

# An Enterprise Intelligent System Development and Solution Framework

Rajendra M. Sonar

**Abstract**—The recent trend has been using hybrid approach rather than using a single intelligent technique to solve the problems. In this paper, we describe and discuss a framework to develop enterprise solutions that are backed by intelligent techniques. The framework not only uses intelligent techniques themselves but it is a complete environment that includes various interfaces and components to develop the intelligent solutions. The framework is completely Web-based and uses XML extensively. It can work like shared plat-form to be accessed by multiple developers, users and decision makers.

**Keywords**—Intelligent System Development Framework, Web-based Intelligent Systems, Retail Banking.

## I. INTRODUCTION

THERE are various tools, shells, architectures, models, and methodologies have been suggested, developed and implemented to develop hybrid intelligent systems [1-9]. The framework described here has been advanced version discussed in [8] to develop enterprise level business solutions that are backed by or enhanced with hybrid intelligent techniques. It uses expert system [10], case-based reasoning [11] and neural network [12] and their hybrids as core intelligent techniques. The major software components and interfaces of the framework has been implemented using C# on ASP.NET [13] environment. The following sections elaborate more on this framework at higher level rather mechanics at detailed level like how individual components work and how systems are integrated.

## II. THE FRAMEWORK

As shown in Fig. 1, the framework has been divided into four layers depending upon functionality: 1. Database Access Layer, 2. Intelligent Systems Layer, 3. Presentation Transformation Layer, and, 4. Session layer. There are various components and interfaces in each layer. The components implement core functionality while interfaces provide interactive environment to configure and manage the components and models. For example, the inference engine is implemented as a component: Expert System Engine, while the interface: Rule Manager facilitates to retrieve, build,

modify and store rule-based expert systems. The database layer does the job of extracting, mapping, transforming and manipulating the data from various databases. The second layer plays the role of business intelligence and implements intelligent techniques: expert system, case-based reasoning and neural network. It has interfaces to configure and model the systems and manage various intelligent system sessions. The third layer acts like a middle tier between presentation services and business intelligence layer. The primary functions include transforming the data into HTML (Hyper-Text Markup Language) [14] format and facilitating user input. It consists of components and interfaces to build various input templates, HTML reports and so on. The fourth layer includes components and interfaces that are common across three layers. It holds repository and definitions of global variable objects logically put together into groups that are common and referenced across component and interfaces in other layers. For example, *Customer* can be a logical group indicating customer variable objects like *Customer.Name*, *Customer.Age*, etc. The *Customer.Age* can be used as a feature in CBR system or fact in expert system Similarly, Common Session Data holds the data at run-time to be accessed and manipulated by intelligent systems, database and presentation layers.

The framework uses a common notation *Case* and attribute-value XML (eXtensible Markup Language) [15] format to represent a database record (retrieved or to be updated), an expert system input and output session, an example in neural network as well as a case in CBR [8]. Making it uniform data structure for data access and manipulation across all layers.

### A. Database Access Layer

Components in this layer help to maintain various database connections simultaneously. The core engine converts SQL (Structured Query Language) responses into XML and XML data into SQL queries. It can fetch the data from more than one database and update also. The queries can be clubbed in batch. Data can be integrated or merged from heterogeneous databases. Integration means the same type of data (structurally) is coming from different sources. For example, a typical decision making application on higher level may need unified view of data from different databases located at different branches. Merge option is used when data is spread across different databases. For example, customer profile is stored in different databases: personal information in one

Rajendra M. Sonar is with Indian Institute of Technology Bombay, Powai, Mumbai-400076, India in Shailesh J Mehta School of Management (e-mail: rm\_sonar@iitb.ac.in).

database and transactions in another database. It is join operation across databases. The field names are mapped to variable names and data can also be transformed if required. The Variable-Database Mapping Interface is used to set mapping and transformation options.

