Journal of ICT, 2 (2), pp: 65-86

A DISTRIBUTED AGENT-BASED ARCHITECTURE FOR CUSTOMER RELATIONSHIP MANAGEMENT IN E-ENTERPRISES

N. R. Srinivasa Raghavan, and Dnyaneshwar Pawar***
Department of Management Studies
Indian Institute of Science
Bangalore 560 012, INDIA

*Email: *raghavan@mgmt.iisc.ernet.in, **pawar@bharathi.mgmt.iisc.ernet.in*

ABSTRACT

The chief objective of Customer Relationship Management (CRM) is to acquire, retain and increase the profitability and lifetime value of customers. This requires a better understanding of customer needs and expectations. Customer information acquired through transactions with the customer is stored in databases which contain valuable information about customers which could be used to business advantage. Typically, databases in organizations are distributed and as such, the data mining tool for such databases should have a distributed architecture. In this paper, we propose the design and implementation issues of an agent based architecture for data mining of such databases where individual agents communicate using KQML. We have defined certain performatives that have been implemented for this application. In this paper we present the design and implementation of the facilitator agent, the broker, the data mining agents and implementation of the performatives that we have defined for this application. Our implementation is based on Java servelets with KQML as the language used for agent communication. Sample runs on known algorithms like *apriori* are performed and the runs demonstrate the proof of concept for the distributed agent based model.

Keywords: software agents, data mining, customer relationship management, distributed systems, UML.

1.0 INTRODUCTION

Businesses some time ago mostly concentrated upon products that are marketed without much regard to customers. But in the era of E-business, enterprises need to evolve marketing strategies that are more customer centric, wherein they are trying to serve their customers better. Better service can be provided to customers by understanding their needs and expectations. Also the companies need to focus more on a specific *high profile* customer group by selling them more goods or services. This strategy requires that they should be able to identify these customers.

This change in marketing paradigm from product oriented marketing to customer oriented marketing requires a better understanding of customer needs. One way to understand of the customer is to study their buying patterns and their demographic details. Customer data is collected when customers make transactions with the organization. Since the amount of customer information is enormous, and databases are distributed, a special tool is required for the analysis of such information to extract business useful knowledge.

We have designed and implemented such a tool that has a distributed architecture and would be useful in such situations. The tool uses data mining algorithms to extract knowledge about customers from the database. We have designed the system such that the data mining algorithms can be changed or new ones can be added as and when required, with little modification to the existing system. This is primarily due to the object-oriented nature of the design. The system is also scalable with respect to the number of database servers that are included in the system.

2.0 RELEVANT WORK

Extensive research is going on in the area of data mining to find efficient algorithms for solving the problems of data mining more efficiently. The work presented in Agrawal and Srikant (1994) gives an efficient algorithm for discovering association rules in databases. Kargupta et al. (1997) describes use of distributed architecture for mining of text databases. Our work concerns more of the itemsets as the contents of the databases and not just text. There are some algorithms for mining sequential patterns in databases; Agrawal and Srikant (1995) describes one such algorithm.

In Choy et al. (2002), the authors present an intelligent customer supplier relationship management system (ISRMS) using the case based reasoning (CBR) technique to select potential suppliers. Shaw et al. (2002) discuss AgentRAIDER, which is an ongoing research project at Texas Tech University designed to develop a comprehensive architecture for an intelligent information retrieval system with distributed heterogeneous data sources. The system is designed to support intelligent retrieval and integration of information from the Internet. The agents proposed in this paper include intelligent user information agents, query enhancing agents, analysis and synthesis agents, filtering agents, and searching and routing agents. KQML is used as the language for communication. But the paper does not tailor the requirements for CRM.

In Wilcox and Gurau (2003), the authors identify and analyse the main advantages of the Unified Modelling Language for Business Modelling, presenting a general example of Business Modelling for CRM implementation in online retailing companies. The paper presents an information system modeling perspective for enterprise systems relevant to CRM including, establishing CRM criteria, data processing, data collection, data analysis, etc. The paper does not discuss issues related to data mining. Yuan and Chang (2001) present a mixed-initiative synthesized learning approach to better understand customers and to provide clues for improving customer relationships based on different sources of web customer data. The approach is a combination of hierarchical automatic labeling SOM (self organized map), decision tree, cross-class analysis, and human tacit experience. The objective of this approach is to hierarchically segment data sources into clusters, automatically label the features of the clusters, discover the characteristics of normal, defected and possibly defected clusters of customers, and provide clues for gaining customer retention. This article is a data mining approach intended for web-based CRM. Our agent based architecture can accommodate their algorithms and hence represents an attempt at integrating the data gathering and data mining issues.

