Knowledge Processing in Contact Centers using a Multi-Agent Architecture

CLAUDIU IONUT POPIRLAN
Department of Mathematics and Computer Science
University of Craiova
A.I. Cuza 13, 200585 Craiova
ROMANIA
popirlan@inf.ucv.ro http://inf.ucv.ro/~popirlan

Abstract: - The explosion of multimedia data, the continuous growth in computing power, and advances in machine learning and speech and natural language processing are making it possible to create a new breed of virtual intelligent agents capable of performing sophisticated and complex tasks that are radically transforming contact centers. These virtual agents are enabling ubiquitous and personalized access to communication services from anywhere. As contact centers grow and become more complex in their function and organization, the knowledge processes become more formal to ensure consistency of advice and efficiency. This paper suggests a multi-agent approach for distributed knowledge processing and discusses the use of enhanced mobile agent architecture (EMA) [12] in context of contact centers to advance and frame future discussion of these knowledge intensive environments. We prove the benefits of the enterprise resource planning (ERP) management using multi-agent systems for contact centers with distributed knowledge, considering an adequate case study and providing experimental results.

Key-Words: - Multi-Agent Systems, Software Agent, Mobile Agent, Contact Centers, Distributed Knowledge, Enterprise Resource Planning (ERP).

1 Introduction

The evolution of communication services over the past century has spawned a broad new industry known as electronic contact, which provides electronic communication mechanisms between people and businesses or organizations. A contact center (also referred to as a customer interaction center or e-contact center) is a central point in an enterprise from which all customer contacts are managed. The contact center typically includes one or more online call centers [5] but may include other types of customer contact as well, including e-mail newsletters, postal mail catalogs, Web site inquiries and chats, and the collection of information from customers during in-store purchasing. A contact center is generally part of an enterprise's overall customer relationship management (CRM).

At present, most actionable knowledge about management of enterprise resource planning (ERP) systems encompasses the implementation process and it relates to software modules used in back-office or production environments. For example, in [8] is described an integrated architecture and a conceptual framework for multi-agent ERP system, a prototype multi-agent enterprise resource planning (MAERP) system that utilizes the characteristics and capabilities of software agents to achieve enterprise wide integration. A software agent is a self-contained, autonomous software module that performs assigned tasks from the human user and interacts/communicates with other applications and other software agents in different platforms to complete the tasks. However, management knowledge needs are changing as ERP adopters begin to face post-implementation performance challenges and as ERP systems extend into customer-facing or front-office work environments where the boundary between the organization and the outside world is porous and interactive.

Agent-based systems [11] claim to be next generation software capable of adapting dynamically to changing business environment and of solving a wide range of knowledge processing application. Software agents are sophisticated computer programs that act autonomously on behalf of their users to solve complex problems, and a multi-agent system is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver. Although sophisticated software agents can be difficult to build from scratch due to the skills and knowledge needed, the widely available agent construction toolkits may provide a quick and easy start to building software agents without much agent expertise. For example, JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases [1].

Since 1950s when transaction processing systems were first introduced, information systems have been successfully implemented in different functional areas, each with its own database and data architecture, to support decision making. Although such functional information systems have matured in terms of functionalities over the years of testing, modification, and maintenance, these systems have also caused problems such as data redundancy, information inconsistency and/or inaccuracy, and high system maintenance costs. In addition, such information systems are likely to result in poor decision quality due to the lack of cross-organizational perspective and communication difficulty. Consequently, the whole organization may lose its competitive edges because it does not realize its full potential.

In the late 1970s and early 1980s, the need for enterprise wide integrated systems intensified as global competition became inevitable, and product customization and innovation became important factors to retain customers and subsequently to gain market share. Systems thinking based management philosophies such as total quality management and just-in-time systems were introduced, which necessitated the management of relationships among functional areas and cross-organizational processes.

The development of such systems slowly evolved from standalone systems (e.g., a standard inventory control system) to material requirement planning/manufacturing resource planning systems, and subsequently, to enterprise wide systems to include other functional areas such as sales and marketing, financial accounting, and human resource management. However, attempts to provide a complete enterprise wide software solution were not successful until the mid-1990s due to technical complexity, lack of resource availability, and unclear vision.

In the mid-1990s, the Gartner Group coined the term “ERP” to refer to next generation systems which differ from earlier ones in the areas of relational database management, graphical user interface, fourth generation languages, client–server architecture, and open system capabilities. The integration is normally implemented through the use of a common database among subsystems. All the subsystems “talk” directly to each other, and the data are made available in real time. The information is updated as changes occur, and the new status is available for everyone to use for decision making or for managing their part of business. The decisions made in different functional areas are based on the same current data to prevent nonoptimal decisions from obsolete or outdated data. Expected benefits from ERP implementation include lower inventory, fewer personnel, lower operating costs, improved order management, on-time delivery, better product quality, higher productivity, and faster customer responsiveness.