The interface SQL Builder helps to build and test the database queries interactively. Dynamic Query Interface is used to define various database queries to fetch the required data at run-time during execution without writing the code explicitly in the business logic in expert system. The dynamic queries can bring a piece of data or data sets in batch from the database to be manipulated. This saves writing explicit database code inside expert system. Filters are used to access or restrict to only required data. The expert system engine populates variable values at run time into filter to filter the dynamic query results. For example, a query can be defined to populate customer profile into session data by using a filter that takes *Customer ID* as an input at run time. This layer also plays very important role in setting and controlling access rights to access the data. The framework enables to develop predefined set of queries for accessing and manipulating data at run time by various components. The access rights are

defined on the following parameters: *Logical Database ID*, *User Role*, *Intelligent System Type (ES/ANN/CBR)*, *Intelligent System ID*, *Application ID* and *Type of Access* (read only/update). It can be further managed for individual user access, depending upon certain criteria etc. at the application at run-time. This is done using query filter that may contains system variables like *User ID*. This ensures that a user accesses and manipulates only authenticated and permitted data.

*B. Intelligent Systems Layer*

This layer has rule-based expert system engine, CBR engine and implementation of multilayer feed-forward network with error-backpropagation learning algorithm [12]. The expert system engine supports backward as well as forward reasoning. The CBR engine supports structural and conversational CBR applications [11]. All intelligent systems use XML for input, output as well storing rules, case-schemas, neural network configuration etc. The XML schemas are simple ones mapping each structure into appropriate format as shown in Fig. 2 (a), (b), (c) and (d).

