A COLLABORATIVE PERSPECTIVE OF CRM

Mărginean Nicolae
Bogdan-Vodă University, Faculty of Economics, Cluj-Napoca, Spinoasa 14 street,
e-mail: nicolae1976@yahoo.com, telef: 0745/318321

Today, companies are becoming more customer focused, trying to adapt their business systems in order to be more responsive to the ever-changing needs of customers. CRM software gives a new face to the relationship with the customer, using the new technology. We propose to include a collaborative component in CRM, a multi-agent system that will automate and optimize some tasks specific to a CRM. We will use the ontologies to store the knowledge about our domain of interest: relationship with customers. These ontologies will be incorporated in the website of the organization and within all websites of the customer. So, handling the ontologies, agents' community will change information between organization and customers in an automatic manner, in order to support all activities implied in the relationship with customers.

Keywords: ontology, software agents, collaborative, website, CRM, OAA, OWL, facilitator

Cod JEL: M15

Introduction

Nowadays, IT&C support all activities of the business. Therefore, e-business technology enables companies to link their IT systems in an efficiently and flexible manner, being closer to business partners in order to better satisfy their needs and expectations. Among the business partners there are the customers. Because companies try to increase their revenues and because these revenues are the result of the business relationships with customers, today, companies are becoming more customer focused, trying to adapt their business systems in order to be more responsive to the ever-changing needs of customers. CRM software gives a new face to the relationship with the customer, using the new technology.

Customer Relationship Management

In accordance with [Fotache, 2004], Customer Relationship Management (CRM) is an integrated sales, marketing and service strategy who can help organizations to manage better customers' relationships. On the same idea, [Pulevska, 2008] says that CRM integrates sales, marketing and service strategies in order to optimize the customer benefits and relationships at long sight. Also, CRM technology is used to learn more about customers' needs and behavior in order to develop stronger relationship with them. CRM software supports all the previous ideas, allowing to the employees from the company departments to share and access pertinent information about customers (like products owned, prior support calls and so on) and their interactions. Interactions with the customers are generally stored in a customer contact histories, in a database format. Using this database, it is eliminated the need to obtain information directly from the customer.
As you see in the Figure 1, in the architecture of a CRM, between customers and database, we propose to include a collaborative component, a multi-agent system that will automate and optimize some tasks specific to a CRM.

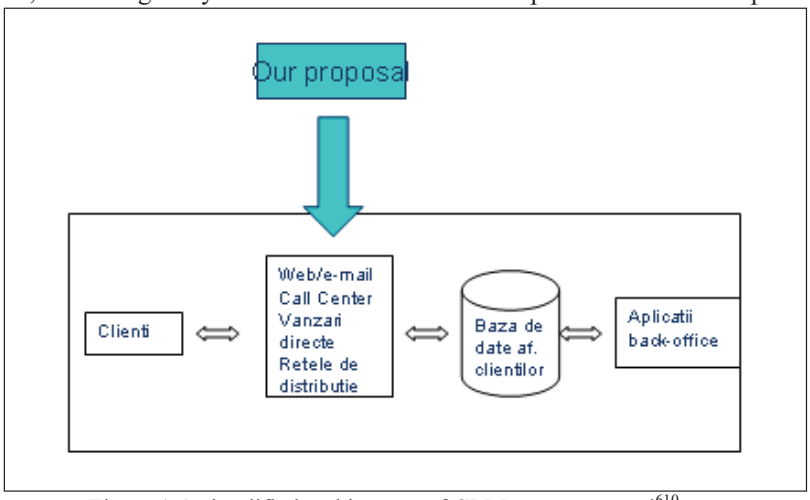

Figure 1 A simplified architecture of CRM + our proposal[610]

610 adapted from Fotache D, Hurbean L, "Solutii informatice integrate pentru gestiunea afacerilor-ERP", Ed. Economica, Bucuresti, 2004, pag. 103

The architecture of the proposed component = OAA agents + OWL ontologies

We have developed our component using Open Agent Architecture, a framework developed at the Artificial Intelligence Center of SRI International, an environment that allows creation of a flexible and dynamic community of software agents, providing a mean for integrating heterogeneous applications in a distributed infrastructure.

