corporate priorities or on mathematical rankings considering each factor’s impact on the result.

4.3. Rule Templates
Blaze Advisor supports the creation of reusable rules management templates that can incorporate entire rule services with multiple steps, functions, and rulesets. Templates help enforce the consistent deployment of rule changes as well as provide the foundation for rule maintenance applications—allowing rules to be modified or created within a strict set of constraints appropriate to satisfy different users or tasks. Templates are exposed through a Web interface so that end users can update the values, choosing from a list or range of values that is set by the developers. Business users do not need to know whether they are editing rules, or functions, or any other part of the project, as the rule maintenance environment does not show any of the syntax. The whole spectrum—from simple rules parameterization to full condition and action selection—is supported. Templates and associated rule maintenance applications are able to reference any property, object, rule, database table etc. in Blaze Advisor.

(Pre-built templates offered for specific industries are covered in Section 10.)
4.4. **Rule Syntax Checking**

Complete rule syntax checking is automated through a pre-compilation step with interactive links to problematic code.

4.5. **Rule Testing and Tracing**

Extensive rule testing and tracing capabilities allow you to trace all rule execution operations in real-time, with interactive test values, user-defined breakpoints, traces, event monitors, single-step operation, and graphical depiction of rule execution.

An analysis tool allows checks for potential conflicts and unwanted interdependencies between rules. And the developer has control over what tests will be run and which rulesets will be checked. Results may be seen on screen or included in project reports (described below).

Rule definitions may be viewed from dedicated rule editing windows, or may be viewed in system reports (both hyperlinked and static). Rules may be accessed from online navigation tree that allows a jump to a named rule, or from within rulesets accessed by clicking on task boxes in the visual ruleflow.

An Execution Browser lets you follow the execution of a rule agent graphically. By displaying a tree of possible execution paths and, as the rule agent is executing the project, it highlights the current execution path. It also provides the ability to interactively build dependency trees showing relationships between rules, parameters, and data objects.

A reporting facility lets you run reports describing the complete structure of a rule service, with detailed documentation of individual elements including ruleflows, decision tables, functional calls, and rules. You can review reports online (with interactive jumps between objects), or formatted with tables of contents for hardcopy or archival documentation. Rule conflict detection is included to point out potential problems with inconsistent, incomplete, or self-referential rules. Finally, all debugging and testing operations can be performed on the production system in a separate processing thread while end-user operations continued unaffected.

4.6. **Multi-User/Team Development**

Multi-user/team development is supported through shared repositories and check-in/check-out capabilities. Users can work offline and synchronize changes with the shared repository. The repository prevents multiple users from making changes at the same time or overwriting revisions by providing a way for users to check out and check in rule service components. Information about an older version can be retrieved and viewed simultaneously with the current version. And previous versions may be promoted into production to undo errors and problems.

Blaze Advisor’s built-in versioning system manages rule changes, including logging historical content changes of rules, rule services, and rule sets. Author, date and other comments are also captured. You can view the information from within the IDE, from a Web-based RMA, or as exported to a data file. The rule server monitors changes in the repository and updates affected rule services without interrupting production operations.

Blaze Advisor Repository also provides important management properties. For example, companies can add their own meta data to a project, thus allowing them to store additional information pertinent to the rules project (i.e., beyond basic information like author, date, and comment, etc.)—such as the stage of development, the office in charge of maintaining the specific code, and so on. These management properties can be queried on and used to determine which files should be moved to testing and production servers.

Companies can also use third-party tools should they have more familiarity with a different system. Blaze Advisor supports file-based, LDAP, and database repositories that will handle file access and versioning, including CVS, PVCS, SourceSafe, and ClearCase.
4.7. Rules Management and Maintenance
A number of wizard-driven development tools for building rules, rulesets, global rule templates, and importing definition classes are provided. Wizards for developing decision trees, decision tables, and scorecards are also available.

