Improvement in the distribution of services in multi-agent systems with SCODA

Jesús A. Román[a], and Sara Rodríguez[b]

[a]Department of Computer Science and Automation. EPS of Zamora. University of Salamanca. Avda. Cardenal Cisneros, 34, 49022, Zamora, Spain.

[b] Biomedical Research Institute of Salamanca/BISITE Research Group, Calle Espejo, s/n, 37007 Salamanca, Spain

zjarg@usal.es, srg@usal.es

KEYWORD

Multiagent Systems; Distributed Services; Specialized Communities; SOA; Architecture

ABSTRACT

The distribution of services on multi-agent systems allows it to reduce to the agents their computational load. The functionality of the system does not reside in the agents themselves, however it is ubiquitously distributed so that allows you to perform tasks in parallel avoiding an additional computational cost to the elements in the system. The distribution of services that offers SCODA (Distrib-uted and Specialized Agent Communities) allows an intelligent management of these services provided by agents of the system and the parallel execution of threads that allow to respond to requests asynchronously, which implies an im-provement in the performance of the system at both the computational level as the level of quality of service in the control of these services. The comparison carried out in the case of study that is presented in this paper demonstrates the existing improvement in the distribution of services on systems based on SCODA.

1. Introduction

The service-oriented computing (SOC) uses the services that allow the development of distributed applications quickly, operable, scalable and massive (Papazoglou *et al.*, 2007). In this aspect, services are autonomous, independent of the platform and weakly coupled to the system that invokes these services.

The key to carry out this paradigm are Service Oriented Architectures (SOA) (Papazoglou *et al.*, 2007). SOA (Rosen *et al.*, 2008) is the way for the development of systems composed of distributed applications and services (Papazoglou *et al.*, 2007). SOA provides a set of guidelines, principles and techniques, through which the information and business processes, can be rearrange and distribute effectively raising the level of competitiveness (Papazoglou & Georgakapoulos, 2003). The definition of service is crucial to the effectiveness of a service-oriented architecture, thus, a service can be defined as a function is clearly formulated, self-contained and independent of the context in which it is run (Papazoglou & van den Heuvel, 2006). The implementation of SOA is strongly linked to Web Services, these being a way to develop these architectures, but not the only (Zimmermann *et al.*, 2005).

The development of multi-agent Systems (MAS) based on service-oriented computing, where the services rendered by the agents are executed in a distributed manner and on demand, endows these systems with greater levels of flexibility, scalability and a better distribution of computational load, which in centralized systems (Shen & Norrie, 1998; Camarinha-Matos & Afsarmanesh, 2007; Voos, 2006). SCODA (Specialized and Distributed Agent COmmunities) (Román *et al*., 2013) uses distributed services so that its execution is optimized by the agents that make up this architecture. It also presents a high tolerance against faults with regard to the monitoring of the services it offers, in a way that provides a control added on these services, and therefore on the entire system.

Advantages	**Disadvantages**
• Lower computational load on agents • Capacity to run multiple services in parallel • Reuse of functionalities • Independence of the programming language • Management and error control independent of the agents	• Need for robust communications • Need to Know the location of the service

Table 1. Advantages and disadvantages of the distribution of services in multi-agent Systems

Table 1 shows the advantages and disadvantages that have the distribution of services in a multi-agent system. It can be seen that have a higher weight the advantages over the disadvantages, since the yield on the multi-agent system increases because the computational load in its members is lower, in addition to providing a separate error handling. With regard to the need for robust communications, is an external factor to the system because it generally depends on the physical environment in which they act. The need to know the location of the service is mitigated through publishing directories of services provided by the multi-agent system itself. In SCODA there are different directories where you publish the location of the services they offer their smart Communities Specialized (Román *et al*., 2013), in addition to optimize the requests of services through the capacity to calculate the number of replicas of the services in real time.

From the distribution of services proposed in SCODA is implemented an improvement in regard to the distribution of the computational load on the agents and services, as well as on its control. To demonstrate this improvement in the distribution of services is carried out a case of study in which proposes the prediction on different time series of demand for food products. The comparison is performed with a MAS (multi-agent system) in the same conditions of implementation and deployment that the system based on SCODA. In the MAS, the agents run the required services centrally and individually.

