A WEB-BASED EXPERT SYSTEM FOR VEHICLE REGISTRATION

Alan T. Demmin and Du Zhang

Department of Computer Science
California State University
Sacramento, CA 95819-6021
atdemmin@innercite.com, zhangd@ecs.csus.edu

Abstract. *In order to satisfy an increasing need to respond rapidly to changing business requirements and knowledge, and to reduce implementation costs, the mainstream business world is progressively relying on the expert systems technology. In addition, recent e-business initiatives are requiring companies to provide "web-enabled" systems in order to maintain a competitive edge in their marketplace. In this paper, we describe our experience in capturing the business logic and implementing a web-based expert system for vehicle registration fee computation at California Department of Motor Vehicles. The system has been implemented using the Blaze Advisor rules engine and has been deployed to Sun's J2EE Reference Implementation of a Java application server. Our work has shown that using expert system technology to implement a web-based, business-driven solution is certainly a viable and promising option. Deployment of the rule service as an Enterprise JavaBean exposes the rulebase to various types of business clients, and provides a consistent and adaptable implementation of the business domain.*

Keywords: business rules, web-based expert systems, Java technology, vehicle registration.

1. INTRODUCTION

Business rules are declarative statements that define or constrain a business domain and are often implicitly defined within the manual and automated processes of a company. "Business rules are directives intended to guide or influence business behavior" [10]. The business rules for a company may or may not be formally documented and may only be available as learned or perceived knowledge of the business personnel. Over the last decade, companies across all industries have shown an expanding interest in formally defining and documenting their business rules and establishing them as official artifacts for the company. In fact, definition and documentation of business rules have proven to help clarify business objectives, streamline business processes, and even improve a company's definition of new or changing business requirements. This helps when justifying business decisions, training new business personnel, and can even reduce the time necessary to deliver software modifications to the production environment.

While there are distinct advantages in simply documenting business rules, the eventual expectation leads to an implementation of the business rules into an automated solution and to make them available to all business processes. In particular, recent e-business initiatives have shown that companies need to rapidly implement changes in order to maintain a competitive edge within the marketplace. This is setting the stage for the integration of expert system technology and intelligent agents within existing business infrastructures.

The purpose of this study is to investigate the feasibility of utilizing an inference capability to administer the business logic for an enterprise within a web-based architecture. A subset of the business domain for vehicle registration fee computation at the state of California Department of Motor Vehicles (DMV) is selected for this study, in particular, the fee computation logic for vessel vehicle types is chosen for the prototype implementation. The business rules for this domain were gathered, organized, and codified. Blaze Advisor rules engine [26] is utilized for the expert system development. The rules service was then deployed to a Java 2 Enterprise Edition (J2EE) web application server and a web browser interface was developed to allow users to inquire about registration fees that are assigned for various transactions.

There are several reasons behind our choice of the DMV project. The first author works as a consultant at DMV and has an intimate knowledge of the problem domain. The current fee processing systems at California DMV were first developed in the 1970's and have been modified many times over the past three decades. The online system used at the local field offices was implemented using the Event Driven Language (EDL), which is the assembly language of the IBM Series/1 machines. The headquarters batch system, which processed the daily transactions and updated the main database during the nightly batch cycle, was implemented in the Cobol for IBM mainframe running the Multiple Virtual Storage (MVS) operating system. Because separate teams at the DMV maintain the two systems, a single change in legislation or business policy can contain different interpretations. Over the years, the two systems have diverged significantly and can sometimes produce inconsistent results. In addition to the potential inconsistency in the two fee computation systems, there is also a significant need for a more modern web-based solution.

The results of this study indicate that using expert systems technology to implement a web-based, business-driven solution is certainly a viable and promising option. The ease of this implementation relies heavily on the supporting tools that are supplied with the development environment. The prototype system we developed allows multiple platform deployment, distributed object access, and graphical or model-driven maintenance of its rule base.

The remainder of this paper is organized as follows. Section 2 briefly describes related work that is currently being pursued in the area of business rule methodology and expert systems technology. Section 3 discusses the design strategy for the project. Section 4 deals with some of the implementation issues. Performance evaluation of the web based expert system through some common use cases for vehicle registration is the focus of Section 5. Comparison with an alternative approach (JESS, Java Expert System Shell [33]) is provided in Section 6. Finally, in Section 7 we conclude the paper with remarks on possible future work.