Templates can be used to automatically generate Web-based rule maintenance applications (RMAs) allowing the use of a company's desired terminology, value choices, and interaction styles for non-technical users. (See Section 3.2.)

4.8. Rule Logic Validation and Testing
Blaze Advisor provides a comprehensive set of testing and debugging tools:

♦ Rule Execution Browser—allows you to follow the execution of a rule agent graphically by displaying a tree of possible execution paths and, as the rule agent is executing the project, highlights the current execution path.

♦ Project Reports—available to help developers identify potential conflicts with a project. The report is generated as linked HTML.

♦ Performance Testing—to identify potential problems in the execution of the rule-base with respect to performance. The tool provides both hot-spot report to aid identification and execution path to identify the specific process or processes.

♦ Regression testing—can be performed by storing test inputs in a persistent data source and having the system read them in for debugging and QA.

♦ Integration with RuleTrack (a tool for capturing and managing business rules that is available from Business Rule Solutions).

♦ All tools available through the development environment can be used to test and debug a live Rule Server whether it is in a test environment or production.

4.9. Rule Logic Verification
Conflict reports analyze potentially conflicting or otherwise invalid rules. You can analyze rules interactively in the IDE or by generating project reports. Testers can select the conflict tests for inclusion in a report and the rule sets for testing. This narrows the amount of output that needs to be perused.

4.10. Conflicts and Priorities
Blaze Advisor uses the RETE rules inferencing algorithm, which is capable of both forward and backwards chaining. Each rule has a priority setting where you can make one rule a higher priority over another. Blaze Advisor supports the concept of rule-priority and the models associated for evaluation of layers of rules from general to more specific (or general to layers of exceptions). A combination of rule-repository architecture, rule-flow and rule-templates hides the complexity of these multi-level rules from the business analyst who can directly access the “general” or “exception” rules using Web-pages to make necessary rule changes.

There is no limitation on the number or complexity of special rules or exceptions that can be applied to any product or scenario. All rules whose conditions are met will be executed by default, but you can control the order of execution with priorities, and you can have rules that cause you to exit from the ruleset without executing the rest of the rules.

4.11. Decision Explanation/Auditing
Blaze Advisor provides a facility for recording logging and auditing information. Recorded information is configurable to meet the requirements of the specific deployment. Examples of information which can be recorded include (but is not limited to):
Recorded information can be used to report on the activity of the rules engine and the progress of each decision. The amount of information recorded can also be determined at runtime to increase levels of audit information, potentially to assist in debugging activities. Such information can be used in conjunction with Blaze Advisor’s debugging tools including the execution flow browser where detailed interrogation of a decision process is required (see Sections 4.4 and 4.5.)

5. Simulation

Rules may be tested within the development environment with interactive inputs or by reading test data files. In the production environment, candidate rules may be tested in parallel with production rules through the use of parallel ruleflow branches.

Regression testing can be performed by storing test inputs in a persistent data source and having the system read them in for debugging and QA.


6.1. Document Generation

Blaze Advisor provides rule project, performance, and conflict reports. Project reports show object model components, ruleflows, rule syntax, conflict analysis, cross reference information, comments, and more.

Performance reports analyze rule times and number of calls on rule processing for both locally running rule projects and deployed rule services. Timer options allow data gathering over multiple sessions to evaluate averages, totals, minimums, and maximums. Remote performance tracking allows users to start monitoring the performance of a rule service on the fly without have to take down the server.

Conflict reports analyze potentially conflicting or otherwise invalid rules. Rules can be analyzed on demand in the IDE or can be reported in complete project reports. Testers can select the conflict tests to be included in a report and the rulesets that should be tested. This narrows the amount of output that needs to be perused.

6.2. Document Management

Blaze Advisor does not provide any inherent document management capabilities—when reports are created, the user is prompted for the location to store the files. There is no management of the reports beyond that.