Researchers involved in agent research have used a variety of terms and offered definitions, explicating their use of each term. Although there is no general agreement as to what constitutes an agent, many studies ([1], [2], [10], [11], [13], [16]) provide taxonomy of autonomous agents and establishes how they are different from a computer program. In the literature, the term software agent is also referring to as “agent,” “autonomous agent,” intelligent agent,” and “business agent.” An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [11].

In the general field of agent-based systems we can identify two major diverging directions, agent theory and industrial applications. On the one hand there is considerable work on formalization of multi-agent systems, e.g., commitments, capabilities, know-how.

Advanced logics which capture essential properties of agents like concurrency, time dependent behavior or inconsistent states are under development. Formal specification techniques like Z are applied to formally capture those systems. On the other hand, there exists a number of successful implementations dedicated to tackle real-world problems like flight control, manufacturing resource allocation, and information retrieval in large databases. Significant research and development into mobile agency has been conducted in recent years, and there are many mobile agent architectures available today. The AgentLink project [17] maintains a list of ongoing projects with regard to any kind of agent-related topics and also maintains a list of available agent toolkits. Nevertheless, several issues still need to be faced to make the mobile agent technology widely accepted: secure and efficient execution supports, standardization, appropriate programming languages and coordination models.

Significant research and development into mobile agency has been conducted in recent years, and there are many mobile agent architectures available today.

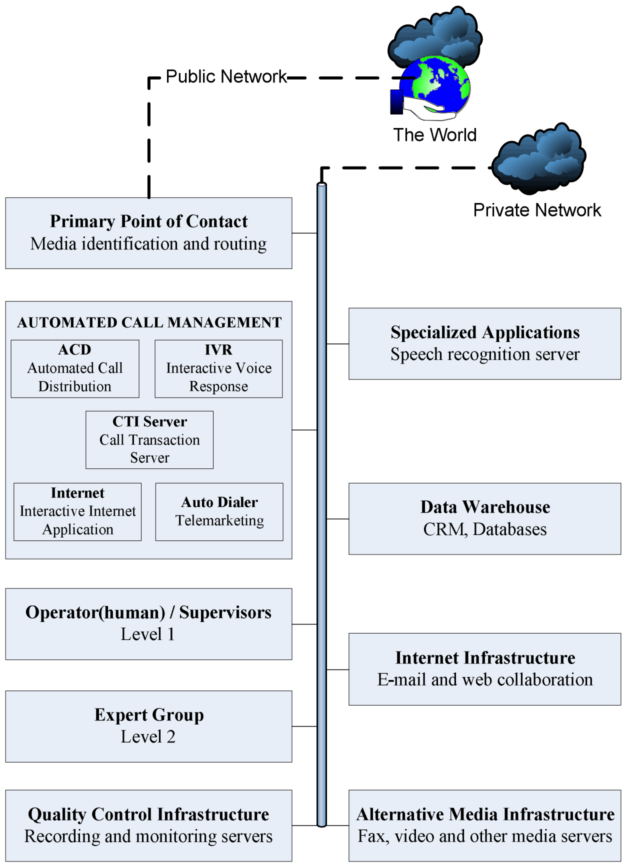
In this paper is presented a multi-agent approach using EMA architecture [12] for distributed knowledge processing, in context of contact centers with enterprise resource planning (ERP) management. We introduce a multi-agent model for the analysis and process of contact center knowledge infrastructures.

2 Contact Center Description

A contact center would typically be provided with special software that would allow contact information to be routed to appropriate people, contacts to be tracked, and data to be gathered. A contact center is considered to be an important element in multichannel marketing. Contact centers are classified by size and analyzed in relation to specific process-oriented dimensions including maturity stages. The model also points out the kinds of knowledge management (KM) interventions that are successfully established in contact centers.

The contact center architecture that includes the related components is described below and can be visualized in Figure 1. The role of the view-related components is as follows:

- Automatic Call Distribution (ACD) is a software feature of the telephone system that routes a call to groups of operators (also called a *queue*) based on first-in, first-answered criteria. The guiding principle is that the caller who has been waiting the longest will be first the caller routed to the next available operator.
- The goal of ASR is to accurately and efficiently convert a speech signal into a text message, independent of the input device, speaker, or the environment.
- Although automation platforms have the flexibility of playing prerecorded prompts, TTS is generally invaluable for reducing the cost of prompt generation and for speaking dynamic contents that are highly variable, such as e-mails, medical and insurance records, names, and addresses. The goal of TTS is to render individual words intelligibly and achieve a prosody that sounds natural to the listener.
- Human speech is an important biometric feature for actively or passively authenticating a person’s identity. In many respects, speaker recognition (SR) technology has advanced to a point where it can be shown to be superior to human performance. Whether it is security of personal records, such as bank account information, or security against identity theft or within public facilities like airports or shopping malls, everyone is concerned about their safety and the safety of their information [4].
- Although interest in ASR and TTS can be traced as far back as the late 1700s, when Von Kempelen created a mechanical talking machine that mimics the human voice apparatus, research in SLU only began in the early 1970s. This era saw the introduction of the Advanced Research Projects Agency (ARPA)-funded Speech Understanding Research (SUR) program. The technology became more widespread in the 1990s, when speech recognizers became powerful enough that they could operate in real-time with relatively reasonable accuracy. SLU makes it viable to adopt natural language dialog applications without having to achieve perfect recognition accuracy and without dictating what users should say, as present in traditional, system-initiative dialogs. In SLU, the goal is to extract the meaning of the recognized speech to identify a user’s request and fulfill their need. There are essentially three components in the VoiceTone SLU system. A preprocessor normalizes the input word lattice by removing disfluencies and applying morphological and synonym rules. An entity extractor is a sort of shallow parser that is written in a grammar rule notation. These grammars [3], compiled as finite-state transducers, are used to capture domain-dependent as well as domain-independent entities.


Public Network
The World
Private Network
Primary Point of Contact
Media identification and routing
AUTOMATED CALL MANAGEMENT
ACD
Automated Call Distribution
IVR
Interactive Voice Response
CTI Server
Call Transaction Server
Internet
Interactive Internet Application
Auto Dialer
Telemarketing
Operator(human) / Supervisors
Level 1
Expert Group
Level 2
Quality Control Infrastructure
Recording and monitoring servers
Specialized Applications
Speech recognition server
Data Warehouse
CRM, Databases
Internet Infrastructure
E-mail and web collaboration
Alternative Media Infrastructure
Fax, video and other media servers
Fig. 1: The Contact Center Structure

2.1 Contact Center Management

A contact center offers a service, delivered through telephone calls with clients. Service level can be defined as the degree of satisfaction of callers with the offered service. This service level consists of many different aspects, related to the quality of the answer, the waiting time of the customer, etc. Some of these are hard to quantify, such as friendliness of the agent, others are more easily quantified.
The main indicator for efficiency, in contact center, is the productivity [7], measured over a certain period (for example, a week). It is usually given as the percentage of time that an agent is working of his or her total scheduled working time:

$$Productivity = \frac{TWT}{TWT + TA} 100\% \qquad (1)$$

where TWT is *total working time* and TA is *time available*.

The percentage of calls that is answered in less than a certain fixed waiting time is sometimes called the telephone service factor (TSF). Another commonly used waiting time metric is the average speed of answer (ASA). The *Erlang C* formula [7] gives the TSF, and can be used to compute the average waiting time for a given number of human agents (operators), service times and traffic intensity.

2.2 Typical Process in a Contact Center

The following introduces a typical process in a multi-agent call-center. Please note this process is meant to be illustrative rather than comprehensive. The customer dials the call-center number and is greeted with a number of options that include the following:

- a recorded message followed by the placement in a telephone queue managed by an Automated Call Distribution System (ACD);
- an Integrated Voice Response (IVR) that offers the caller different options where caller interacts with the IVR using a touch-tone telephone or voice control;
- the call is immediately directed by an ACD to an agent who manages the query. If the agent cannot personally resolve the query they direct the call to someone who can answer the query.

To better understand the process of a call-center that provides knowledge-based support (responses) to queries from a customer base, the following general theoretical schema, the Query-Response Cycle [15] is proffered, as shown in Figure 2.

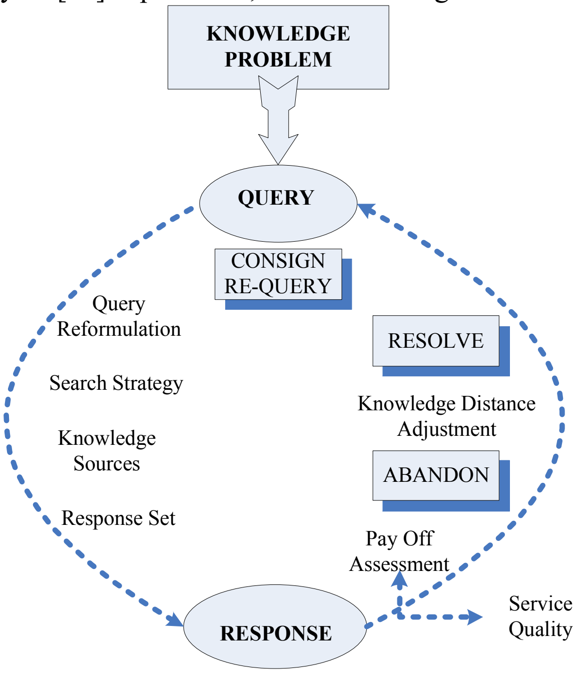


Fig. 2: Query-Response Cycle

To manage call centers, or more generally, contact centers, effectively, one needs to have multiple skills. Roughly speaking there are those skills which are unique to the product that is delivered, and there are those skills that are needed in virtually any call center.

