Overview of the Tranfinity Server Business Rules Engine

The Tranfinity Server Business Rules Engine from enTechnia is a software product that allows the user to introduce sets of rules into almost any part of their computer processing, and ensure that these rules are enforced correctly and uniformly.

This is accomplished in a way that makes the rules visible and accessible to any authorized person within the organization – not just to the IT people – and the rules can be changed easily to reflect changing business conditions. These modifications can usually be made without the need to involve the organization's IT department, or the need to perform conventional computer programming on the core application that runs the enterprise.

The Business Rules Engine consists of several software components that can be configured to work seamlessly with the organization's existing IT environment, and can be extended to cover any industry. Templates of basic Rules can be used as a starting point for several application areas, and these can be modified to tailor-fit the specific needs of any organization.

The Business Rules Engine can also be used to develop applications that can adapt quickly to changes in the business environment, providing a distinct competitive advantage to the organization using it. Some of the advantages of using the Business Rules Engine are:

- Lets you separate business logic from application logic.
- Reduces cost of application maintenance.
- Shortens development time for complex business logic.

The Business Rules Engine resides on a Server that can be accessed through the organization's internal network (intranet) or, if desired, globally over the Internet. The software is very robust and capable of accommodating a very large number of concurrent users. It utilizes industry-standard technology such as http, TCP/IP, and XML for communication, which allows any platform to access the services.

Purpose and uses of the Business Rules Engine

The primary purpose of the Business Rules Engine is to allow an organization to enforce its business rules easily and automatically. For example, a business may want to adopt a certain Discounting policy for customers that meet certain criteria. This can always be accomplished by having the IT Department program these Business Rules into their core software applications, but there may be drawbacks to this: Modifying major applications often leads to a situation where these Business Rules become not easily accessible by management or other non-IT personnel, who are often the decision-makers as to when a rule will change. Whenever an update or version change of the now-modified core application is to be installed, much care and time is needed to ensure that the business rules code will perform the same task without upsetting the new version of the application.

With the Business Rules Engine, the business rule parameters can be stored in a Database (the 'Business Rules Repository') and the Rules themselves reside in a directory separate from the core application. The Rules or parameters can be inspected and/or modified, often without the need for 'traditional' programming. Business rules can be either simple or very complex, and using tables to hold the parameter values allows changes to be made while keeping the "Rule" code untouched. Both the parameters and the Rules may be created or modified using a simple graphical tool that will allow any trained person(s) to easily perform changes or additions.

The Business Rules Engine is accessed from another application using enTechnia's API (Application Programming Interface) that provides a consistent method of communication between the applications. When business rules need to be enforced, the core application accesses the Business Rules Engine and requests it to apply the desired rule(s), passing it any information necessary for the correct enforcement of those rules. In the Discounting example, the application would pass data about the current customer. The Rules Engine would then proceed to apply the criteria in the rule, and enforce the Discounting Policy, as appropriate.

The greatest advantage of this method of enforcement is that the Business Rules are external to the application and can be viewed, maintained, tested and applied without affecting the core application. All the application needs to do is to access the Rules Engine and exchange information with it. If the Business Rules change (i.e.; the revenue requirements for receiving an "A" level customer discount has changed) the core application need not be modified. A trained managerial-level or other administrative person can use the graphical tool to make this change. This frees the IT Department from the responsibility of modifying and retesting the main application each time some business conditions change.

A secondary but equally important purpose of the Business Rules Engine is to allow the actual development of new applications, where again, the application and business logic can be separated from the presentation logic. For example: One can develop a set of browser-based input forms (Web pages) and store all the business and Database access logic in the Business Rules Repository in the form of Rules. The Web pages can access the Engine via JavaScript or VBScript. The Engine can do all the "heavy lifting" in terms of processing the entered data.

If there is a need for very high speed or for very elaborate programming, and the IT resources are available, rules can be programmed in traditional programming languages and stored in libraries that the Engine can use (DLLs in MS Windows and soon Shared Library Modules in Linux). If the IT resources are not available in-house, this work can also be performed by enTechnia, as a separate project.

Accessing the Business Rules Engine from an application can be accomplished via XML using either http or TCP/IP. Any computer language such as C++, Delphi, VB Script, or JavaScript that can create ActiveX controls can communicate with the Engine.

Business Rules Engine Description

The main component of the Tranfinity™ Server Business Rules Engine is the 'Business Rules Executor' which is a multi-threaded software module usually residing on a Server (though it can run locally on a machine if this is desired). Any Client software which can format XML messages can access the Executor using http or TCP/IP communication methods.

When the Business Rules Executor is started it loads into memory a set of rules from the Business Rules Repository Database. Which rules get loaded initially is configurable, so that often-used Business Rules can be stored in memory and immediately available. Other Business Rules can be loaded as needed by an 'Object Manager' that manages available resources to achieve optimum performance as configured by the user.

Each Rule is assigned a unique number (numeric id) and a name when it is first created and placed in the Repository.

When a Client application needs to apply a rule (or to execute any piece of business logic that can be stored as a business rule), it calls the Server, passing it the unique numeric id of the desired Rule and any data items that are needed so that the criteria for the Rule can be evaluated.

The Server receives the request and starts a new thread to service it. (The Server can accept multiple requests and be processing them simultaneously, since a different execution thread services each request independently). Each request can have its own copy of data, or data can be shared between requests under control of the calling application.