We have chosen it due to its facilities for agent interaction based on delegation to a special agent named facilitator. This agent coordinates the agent community allowing parallel execution, error treatment, detection of conflicts that can occur in problem solving and so on. Also, the facilitator provides a global data store for its client agents, which allows them to adopt a blackboard style of interaction. A system configuration is not limited to a single facilitator. The facilitator maintains a knowledge base that contains all the capabilities of the agents' community [Martin, OAA].

The client agents are usually specialized in providing a set of services and acts as a client of some facilitators. These services are named the capabilities of a certain agent. The declarations of the capabilities are referred as solvables. It is distinguished two types of solvables: procedure solvables and data solvables. A procedure solvable represents a procedure that acts in some manner whereas a data solvable offer a connection to a collection of data. A procedure solvable is handled by a declared handler whereas this is not necessary for a data solvable. Data solvables represent a dynamic collection of facts, which may be modified at runtime.

All the client agents inform its facilitator about the services they can provide. When is needed a service of a certain client agent, the facilitator send a request to it, expressed in ICL. The agent takes the message, parses it, processes it using his capabilities, and returns an answer to the facilitator [Martin, OAA].

When a client agent requests a services offered by the agents community, it send an ICL message to the parent facilitator in order to solve his requirement. With other words, the agent delegates his task to its facilitator. Thus, is not necessary to specify an particular agent to handle the call, the facilitator controls that aspect, obtaining the response to his requirements.

In OAA, the agents can be developed using many languages as Prolog, C, Java, Visual Basic and so on. Every developed agent must incorporate in his code an agent library. For all reminded language is available a specific library.

Also, OAA provides a mechanism of triggering some actions or procedures when some set of conditions is met. Thus, using triggers, each agent can monitor either locally or remotely some events that can appear at its facilitator or peer agents.

We will use the ontologies to store the knowledge about our domain of interest: relationship with customers. We will use OWL because allows us to publish and share ontologies in the WEB. It has a XML-based syntax, is an extension of RDF Schema, using the RDF meaning of classes and properties, having incorporated many primitives to support the richer expressiveness. All the concepts of the domain, the relationship between them and their instances will be located in the ontology. The knowledge of the system will be represented using OWL ontologies because it can be incorporated in websites, so that web resources becoming more accessible to automated processes achieved by the agents.

In accordance with [Horridge, 2009], OWL is a language for defining and instantiating WEB ontology. The main components of OWL ontology are classes, properties and the individuals. Individuals, also known as instances, represent objects in the domain in which we are interested. The links between individuals or relationship between them are named properties. There are many types of property:

-Object properties are relationships between two individuals. Object property may have a domain and a range specified. Properties link individuals from the domain to individual from the range. Also, there are many types of object property. If some property links individual "a" to individual "b", then inverse property links individual "b" to individual "a". If a property is functional, for a given individual, there can be at most individual that is related to the individual via the property. If a property is transitive, and the property relates individual "a" to individual "b", and also individual "b" to individual "c", then we can infer that individual "a" is related to individual "c" via property. If a property is symmetric and the property relates individual "a" to individual "b" then individual "b" is related to individual "a" via property. If a property P is antisymmetric and the property relates individual "a" to individual "b", then individual "b" cannot be related to individual "a". A property P is reflexive when the property relates individual "a" to itself. If a property is irreflexive, and individual "a" is related to individual "b", then individual "a" and individual "b" are not the same.

-Datatype property links an individual to the data values.

-Annotation properties are used to add information to all components of ontology (classes, individuals, properties).

The classes are concrete representations of the diverse concepts and describe all conditions that must be satisfied by a set of individuals that will become the member of the class. Classes are organized in the form of superclass-subclass hierarchies, named taxonomies. It must note that some classes may be disjoint, so that an individual cannot be an instance of more than one of these classes.