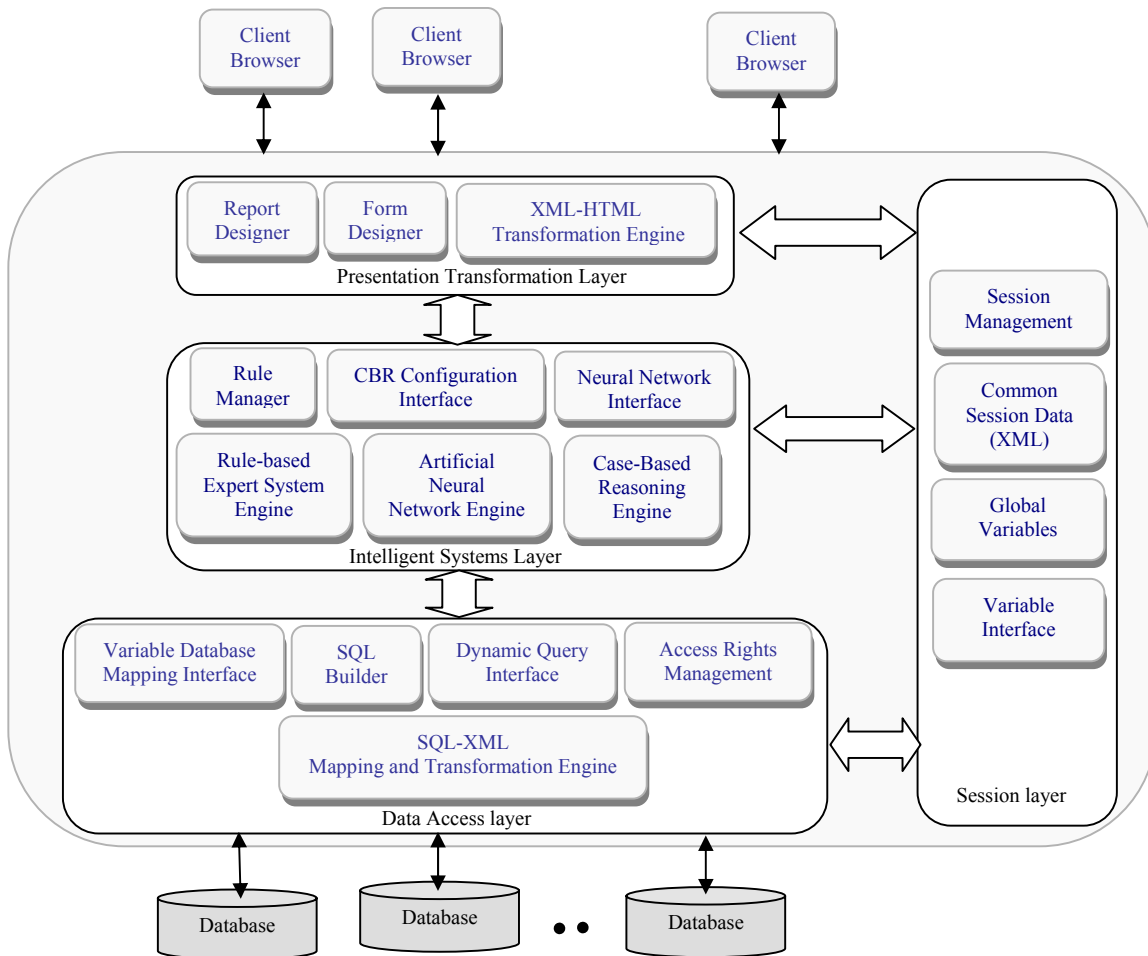


Fig. 1 Hybrid Framework

At run-time, these systems take input and output through Common Session Data, which acts as a sharing mechanism. The session data is hold in XML format and follows common case format so that components can exchange the data amongst them. Each system can work independently or can work in integrated mode. In independent mode, system can take input from session and can send output to session again. The systems can be integrated in modular way or even in host-assistant mode. For example, expert system can act as host controlling execution of CBR and neural network. Various interfaces help to manage and configure intelligent systems. For example, Rule Manager is used to manage the rules. It helps to build and modify the rules, validate syntax and semantics the rule and so on. Similarly, Case Configuration Interface helps to case matching similarity logic.

```
- <ANN>
  <ANN_ID>Product_Classifier</ANN_ID>
  <FileName>ClassAnn.xml</FileName>
  - <Variables Type="Input">
    <Variable>Customer.Age</Variable>
    <Variable>Customer.Income Group</Variable>
  </Variables>
  - <Variables Type="Output">
    <Variable>Customer.Product Bought</Variable>
  </Variables>
</ANN>
- <ANN>
```

(a) Partial ANN XML structure

```
- <ANN>
  - <Layers NoOfLayers="3">
    - <Layer No="0">
      <NoOfNeurons>2</NoOfNeurons>
      <TrFunction>L</TrFunction>
    </Layer>
    - <Layer No="1">
      <NoOfNeurons>4</NoOfNeurons>
      <TrFunction>B</TrFunction>
      <TrFunctionConstant>4</TrFunctionConstant>
    </Layer>
    - <Layer No="2">
      <NoOfNeurons>1</NoOfNeurons>
      <TrFunction>L</TrFunction>
    </Layer>
  </Layers>
  - <Matrices Type="Weight">
    - <MAT No="1">
      - <Matrix>
        <Rows>4</Rows>
        <Cols>2</Cols>
        - <Row Label="0">
          <Col No="0">-0.327070767358834</Col>
        </Row Label="0">
      </Matrix>
    </MAT No="1">
  </Matrices Type="Weight">
</ANN>
```

(b) Partial ANN XML structure

```
- <CaseSchema>
  <Case_ID>Credit Case</Case_ID>
  <CBRTType>Structural</CBRTType>
  - <Options>
    <NumberOfMatchingCases>10</NumberOfMatchingCases>
  </Options>
  - <CaseElements>
    - <CaseElement>
      <Variable>Customer.DebtToIncomeRatio</Variable>
    </CaseElement>
  </CaseElements>
  - <Types>
    <Type>Key</Type>
    <Type>Output</Type>
  </Types>
  <Weight>0.05</Weight>
  <SimilarityMeasure>Numeric</SimilarityMeasure>
  <IndexLevel>0</IndexLevel>
  <MaxCases>25</MaxCases>
</CaseSchema>
```

(c) Partial CBR XML structure

```
- <RULE NO="38" ID="CreditRating.Job Status Good 3">
  <IF>Customer.Has Job IS Yes AND Customer.Experience >= 0 AND
  Customer.Occupation IS [Salaried: Govt,Salaried: Public Sector,Salaried:
  Multinational] AND GET_CODE(Customer.Designation) IS
  [Clerical,Supervisory] AND Customer.Job Type IS "Term Contract-5 Years"
  AND Customer.Loan Period <= 5</IF>
  <THEN>CreditRating.Job Status IS Good</THEN>
</RULE>
- <RULE NO="39" ID="CreditRating.Job Status Good 4">
  <IF>Customer.Has Job IS Yes AND Customer.Experience >= 10 AND
  Customer.Occupation IS [Salaried: Private Limited,Salaried: Small Firm] AND
  GET_CODE(Customer.Designation) IS [Clerical,Supervisory]</IF>
  <THEN>CreditRating.Job Status IS Good</THEN>
</RULE>
```

(d) Sample Rule in XML format

Fig. 2 XML Formats for Intelligent Techniques

C. Presentation Transformation Layer

Traditionally expert systems have been used to work in Question & Answer (Q&A) mode. This may not appropriate when the number of user input required for a particular application are large. On the other hand, if lots of required user inputs are put in one data entry form, it may be crowded and lot of logically unrelated pieces may be put together. Along with Q&A, the framework facilitates to have applications comprised of small modular forms, each one containing inputs that are logically related. Similarly, reports can also be developed in modular way. This layer takes care of presentation of output by converting XML data into HTML form. Similarly it converts data sent by Internet browser into XML.