2. RELATED WORK

2.1. Business Rule Methodology

Business Rule Solutions, LLC is a leading company that focuses on business rule solutions and methodology [25]. The concept of a fact-model has been introduced within the business rule methodology called Proteus™ [25]. A fact model is a way of modeling the business terms and their relationships and is similar to the entity-relationship diagramming that is used for relational database design.

Knowledge Partners Incorporated (KPI) is another leader in the business rules approach [30]. A formal methodology called the STEP principles (for Separating, Tracing, Externalizing, Positioning rules for change) is proposed in [11].

The *Business Rules Forum* is an annual conference for the business rule solutions and methodologies. This conference has grown substantially over its five-year history and continues to attract organizations in all industries, both in local and federal governments as well as the private sector. The focus is primarily on the discussion of business rules as true artifacts for organizations and the importance of following a formal business rule methodology when defining the rules for an enterprise.

2.2. Rules Engine Technology

Blaze Advisor is an industry-leading decision building system that employs rules engine technology to deliver automated business solutions [26]. The platform is 100% Java-based and offers deployment capabilities to most platforms available today. It includes many interfacing capabilities to allow integration with virtually all modern and legacy systems and employs the use of XML. Blaze Advisor supports both forward and backward chaining. There are tools for analyzing the rulebase to locate inconsistencies, rule conflicts, and even circular logic. We use the Blaze Advisor rules engine to develop our prototype system.

ILOG JRules is also a 100% Java-based rules engine, geared toward providing business-driven solutions. [28] Similar to Blaze Advisor, JRules is a product that includes graphical user interfaces for building and modifying rule bases. It also includes capabilities to deploy the automated solution to various platforms and with differing interfaces.

JESS (Java Expert System Shell) is a scientific Java-based expert system shell that is an extension to the CLIPS system developed by NASA in 1984 [33]. Since JESS makes the CLIPS system Java-compatible, it has opened the door to providing many of the multiple platform and multiple interface capabilities of the business driven solutions.

IBM CommonRules. IBM started the Intelligent Agents Project at IBM T.J. Watson Research Center in 1994 to create intelligent agent technology that would be highly reusable, extensible, and easily integrated into a variety of distributed applications [18]. In 1997, this project divided into a few sub-projects, one called Business Rules for Electronic Commerce, which included an investigation into rule based technology for business processes involving both business-to-consumer (B2C) and business-to-business (B2B) e-Commerce. This research was particularly concerned with the creation of a core rules technology that could be reused with various heterogeneous applications to share common rules information. The resulting technology is known as IBM CommonRules [19], which is a Java library of classes to be used as a development platform for rule-based applications and includes a data interchange format defined as an XML schema called Business Rules Markup Language (BRML) [18].

2.3. Rule Standards and Specifications

Business Rules Group. The Business Rules Group [23] is an organization of practitioners in the field of systems and business analysis dedicated to defining and supporting the specifications on business rules and how they are related to enterprise organization and systems integration. This group was initially a project within GUIDE (Guidance of the Users of Integrated Data-processing Equipment) International [27] and now exists as its own organization.

OMG Business Rules Working Group (OMG BRWG). The Object Management Group (OMG) [31] is an organization established in 1989, dedicated to the establishment of formal specifications for the software industry that are vendor independent and are directed at the standardization and interoperability of object oriented technologies. The Business Rules Working Group (or Business Rules Special Interest Group) is a sub-committee of the OMG Architecture Board, working to establish formal specifications on business rules and/or rules engine technology. The BRWG recently issued a Request for Information (RFI) on Business Rules Expression.

Java Community Process - JSR94. The Java Community Process (JCP) is an open organization of Java developers and licensees whose goal is to expand and enhance the Java technology platform and its associated specifications [29]. JSR094 is a specification request to define a formal interface for a Java Rule Engine API to be used as a standard interface for both a Java 2 Standard Edition application and within a J2EE application server. This specification is a first step in standardizing the interface to a rules engine within the Java platform and includes high-level classes within the `javax.rules` package structure and defines classes for a Rule Service Provider, a Rule Runtime, and a Rule Session, among others.