The article in Liang and Huang (2000) studies how Intelligent Agents can be used to facilitate electronic trading. The authors review the activities and structures of electronic markets. This is followed by an analysis of agents useful for electronic commerce. A three-layer architecture for organizing intelligent agents for ecom is developed. Finally, application of the framework to support ecom and related issues are presented. The paper is oriented more towards E-trading, but represents an application of agents to E-commerce. Finally, the authors in Stolfo et al., (1997) present the description of the system called JAM,

which is a set of learning programs written in Java, and tailored towards fraud and intrusion detection in financial information systems. Their approach however, does not use KQML which is the most popular language for agent communications. Our approach is similar in spirit, but is tailored to CRM and uses KQML.

2.1 Customer Relationship Management

The chief objective of customer relationship management is to acquire, and increase the profitability, and lifetime value of customers. Customer Relationship Management mainly concerns (Kalakota et al., 2003):

- Customer Segmentation,
- Customer Profiling,
- Product Affinity Analysis,
- Cross Selling.

Customer segmentation involves partitioning the customer database into different segments. For example, the partitioning may be done with respect to customer demographic information, customer purchase patterns, etc., so that customer records that have similar attributes are grouped together. Such homogeneous subpopulations can be targeted for specialized treatment. Once the sub-populations have been identified we need to understand what distinguishes one customer segment from the others in the database in terms of their demographic details, so that we can classify new customers into one of the segments depending upon their attributes. Product affinity analysis involves identifying which products tend to sell well together (also known as *Market Basket Analysis*), and which products tend to sell in a sequence which is called *Sequential Pattern Analysis*. By studying the purchase patterns of the products frequently purchased together, one can leverage the existing customer base by selling them more products. This is known as *Cross Selling*.

Data mining is the process of extracting hidden, unknown and useful relationships from a database. Database mining can be considered as the confluence of machine learning techniques and the performance emphasis of database technology (Agrawal et al., 1993). The three major problems in data mining, namely classification, association rule mining and temporal mining can be used to address the CRM issues of segmentation, profiling, etc., which were discussed above.

Our attempt in this paper, is to use object oriented modeling in conjunction with the notion of software agents, to architect a distributed system which can assist data mining tasks. We do not intend to develop a new algorithm, instead, our framework can be used by data mining analysts to deploy their algorithms.

3.0 ARCHITECTURE OF THE SYSTEM

We decided upon the architecture of the system by keeping in mind the following requirements of the system.

- A distributed architecture,
- Data mining algorithms to generate useful knowledge,
- Scalable with respect to the number of database servers,
- Provides a variety of algorithms that has direct use in CRM, and
- Has a web based GUI so that it can be accessed from anywhere.

Since typically the databases in companies are distributed it is natural to have a system to perform data mining operations that has a distributed architecture. Thus the architecture of our system is agent based and distributed. Figure 1 shows the architecture of the system.

Structurally, the system consists of a central agent called the facilitator agent, and individual agents called data mining agents, that are run on different machines. The facilitator agent is responsible for interaction , with users for input and displays output in an appropriate form. The facilitator agent is responsible for controlling the communication among the agents in the system. Thus, it acts as a coordinator. This architecture provides for the *parallel execution* of the input queries given by the users because the processing of the queries is simultaneously happening at the different machines.

The agents running on different machines accept performatives from the facilitator and perform operations according to it, which involves accessing the database, running the requested data mining algorithm, etc. All the agents talk in Knowledge Query and Manipulation Language (KQML). They support special performatives for data mining operations. The Broker agent keeps the database of the services and the addresses of the agents that provide that service. Initially, all the data mining agents *advertise* the services they intend to provide to the Broker which is achieved by sending the KQML performative *advertise* to the Broker.