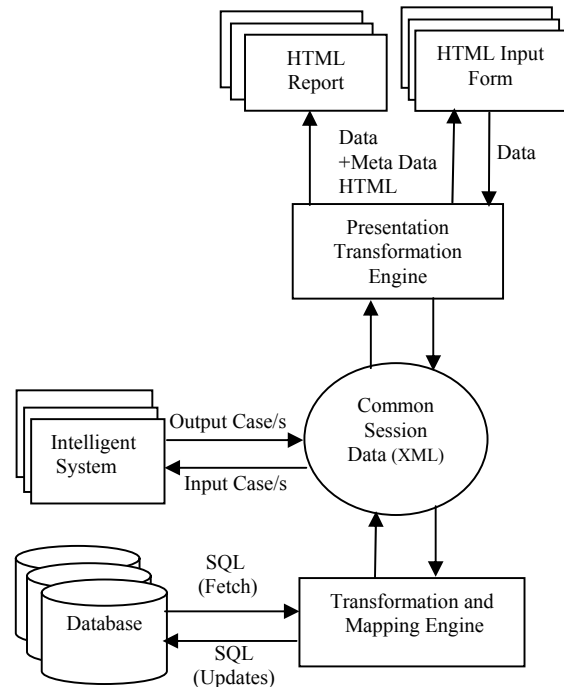


Fig. 3 Sharing Mechanism through Session Data

The interfaces: Report Designer and Form Designer help to design various formatted output reports and input form templates to populate, enter and validate data. Output reports include SQL results, most matching cases, expert system output or neural network data and results. Form Designer is used to develop modular web forms which can be invoked using expert system either to enter problem case for CBR or

input for expert system itself. These forms are HTML based and contain JavaScript code to control validation, etc. For example, a form to get personal customer information, a form to get job and salary related parameters, etc. The forms and reports can be invoked through expert system logic whenever required or asked or depending upon certain criteria. For example, if a customer holds a credit card, then input form asking for credit card details can be invoked other wise some other form depending upon payment type can be invoked. The Report Designer is used to generate various report templates which are used to view HTML output on browser. The report templates are of different types and contain JavaScript code to populate into the template, manipulate and format the data sent at client browser by the server components.

**D. Session Layer**

Global Variables is the repository of variable objects (like data dictionary) that are used across various components and interfaces. Each variable object describes its ID, data type, type of variable and so on. Variable Interface helps to manage the variables. This layer manages the session with users. It holds the data to be shared by all the other three layers as

shown in Fig. 3. For example, whenever a database query is fired, it populates data into session (the format is shown in Fig. 4). This session data can be populated into the input form for editing or modifying or into the report to be displayed at client browser. This makes it easy to exchange information without writing code fetch data from databases and populate it into the variables explicitly. Similarly update the data into the databases from the variables. The framework takes into consideration access rights while fetching and updating.

```

- <Case No="5">
- <CaseData>
- <Element>
  <Name>Customer.Age</Name>
  <Value>35</Value>
</Element>
- <Element>
  <Name>Customer.ID</Name>
  <Value>E2344</Value>
</Element>
    
```