2. Related Work

The ability to distribute the computational load in a MAS implies a division of tasks between the agents that comprise it. In addition, if the bulk of computational load runs in a distributed manner in the form of external service which provides to the system a significant advantage in the implementation of tasks whose computational cost is high, or where there is a significant amount of requests simultaneously. That is why SOA (Service-Oriented Architecture) or CORBA (Common Object Request Broker Architecture) are considered as valid models taking into account the integration and performance as important aspects due to the functionalities are created outside of the system, i.e. as external programs linked to the system (Ricci *et al.*, 2007). This type of architectures are seeking a interoperability between different systems, distribution of resources and independence of the programming languages (Cerami, 2002).

The idea that a MAS runs their services on a distributed has to take into account an effective communication between the agents and the services to run. That is why the philosophy of SOA is completely validated in its application to this type of systems.

SOA proposes a model based on a set of services between which there is effective communication that can involve from a simple exchange of data, up to the possible coordination between these services (Cerami, 2002). This philosophy is desirable in the MAS which has distributed services, so that the coordination between the agents and the services allow a coordination and control over the same. In this aspect, the capacity that allows a multi-agent system to carry out a control of their distributed services is an advantage with regard to the solution of problems in this service, as a malfunction that service (Román *et al.*, 2013; Román *et al.*, 2011).

Another very important aspect to take into account is the amount of replicas of a service required at run time required for the system to be able to attend to all the requests in an optimal way. This number of replicas of service can be calculated from the following formula:

$$N_R = \frac{N_S \times T_{MAX}}{P \times Th} \tag{1}$$

Being:
N_R, the number of replicas necessary.
N_S, the number of services requested in a period of time P.
T_{MAX}, the maximum execution time of service.
P, a period of time.
Th, the number of simultaneous threads on a service.

Depending on the number of existing replicas of a particular service, the system must have the ability to select the service that hold lower computational load, optimizing its operation.

All of these capabilities which are desirable in the MAS which runs their functions in a distributed manner are found in the SCODA components (Román *et al.*, 2013; Román *et al.*, 2011), and especially in their SIC (Specialized Intelligent Communities), which perform their services in a distributed way and in the parameters established above.

In the SIC is present the ability to distribute and select your job in the smart way. They are composed of an controller agent (*CommunityController*), and a team integrated by a planner agent and another executor (*PlannerAgent* and *ExecutorAgent*), that are instantiated at run time when it is needed and released at the end of its work. The own internal architecture of the Community and the philosophy that was being pursued in regard to the instantiation and release of agents under demand implies that SCODA is considered as an efficient architecture in regard to the distribution of its services and the management of internal resources of the platform that runs it. These SIC run as autonomous systems, so that an implementation based on SCODA can be composed of one or several SIC and make use of the services they provide separately, or use multiple SIC and their services in a scaled way and coordinated, and thus to have the capacity to solve big problems in a collaborative way.

Example of individual Specialized Intelligent Communities
1. *External Application → Request Service (translation)*
2. *Communicator → Receive Request (translation)*
3. *Communicator → Select Community(translation)*
4. *Translation Community → Run Service (translation);Send Request (translation)*
5. *External Application → Request Service (dictionary)*
6. *Communicator → RecibirSolicitud(dictionary)*
7. *Communicator → Select Community(dictionary)*
8. *Dictionary → Run Service(dictionary); Send Request (dictionary)*

Code 1. Example of implementation of Specialized Intelligent Communities working individually

Figure 1 shows the possibility that the SIC can run individually, without which there is a collaboration between them. It can be seen that there are N independent SIC, and each one of them is associated M services. The requests that come in SCODA, shall be dealt with by the SIC that is able to give an answer. Thus, Code 1 is proposed as an example where shown in an easy way what is exposing. There are two SIC, a that provides translation services and the other services of dictionary. External applications request to SCODA some translation and dictionary services without any relation, which implies that the corresponding SIC responds independently.