6.3. Interfaces to Third-Party Document Management Systems

No specific interfaces to third-party document management systems are provided; however, integration of Blaze with major document management systems products (e.g., Interwoven) is possible using API calls.
7. Development Environment

Blaze Advisor Builder provides a graphical IDE for building and maintaining rule-based applications. (See Sections 3.1 and 3.2 for more on Blaze Advisor Builder).

Of particular importance are the Blaze Advisor Quick Deployers—wizard-based facilities designed to simplify deployment of Blaze Advisors applications. Quick Deployers automate code generation to create configuration files for immediate deployment in the products of major vendors. Wizards collect deployment information and generate appropriate installation files for any supported application server, ant build files, Web Services, and configuration files. Quick Deployers ask a series of questions that determine the specific configuration of the production environment and the desired operation of the rule service. They can deploy rule services as session beans in an EJB container, packaged and deployed for use with most of the major J2EE application servers (such as WebLogic, WebSphere, Oracle iAS, Sun ONE, and jBoss). If the selected EJB container supports the features, you can choose to run as a Message-Driven Bean, a stateless service, or a stateful (synchronous or asynchronous) service. They can also deploy as a .Net assembly. In addition, Quick Deployers also can generate deployments for processing in Java, Microsoft Transaction Server, JSR-94 rule engine interface compliance, and Web Services.

Quick Deployers are driven by a plug-in architecture, so customers can also create their own customized deployment specific to their environment and have use of that custom deployment repeatedly through the wizard. In addition, new plug-ins to support new environments can be made available to customers immediately instead of waiting for the next release of Blaze Advisor. (For more on Quick Deployers see Section 2.1)

7.1. Programming Languages and Support for Open Standards, Components and Frameworks.

The Blaze Advisor IDE is written in Java; a version written in J# is also available for users wanting a .NET version of the product. A version is also available that can read from and write to COBOL copybooks.

Blaze Advisor is designed to work with any structured data format, including Java and .Net objects, SQL databases, XML documents/schemas, and message queues. Blaze Advisor also provides extensive XPath operations for the interrogation of XML strings and objects. It can receive and return or issue XML strings and objects based on the structure used across the architecture. Native support for database operations is also built in to the rule language.

The Blaze Advisor Rule Server is a 100% Java object conforming to both J2EE (1.2) and Web Services open standards. Supported deployments include J2EE Enterprise Java Bean (EJB), Model Driven Bean (MDB) or Web Service. Other configuration options available include deploying the server as a COM/MTS object, a .NET component with a C/C++ interface, or within EAI layers such as MQ Series, JMS and RMI.

7.2. Software Models and Code Generation

Blaze Advisor Java can generate deployment code for deployment as a Java object, a J2EE session bean, message driven bean, or Web Service. Blaze Advisor .NET allows you to create a .NET deployment, generating client code in C#, J#, or Visual Basic. Blaze Advisor for COBOL can generate an entire rule project as COBOL code, complete with the ability to read from and write to COBOL copybooks.

7.3. Maintenance and Update Support

The repository supports rule service deployment for multiple applications, ensuring consistent enforcement of business policies and procedures across an enterprise. Rule structures, rule services, allowed value controls, and audit trail data (such as “last modified date”, “last modified by”, and
“reason changed”) are stored in the repository, where the rule server monitors changes in the repository and updates affected rule services without interrupting production operations.

The repository also has release management facilities, so that you can save off a set of files as a specific release of the application so that subsequent changes to the files do not affect the released version of the product. Configuration options include saving XML rule files in a LDAP system that supports JNDI, a database that supports JDBC, or saved as flat files. (For more on Blaze Advisor Repository see Section 2.1)

7.4. Integration with Third-Party Application Development Environments
Blaze Advisor Builder is a standalone Java application. It does not integrate with any other ADEs; however, it is possible to use Blaze Advisor’s RMA API calls to integrate the business user view into those environments.

8. Methodology Support
Blaze Advisor does not provide any specific support for any particular (public) business rules development methodology. [IS THIS CORRECT? PLEASE EXPLAIN, IF OTHERWISE]
However, product documentation includes a section covering best practices: development, including roles and responsibilities; repository management; security; repository content organization; authoring practices and patterns; rule writing; release management; quality control; deployment and execution; and performance tuning.