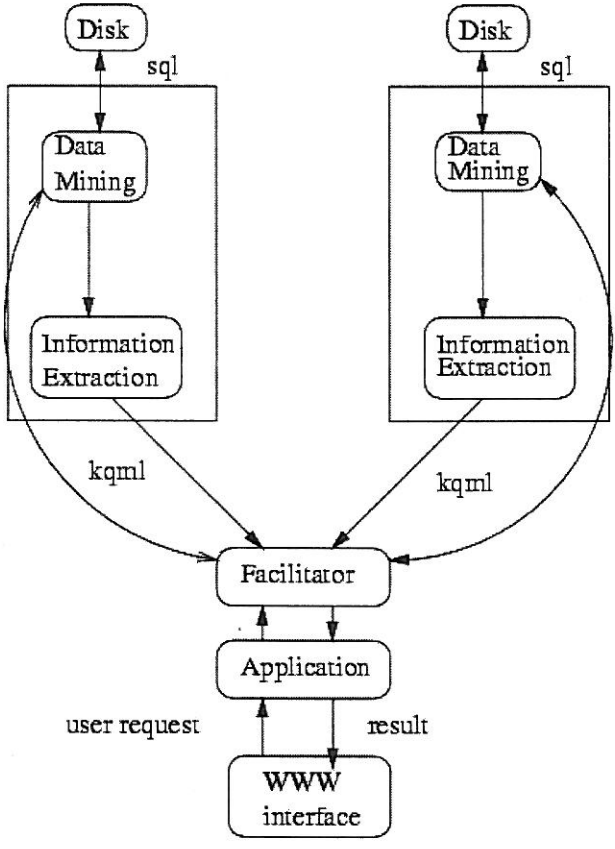

Fig 1: Architecture of the system

The facilitator, while looking for an agent that can provide the requested service, contacts the Broker which gives the addresses of the agents that provide that service. When a new database server is to be added to the system the data mining agents at that server will just *advertise* their services to the Broker, which then gives the address of this agent for the requested services.

The user gives a query to the facilitator that then interprets it and decides the performatives and the parameters that are required to send with the performative so as to satisfy the user's query. The data mining agents always "listen" for any performative from the facilitator and upon receipt of the

performative they process it or give an appropriate error message. The data mining agents when started register the performatives that they want to process with the broker. Thus, when the facilitator wants to send some performatives to the agents, it first contacts the broker to get the name of the agents which can process the performative.

At this point, we mention an important deviation of our approach from traditional data mining. While traditionally, data mining is performed on huge data warehouses, it does not facilitate the mining portions of the warehouse that the user is interested in. Also, we feel that support and confidence factors are not sufficient for CRM applications. For instance, consider a personal computer reseller. Let us assume that mining his transaction databases has resulted in an association rule of the type: motherboard purchases imply hard disk purchases with a support of 0.8 and a confidence of 0.9. This rule makes business sense only if the hard disk transactions have significant dollar value! In our work, we weed out rules without much business value by providing querying mechanism prior to data mining.

3.1 Agent Communication

There is extensive literature available on software agent based systems. The interested reader is referred to Cost, Finin, Labrou, Luan, Peng, Soboroff, Mayfield, and Boughannam (1998). Our system is basically a set of agents working in co-operative manner so as to achieve the common goal of mining the whole distributed database and presenting it to the user. For the smooth working of the agents we need to decide upon a protocol for communication. Moreover the protocol should be a standard which provides inter-oparatibility. For this reason we chose KQML as a language for agent communication.

3.2 Knowledge Query and Manipulation Language

The individual agents in the system communicate in KQML (DARPA, 1993). KQML is a language for programs to communicate *attitude* about information such as querying, stating, believing, requiring, achieving, subscribing and offering. KQML is indifferent to the content of the information itself so that a KQML expression may contain sub-expressions in another language that both communicating agents understand. A KQML message is called performative, in that messages are required to perform some action by virtue of being sent. A KQML performative consists of a string of the form key=value pair, where key is the reserved performative keyword and the value is the value assumed by that

keyword or parameter. The pairs are seperated by a colon. The examples of reserved parameter are in-reply-to, reply-with, sender, receiver, etc.

The types of performatives that are defined in the KQML standard are as follows:

- Basic informative performatives: tell, deny
- Database performatives: insert,delete
- Basic responses: error, sorry
- Basic query performatives: ask-if, ask-about, reply
- Multi-response query performatives: stream-about, stream-all
- Basic effector performatives: achieve
- Generator performatives: ready, next, rest
- Capabality definition performatives: advertise,
- Notification performatives: subscribe
- Networking performatives: register, forword
- Facilitation performatives: broker-one, broker-all

Here is an example of a KQML performative "advertise".
advertise
: *content* < *performative* >
: *languageKQML*
: *ontology* < *word* >
: *sender* < *word* >
:*receiver* < *word* >

This performative tells the receivers about the availability of service defined by the performative in < *performative* > which could be any KQML performative.