Rule Markup Initiative. The Rule Markup Initiative as another open network of individuals from both the business and academic world, dedicated to establishing a formal XML schema for a Rule Markup Language that can be shared among rules engine vendors [32]. This will produce a standard set of XML tags to be used for rules representation, translation, and interaction between heterogeneous rule processing systems. As the formal specifications from the OMG and JCP work to provide standards for object management and the Java platform specifically, a formal rule markup will provide a standard way for businesses to share rules and for vendors to provide a consistent representation for rules interchange.

Java 2 Enterprise Edition (J2EE). It is also important to mention the deployment environment that has been chosen for this project. The J2EE specification [34] provides for a standard platform that supports component architectures, with consistent application programming interfaces used to interact with the various application services. The J2EE specification is becoming the industry-leading model for application servers based on the Java platform and will most likely be the industry standard for all object oriented enterprise applications in the future.

3. DESIGN

The design of our rule based solution for the DMV application requires a thorough definition of all terms, facts, and rules that comprised the knowledge base. In addition, the underlying infrastructure and platform that would support the enterprise system are also considered carefully during the design stage.

3.1. Object Modeling and Term Definition

This system has been designed using the principles of object-oriented programming and relies on the use of model-driven architectures. UML class diagrams were developed for the input/out object model and the high-level interface design. This object model was then used for the term definitions within the rulebase.

One of the first design considerations of any software system is the identification of data elements that will be used as input to the system, and similarly, the data elements that will be returned or updated by the system. When designing rule-based systems, these data elements are the terms and facts that are part of the knowledge base. The first step in designing a rulebase is to identify all of the data elements or terms within the knowledge and assign appropriate term names. The importance of this effort must not be overlooked. A term name can make a significant difference in the comprehension of the rules that utilize these terms.

It was important to first develop a list of terms and/or object attributes that would be used to develop the initial fact list for the rulebase. After interviews with the business users and analysis of existing programs, the input data elements were grouped into the following objects:

- FeeTransaction - the type of transaction being requested
- FeeVehicle - the vehicle that is involved in the transaction
- FeeOwner – the owner of the vehicle that is involved in the transaction

The information would be processed through the fee computation rules to determine a list of fees for the transaction being requested. In addition, some related fields are also returned as part of the solution. This information was grouped into the following object:

- FeeResult – the resulting fee list and related information for the transaction

After gathering all of the necessary information for each object type and gaining formal acceptance on all attribute and term names, a UML class diagram was developed to represent the object model. Since the fee computation system was being developed as a request/response system, the FeeRequest and FeeResponse classes were defined to encapsulate all of the functionality necessary to translate data between a serialized and de-serialized format, to match requests with responses, and to designate which rule service will be accessed within the rules engine. FeeRequest and FeeResponse are composite objects which contain all of the sub-objects used as input and output data within the rule base.

3.2. Business Rulebase Design

Once the terms and object model were defined, a formal methodology was employed to define the business rules. We chose the STEP™ process [11] to separate, trace, externalize, and position rules for changing business requirements. This process began by interviewing business users and business analysts to gather facts and rules about vessel registration fee computation. The process also included the use of rule mining techniques to extract business rules from the existing legacy programs.

This methodology led to a better understanding and formal definition of the fee computation business process. When a business process can be formally defined with a series of tasks and sub-processes, this can be represented within a rulebase as a ruleflow. A ruleflow governs the business process and organizes sections of the knowledge base into logical partitions or processing steps. When working with the Advisor Builder product, the ruleflow is created and maintained using a graphical editor, much like creating a data flow diagram or the structure chart of a computer program.

When designing the ruleflow for the business process, two main aspects of the business were carefully considered in order to provide a well-organized system:

1. logical separation of vehicle types
2. logical separation of fee types

Once the object model and business process were defined, the rules within the process tasks were created. As knowledge of the fee computation process grew, ruleflows were enhanced and refined logically, term names were improved to represent their true business meaning, and the object model was modified for clarity and efficiency. Due to space limitations, we skip the details of following the STEP™ approach. Interested readers may refer to [35].