9. Integration
9.1. BPM Suite Integration
Blaze Advisor has demonstrated integration with BPM products from Fujitsu Software, FileNet, Lombardi Software and Savvion. Integration for Blaze Advisor is built into the Fujitsu Software product, and similar joint development work is in progress with FileNet.

9.2. Production Systems and Third-Party Applications
Blaze Advisor’s open architecture communicates with external systems and data sources via CORBA, MQ Series messaging, EJB, JDBC, COM, .NET, Web Services, XML, JMS, and other middleware transports such as Tibco Rendezvous.

9.3. Web Services/SOA Deployment
Quick Deployer Wizards supports Web Service generation for the following EJB Containers:
  ♦ iPlanet 6.5
  ♦ WebLogic 6.1, 7.0 and 8.1
  ♦ WebSphere 4.0 and 5.0
  ♦ Sun ONE 7.0
  ♦ Oracle 9iAS Release 2 (9.0.3)
Generated files for the Web Service are the same as for an EJB, but include additional files to support the container’s Web Service implementation. (i.e., Blaze Advisor supports the same set of capabilities around deploying in an SOA as the application server that a customer chooses to use.)

9.4. Messaging and Middleware Integration
Quick Deployer wizards can generate container-specific files for the following EJB containers:
  ♦ iPlanet 6.5
Blaze Advisor communicates with external systems and data sources via MQ Series messaging, JMS, and other middleware transports such as Tibco Rendezvous.

9.5. Database Support
Blaze Advisor supports any relational database vendors with JDBC database drivers, including Oracle, Microsoft, IBM, Informix, and Sybase.

9.6. Predictive/Analytic Modeling Support
The ability to generate rules from data for use with Blaze Advisor is facilitated through the use of Fair Isaac’s Model Builder tool (optional). You can also use Model Builder to create optimized scorecard models and decision trees, and for identifying policy weighting factors and thresholds for building more optimized business rules. Blaze Advisor has built-in capabilities to bring in models from Model Builder.

10. Templates and Frameworks

10.1. Horizontal and Vertical Industry Templates, Frameworks, and Pre-Built Rule Bases
Fair Isaac offers three industry-specific templates. In Insurance, Quick Start for Insurance Underwriting includes rules built around the ACORD Standards and a framework enhanced by lessons learned from Fair Isaac's successfully deployed BRM applications. Similar offerings are available for Mortgage and Healthcare.

10.2. Domain-Specific Knowledge Products
These offerings are primarily made available as consulting engagements where the framework code is included as part of the offering.

10.3. Domain and Industry-Specific Packaged Rule-Based Applications
Fair Isaac plans to eventually develop the above-mentioned frameworks into full packaged applications. (Note: Many of Fair Isaac’s existing applications have added Blaze Advisor as a rules engine component, such as Falcon Fraud Manager.)

11. Administration and Security
Blaze Advisor provides a built-in authorization manager that allows role and file authorization levels to be set independently. An API for user authentication and authorization is also provided. This facilitates integration with existing security infrastructure systems (i.e. single sign-on, LDAP, etc.), allowing security administration for Blaze Advisor within the existing infrastructure.

12. Platforms
The production system is certified to run on hardware from IBM, HP, Dell, Sun, or Windows-based PCs/workstations/servers. The execution engine can be deployed on mainframes, Unix, or PCs,
13. Scalability

13.1. Vertical Scalability
Blaze Advisor provides automatic scalability for thousands of client sessions. Performance scales linearly as the number of CPU’s is increased.