4.0 JACKAL: INFRASTRUCTURE FOR KQML AGENTS

All agents talk in KQML and hence they require a communicator facility that will send and receive KQML performatives and provide some additional required services. Jackal (Cost et al., 1998) is such a package that provides methods for agents written in Java, to send and receive KQML performatives and services, and KQML Naming Service (KNS). The basic facilities provided by Jackal are:

- Sending and receiving KQML messages.

- Implements KNS and Addressing.
- Provides conversation based dialogue management.

Jackal is implemented in Java at the University of Maryland, Baltimore. The Jackal's functionality is accessed through a class instance which can be shared among components of an agent. Jackal design is based on KQML Naming Service (KNS), an evolving standard for resolving agent names in hierarchically structured dynamic environment. This means that agents are required only to deal with symbolic agent names and may leave issues such as physical address resolution and alias identification to the Jackal infrastructure. KNS adds a communication layer in which symbolic names are mapped to actual internet addresses. It offers support for dynamic group formation and disbanding, maintenance of persistent and distributed agent identity. KNS specifies protocols for agent addressing and naming, authentication, aliasing and domain registration.

5.0 PERFORMATIVES

We have defined and implemented some performatives that are especially required for the kind of applications we are considering. These performatives are:

5.1 Associate

The following performative for association tells the receiver that he wants to know the association rules from the receiver's database as specified by the expression in the *:content* field. The performative allows the sender to communicate the receiver about its intent for consideration of the subset of the database specified in the content field. This performative also should convey the minimum confidence factor and the support that the association rule must have for the rule to be useful. For precise specification of the rule, an unambiguous syntax for specifying the association rules must be decided upon by the sender and receiver.

associate
:sender < _ *word* >
:receiver < _ *word* >$
:content < _ *data data value minconf*
min conf minsupp min supp >

For example, suppose that the user requests *association* operation that requires *all* association rules with *milk* in antecedent and *butter* in consequent part of the rule with the requirement that the rules should have minimum 0.7 and 0.8 as *confidence* and *support* factor respectively. Then the association performative that the facilitator agent might send for this input is:

:sender <facilitator.ans>
:receiver <dm-agent-ruby.ans>
:content < min conf 0.7
min supp 0.8
ante milk
cons butter >

5.2 Classify

This performative tells the receiver that the sender wants the characteristics of segments specified in Content: field. The characteristics may be the attribute ranges that the receiver will reply with for each segment. For example: the following classify performative tells the receiver to give the characteristics of the segment on customer income groups.

classify
: sender < word >
: receiver < word >
: content < expression_specifying_segments>

5.3 Forecast

This performative tells the receiver to give the forecasted value of the variable specified in the content field at the (future) time specified in the content field.

forecast
: sender < word >
: receiver < word >
: content < variable var_name, time time_value >

The reply to such a performative could be one of the following:

- Reply: The receiver will reply with the reply performative with the string representation of the result of the performative that the sender sent.

- Error: The receiver will reply with an error performative if that performative is not supported or the service cannot be done.

6.0 DESIGN AND IMPLEMENTATION

The system has been designed in UML. We have implemented the above described system in Java. We have identified the following requirements of the system that guided the design process.

- One should be able to add a new database machine to the existing system with little changes.
- The flexibility to change the data mining algorithms implemented.
- The ability to provide results of the queries to the user in an aggregate form.
- The user should be able to select any machine for queries for a particular input.
- The browser should be the user interface to the system.

6.1 Design

The class diagram for a data mining agent is as shown in the Figure 2. All the Data Mining agents have the same design even though they implement different algorithms and provide different services to the facilitator agent. The Intercom class is the gateway of the agent to the outside world since all the communication details are handled by this class. They have a common class named Communicator. This communicator interprets the KQML performatives received from the facilitator agent. This class then instantiates the Algorithms classes depending upon the request received from the facilitator. This class also takes care of sending the results of the queries to the facilitator. The required methods to create the representation of the results of the data mining operations in a form suitable for sending it to the facilitator (in the form of a KQML performative) are declared by the Result interface. The methods declared by the Result interface are to String(), and insert(). The result class for a particular application has to implement this interface so that it can be easily integrated in the system. Figure 3 shows the objects present in the system at the instant a request for the association discovery is being processsed.

6.1.1 Implementation of Facilitator

The facilitator agent takes the input from the user, and then issues the required performatives to the appropriate data mining agents on the specified machines. The facilitator is implemented in Java and it uses Java Servlets for handling the queries from the user through the web. The facilitator has to interpret the queries given by the user and then send the required performatives to the specified machines. The facilitator will aggregate the results returned by the different data mining agents to present them to the user.