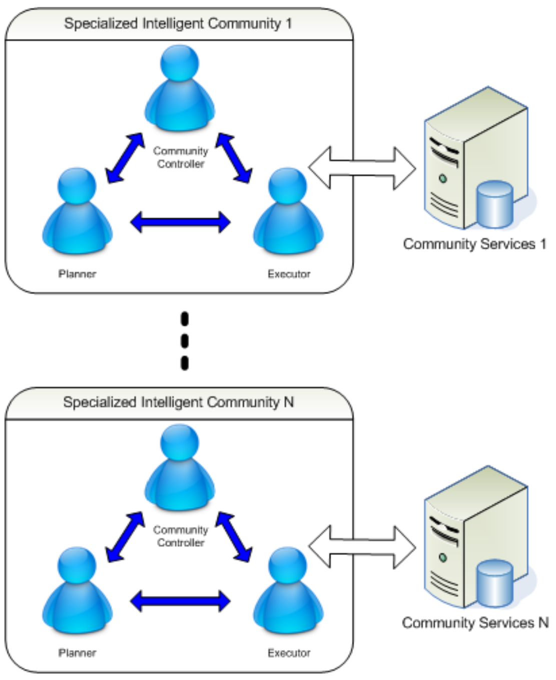


Figure 1. Specialized Intelligent Communities working independently

Figure 2 shows the detail of several SIC which collaborate in a scaled way to get results that individually would not achieve. As an example we propose the SIC of the previous case, running translation services and other services of dictionary. In this case we have an external application will request a service of dictionary and translation of jointly, which implies that there has to be a collaboration between the two SIC to achieve meet the request made by the external application.

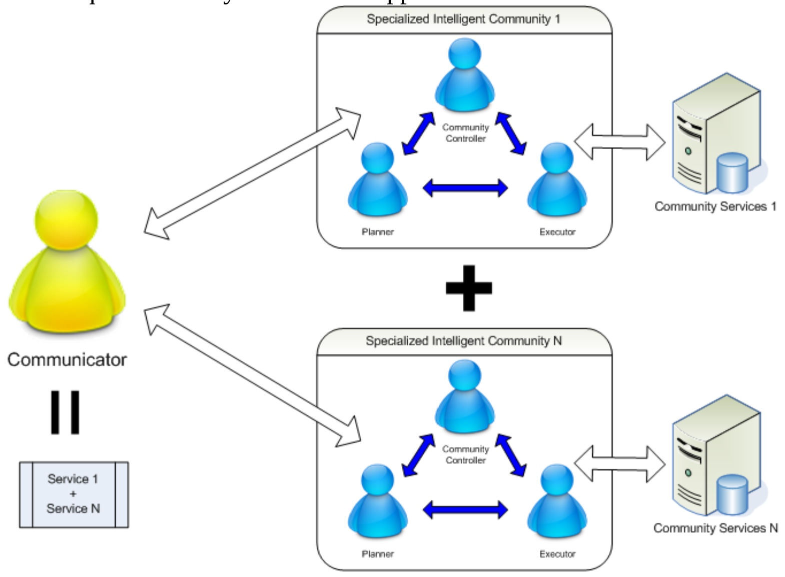


Figure 2. Specialized Intelligent Communities working in a scaled way

Code 2 shows the detailed collaboration between the SIC that offer translation and dictionary services. This is a scaled collaboration planned by the CommunicatorAgent, which can select the Specialized Intelligent Communities needed to achieve the global objective using a type of *DirectoryFacilitator* that stores the SIC and their respective information for future use.

Example of collaborating Specialized Intelligent Communities
1. External Application → Request Service(dictionary+translation)
2. Communicator →Response(dictionary+translation)
3. Communicator → Select Community(dictionary)
4. Community Dictionary → Run Service (dictionary);Send Response(dictionary)
5. Communicator →Response(dictionary)
6. Communicator → Select Community (translation, DictionaryResponse)
7. Community Translation → Run Service (translation, DictionaryResponse); Send Respuestas(translation, Respuesta dictionary)
8. Communicator → Enviar Respuesta (dictionary+translation)

Code 2. Example of Specialized Intelligent Communities working in collaboration

From the explained above concerning the structure and functioning of the SIC, in the following point introduces the distribution of services in the SIC and explain its management of control as well as the fault tolerance on these services.

3. Services Distribution in Specialized Intelligent Communities

The services of the SIC represent the functionalities it offers any system based on SCODA, being these services to the communities where is the computational processing offered by the system. These systems are accessed in a distributed and ubiquitous way. These systems are accessed in a distributed and ubiquitous manner so that the agents of SCODA are released of computational load, so that, SCODA is considered as a lightweight architecture. The services of the Community are permanently assets to receive requests from its corresponding SIC and are designed for your access remotely via sockets, due to its ease of deployment and its good result in other jobs as they are (Sunwook *et al.*, 2009; Balaji *et al.*, 2006; Douglas & Pai, 2006). The services are organized by categories in terms of their specialization, i.e. services relating to a particular functionality are grouped together for their access by the specialized community services that offers these, in this way the principal aim is to find the efficiency at the time of selecting the parameters required for a correct execution, as well as greater speed to find its location.