For rule acquisition, two main pieces of business documentation specific to fee computation include:

1. State of California Vehicle Code
2. California Department of Motor Vehicles Registration Manual

Rule acquisition also involved the use of a technique called rule mining [20]. Rule mining is a process by which an existing procedural program is analyzed and the business rules are discovered, extracted, and documented. An effective method is to identify program output data and perform searches within the program for statements that modify this data. In turn, any interim data elements that are used in the modification of the output data are also researched. Eventually, the statements are traced back to the input object model. The resulting list of statements contains all of the program code that is used to develop a single output field. This process is called "program slicing" and was used to help start the process of rule gathering. After program slicing was performed, the business users are often able to help determine the true business meaning that supports the implementation in the corresponding program code.

The deliverables from this stage of the design are business rule design documents which include the ruleflow and design diagrams along with the rules organized into rulesets and documented in natural business language.

3.3. Technical Interface Design

The business logic for the fee computation application has been designed as an Enterprise Java Bean (EJB) on the server side of a multi-tier environment, and is to be offered as a service within a J2EE application server [34]. This service is accessible from various clients, including HTML web browsers, Java Applets, Java Servlets, Java Applications using the Java Remote Method Invocation (RMI) protocol, and other J2EE Clients (including wireless, hand-held devices). Since Blaze Advisor comes pre-packaged with deployment capabilities that turn any rule service into an EJB service, we choose Blaze Advisor's deployment model for the project.

The Blaze Advisor deployment positions the rulebase for changing business requirements by allowing an update of the rulebase to occur in a separate repository within its own component of the application. In fact, an update to the rulebase can occur without any changes to the client application, servlet, or EJB. The rulebase is stored in a repository file that gets loaded when the EJB is initialized. Advisor Builder and Advisor Innovator are two tools that allow direct update and deployment of the rulebase. XML has been used to communicate the response information returned from the FeeController servlet back to the web browser. In addition, Blaze Advisor uses XML for its internal representation of the rulebase. A formal XML schema has been developed for the FeeRequest and FeeResponse objects. These schemas will be used when the data is serialized into XML documents. The documents are then easily processed by Java components using a Java XML parser.

4. IMPLEMENTATION

4.1. Object Model Creation

A JavaBean is a class that follows a specific set of implementation constraints in order to provide consistent functionality as a Java component that can then be easily manipulated by other systems. Specifically, a JavaBean has the following properties [1]: (a) It is a public class, (b) It contains a public constructor with no arguments (self-initialization), and (c) It contains public access methods (getXXX and setXXX) for each property.

The Blaze Advisor rules engine utilizes JavaBeans to interact with data that will be manipulated outside of the rules engine. For instance, all objects that are passed from a Java application into the rules engine, are passed as JavaBeans, so that the rules engine can manipulate the data using the public accessor methods. Similarly, all output data that will be returned back to the calling application, will also be passed back as JavaBeans. In order to create the object model for the application, a Java package structure was first defined as follows.

- edu.csus.ejb.feeserv - fee computation server (Enterprise JavaBean)
- edu.csus.feecomp - fee computation objects (JavaBeans)
- edu.csus.feetest - fee computation test clients
- edu.csus.servlet - fee controller servlet
- edu.csus.util - utility classes

Since the fee computation system is designed as a request/response system, the FeeRequest and FeeResponse classes are defined as composite objects that will hold all of the data objects for input and output respectively. The data objects are implemented as JavaBeans and can be manipulated within the rulebase.

- edu.csus.feecomp.FeeRequest (composite input class)
  - edu.csus.feecomp.FeeTransaction
  - edu.csus.feecomp.FeeVehicle
  - edu.csus.feecomp.FeeOwner
- edu.csus.feecomp.FeeResponse (composite output class)
  - edu.csus.feecomp.FeeResult
  - edu.csus.feecomp.Fee (list of fees)

Java classes are imported into the Advisor project using the Advisor Builder graphical user interface. In order to begin development of the fee computation rulebase, an Advisor project was created using Advisor Builder. When a project is initially defined, it contains no rules or other entities. From this point forward, a project can be manipulated using the graphical interface to add and edit classes, objects, ruleflows, rulesets, rules, functions, variables, and patterns. In addition to the imported classes, it was necessary to define internal classes for use within the rulebase.