Blaze Advisor is architected to run multiple simultaneous rule execution processes in separate threads. Each thread is dedicated to handling the complete rule service processing for a service requestor (which may be a physical requestor, such as a Web client in an interactive Internet application, or a logical requestor, such as a queued service request in a batch application). Each thread is an instantiation of the Rule Engine processing code held in memory—Fair Isaac refers to these threads as “Rule Agents.” Only the context information is maintained in these agents, which makes their instantiation fast and cost effective. The rules structure is shared by all Rule Agents participating in a rule service. The number of Rule Agents and their system properties are configurable by the system administrator.

Administrators can even make configuration parameters dynamic, based on load factors as seen by an application server. Each Rule Agent that is running and available uses system memory, but adds more parallel load capacity to the overall application. This architecture ensures that Blaze Advisor is infinitely scalable to handle any desired load volumes, simply by adding more processing threads and spanning multiple processors and servers if needed.

13.2. Horizontal Scalability
Blaze Advisor provides an enterprise rules repository for native deployment on multiple platforms (Java, .NET, COBOL). A single repository offers the ultimate flexibility for interactions between applications, data, and rules. This architecture makes rule maintenance and deployment far less complex for accessing and executing decision logic in many types of environments with no translation or conversion steps necessary. A single repository accommodates multiple projects simultaneously and delivers a high degree of business logic re-use and extension across the enterprise. This promotes increased accuracy of your decision management policies and lowers development costs and time.

A built-in versioning feature manages historical content of rules and rule services including author, date and additional comments. Rules created or changed in a RMA or in the IDE are stored in the rules repository.

The repository can be used to prevent multiple users from making changes at the same time or overwriting revisions by providing a way for users to check out and check in rule service components. The system allows offline work with synchronization upon re-linking to the shared repository. Both IDE and RMA changes can be tracked with automatic assignment of incremental version numbers and audit information. The collection and reporting of audit information is a configurable element of the version control and security features of the repository, allowing administrators to review alterations to the rulebase, tracking who made the changes, when, what was changed and additional management information as required. Other third-party tools may be used as well. (For more on Blaze Advisor Repository see Section 4.6.)


14.1. Industry Standards Initiatives
Fair Isaac leads the OMG Production Rule Representation submission team, is a supporter of the OMG SBVR, and advises on BPMN/BPDM at OMG. Fair Isaac is also involved in business rules initiatives with the W3C and supports the JCP JSR-94.
14.2. **Business Rules Consortiums, Groups and Conferences**
Fair Isaac is a primary sponsor of BRForum and EBRC (refer to their web sites), and is active in certain domain standards bodies (eg ACORD, PMML). Conference presentations and support include BPM conferences, DAMA, and Semantic Technology.

15. **Pricing**

**PLEASE PROVIDE STARTING PRICING ($) FOR BLAZE ADVISOR**

16. **Company, Product Positioning and Support**

Blaze Advisor and its optional tools offers one of the most comprehensive set of products currently available for building and managing BRM applications.

16.1. **Company Background**
Fair Isaac was founded in 1956 and is a leading provider of analytic development tools and applications. The company markets its various offerings as “Enterprise Decision Management” products. These products include predictive models; customer data management and data analysis services; and rule-based decision management systems. Fair Isaac also provides tools and services that help businesses develop and deploy their own systems for enterprise decision management.

Many of the leading government agencies, banks and credit card issuers rely on Fair Isaac solutions, as do insurers, retailers, telecommunications providers, and healthcare organizations.

Fair Isaac serves global markets through offices in nine countries and has approximately 3,075 employees. Of these approximately 130 are in BRM (100 product development and 36 in product sales. The company also has about 230 employees in consulting and over 100 in support.

16.2. **Positioning**
In the BPM space, the industries most represented in Fair Isaac’s customer base are insurance and finance, followed by government, healthcare, and telecommunications; however, the company has customers across a wide variety of industries.

16.3. **Educational Training/Product Training**
Both on-site and scheduled training is offered for Blaze Advisor directly at the fundamental and advanced levels. Training is live, with instructors available to assist students. Students have access to training computers and hardcopy training guides. Blaze Advisor software also comes with numerous code examples and tutorials in HTML descriptions and code files.