One way to describe a class is to define it as a subclass of the existent classes. Another way to describe a class is represented by the restrictions. It can be distinguished three main categories: quantifier restriction, cardinality

restriction and “hasValue” restriction. Quantifier description can be divided in existential restrictions and universal restrictions. Existential restriction describes a class of individuals that have at least one relationship among a specified property to an individual that is a member of a specified class. Universal restriction constrains the relationship along a given property to individual that are members of a specific class.
Regarding cardinality restriction, it can be distinguished a minimum cardinality, a maximum cardinality and a specified cardinality.
A hasValue restriction describes the set of individuals that have at least one relationship along a specified property to a specific individual.
OWL ontology is processed by a reasoner and offers the next services: testing whether or not one class is a subclass of another class, the building the inferred ontology class hierarchy and the verifying the consistency checking of a class.
Also, one class may be obtained through applying intersection, union or complement operators against the existent classes. The same OWL allows classes to be defined by listing the individuals that are the member of the class.
A little part of ontology that presents the software licenses as merchandise traded by the organization, and which will be incorporated in the website of the organization, is listed below.

```
<rdf:RDF>
  xmlns:owl=”http://www.w3.org/2002/07/owl#”
  All Namespaces

<owl:Ontology rdf:about=””>
   <owl:imports rdf:resource=”www.ubv.ro/ontologie”/>
   The header of ontology: version, import, comment
</owl:Ontology>

<owl:Class rdf:ID=”marfa”/>
<owl:Class rdf:ID=”licenta”>
    <rdfs:subClassOf rdf:resource=”#marfa”/>
</owl:Class>
<owl:Class rdf:ID:=”licentalimbaje”>
   <owl:intersectionOf>
   <owl:Class rdf:about=”#licenta”/>
   <owl:Restriction>
         <owl:onProperty rdf:resource=”tip”/>
         <owl:hasValue>
              <xsd:string rdf:value=”limbaje”>
         </owl:hasValue>
   </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
</rdf:RDF>

<owl:oneof rdf:parseType=”Collection”>
<owl:Thing rdf:about=”Microsoft”>
<owl:Thing rdf:about=”Borland”>
</owl:one of>

<licentalimbaje rdf:ID=”Prolog”>
```

In order to develop an ontology we must follow the next steps[611]:

- determine the domain and scope of the ontology;
- consider reusing existing ontologies;
- enumerate important terms in the ontology;
- define the classes and the class hierarchy;
- define the properties of classes;
- create instances;

The functionalities of our agents’ community will be achieved using the multi-agent platform Open Agent Architecture, with one facilitator, with delegation style, and the language Sicstus Prolog for creating every agent. The OWL ontologies will be built using Proteje 4 and the Prolog-style quering will be achieved using the PrologTab plug-in of it. OWL ontology will be processed by a reasoner named Pellet.

611 http://protege.stanford.edu/publications/ontology_development/ontology101.pdf