Fig. 4 Common Attribute-Value Format

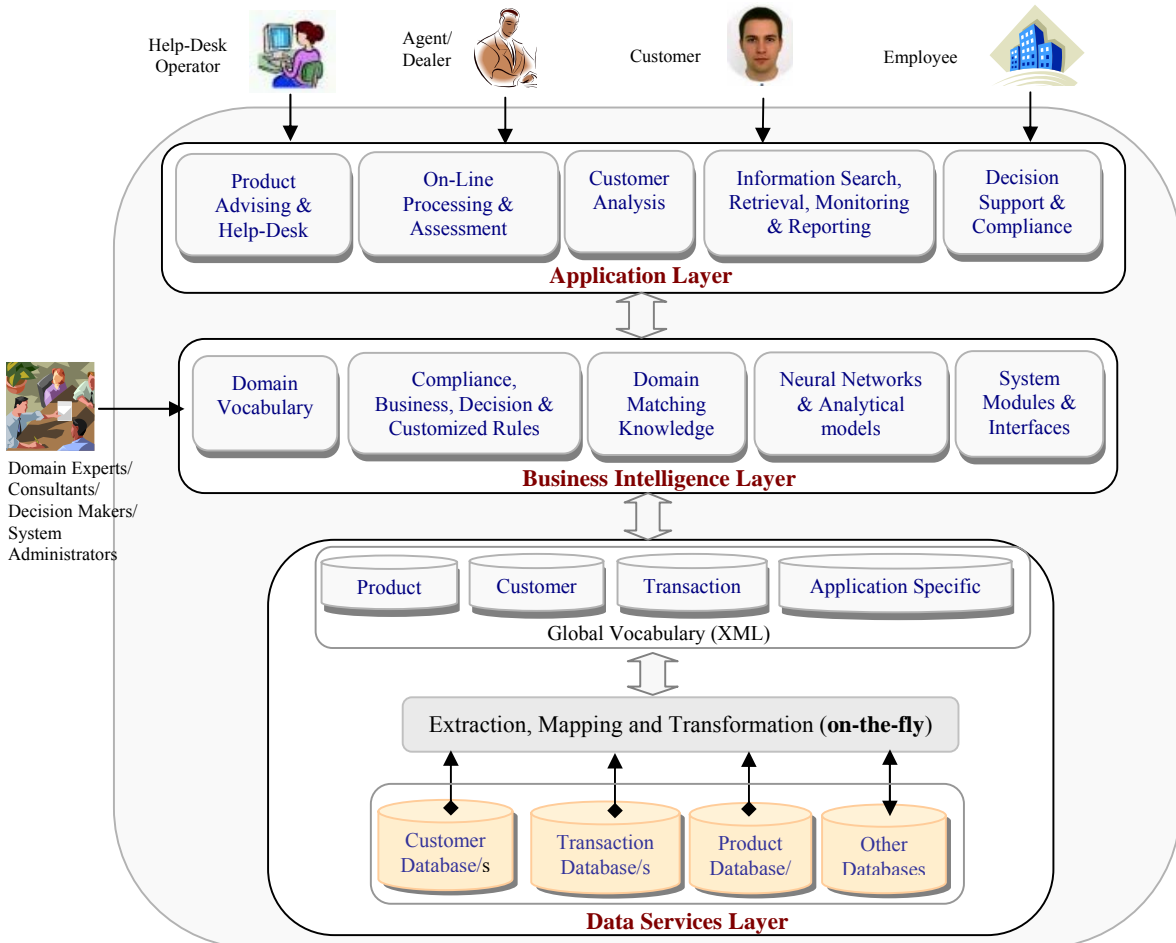


Fig. 5 Generic Solution Framework

### III. SOLUTION FRAMEWORK

The framework discussed in the previous section has been used to address practical solutions in various domain areas. Since the framework integrates various intelligent techniques, it can address wide-range of generic applications. As shown in Fig. 5, the solution framework is divided into three layers. 1. Data Services Layer: it represents various databases connected to store and manipulate data, and, mapping and transformation logic. 2. Business Intelligence Layer: this layer represents various models, set of rules, matching knowledge used by intelligent techniques while solving the problems. 3. Application Layer: these are various applications that use or enhanced with expert systems, case-based reasoning systems, neural networks in modular way or hybrids of them. The following sub-sections describe the solution framework.

#### A. Data Services Layer

In practical customer oriented business applications, commonly used data includes detailed customer profiles, transactions, product information and so on. It is possible to model entities or data objects such as customer and transaction on a common generic model that can be used in many applications. Like a typical customer profile includes various facts or attributes about the customer such as Name, Address, Age, Sex, etc. The generic transaction data object includes transaction attributes like Date, Time, Amount, Type, Location, etc. Global Vocabulary contains logical descriptions and details about each such attribute that includes name of the attribute, data type, valid values etc. The Global Vocabulary is independent of database schema. The Database Access Layer of the framework maps the database schema to Global Vocabulary. This helps to develop solutions that are not dependent of database schema. The intelligent systems and other components use Global Vocabulary rather than database fields themselves.

Global Vocabulary itself logically divided into various groups like Customer, Transaction, etc. Logical grouping helps to define and manage access rights to user, roles etc. as well as defining mapping and transformation logic.