The Executor software retrieves the desired rule and starts executing its instructions. When it is done, successfully or otherwise, it returns a completion code, and any data items requested by the caller.

Rules are, therefore, mini programs, written in a simple "language" (a variation of Basic). With the aid of the graphical Editor tool, even non-programmers will be able to modify the rules as the parameters change. These rules can invoke other Rules (including 'built-in' Rules provided as part of the Engine or written by the user in traditional programming languages). Rules can also execute scripts written in a Visual BASIC-like script language or in a JavaScript-like script language. All the resources of the system are available to the Rules.

Entire templates of basic Business Rules appropriate for a specific application (e.g. General Accounting) can be modified with the graphical Editor tool or used "out-of-the-box". These templates (called "Rule Books") can be loaded in their entirety or partially as needed.

A Typical Application

One of the earlier applications of the Business Rules Engine was an implementation of a Rating Engine for calculating the freight necessary for shipping a package from any point within the United States to any destination, domestic or international.

This application can be used not only in an company's warehouse for shipping product out, but it can also be used in conjunction with a Web site that takes orders from customers across the world and allows those customers to select the best way to have their order shipped to them.

All the rules for calculating the freight for a package of a given weight sent from a given zip code to a given destination were entered into the "Rating Rule Book" in the form of Rules. All restrictions applied and facilities offered by different Carriers (UPS, Fed Ex, etc) were included in the Business Rules.

A Client calling the Server would provide information about the origin and destination addresses, the weight and dimensions of the package and the desired method of shipment. This could be a specific request, such as "UPS ground" which would then return an exact cost for the freight, or it could be a general request such as "Overnight" which would return a list of several methods of shipment that would satisfy the "Overnight" requirement, together with the cost of each.

A typical Client could be a browser-based set of input forms with a human being entering the information at their desktop, or an application program such as an invoice generator that needs to calculate the freight cost for a given customer order to include it in the invoice. In the latter case, the invoicing application is extended to call the Business Rules Engine Server passing the necessary information and receiving back the results using XML. Freight rating rules change often as Carrier fees change, new services are introduced, and old services are modified. Keeping all the Carrier-related rules isolated makes the maintenance of the invoicing application much easier. (For an annual fee, the carrier rules are maintained by enTechnia, and updated to the user base with each change).

Business Case:

Using the Tranfinity Server in a Point Of Sale application.

A franchiser owned retail convenience outlets located in a number of states and counties. Most of the stores offered alcoholic beverages for sale, though the differences in state law between the varying locations made it difficult for the owners and managers to set a corporate policy on how to enforce the sale of alcohol and ensure that it was sold only to those persons who were of legal age in the state that the store was located.

These stores were among many of the convenience store and liquor stores in a region that were targeted by local officials of Alcohol, Tobacco and Firearms for "sting" operations. Either non-caring or unscrupulous employees in two of the stores sold alcohol to under-age customers without checking the ID of the buyer. The result of the charges for the owners were not only stiff fines, but the loss of revenues from sale of alcohol at the two stores for a number of months (which represented a significant portion of the daily business). Part of the process of re-gaining the license to sell alcohol was showing that a process was put in place to help stop these illegal sales from happening again. The owners decided to implement a software solution to work with their Point Of Sale system.

Before the incident with the underage sales, the franchiser was in negotiations to acquire a number of stores that were using a different point of sale system. It was a proprietary system, running on Unix – which was completely different from the Windows-based package that was being used in the existing stores. Licenses for the Windows package for the new stores would be costly, but was planned for in the near future. The reduced of revenues from the loss of the license to sell alcohol at the two stores, as well as a slowing economy delayed the purchase of the additional licenses for the Windows POS system. On top of that, one of the stores being purchased also was involved in a sale of alcohol to a minor, and there was a need to implement the same "proof of process" to regain a liquor license.

For far less than the price of the licenses for the Windows POS system, was the price of a single license for the Tranfinity Server. The Windows-based POS had no means for enforcing any "age" or date-based rules, other than some inventory management functions. The same was true with the older Unix software. Since both systems were going to require modification, it was fairly simple to have code created that would make an XML call to an I.P. Address and port on the network.

For all alcohol sales, (identified by the scan of the SKU) the checker would have to enter a date of birth from the buyers ID. The POS system would send over a XML transaction that included the Store ID and the date of birth entered by the checker. The Tranfinity Server, running on the franchiser network in their corporate headquarters would receive the XML data, and identify the location of the store from the Store ID that was sent with the date of birth. A reference to the law for the location (Age required to purchase alcohol) was checked against a table containing all of the stores and the according laws for their address. The date entered was checked against the current date, and the if difference was greater than what was the minimum to buy alcohol, the POS system(s) would allow the transaction to continue. If the age was less than what was required, a message was presented to the checker, who was instructed in the message to remove that item from the sale before being allowed to continue.

The solution was less expensive than modifying the Windows-based code to perform the check, and much less than the cost of upgrading the newly acquired stores to the Windows POS system. The data for the rule could be easily maintained by IS Administrators at the corporate headquarters, on one server that functioned for both systems. Once the funds were available to upgrade stores from the Unix to the Windows POS package, the solution was already in place, and there were no additional charges to maintain that solution for system version upgrades. In addition to saving money on the solution, it assisted in getting the liquor license back for the affected stores, increasing the incremental revenues and overall profits.