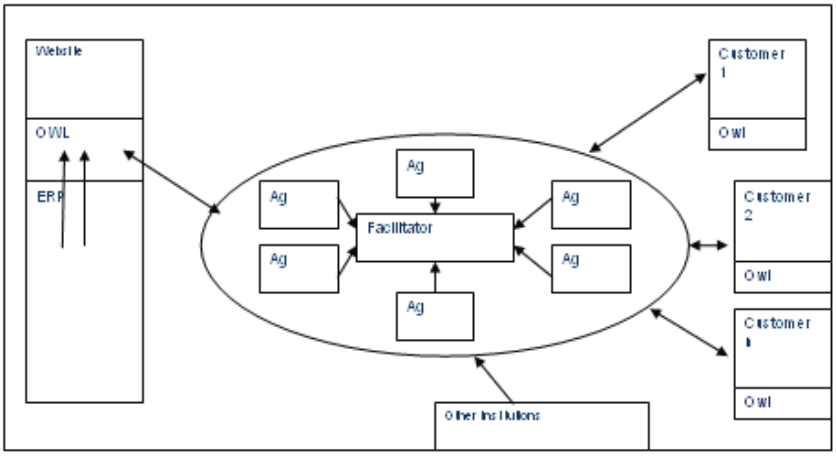

Figure 2 The proposed framework

According to the previous framework, the mechanism of proposed architecture gravitates to the virtual community of agents, which, in a collaborative manner, tries to solve the problems regarding the relationship with customers. We have identified a lot of agents, all connected to a facilitator, everyone having a specific role within the system:

Task agents – solve one of the next task specific to a CRM:

-support for planning of the marketing campaigns;
-support for designing and executing the marketing campaigns;
-sending campaign-related material to customer;
-tracking, storing and analyzing campaign statistics, including tracking responses and analyzing trends;
-scheduling sales call or mailings;
-tracking responses to/from the customers;
-generating many reports;
-the analyzing of the customer behavior in order to make decision regarding the products and services, -data mining technology being a good alternative for this aim;
financial forecasting and customer profitability;
support for analyzing the sales performance;
support for analyzing the customer trends;

Customer collector agents – gather all possible information from the website of the customers where is present an OWL ontology. Within ontology we can find out much information that identified each customer and characterized him.

Provider information agents – takes all possible information from the ontology developed by the organization in order to provide them to the customers. In this ontology will be presented all information and services provided by the organization in relationship with customer.

Conclusions

Relationship with customers is one of complex problem that an organization is confronted. Every complex problem can be easily solved if it is split in subproblems or modules. The paradigm of collaborating software systems is a good alternative to tackle large and difficult problems. The collaborating-software paradigm is a „divide-et-impera" approach to the development and maintenance of large and complex software applications.

The delegated computing ability of OAA enables both human users and software agents to express their requests in terms of what is to be done without requiring specification of who is to do the work or how it should be performed. Programming with delegation is very advantageous because reduces the dependencies among the agents of the community. Thus, is reduced the complexity for users and agents. Also it is encouraged reusing across applications and domain because interagent interactions are not pre-defined.

A CRM software developed with a distributed agent architecture allows the construction of systems that are more flexible and adaptable. In any moment it can be activated, deactivated, added, replaced and deleted agents that have associated specific task, in any moment agents can be created in multiple programming languages and interface with existing legacy systems, the solving problems are achieved using distributiveness and parallelism.

The presented OWL ontologies assure adequate knowledge at the right place and right time. That is the noble purpose of the modern knowledge management. The change of organizations management vision by giving information the role of main source will lead to major changes in the structure of any computer system. Software agents can query these ontologies and find out the desired information in order to solve the associated tasks, conferring to the CRM system more automation. Also, an OWL ontology can be incorporated in a web page, so

that all the knowledge are available for all interested persons or capable software, conferring to the CRM system more transparency and opening.

References

1. Fotache D, Hurbean L, "Solutii informatice pentru gestiunea afacerilor – ERP", Ed. Economica, Bucuresti, 2004
2. Fotache D, "Customer relationship management", Rev de Informatica Economica, nr 2(30), ASE Bucuresti, 2004
3. Horridge M, "A practical guide to building OWL using Proteje 4 and CO-ODE tools", University of Manchester, 2009
4. Pulevska-Ivanovska L, "CRM in Macedonian Telecommunications", The Annals of the "Stefan-Voda" University nr 8, Suceava, 2008
5. Martin D, Cheyer A, Moran D, "The open agent architecture: A framework for building distributed Software Systems", Available online: http://www.ai.sri.com/oaa
6. Noy N, McGuiness D, "Ontology development 101: A guide to creating your first ontology", Available online:http://protege.stanford.edu/publications/ontology_development/ontology101.pdf
7. en.wikipedia.org
8. www.ai.sri.com/oaa
9. proteje.standford.edu