3 EMA Architecture for Contact Centers

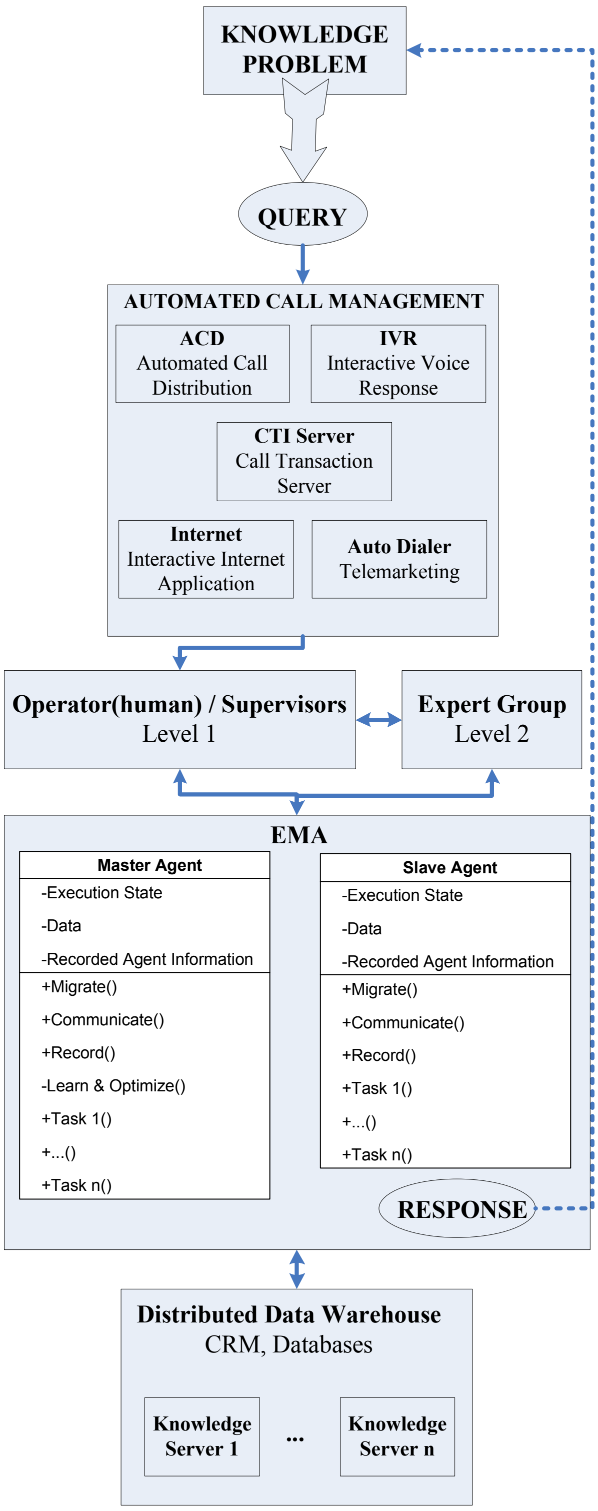

KNOWLEDGE PROBLEM
QUERY
AUTOMATED CALL MANAGEMENT
ACD Automated Call Distribution
IVR Interactive Voice Response
CTI Server Call Transaction Server
Internet Interactive Internet Application
Auto Dialer Telemarketing
Operator(human) / Supervisors Level 1
Expert Group Level 2
EMA
Master Agent
-Execution State
-Data
-Recorded Agent Information
+Migrate()
+Communicate()
+Record()
-Learn & Optimize()
+Task 1()
+...()
+Task n()
Slave Agent
-Execution State
-Data
-Recorded Agent Information
+Migrate()
+Communicate()
+Record()
+Task 1()
+...()
+Task n()
RESPONSE
Distributed Data Warehouse CRM, Databases
Knowledge Server 1
...
Knowledge Server n

Fig. 3: Using EMA in Contact Center

In [8] the proposed *Enhanced Mobile Agents (EMA)* architecture improves the basic structure of mobile agents and their advantages can be highlighted very clearly in case of master/slave mobile agents system. This system includes one Master Agent and several Slave Agents [9]. The Master Agent receives complex requirements (interrogations) and creates specific Slave Agents in order to process parts of a certain complex activity. In this case, the Master Agent may use *Recorded Agent Information (RAI)* component to adapt its behavior and/or the actions directed to Slave Agents in order to improve the solving of assignments. Also, Master Agent can use its RAI component for the same purpose and at the same time can provide access to it. We proposed EMA architecture for distributed knowledge processing in the context of contact center, as shown in Figure 3.