16.4. **Product Support**
A dedicated team of engineers provide technical/product assistance for clients from solution deployment through production. Product Support teams provide clients with a single Fair Isaac point of contact. Support teams also have multilingual skills. To better assist in phone conversations with non-English speaking clients, Fair Isaac may engage translation services during the call (currently available through the AT&T Language Line Service).

Support engineers are trained in problem analysis. They also retain ownership of the incident and involve developers or Professional Services specialists as needed in the resolution process. Inquiries are logged, tracked, resolved and closed only with client agreement.

Various level of support for Blaze Advisor are offered, including general (within specific business hours), 24-hour services (based on maintenance contract) and “Temporary After Hours” support available as an add-on service. In addition, Fair Isaac provides online support.

16.5. **Consulting Support**
Fair Isaac Professional Services is a stand-alone business segment with over 400 professionals delivering solutions to organizations worldwide, and with personnel spanning 16 offices in 10
countries. It is currently engaged on nearly 400 projects and has delivered solutions for 46 of the world’s top 50 financial services institutions.

Projects are guided by and delivered within the context of a structured delivery methodology (The Fair Isaac Business Integration Methodology). This methodology consists of a series of detailed steps, tools, and templates encapsulated within a framework that follows an industry standard approach to addressing core project management processes. All projects undertaken are also supported by a centralized Program Management Office, which is responsible for ensuring adherence to methodology principles, facilitating project staffing, and consolidating weekly reporting of project status.

17. Case Study: California Department of Motor Vehicles (DMV) Revolutionizes Vehicle Registration Using Blaze

17.1. Context
The California Department of Motor Vehicles (DMV) is responsible for collecting approximately US $4.1 billion annually in vehicle registration fees. Centralized computer systems in Sacramento, California communicate with local systems across 167 field offices throughout the state to handle the complex task of calculating registration fees for the nation’s largest population of autos, trucks, motorcycles, vessels and other vehicle types.

17.2. The Challenge
In 2000, the DMV realized that they needed to update and converge the two separate vehicle fee systems, and move to a modern system that would meet the state’s strategic realignment towards eGovernment with future public access via the Internet. The DMV began the Vehicle Registration Fee Computation (VR Fee Comp) project to consolidate and streamline the fee generation processes that were implemented across two different computing platforms, two computing languages and two distinct systems:

The DMV Automated fee system (DMVA) is deployed on servers at each of the DMV’s 167 field offices and its headquarters. It processes customer initiated vehicle registration transactions in real time in IBM’s proprietary Event Driven Language (EDL).

The DMV Batch fee (DMVB) systems are deployed on mainframe computers at the Teale Data Center, where renewal notices are generated and the Remittance Processing system handles most of the “DMV by mail” COBOL.

Because of the two different computing platforms, changes and updates required two separate development efforts, two different analyst teams, and two different databases. This made it difficult to coordinate changes and ensure consistency between the two systems.

“Most of the computer programs composing the DMV legacy fee systems have been used for more than three decades with constant updates and workarounds grafted into the code by multiple people over the years,” says Jerriianne Seitz, Data Processing Manager at the California DMV. “Making changes to the system was no easy task as one change could introduce numerous side effects.”

Due to the complexity of the programs and duplication of effort required to make changes to two separate systems, the DMV was challenged to meet legislatively mandated deadlines for fee changes. Even minor changes required extensive analysis and programming efforts by the legacy system’s development staff. There also was a real and practical need to update the systems as they had met their physical limitation; the DMV reached a point where they couldn’t add more statements to the system.
17.3. The Solution

After discovering all of the places where Vehicle Fee calculation rules had been programmed, the VR Fee Comp team’s analysis uncovered that instead of simply recoding the existing systems, they needed to consider newer technologies and methodologies for managing their business processes.

The team recognized that BRM software would enable the separation of business logic, policies and processes from the actual application programming. The ability to give the power of business policies and business changes to analysts instead of programmers was a monumental change to the current application development process for the DMV, as often a limited amount of programming resources resulted in latency to system updates. In addition to faster system changes, a business rule system could enforce the greatest levels of compliance with the legislative mandates.