4.2. Rulebase Implementation

The Blaze Advisor rules engine development environment uses its own proprietary programming language called Structured Rule Language (SRL) to define the rulebase. The Advisor Builder tool allows the rules engineer to work with the rulebase in a graphical manner, which automatically generates the SRL code. The engineer can also work directly with the SRL source code.

A rulebase is implemented within Advisor Builder in much the same manner as the design work discussed in the previous section. First, the order of tasks in the business process is defined as a ruleflow, Rulesets and functions are then implemented for the tasks in the ruleflow. Rulesets are a group of related rules that will be evaluated, scheduled, and fired by the Rete algorithm. The Rete algorithm has been implemented within Blaze Advisor as a set of Java classes which is not explicitly exposed to the users of the product.

When a test of the rulebase is performed (see Section 5), execution begins at the start of the ruleflow. As tasks in the ruleflow are encountered, the implementation of a task is performed. If the task implementation is a function, it is executed in the same manner as a procedural function or sub-routine. If the task implementation is a ruleset, the working memory and pattern network for the Rete algorithm is created with the elements and patterns that exist within the ruleset. The evaluation stage is performed and rules are added to the agenda. After evaluation is complete, a selected rule is fired and the knowledge base is updated. This process continues until the agenda is clear for the current ruleset. Processing then continues with the next task in the ruleflow. When the entire ruleflow has been processed, the results can be displayed to the user or returned to the calling module. Figure 1 illustrates sections of the fee computation ruleflows and rulesets within the Advisor Builder project.

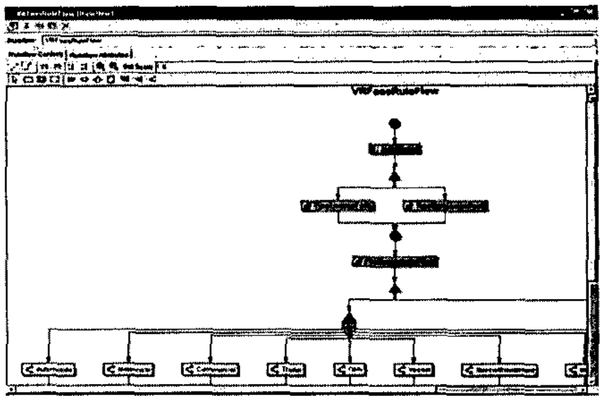

Figure 1. Vehicle Registration Fee Ruleflow.

4.3. Interface Implementation

To integrate the rules engine as part of an enterprise solution, the rulebase must be deployed into a Java application environment. There are a few ways to develop a runtime version of the rulebase and here we choose to deploy the fee computation rulebase as an EJB in a J2EE Application Server. This is done through a package called Advisor Server in the Blaze Advisor suite of tools, which is a deployment runtime system to expose the rulebase to Java application clients and has support for multiple threading and concurrency control through the concept of a "Rule Agent".

5. PERFORMANCE EVALUATION

Three test transactions have been executed each using a different environment (the Advisor Testing environment, the web application client, and the Java application client). The performance evaluation of each test transaction is described in the subsections below.

5.1. Advisor Builder Testing

The Advisor Builder development platform not only provides a graphical user interface for designing and building a rulebase, but also includes the ability to run test transactions, and even a full suite of debugging capabilities that are common with most programming language development tools. This includes defining input test objects, stepping through rules execution, watching term values, setting breakpoints, and printing output to the screen.

The Advisor Builder development platform also provides some reporting and monitoring tools to help determine the correctness, completeness, and performance of the rulebase. Information is generated for every ruleflow, ruleset, function, object, and variable within the project and how they relate to each other. A conflict analysis is also generated to show potential errors within the rulebase. The performance reports will display statistical information for the following static and dynamic categories: number of rules, number of entities considered in the rules, maximum depth of ruleflows, total number of properties tested in the rules, total number of rules fired, and total number of predicates evaluated.