The proposed multi-agent ERP architecture is composed of:

- a set of Master Agents (a coordinating agent for each query), and
- a set of Slave Agents (data collecting agents).

Figure 3 illustrates the abstract level of multi-agent system architecture with coordination agents serving as the representatives of each knowledge problem (query) and communicating with each other over the network. The data collecting agents (Slave Agents) perform specific tasks for a coordination agent (Slave Agent). Next, various functions/responsibilities undertaken by each type of software agent are discussed.

3.1 Master Agent - The coordination agent

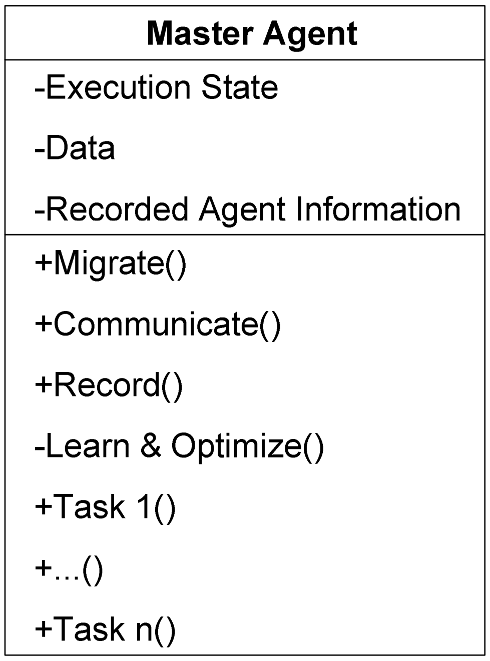


Fig. 4: Master Agent

The Master Agent (MA) is the heart of this multi-agent ERP architecture, is the representative of the knowledge problem when communicating with other coordination agents. The major responsibilities for the MA include:

- receiving instructions and reporting to the human operator in contact center, through an interface;
- assigning data collection to and receiving data from the Slave Agents (data collection agents);
- assigning tasks to, and receiving feedback from Slave Agents;
- communicating with and providing requested data to other MA.

With their domain knowledge, the Master Agents have the ability to monitor, communicate, and collaborate with other agents, react to various requests, as well as assign tasks to proper Slave Agents.

3.2 Slave Agent - The data collecting agent

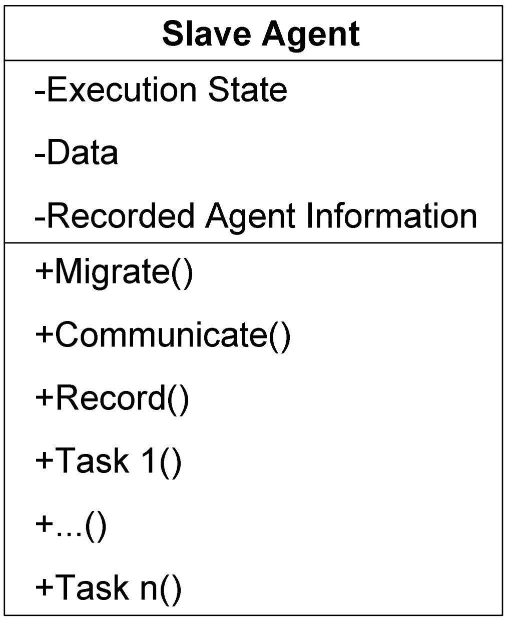


Fig. 5: Slave Agent

The objective of the Slave Agent (SA) is to query specific database(s)(distributed) within the contact center servers and obtain the information requested by its own Master Agent. It possesses specific domain knowledge needed to carry out its tasks. The *intelligence* in the Slave Agents identifies invalid data and missing values so that the data is complete and applicable when being returned to the Master Agent. However, the structures or abilities of SA's need not be the same in different contact centers because each contact center may have a different database management system (DBMS) or data warehouse.

The responsibility of a Slave Agent is to:

- retrieve information requested by its own Master Agent;
- query distributed databases within the contact center;
- performing data analysis by running specific program and/or algorithm;
- reporting the results back to the Master Agent.

The system must be able to process the data from a distributed knowledge base. In this respect, the Slave Agents visit, one after another, all or a part of the servers to whom they ask for certain information. When an SA gathers all the knowledge requested by its MA it returns home and shows the results. The system must function independently of the server addresses. Thus, in a situation where each server has a dynamic IP address, the servers list maintained by the agent must also be dynamic. The solution for this list to reflect at any moment, as exactly as possible, the addresses of the servers is to constantly update it.