Any BRM system would have to be usable by non-technical analysts responsible for overseeing legislative compliance. It would also require an intuitive interface that would allow comprehensive control and testing of rules without obscure programming syntax. From a technical standpoint, it must run quickly and scale to handle massive numbers of transactions on a variety of systems, from the largest mainframes to office servers. And it would have to do all this without requiring replacement or rewriting of the vast majority of the legacy applications in place throughout the DMV infrastructure. According to Seitz, “Many people within the organization felt our rules were so complex that a third-party software package wouldn’t be able to handle the task. We needed to evaluate and demonstrate functionality and performance in any solution we planned to bring in.”

Upon delving into the project, however, the DMV realized the necessity existed to also conduct a best practice review of their business policies and practice methodologies. The resulting clearer understanding of their decision infrastructure proved to be an invaluable process. They examined a business rules methodology, created a blueprint of their business processes, and looked at normalizing and understanding the terms and policies that were already in place. It was necessary to look at existing business policies, the legislation that affects them, and how a consolidation of look-ups and interrelations could be architected to simplify future updates.

The DMV’s Vehicle Registration team reengineered a vehicle registration fee system that would work in a manner that would be transparent to end users. The revised solution resulted in a Blaze Advisor, IBM WebSphere application server, and a J2EE solution operating on a mainframe.

The development team, however, faced the challenge of having little or no experience in Java software development. Most of the team was experience with COBOL and Assembler programming. Fair Isaac Professional Services offered Java software training and on-site assistance during the pre-project implementation phase. The DMV was able to move smoothly from design and structure of conceptual rules to development and execution without fear of overlooking or misinterpreting key decision factors.

Leveraging the power of Blaze Advisor and its RMAs, the DMV successfully gives the non-technical analysts responsible for overseeing legislative compliance the ability to ensure proper implementation of the policy rules across the DMV vehicle registration fee systems without having to become programmers. They also defined templates for certain types of rules that business analysts should be able to create on their own. These maintenance facilities are available through automatically generated Web pages that eliminate formal rule syntax and custom editing environments.

17.4. Result/Benefits

The DMV has successfully developed a revolutionary solution that boasts a Blaze Advisor business rules system as the cornerstone of its vehicle registration fee systems.

The first phase of the project implemented the new technical environment and fee business rules for vessels (i.e., boats and other watercraft). Vessels were selected for the first implementation because it
represented a lower volume transaction count and therefore a lower impact to DMV field office operations. This phase was completed and successfully released to production in March 2003.

The second phase of the project implemented the new fee business rules supporting expedited legislation for the Vehicle Registration Penalty (Reg penalty) fees which could not be implemented in the legacy fee system due to its limitations. This phase went live across the DMVA in April 2003. The Reg penalty fees system processes 60,000 to 75,000 business transactions per day.

The third phase of the VR Fee Comp project went into production in February 2005. Phase three implemented the rules and fees for autos, commercial vehicles, trailers, motorcycles and off-highway vehicles. It required over 2,000 rules and numerous rule-sets, rule flows, and tables and processes for several hundred thousand business transactions per day. While many projects boast tens of thousands of rules, the DMV team’s thorough methodology resulted in an effective rule-set of 2,100. With an efficient use of intermediate rules, the DMV proudly reduced the overall rule-set that results in less coding. It also provides better visibility into the rules that are in effect, an understanding of the levels of business change and how changes to the rules will interact and affect subsequent business processes. The DMV’s new system leverages business rules to support mandated and changing legislative laws.

Thus far the fee comp team has focused on updating the DMVA aspect of the project. With the deployment of the fourth phase of the project to production the team moved towards the consolidation of DMVA and DMVB. The DMV’s new system leverages business rules to support the mandated and changing legislative laws.