5.2. Web Application Client

The web-based deployment of the fee computation rulebase was the main goal of this project and it has been successfully implemented within Sun Microsystem's J2EE Reference Implementation, a Java application server. The J2EE server was installed to a desktop machine running Microsoft Windows XP. The fee application was installed into the server and a URL was created for public access through the web server. Figure 2 displays the results.

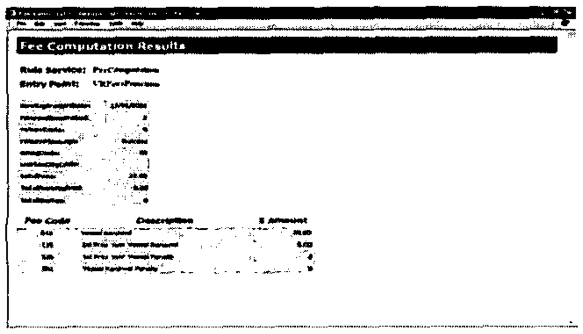

Figure 2. Web Application Output Display.

The performance of the web application has a response time within one second. The rules engine itself consumes two hundred to three hundred milliseconds, and the network traffic can affect the final response time.

There is a startup cost associated with the rules engine in the form of loading external and internal table information and setting up the Rete network. This generally takes about fifteen seconds during the first invocation of the EJB. Once the EJB is initialized, it remains in memory for the EJB container to process future requests.

5.3. Java Application Client

The Java application client was an extra implementation component for this project that shows the advantages of deploying an application as an EJB. The web application provides a user interface to the rules application, while the Java application provides a batch or program interface to the same rules application. This allows the deployment of a business solution to be accessible and consistent through many different means of communication. To demonstrate this type of client, a Java class was written that would read the request data from an external file, invoke the rules application, and write the response data back to an output file.

The Java client can achieve a better performance than the web application, since it performs a direct access to the EJB and bypasses the FeeController servlet. Again, this client application could be used to interface with other areas of an existing system and provide access to the same rulebase that is used over the Internet.

6. COMPARISON TO JESS IMPLEMENTATION

As part of the study, we compare the Blaze Advisor rules engine implementation with the JESS implementation in the following three aspects: rule base development, interface development, and testing and deployment. Through the comparison, we obtained some useful conclusions. The following subsections summarize some of the results. Interested readers may refer to [35] for details.

6.1. Rulebase Development

Since both Blaze Advisor and JESS are Java-based implementations of the Rete Algorithm, the interfaces and deployment possibilities are similar. However, the rules syntax for the Advisor is based on their own proprietary language (SRL), while JESS accepts the syntax of the CLIPS expert system shell. These two languages are actually very different in their approach to representing declarative rules and asserting/retracting facts as the inference engine executes its processes.

One of the issues to address in the translation from the Advisor to JESS was to determine how external Java objects would be integrated or interfaced with the expert system. Java classes that are defined outside of the rules, are imported into the Advisor Builder using features within the tool. Once imported, their public members may then be referenced directly within the rules in the same fashion that internal classes and objects are referenced.

For JESS, the process for linking external classes into the expert system is called "binding". A local or global variable is bound to a class instance and all class members are available using specially defined constructs for JESS. However, in order to use these objects within the rulebase in the same manner as other objects, variables, and atoms, a special construct called "definstance" must be used, which will generate a template for a fact within the knowledge base that links the external object to the fact. This is the only way to gain the ability to use members of an object in the antecedent of a rule.

Another important feature of the Advisor rulebase that needed to be converted to JESS, was the creation of internal objects that assist with fee computation. In particular, the FeeTable is an object that stores fee amounts, descriptions, and effective dates for the specific fee codes used by vehicle registration. During the process of fee computation, if a certain fee qualifies for assignment, the proper amount must be determined from the FeeTable. For the Advisor implementation, this table was easily setup as an association table, which is similar to a Java hash table. When a fee code qualifies for assignment, a function is used to examine the FeeTable object and return the appropriate fee. For the JESS implementation, it was easier to define the entire table as a set of rules that are pattern matched by fee code. The appropriate FeeTable entry is found by using the inference engine, instead of a function call.