3.3 Multi-Agent System and Agent Collaboration

The computing paradigm of multi-agent systems (MAS) has its origin in both distributed artificial intelligence (DAI) and object-oriented distributed systems. There is no consensus on the definition of software agents or of agency, and some people go so far as to suggest that any piece of software or object that can perform a specific given task is an agent. However, the prevailing opinion is that an agent may exhibit three important general characteristics: *autonomy*, *adaptation,* and *cooperation.* By "autonomy" we mean that agents have their own agenda of goals and exhibit goal-directed behavior. They are not simply reactive, but can be pro-active and take initiatives as they deem appropriate. In this sense, agent systems can be viewed as a generalization of the client-server model in that each agent can be both a client and a server and can provide and request services to and from others. Adaptation implies that agents are capable of adapting to the environment, which includes other agents and human users, and can learn from the experience in order to improve themselves in a changing environment. Cooperation and coordination between agents is probably the most important feature of multi-agent systems. Unlike those stand-alone agents, agents in a multi-agent system collaborate with each other to achieve common goals. In other words, these agents *share* information, knowledge, and tasks among themselves. The intelligence of MAS is not only reflected by the expertise of individual agents but also exhibited by the emerged collective behavior beyond individual agents. From software engineering point of view, the approach of MAS is also proven to be an effective way to develop large distributed systems. Since agents are relatively independent pieces of software interacting with each other only through message-based interagent communication, system development, integration, and maintenance become easier and less costly. For instance, it is easy to add new agents into the agent system when needed. Also, the modification of legacy applications can be kept minimum when they are to be brought into the system. Aside from adding communication capabilities to a legacy application, nothing else is required to change.

Cooperation and coordination of agents in a MAS requires agents to be able to understand each other and to communicate effectively with each other. The infrastructure that supports agent cooperation in a multi-agent system is thus seen to include at least the following key components:

- A common agent communication language (ACL) and protocol.
- A common format for the content of communication.
- A shared ontology.

With a common communication language, content language, and a shared ontology, agents can communicate with each other in the same manner, in the same syntax, and with the same understanding of the world. In addition, to make agent collaboration more efficient and effective, some service agents are often created in multi-agent systems. One type of a service agent is the Agent Name Server (ANS). The ANS serves as the central repository of physical addresses (in the form of the chosen transport mechanism) for all involved agents. It maintains an address table of all registered agents, accessible through the agents' symbolic names. Newly created agents must register themselves with the ANS with their names, physical addresses and possibly other information by sending to the ANS a message with the performative register. The ANS maps the symbolic name of a registered agent to its physical address when requested by other agents.

Another type of a service agent is the Facilitator Agent (FA) which provides additional services to other agents. A simple FA is a Broker Agent (BA). The BA serves, to some extent, as a dynamic information hub or switchboard. It registers services offered and requested by individual agents and connects dynamically available services to requests whenever possible. Agents register their available services by sending messages with the performative advertise, and request services by sending to the BA messages with brokering performatives such as recommend-one. In both cases, the description of the specific service is in the content of the message. In a reply to a recommend-one message the BA will send the symbolic name of an agent which has advertised for being able to provide the requested service at the BA, or sorry if such request cannot be met by current advertises.

In this architecture, an application will define events it is interested in (e.g. changes in process rates, yield, material due dates) and have programs (agents) to monitor such events. When those events occur, the agents will notify the concerned applications. The monitoring/notification architecture is in sharp contrast to the polling architecture used by many existing systems where reports are periodically generated from production databases. These reports are interpreted either by humans or by other computer programs. The polling model has several major weaknesses:

- Reports cannot be generated in real-time.
- Pre-scheduled reports may not catch critical events early enough to make corrective actions.
- Not all information is logged for reports.

A natural way to implement the monitoring/notification architecture is to use Broker agents (BA) to track the agents which can provide monitoring service for a type of event another agent is interest in. The ability of the BA to dynamically link services with requests in real-time would greatly increase the flexibility toward manufacturing integration and interoperation.

4 Case Study and Experimental Results

In order to illustrate the proposed multi-agent architecture for distributed knowledge processing in a contact center, we will explain the setting and describe a scenario to answer a specific query:

A client needs to determine if the company can accept an order from customer X from city C for N units of product K at price P by Tuesday?

For simplicity, let us assume that the multinational company has four working point in four different locations: Romania (accounting department), Bulgaria (production department), Czech Republic (logistics/distribution department), and Greece (marketing department). Each working point has its own information system, database, and data architecture. The Contact Center mainframe is located in Romania and the client is located in Greece.