Coupled with the IBM’s WebSphere Applications Server, Blaze Advisor positions the DMV to make future system enhancements more easily and to add centralized and consistent rules to delivery channels such as self-service websites and telephone response systems. “The implementation of the VR Fee Calculator on the Internet further supports our strategic business and IT goals by improving the integrity and quality of its products and services to customers,” added Seitz. “The new rules powered system and resulting business practices should continue to lower our operating costs by decreasing the number of telephone calls and transactions in the CA DMV Field Offices and Telephone Service Centers.” The system went live on June 30, 2005 as the main feature on their webpage and is a huge success. Now citizens can go see how much it would cost to register a vehicle or boat in California before purchasing it and will be able to see how much their registration fees will be in future years. To see the application run live, visit https://vrir.dmv.ca.gov/ feecalculatorweb/select calculations.do.

18. Company Offices

USA
Corporate Headquarters
901 Marquette Avenue
Suite 3200
Minneapolis, MN 55402 USA
Tel: +1 (612) 758 5200
E-mail: info@fairisaac.com

Brazil
Rua Luigi Galvani, 200 - 8º Andar
Brooklin Novo
04575-020 São Paulo – SP
Brasil
Tel: +55 (11) 5501 9222
Fax: +55 (11) 5506 7703
Email: Lac_Info@Fairisaac.Com

Canada
The Exchange Tower
130 King Street West, Suite 1800
Toronto, Ontario, M5X 1E3, Canada
Tel: +1 (416) 865 3390
Fax: +1 (416) 865 3379
Email: Info@Fairisaac.Com
Germany
Fair Isaac Corporation
Maximilianstr. 35
D-80539 München
Deutschland
E-mail: emeainfo@fairisaac.com

India
Fair Isaac India Software Pvt. Ltd.
Titanium Building
135 Airport Road
Kodihalli
Bangalore 560017
India
Phone: +91 80 51371700

Japan
Meiji Yasuda Seimei Bldg. 5F
2-1-1, Marunouchi, Chiyoda-ku
Tokyo 100-0005 Japan
Tel: +81 3 6212 2770
Fax: +81 3 6212 2769
E-mail: Info@Fairisaac.Co.Jp
Malaysia
Unit 902C, Level 9, Uptown 5
5 Jalan SS21/39
Damansara Uptown
47400 Petaling Jaya
Selangor Darul Ehsan
Kuala Lumpur,
Malaysia
Tel: +60 3 7727 5413
Fax: +60 3 7727 5244

Singapore
No. 8 Shenton Way
#40-01 Temasek Tower
Singapore 068811
Tel: +65 6323 9700
Fax: +65 6323 3449

Spain
Fair Isaac
Calle Nuria 36, Planta 2
28034 Madrid, Spain
Tel: +34 917355020
Fax: +34 913721534
E-mail: Madrid_Info@Fairisaac.Com
This BPTrends Product Report is produced by BPTrends as a service to our members and readers. Our objective is to present the various options to organizations when they are choosing BPM software tools - not to endorse or rank the “best” BPM products. All product reports are produced in the same manner: BPTrends prepared a detailed Questionnaire that includes a set of core criteria. All features that extend beyond the core are treated as extensions of the core that render the tool more useful for the development of specific types of process work or for the development of BPM applications. This provides our members and readers with a means of comparing the core features of the products within a specific category. All participating vendors are asked to complete and return the Questionnaire to BPTrends. BPTrends reviews the vendor questionnaires and edits and produces the final reports. BPTrends does not do any product testing.

All rights to this report are reserved. Members and readers of BPTrends are hereby granted permission to download, reproduce and distribute copies of this report, for non-commercial purposes only, in the format in which it appears here. This report may not be reproduced, in whole or in part, without the express written consent of BPTrends.

Vendors whose products are described in these reports have established trademarks and registered marks for their products. Similarly, products by other vendors are sometimes referred to in these reports and these product names are also protected by trademarks or registered marks.

Carolyn Potts

Managing Editor

BPTrends. LLC