#### B. Business Intelligence Layer

This layer has various intelligent techniques. Depending upon the applications addressed various intelligent systems can be developed. There can be quite a number of intelligent systems (expert systems, CBR-schemas and neural networks) and logically separated into application groups like Customer Analysis, Credit Rating etc. Each intelligent system has a unique identity called Intelligent System ID, and used to operate on or execute that system, for example RUN\_CBR(Credit Case) would run CBR whose ID is Credit Case. This brings modularity in developing intelligent systems as well as user access (including role-based access) rights can be set to access, change intelligent systems. For example, if the Branch Manager logs in, he/she is supposed to access only Customer Analysis application, other users may not access to

this application. The models, case-schema, rule-base are stored in XML formats as discussed earlier.

#### C. Application Layer

Depending upon type of solutions various applications can be developed that would be part of a solution. For example, Retail Banking [16] Solution contains suit of applications that are used for Loan Approval, Credit Rating, Transaction Monitoring, Product Help Desk, etc.

Similar to generic data models like customer, transaction, there can be generic applications like Customer Analysis, Monitoring and Retrieval, Searching, etc. These applications use or invoke various intelligent systems from Business Intelligence Layer. For example, Credit Rating application may invoke expert system, neural network and case-based system in modular way. Expert systems would detect inconsistencies in data provided and do the qualitative evaluation of customer. Neural network would do credit scoring and CBR system would match profile of the customer with past ones and would determine likely behaviour or default probability. Results would be combined to conclude.

Applications can be accessed at various levels, by internal as well as external people. For example, a Help-Desk application can be accessed by the help desk operators sitting at Call Center. While, Customer Analysis application is accessed by the branch manager who's job is to assess the customer.

### IV. PROTOTYPE SOLUTION

There are various solutions that can be developed using the framework. We are developing a prototype solution for Retail Banking being implemented in one of the public sector banks in India.

The typical retail banking functions other than transaction support, have been broadly classified into following broad areas: a) Customer Analysis: understanding and analyzing customer well, their preferences; profiling and segmentation; looking for right marketing, cross-selling opportunities and up-selling opportunities to right people and so on. b) Decision Support: evaluating customer, credit rating and scoring etc. c) Help-Desk and Self-Service: providing help to customer on products and services; facilitating them to explore information such as statements, past transactions and so on. d) Retrieval, Monitoring and Reporting: continuously monitoring the activities to detect suspicious and abnormal activities, reporting various exceptions, and, sending alerts and reminders; managers and decision makers can access required information quickly and easily.

One of the components of retail banking solution contains end-to-end Retail Loan Processing (especially loans for housing, vehicle, personal, mortgage, education): that includes on-line form submission and assessment, credit rating and approval, monitoring and reporting. The solution would be centralized can be accessed by multiple branches as well customers on-line. Fig. 6 shows an example application using

the modular hybrid approach to Credit Rating. The combined approach using expert system, neural network and case-based reasoning helps in better assessment of customers and increases reliability compared to using only a neural network or expert system approach.