6.1.2 Implementation of Facilitator

The facilitator agent takes the input from the user, and then issues the required performatives to the appropriate data mining agents on the specified machines. The facilitator is implemented in Java and it uses Java Servlets for handling the queries from the user through the web. The facilitator has to interpret the queries given by the user and then send the required performatives to the specified machines. The facilitator will aggregate the results returned by the different data mining agents to present them to the user.

For integrating the results of an association discovery operation we implement an algorithm described in Pudi and Haritsa (2002). The sistem, being centrally managed, is suitable for this kind of implementation too. This is because the algorithm described in Pudi and Haritsa (2002) requires more frequent but a structured communication between the facilitator agent and the data mining agents. For example, the facilitator may convey its request of *negative border closure* to the data mining agents by a special performative. Also, the advantage in using that algorithm for this purpose is that the algorithms, that were used by the individual agents may be different without affecting the result of the overall system.

6.1.3 Implementation of Broker

The broker agent is responsible for providing the facilitator the addresses of the data mining agents that provide the services. It is implemented in Java and it maintains a data of the addresses of the machines and the performatives that they implement. This then redirects the *recommend-one* kind of requests from the facilitator to the appropriate machines who provides the services required by the performative requested.

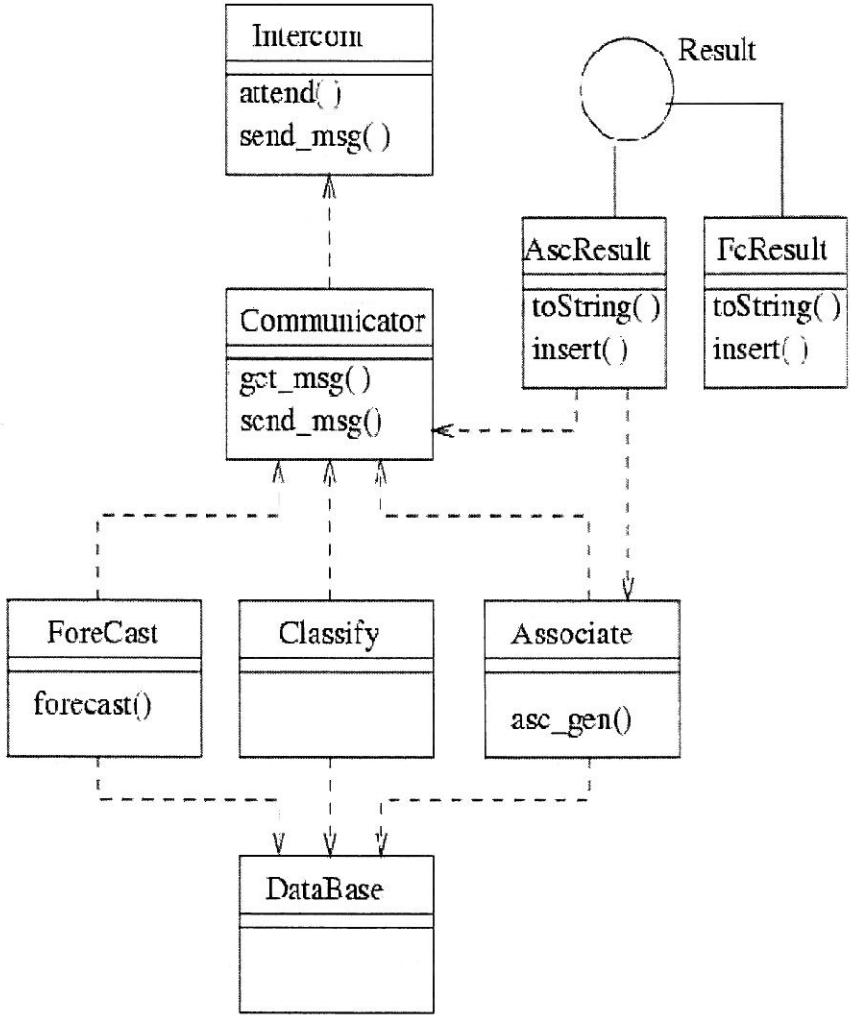

Fig 2: Class diagram of DM agent

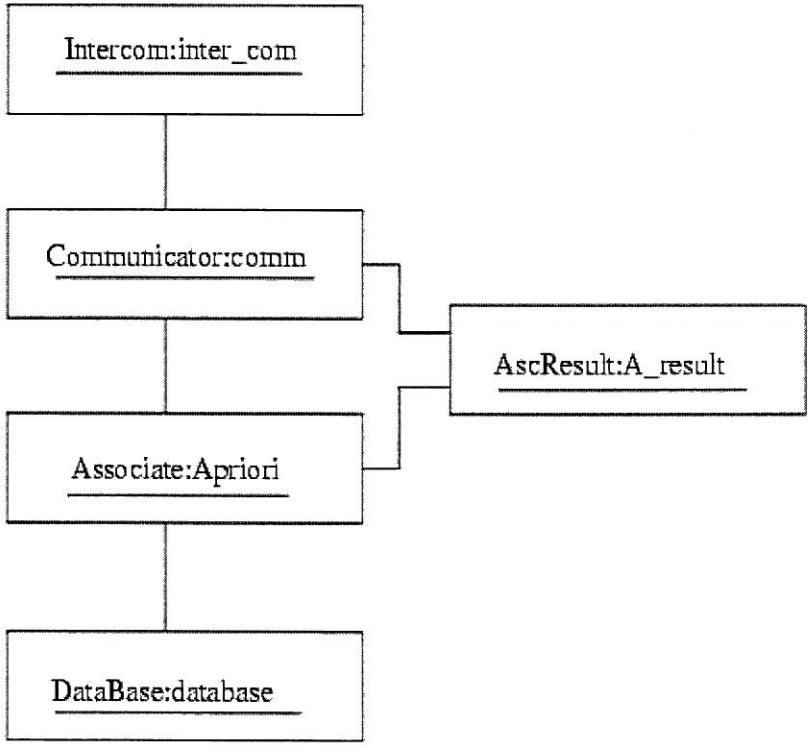

Fig 3: Object diagram of DM agent

This arrangement gives the system enough flexibility to add new machines to the existing system or change the addresses of the existing machines. This is because when a new database server is added to the system it will just register the services it provides to the broker agent. Thus, when a request for such service comes to the broker it can redirect the requisitioner to the appropriate server. The only constraint on the server side is that it should understand the KQML messages, and send the result of the data mining operations in the appropriate form.