The services of the Community, as well as another type of distributed services such as Web Services and Services Oriented Architecture (Papazoglou *et al.*, 2007) must have mechanisms for publication and discoveries of services. Also have to possess an updated directory of the same so that they are accessible when you require (Leymann, 2002) (Papazoglou & van den Heuvel, 2006), this is the reason why SCODA has a flexible services directory, from where the services can be invoked, modified, added and deleted dynamically through the SIC that provide them, however, insertion, modification or elimination of the services of the Community is done manually for safety reasons (López, Luck, & d'Inverno, 2006).

Community	Invocation	Predicate	Host	Port	Extra	Description
[Community1]	[invocation]	[type=1; urldatos=data1…]	[community1]	[8998]	[empty]	[description of services]
[Community2]	[invocation]	[type=3; urldatos=data1…]	[community3]	[8787]	[empty]	[description of service]

Table 2. Structure of the service directory for the Specialized Intelligent Communities in SCODA

Table 2 shows the structure of the directory of the services of the SIC. This directory includes data on the name of the Community, how is the service invoking method, the predicate with the required parameters, the host where the service is located and its access port and the description of the service. In addition displays a field with the description of each of the services.

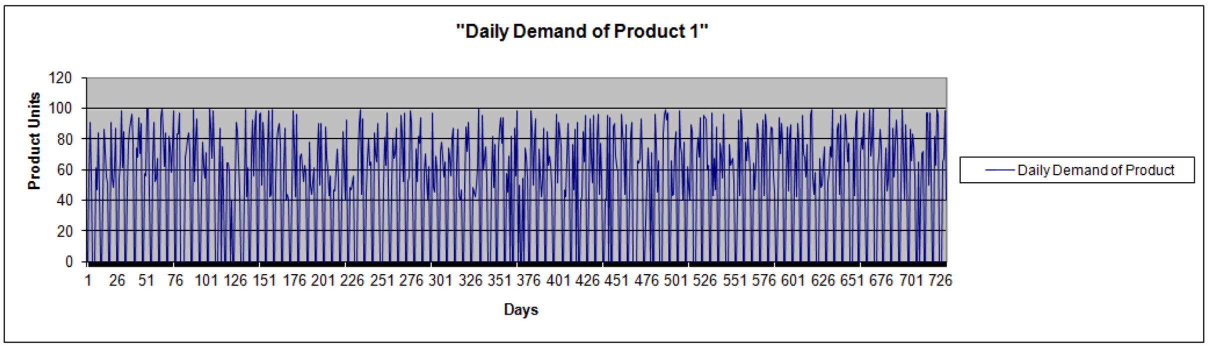
The management of the quality of these services is done by one of the agents that provides SCODA, as is the *QualityAgent*. This agent has the ability to monitor at run time the operation of the services in a way that detects any abnormality in your operation. In addition this agent has the ability to restart a particular service if, in effect, detects that is inactive or that the requests that are made to this service are not answered correctly. From these capabilities offered by the SCODA architecture is carried out an improvement in the Quality of Service (QoS) at the level of fault tolerance on the services that they provide.

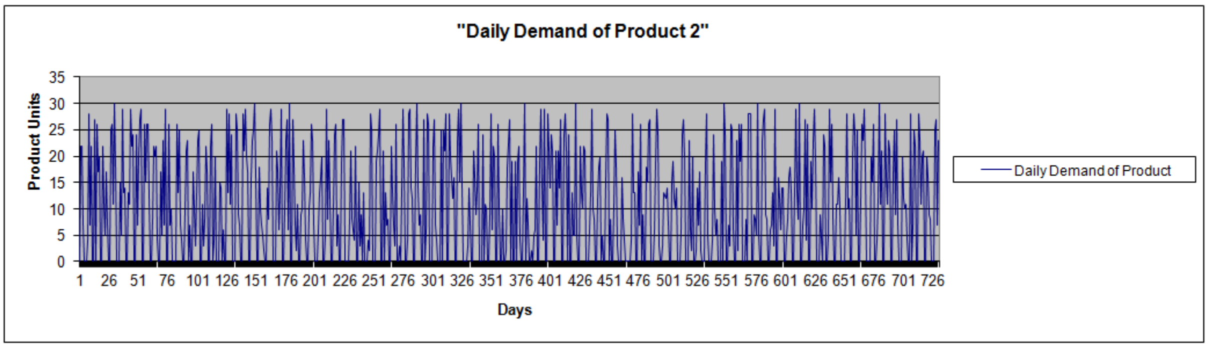
Another of the proposed improvements lies in the efficiency of the distribution of services that perform the SIC. This improvement will be demonstrated in the case of study that is presented in the following point.