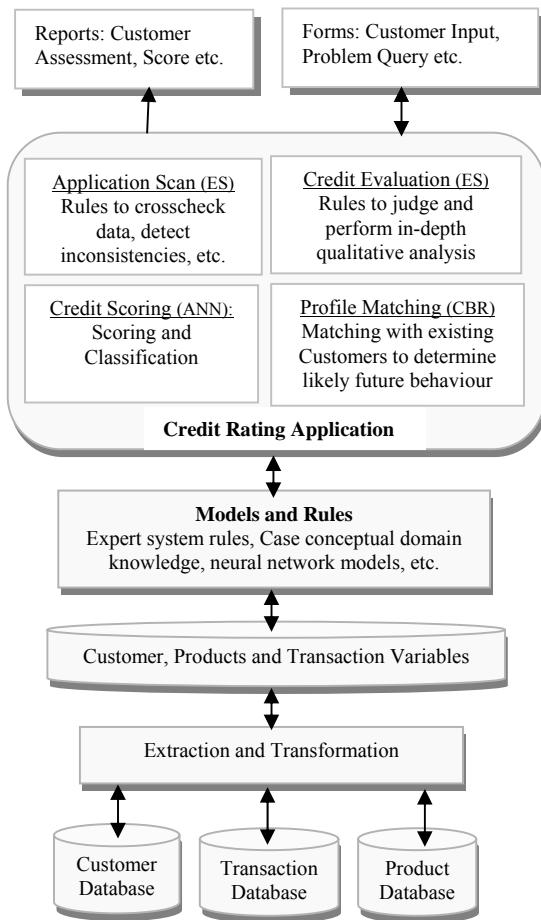


Fig. 6 Hybrid Approach to Credit Rating

## V. DISCUSSION AND CONCLUSION

The framework discussed here shows a layered approach and logical organization of components and interfaces to develop solutions based on hybrid intelligent systems. Use of XML makes all components including database independent of each other. This framework can be used to develop complete solutions that include user interfaces and database access and that are backed by or enhanced hybrid intelligence.

The solution framework suggests a generic way of addressing various applications based on intelligent systems. This is possible because of combination of various intelligent systems enables to solve different types of problems right from product advisory systems to decision support. Either these applications use individual intelligent technique in modular way or solve the problem using integrated approach. Rule-based expert system component includes various functions to control execution of CBR and neural networks

thereby making it possible to develop hybrid intelligent systems in different modes.

This kind of generic framework can easily be customized to address specific solutions based on generic ones because it is modeled on more logical way rather than implementation specific way. The framework is completely web-based. The variables (global vocabulary), intelligent systems and applications are logically separated into groups. This makes it possible to have greater user-based and role-based access control on accessing variables, applications, data etc. The framework can be shared and used, to develop as well as to use the solutions by multiple developers; decision makers and users concurrently over the Internet/Intranet or it can work like Application Service Provider model. Because of built-in extraction, mapping and transformation layer, the applications developed using this framework can be connected to (or on top of) existing databases without any need to restructure or redesign the databases.

## REFERENCES

- [1] Larry Medsker, *Hybrid intelligent systems*, Kluwer Academic Publishers, Boston, 1995.
- [2] Suran Goonatilake and Sukhdev Khebbal, Ed. *Intelligent Hybrid System*, John Wiley and Sons, 1995.
- [3] Jim Prentzas and Ioannis Hatzilygeroudis, "Integrating hybrid rule-based with case-based reasoning", *Advances in case-based reasoning*, ECCBR 2002, pp. 336-349.
- [4] S. Butakov and D. Rubtsov, "Development of hybrid expert system shell automation", *Control, and Information Technology*, ACTA Press, 2002.
- [5] A.V. Gavrilov and N.A. Chistyakov, "An Architecture of the Toolkit for Development of Hybrid Expert Systems", *Automation, Control, and Information Technology*, ACTA Press, 2005.
- [6] Melanie Hilario, Christian Pellegrini and Frederic Alexandre, "Modular integration of connectionist and symbolic processing in knowledge-based systems", *International Symposium on Integrating Knowledge and Neural Heuristics*, Florida, 1994, pp.123-132.
- [7] David Leak, "Combining rules and cases to learn case adaptation", <http://www.cs.indiana.edu/~leake/papers/p-95-09.html>.
- [8] R.M. Sonar, "A Web-based Hybrid Intelligent System Framework", *Intelligent Systems and Control*, ACTA Press, 2004, pp. 254-259.
- [9] R. M. Sonar and A. Saha, "An integration framework to develop modular hybrid intelligent systems", *Frontiers in Artificial Intelligence and Applications*, 69, IOS Press, 2001, pp.1499-1506.
- [10] E. A. Feigenbaum, "The art of artificial intelligence: Themes and case studies of knowledge engineering", *In Proc. of the 5th IJCAI*, Cambridge, MA, 1977, pp. 1014-1029.
- [11] J. L. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann, San Mateo, CA, 1993.
- [12] D.E. Rumelhart, G.E. Hinton and R.H. Williams, "Learning representations by back-propagation errors", *Nature*, 323, 1986, pp. 33-36.
- [13] G. Andrew Duthie, *Programming with Microsoft Visual C#® .NET Version 2003 Step By Step*, Microsoft Press, 2003.
- [14] HTML:<http://www.w3.org/MarkUp/>
- [15] XML:<http://www.w3.org/XML/>
- [16] Almeida, R.A., "Retail Banking: A Focus", IBA (Indian Bankers Association) Bulletin: *Special Issue: Retail Banking*, Volume XXV No.11, 2003, pp. 5-8.