6.1.4 Implementation of Data Mining Agents

The data mining agents implement the actual data mining Algorithms. The data mining agents are also implemented in Java. From the class diagram in Figure 2,

it can be seen that the data mining agent has two parts that handle the two different responsibilities.

1. The communicator that handles the message traffic in and out of the Data Miner. This is also responsible for *context management* with the communicating agent.
2. The actual part which implements the data mining algorithms.

From the diagram it is clear that the data mining agents always listen for a performative from the facilitator, and this happens in the run method of the intercom class. When a new message is received this method sends it to the communicator object which is responsible for context management and interpreting the messages. The communicator object, depending upon the request received, instantiates appropriate (algorithms) objects and executes the required methods from the classes. The Data Miner implements the following data mining operations.

- An algorithm for discovering associations rules in a database, for customer Cross Selling application.
- An algorithm for forecasting the variables in the database for forecasting application.

Figure 4 shows the sequence diagram that corresponds to the class diagram shown previously. This diagram shows the interactions among the various components of the agent in a case where the request for the Association operation occurred.

There are many algorithms for discovering the associations rules in a database. The problem of discovering the association rules from the database is divided into two parts:

1. Find all sets of items (item-set) that have transaction support above minimum specified support. These item sets are called large item sets. For discovering such large item sets we have used the Apriori (Agrawal and Srikant, 1994) algorithm.
2. Using the large item sets we generate the desired rules. For this we have used an algorithm given in Pudi and Haritsa (2002).

What follows is a brief description of the algorithm *Apriori*. This algorithm is for generating of the large item sets in a database (see Table 1).