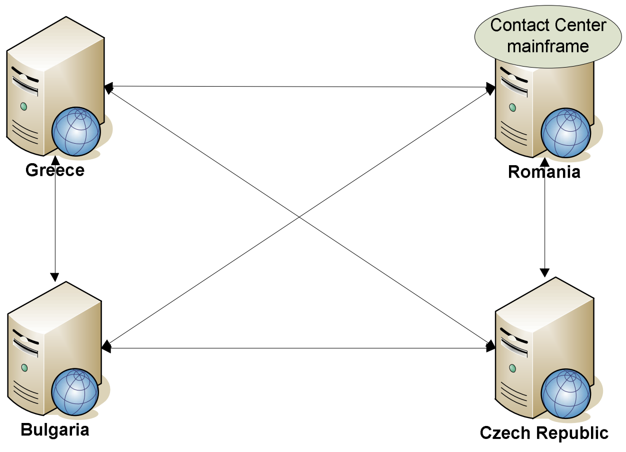


Fig. 6: Network topology and agencies locations

The solution is based on four agencies located in the considered countries (working point) - the one located in Romania is *the home agency*. The network topology is shown in Figure 6. Each node provides an execution environment with different hardware configurations and different operating systems. For implementation we used JADE (Java Agent DEvelopment Framework) [1].

Initially, the home agency will contain only the Master Agent. Based on the assignment it will create at least one Slave Agent to each of the other agencies. To accomplish their mission, those Slave Agents, will migrate to the appropriate agency, retrieve needed data locally, process it, and send their results. Also, each Slave Agent will record information, using RAI component [12], about queries duration, and Master Agent will use this to improve the assignment solving. We choose a simple improvement method which will increase by 1 the number of Slave Agents sent to an agency if the recorded query duration is greater than 1 second (see Algorithm 1).

Algorithm 1. **Assignment Improve**

```
for each Slave Agent do
    if RecordedQueryDuration > 1 then
        NumberOfSlaveAgents++;
    endif
endfor
```

By exercising domain knowledge, Master Agent organizes four tasks concurrently:

- inquire the Slave Agent in production department (Bulgaria) to obtain current inventory status of K;
- inquire the Slave Agent in logistics/distribution department (Czech Republic) for delivery information;
- inquire the Slave Agent in the marketing department for price of product K;
- monitor the status of requested information from various agents.

Due to the numbers and complexity of tasks, two Slave Agents are assigned to the production department (Bulgaria) for the product mix optimization system and for the master production scheduling system. In the logistics and distribution department (Czech Republic), Slave Agent examines a scenario of scheduling N unit of product K delivered to city C by Tuesday. In the marketing department (Greece), Slave Agent queries database and obtains current selling price (P) of product K given order quantity N and returns the result back to its Master Agent.

Upon receiving all the information from Slave Agents, Master Agent exercises its own domain knowledge to evaluate the information and to make recommendations to the client. Based on the overall results, two sets of procedures may be executed by the Master Agent:

- Case 1: All conditions are met for order acceptance
    - Master Agent informs human operator to notify the client that all conditions are met for order acceptance;
    - If the client commits the order, the operator will communicate with the Master Agent to trigger a sequence of actions;
    - If the client rejects the order, the operator will perform a sequence of actions.
- Not all conditions are met for order acceptance. In this case the Master Agent response with negative answer and can list partial solutions (for example Master Agent can evaluate the query without taking into account one of the conditions).

4.1 Experimental Results

The results presented in this subsection prove the efficiency of proposed multi-agent architecture for distributed knowledge processing in contact centers. The assignment implies to process a query that involves finding if the can accept an order that meet the following conditions:

- Customer = Renault Automobile;
- City = Paris;
- Units = 7;
- Product = summer tires;
- Price = 150 €.

In the first run the Master Agent creates only one Slave Agent for each of the Bulgaria, Czech Republic, and Greece agencies. The query duration stored in RAI component [8] of each Slave Agent are presented in Table 1.

	Czech R.	Bulgaria	Greece
	Slave Agents		
	SA-C1	*SA-B1*	*SA-G1*
Duration (sec)	0,7	**1,1**	0,4

Table 1. Experimental results – First Run

The final result which aggregates the Slave Agents results was obtained in 1,5 seconds. When trying to understand the values of execution times we have to keep in mind that the mobile agents' code is executed quasi-parallel.

The second run with the same requirements will look different, because based on information recorded in RAI component [8] of the *SA-B1* agent, the Master Agent will decide to create two Slave Agents - *SA-B1* and *SA-B2* - which will co-operate by splitting the records in two ranges. The results of this test are presented in Table 2.