6.2. Interface Development

Once again, with both the Advisor and JESS being Java-based implementations of the Rete Algorithm, the interfaces to the rules are very similar, and in fact, identical in places. For the Advisor implementation, two application interfaces were developed as mentioned in Section 5. For the JESS implementation, the HTML pages, the Java Servlet, and the Java client use the same source code. Only the business method of the EJB requires modification for the JESS rulebase execution.

Advisor Builder supports the ability to generate much of the deployment code necessary for installation of the rulebase as an EJB within a J2EE application server.

The loading of the Advisor rules file is specified within the deployment descriptor and is automatically loaded through the environment variable `ndconfig.server`. However, this functionality needed to be explicitly created for JESS, by loading the rules files during initialization of the bean using the "bload" construct.

6.3. Testing and Deployment

Some final conclusions on the comparison between the Advisor and JESS involve the ease of testing and deployment using the Advisor development environment versus the manual effort needed to perform the same functions using JESS.

The automated deployment generation that offers many options for Advisor could also be implemented for JESS, however none of it exists today. Businesses require the ability to deploy their solutions to varying platforms and tier structures and with different interface requirements. Some additional work could be pursued to provide these different deployment options as a standard set of library classes within the JESS package. This would allow for a quicker implementation of tested methods for loading rulebase files, interacting with XML data, creation of Stateful and Stateless session beans, and integration with standard messaging API's. In fact, all of the standard J2EE interfaces could be offered along with the newly proposed JSR-94 specification for a standard rules engine API.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we describe our experience in capturing the business logic and implementing a web-based expert system for vehicle registration fee computation at California Department of Motor Vehicles. The system has been implemented using the Blaze Advisor rules engine and has been deployed to Sun's J2EE Reference Implementation of a Java application server. Our results have shown that using expert system technology to implement a web-based, business-driven solution is certainly a viable and promising option. Deployment of the rule service as an EJB exposes the rulebase to various types of business clients, and provides a consistent and adaptable implementation of the business domain.

There are several lessons learned. First, the methodology of discovering, defining, and organizing the business rules became extremely important. By not applying a business rule approach, it is possible to implement a rulebase that is not easily adaptable to a changing business world and will not accurately represent the true business domain. By following a formal business-driven methodology, the rulebase organization was easier to develop and the rules were easily traced to existing business documentation. Reducing the time spent translating the legislation and policies that govern a business into the automated system that runs the business, is a key benefit of creating an effective rule-based system. This will allow a company to rapidly respond to changing business requirements and maintain a competitive edge in the marketplace.

The second lesson is that a new set of rules engineering skills is needed for business analysts and technical specialists working in the field of web-based business solutions through expert system technology. Such skills include: a mastery of the English (or chosen) language, the ability to translate business documentation into sets of well-defined rules, the ability to reduce rules to their atomic state (singular purpose or constraint), communication skills with both business and technical personnel, and an understanding of declarative programming paradigm.

Lesson three is the need of tools and development environments that are geared toward multiple communication or interface strategies and for legacy systems integration. Tools need to be graphical or model-driven, have complete testing and debugging environments and conflict and performance analysis facilities, support automated deployment, and have the ability to generate rules maintenance applications that can be exposed to business users over the Internet.

Lesson four is the creation of a process for drafting, defining, and accepting formal standards and specifications. The success of Open Systems Architectures (OSA) relies heavily on the participation of industry leading experts to propose and refine standards and to ensure that the vendor population will provide compliant solutions.

Possible future work includes: a system for complete vehicle registration. This project focused on a sub-section (vessels) of the complete business application for generating the appropriate fees for vehicle registration. A natural extension would be to include developing the rulebase for all vehicle types (automobiles, motorcycles, commercial vehicles, trailers, off highway vehicles, and special equipment).

Some more general issues for future work include: development of business rule maintenance capability that is accessible through the Internet, a standard XML schema for business rule representation, and a web-based "rule service" that receives both a rulebase definition and a rulebase query, and returns the results of the service back to the user, over the Internet.

REFERENCES

[1] Rahim Adatia, et al. *Professional EJB*, Wrox Press Ltd., Birmingham, AL, 2001.

[2] Scott Ambler, Tyler Jewell, and Ed Roman. *Mastering Enterprise JavaBeans*, John Wiley & Sons, Inc., New York, NY, 2002.