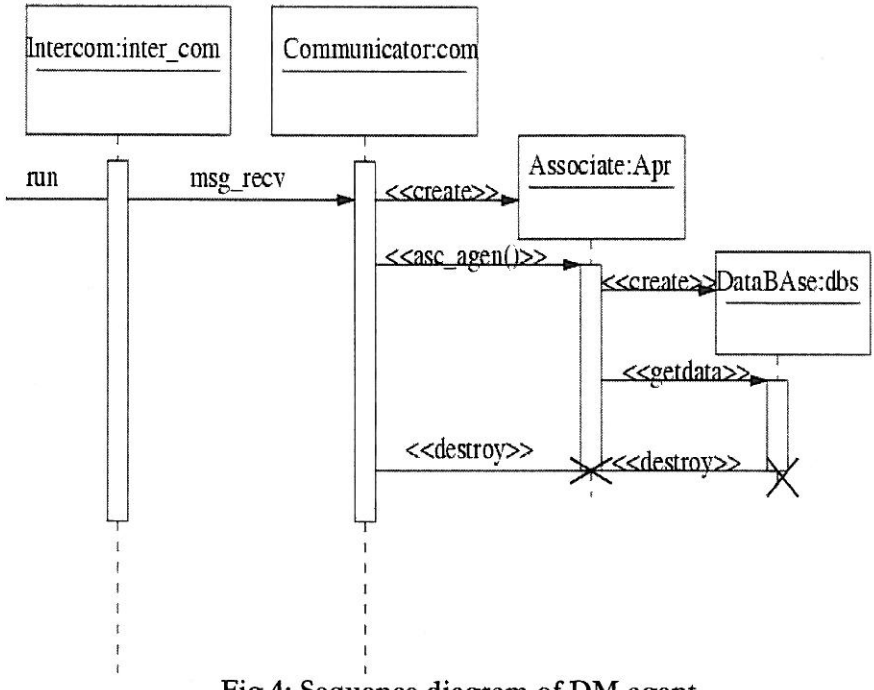

Fig 4: Sequence diagram of DM agent

The algorithm generates the *candidate item sets* to be counted for the next pass by using only the candidate item sets found large in the previous pass. The logic is that any subset of a large item set must be large. Therefore the candidate item sets having H items can be generated by joining large item sets of size $k-1$ items and deleting those that contain any subset that is not large. The *apriori_gen* function takes as argument L_{k-1} the set of all large $(K-1)$ – *itemsets* and returns a set of all large k-itemsets. The function works as shown in Table 2. Subsequently, we use an algorithm given in Pudi and Haritsa (2002) for the generation of the rules from the generated large itemsets. We have also implemented well known methods such as moving average and exponential smoothing for generating the forecast of the variables. The algorithms will be required for the processing of the forecast performative.

Table 1: The Apriori Algorithm

L_1 = *large 1-item sets;*
for ($k = 2$; $L_{k-1} \neq \phi$; $k++$) *do begin*
  C_k = *apriori_gen*(L_{k-1});
  forall transactions $t \in D$ *do begin*
    C_t = *subset*(C_k, t);//*Candidates contained in t*
    forall candidates $C \in C_t$ *do*
      c.count $++$;
  end
  $L_k = \{c \in C_k | c.count \geq min\_sup\}$
end
Answer $= U_k L_k$;

7.0 OVERALL OPERATION OF THE SYSTEM

The system is provided with a web based interface that is the primary source of input to the system. The user gives the type of operation to be performed on the database and also optionally s/he can specify which databases to be included in the data mining operations. The user also specifies some additional parameters that may be required to completely specify the operations s/he is interested in performing on the database. Once these inputs have been received, the facilitator agent starts processing them. It first constructs a valid KQML performative that corresponds to the database mining operation requested by the user, adding the additional performatives to it. Also, it then decides to which data mining agents these performatives are to be sent, depending upon the choice given by the user. The data mining agents always listen for a performative, upon receipt of which they start processing it. The appropriate algorithms classes are constructed and the operations then are executed. The results of these operations are then transformed in a form that is understandable by all the agents in the system by the use of the String() method defined in the Result interface. The results are then sent to the facilitator agent. The facilitator receives results from all the data mining agents, which it then integrates and presents in a suitable form to the user.

insert into C_k
select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
from $L_{k-1}p, L_{k-1}q$
where $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
Delete all itemsets $c \in C_k$ such that some (k-1) subset of c is not in L_{k-1}:
forall itemsets $c \in C_k$ *do*
  forall $(k-1)$ *– subsets* s *of* c *do*
    if$(s \notin L_{k-1})$ *then*
      delete c *from* C_k

7.1 Deployment of the Tool

Figure 5 shows the deployment view for the tool that we have developed. The software for the tool is organized in different packages, each component is written in a separate package. The tool requires that the *ans* (agent name server) package is run on a machine first, and then other components can be started. The *ans* package can run on a separate machine or with any of the components. The next component to deploy is the broker followed by the data mining agents. There is some initial conversation between all the other components and the *ans* package but it has not been shown in the figure.

The initial conversations that take place between the *ans* and the other components in the tool are basically registration requests for registration of these components with the *ans* so that they can be identified by that name. Also the other agents in the system identify all the other components in the system by their names registered with the *ans*.