	Czech R.	Bulgaria		Greece
	Slave Agents			
	SA-C1	*SA-B1*	*SA-B2*	*SA-G1*
Duration (sec)	0,7	0,5	0,5	0,4

Table 2. Experimental results – Second Run

The final result on second processing of the query was obtained in 0.9 seconds.

5 Conclusion

Intelligent business agents claim to be the next generation of model-based solutions for business-to business and E-Commerce applications [8]. This study proposes another application for software agents as an implementation alternative of ERP, in context of a Contact Center solution.

With this approach, we demonstrate that a multi-agent system can effectively improve the contact center efficiency by distributed knowledge processing, by gathering relevant information and knowledge, and more importantly, by agents cooperation in order to arrive at timely decisions in dealing with various enterprise scenarios. We provide a simple illustration to show how the proposed system might work.

Further research is needed to extend the current work, to advance and confirm this model, and to address its limitations. We intend to develop a prototype of this multi-agent system; which can demonstrate that more practical and relevant problems can be addressed successfully.

Acknowledgements: This work was supported by the strategic grant POSDRU/89/1.5/S/61968, Project ID61968 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007–2013.

References:

[1] F.L. Bellifemine, G. Caire and D. Greenwood, Developing Multi-Agent Systems with JADE, Wiley, 2007.

[2] P. Braun, W. Rossak, Mobile Agents: Concepts, Mobility Models, & the Tracy Toolkit, Elsevier Inc. (USA) and dpunkt.verlag(Germany), 2005.

[3] M. Colhon, N. Tandareanu, An Approach for Contextual Translation based on Semantic Schemas, Proceedings od 14th WSEAS International Conference on Computers, ISSN 1792-4251, ISBN 978-960-474-201-1, 2010, pp. 294-300.

[4] N. Constantinescu and G. Stephanides, Identication of parts in identity-based encryption. WSEAS Academic Press, Advandces in Learning, Commerce and Security, K. Morgan Editor, University of Bergen, Norway and J.M. SPECTOR, Syracuse University, USA, Vol. 30, 2004, pp. 177-183.

[5] N. Gans, G. Koole and A. Mandelbaum, Telephone call centers: Tutorial, review, and research prospects, Manufacturing and service operations management 5, 2003, pp. 79-141.

[6] Ion Iancu, Claudiu-Ionut Popîrlan, Mamdani Fuzzy Logic Controller with Mobile Agents for Matching, Proceedings of the 11th WSEAS International Conference on FUZZY SYSTEMS (FS '10), Iasi, Romania, June 13-15,

2010, ISSN: 1790-5109, ISBN: 978-960-474-195-3, pp. 117-122.

[7] G. Koole, Call center mathematics, online book http://www.math.vu.nl/˜koole/ccmath/book.pdf

[8] B.-R. Leaa, M.C. Guptab andW.-B. Yu, A prototype multi-agent ERP system: an integrated architecture and a conceptual framework, Technovation 25, 2005, pp. 433-441.

[9] M. Papazoglou, Agent-oriented technology in support of e-business, Communications of the ACM 44, no. 4, 2001, pp. 71-77.

[10] C.-I. Popirlan, L. Stefanescu, A Mobile Agents System for Intelligent Data Analysis, Proceedings of WSEAS Applied Computing Conference 2009 (ACC 2009), September 28-30, Athens, Greece, Mathematical Methods and Applied Computing, Volume 1, N. Mastorakis, et al. (Eds.), WSEAS Press, ISBN: 978-960-474-124-3, 2010, pp. 663-668.

[11] S. Russell and P. Norvig Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.

[12] G. Stoian and C.-I. Popirlan, A proposal for an enhanced mobile agent architecture (EMA), Annals of the University of Craiova, Mathematics and Computer Science Series 37, no. 1, 2010, pp. 71–79.

[13] N. Tandareanu and C.-I. Popirlan, A Mobile Agents Approach for Knowledge Bases Processing, Proceedings of the Twelfth IASTED International Conference on Intelligent Systems and Control (ISC 2009), Cambridge, Massachusetts, USA, November 2–4, 2009, pp. 27-32.

[14] N. Tandareanu, M. Colhon, C. Zamfir, A Spoken Question Answering System Based on Conditional Knowledge, Proceedings of 14th WSEAS International Conference on Computers, ISSN 1792-4251, ISBN 978-960-474-201-1, 2010, pp. 220-226.

[15] G. Timbrell and B. Shepperd, Improving Advice and Support Services using Applied Knowledge Management Strategies, Proceedings of the Enabling the Information Future Conference, Brisbane, 16-17 October, 2002.

[16] P. Thati, P.-H. Chang, G. Agha, Crawlets: Agents for high performance web search engine, Lecture Notes in Computer Science 2240, 2001, pp. 119-134.

[17] www.agentlink.org