4. Case of Study

The prediction of values belonging to time series of various kinds is a widely discussed problem in the literature (Atiya *et al.*, 1999; Corchado & Lees, 2001; Corchado & Aiken, 2002; Martin-Merino & Román, 2006a; Martin-Merino & Román, 2006b). There are various techniques used to predict time series, from traditional statistical models such as ARIMA and the transfer functions to nonlinear models based on artificial neural networks (Velásquez *et al.*, 2010).

The use of one or the other technique has to do with the ability that it has to process the kind of temporary series selected, thus, in our case the time series corresponding to the demand of food products, is very diverse in terms of the type of product. If for example we take the time series of the orders daily that are performed a product "Product 1", we can see that the series has a periodicity in which their yearly maximum is reached at Christmas and Holy Week, however other products do not present or these maximums neither this periodicity, so it is necessary to use any technique that has a high capacity of generalization of non-linear series. In Figure 3 presents the series of demand for three different products that are sold in the period between the 01/01/2009 until 30/09/2011.





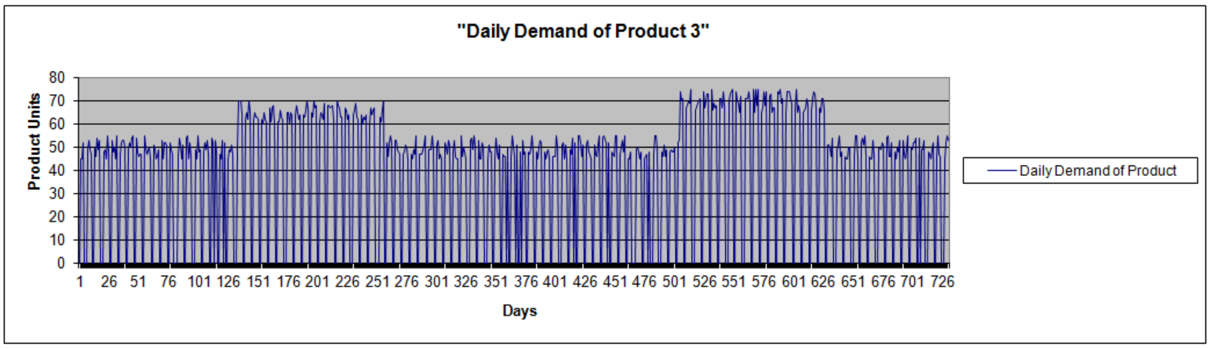


Figure 3. Series of demand for different products

As we can see in Figure 3 the series of the demand for these three products is not linear. In addition, the first two have a character that is not newspaper and the product units defendant are completely different. It is observed in the three charts that there are a great variability of the data on the basis of the day of the year as the weekends and holidays, the demand for products is zero.

In view of these graphs can be deduced that there is a great difference between each one of the products that the company offers. It is also important to point out the difficulty that will be the time to predict the demand for the different products due to the nonlinear character and variability of these series, which implies that the selected technique has to have a high you capacity of generalization.

Within the non-linear techniques, the Multilayer Perceptron (MLP) has been widely used in the literature for the prediction of non-linear series, since they have the ability to approximate any continuous function defined in a compact domain, however, its specification is essentially based on heuristics criteria and expert judgment of modeler, so that, this process is based on a set of critical steps which affect the final result of the model, in terms of their adjustment to the historical data and its capacity of generalization in order to make predictions (Velásquez *et al.*, 2010).

On the other hand, SVM (Support Vector Machines) (Vapnik, 1995; Vapnik *et al.*, 1996) are a type of neural networks which were designed in its origins to carry out the resolution of non-linear problems of classification (Burges, 1998; Belousov *et al.*, 2002). In recent years, SVM have been applied to problems of regression and prediction of temporary series (Vapnik, 1995; Vapnik *et al.*, 1996) due to its great capacity of generalization (Schölkopf, 2002), due to its structure and the methodology used for the estimation of its parameters.