The agents were deployed on machines named topaz, garnet, europa, opal and emerald at the Department of Computer Science, Indian Institute of Science Because the apriori algorithm is well known and because our focus is not on datamining algorithms per se, our validation only pertained to the *proof of concept* of applying agents in CRM. Nevertheless we did encode the algorithm in java and ran the same on well known random data generators which are available on the web for association rule mining. Specifically, we used *assoc.gen* (Quest, 1996). For the sake of completeness, a sample run using the apriori algorithm is given in Table 3. On a dataset of 100000 transactions that were randomly generated

system in terms of adding more database servers (data mining agents). The

Table 3: The truncated results of a sample run on 100000 random transactions

Supp=0.5; Conf=0.7; ave tlen 7; ntrans 100000
The conf is 0.88192165 Rule is— 1—— 4—
The conf is 0.8957324 Rule is— 1 ——9 —
The conf is 0.8810131 Rule is— 6 ——4 —
The conf is 0.87989104 Rule is— 8—— 4—
The conf is 0.88378966 Rule is— 4—— 9—

performatives that we have defined for this application are useful not only in this application but in all those which require the individual system components to interact with each other that involves data mining. In addition we have shown how effective the data mining techniques can be in the real world especially its application in customer relationship management.

There is scope of more work in aggregating the results obtained from the different data mining agents so that the results can be viewed as being obtained by performing the data mining operation on a single large database having the contents of the individual database. In this regard Pudi and Haritsa (2002) describes an algorithm that can be modified for use in integrating the results of the associations discovery problem. The issue that we have not dealt with in this paper is the security in concerns transmitting the results from one machine to another. The communication over the network needs to be made secure as the data being communicated is valuable, and may be of interest to others.

ACKNOWLEDGEMENTS

This work was partly supported by the Department of Science and Technology Grant No. SR/FTP/ET-08/2001, Government of India. The first author wishes to thank the two anonymous referees for their useful suggestions.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: A performance perspective. *IEEE Transactions on Knowledge Discovery and Data Engineering, 5*(6), 914-925.

Agrawal, R., & Srikant, R.(1994). Fast algorithms for mining associations rules. *Proceedings of the 20th VLDB Conference*, (pp 487-499), Santiago, Chile.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Technical Report, IBM Reasearch Division*, Almaden Reasearch Center, San Jose.

Choy, K. L., Lee, W. B., & Lo, V. (2002). Development of a case based intelligent customer supplier relationship management system. *Expert Systems with Applications, 21*, 1-17.

Cost, R. S., Finin, T., Labrou, Y., Luan, X., Peng, Y., Soboroff, I., Mayfield, J., & Boughannam, A. (1998). Jackal: A java based tool for agent development. *Proceedings of AAAI workshop on Software Tools for Developing agents*, (pp 73-82), Minneapolis, Minnesota.

DARPA (1993). *Specification of the KQML agent communication language*. DARPA Knowledge Sharing Initiative External Interfaces Working Group.

Kalakota, R., Robinson, M., & Tapscott, D. (2003). *E-Business: Roadmap for Success*. Addison-Wesley Information Technology Series, Reading MA, USA.

Kargupta, H., Hamzaoglu, I., & Stafford, B. (1997). Scalable distributed data mining using an agent based architecture. In Heckerman D., Mannila H., Pregibon D. and Uthurusamy (Eds.) *Proceedings of High Performance Computing*, R., (pp 211-214), AAAI Press.

Liang, T., & Huang, J. (2000). A framework for applying intelligent agents to support electronic trading. *Decision Support Systems, 28*, 305-317.

Pudi, V., & Haritsa, J. (2002). On the efficiency of association rule mining algorithms. *Proc. of 6th Pacific Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, (pp 80-91), Taipei, Taiwan.

Quest (1996). Quest synthetic data generation code, [On-line]. http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html

Shaw, N.G., Mian, A., & Yadav, S. B. (2002). A comprehensive agent-based architecture for intelligent information retrieval in a distributed heterogeneous environment. *Decision Support Systems, 32*, 401-415.

Stolfo, S. J., Prodromidis, A. L., Tselepis, S., Lee, W., F. D. W., & Chan, P.K.(1997). JAM: Java agents for meta-learning over distributed databases. *Knowledge Discovery and Data Mining*, 74-81.

Wilcox, P. A., Gurau, C. (2003). Business modelling with UML: The implementation of CRM systems for online retailing. *Journal of Retailing and Consumer Services, 10*, 181-191.

Yuan, S. T., Chang, W. L. (2001). Mixed-initiative synthesized learning approach for web-based CRM. *Expert Systems with Applications, 20*, 187-200.