[3] Danny Ayers, et al. *Professional Java Server Programming*. Wrox Press Ltd., Birmingham, AL, 1999.

[4] Kurt Cagle, et al. *Professional XSL*, Wrox Press Ltd., Birmingham, AL, 2001.

[5] Andrea Chiarelli, et al. *Professional JavaScript*, Wrox Press Ltd., Birmingham, AL, 1999.

[6] Joseph Giarratano and Gary Riley. *Expert Systems: Principles and Programming*, PWS Publishing Company, Boston, MA, 1994.

[7] Tom Myers and Alexander Nakhimovsky. *Professional Java XML Programming*, Wrox Press Ltd., Birmingham, AL, 1999.

[8] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1995.

[9] Ronald G. Ross, *Business Rule Concepts*, Business Rules Solutions, LLC, Houston, TX, 1998.

[10] Ronald G. Ross, *Principles of the Business Rules Approach*, Addison-Wesley, Boston, MA, 2003.

[11] Barbara von Halle, *Business Rules Applied*, John Wiley & Sons, Inc., New York, NY, 2002.

[12] Carole Ann Berlioz and Colleen McClintock. "Implementing Business Rules in Java: Part 1" *Java Developers Journal* May 2000, pp 8-16.

[13] Charles L. Forgy, "Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Matching Problem" *Artificial Intelligence* 19 1982, pp 17-37.

[14] Colleen McClintock, and Chris Roberts. "Implementing Business Rules in Java: Part 2" *Java Developers Journal* July 2000, pp 8-14.

[15] Ken Molay, "Business Rule Representations of Java Objects" *Java Developers Journal* Feb. 2001, pp 62-68.

[16] Steve Ross-Talbot, "Java Rules in J2EE" *Java Developers Journal* Sept. 2001, pp 12-16.

[17] Ernest J. Friedman-Hill, "Jess, The Rule Engine for the Java Platform" Online posting. 9 Apr. 2003. Sandia National Laboratories. http://herzberg.ca.sandia.gov/jess/docs/61

[18] Benjamin Grosof, et al. "Business Rules for Electronic Commerce" Online posting. 1999. IBM Research Project at IBM T. J. Watson Research. http://www.research.ibm.com/rules/home.html

[19] Benjamin Grosof, et al. "IBM Release CommonRules 1.0: Business Rules for the Web" Online posting. 1999. Overview of IBM CommonRules 1.0. http://www.research.ibm.com/rules/commonrules-overview.html

[20] Art Moore, et al. "Discovering Rules Through Business Rule Mining" Online posting. 2001. Knowledge Partners, Inc. http://www.kpiusa.com/BRBook/BusinessRules.htm

[21] Margaret Thorpe, "Business Rules Exchange - The Next XML Wave" Online posting. May 2001. XML Europe 2001. http://www.gca.org/papers/xmleurope2001/papers/html/s15-2.html

[22] Margaret Thorpe, "Simple Rule Markup Language" Online posting. 17 May 2001. Cover Pages: Hosted by Oasis. http://xml.coverpages.org/srml.html

[23] Business Rules Group. http://www.businessrulesgroup.org/brghome.htm

[24] Business Rules Journal. http://www.brcommunity.com

[25] Business Rules Solutions, LLC. http://www.brsolutions.com

[26] Fair Isaac, Inc. http://www.fairisaac.com

[27] GUIDE International Corporation. http://www.cbi.umn.edu/collections/inv/guide84.htm

[28] ILOG, Inc. http://www.ilog.com

[29] Java Community Process. http://www.jcp.org/en/home/index

[30] Knowledge Partners, Inc. http://www.kpiusa.com

[31] Object Management Group. http://www.omg.org

[32] The Rule Markup Initiative. http://www.dfki.uni-kl.de/ruleml

[33] Sandia National Laboratories, Java Expert System Shell. http://herzberg.ca.sandia.gov/jess

[34] Sun Microsystems, Java Technology. http://java.sun.com

[35] Alan Demmin, *A Web-based Expert System for Vehicle Registration*, Mater degree project, Department of Computer Science, California State University, Sacramento, May 2003.