In (Velásquez *et al.*, 2010) are shown several advantages of the SVM above other non-linear techniques, including the Multilayer Perceptron:

- Its specification is based on the principle of minimizing structural risk that is equivalent to minimize the upper limit of the error of generalization of the model. Thus, the estimate of a SVM seeks to find both the optimal structure of the model and the optimal values of its parameters, allowing a greater capacity of generalization of the model. The estimation of the parameters of other traditional models, such as the MLP, is based on the application of the principle of minimizing empirical risk which depends primarily of adjustment to the historical data, taking the structure of the model as an input parameter.

- The estimation of the parameters of a SVM is equivalent to the solution of a quadratic programming model with linear restrictions; this implies that the optimal solution is global and only, which represents a clear advantage over models as the MLP that are characterized by multiple local minimums.

However, we must have in account that we need to make the setting of a series of parameters to address each problem in particular. This implies that transforms in a technique with a high computational cost, since the adjustment of the parameters is usually performed through cross-validation, where additionally, you must choose the criteria of this same (Martín-Merino & Román 2006a; Velásquez *et al.*, 2010).

In (Vapnik, 1995; Vapnik *et al.*, 1996; Velásquez *et al.*, 2010) gives a detailed description of the SVM, both of their mathematical formulation as the possible applications that can have. In our case we will adopt a regression approach to carry out the predictions of the different products and with this check the operation of the multi-agent system and its effectiveness, having in account the specifications proposed in the literature quoted above at the time of its implementation and to estimate their parameters. It is precisely this setting parameters which is done dynamically by the SIC having in account the type of product which is necessary to predict its serie of demand.

On the basis of the definition of the problem proposed are executed requests for prediction of different types of product on the system based on SCODA, and on a MAS centralized. Are the systems which must select the prediction model and its parameters in function of the type of product to predict, and thus be able to compare the efficiency of the response.

It is presented a comparison at the structural level of SCODA with the MAS developed for the case. This comparison is made on the basis of a series of features that are related to the functionality of the architecture regardless of the platform for deployment. The MAS implemented consists of a CoordinatorAgent that exercises of coordinator, and an agent ForecastAgent that makes predictions and selects the appropriate model for this. Both systems are deployed on JADEX (Pokahr *et al.*, 2003; Pokahr *et al.*, 2007) and on the same machine where the system is deployed with SCODA. Table 3 shows the differences at the structural level between the selected architectures for the resolution of the problem.

Characteristics	MAS	SCODA
Type of architecture	Based on agents Centralized	Based on agents Distributed
Platform	JADEX	JADEX
Communication Protocols	ACL HTTP	ACL HTTP TCP/IP
Number of permanent agents	2	3+2
Programming Language	JAVA	JAVA
Robustness	There is no incident management	Agent dedicated to incident management Statistics of operation
Reuse of resources	Functionality built into the architecture	Reusable services Reusable Architecture
Resource Distribution	Each agent performs a full-service	Distribution and replication of services The agents perform specialized tasks Instantiation at runtime
Computational load of the system	Centralized	Distributed

Table 3. Comparative of SCODA with a MAS system at the structural level

Analyzing the Table 3 the MAS implements an architecture based on centralized agents and on the contrary in SCODA are distributed. The advantage that offers a distributed architecture is the ability to allocate resources more efficiently. With regard to the protocols of communication used, both architectures use HTTP (RFC 2616, 1999) as external communication and ACL (FIPA, 2005) for internal communication between the agents. However, SCODA uses TCP/IP sockets for communication with external services, which implies that these services will be performed on different machines. Something to highlight is that the MAS is constituted of 2 permanent agents and SCODA of 3+2. In SCODA there is an agent that is responsible for carrying out the management of incidents and errors that occur at run time. SCODA enables complete management of incidents and has a high fault tolerance so that if one of the agents is not running properly, the system would not fall, is the agent which has the ability to boot at run time other agents which solves the error. This feature is absent in the MAS, forcing manually restart the agents that have undergone any interruption. In addition, SCODA provides a daily operations through the that you can monitor the operation of the system and manage the replication capability of services.

In regard to the reuse and distribution of resources, the MAS developed integrates all the functionality in its architecture, so that the execution is performed centrally, thus using resources on the same machine where it is deployed its architecture, and forcing to reprogram the agents for any other development. On the contrary, SCODA distributes its resources, that implies that its architecture is standard on any problem, i.e. always consists of the same agents, and these agents always have the same structure. In addition to the distribution of services allows them to be reused in other developments individually or collaborative.

Finally is made reference to the computational load that exists in the implementation of both architectures. The architecture of the MAS is centralized, forcing consuming the resources needed on the machine where the system is running, and therefore, in operations that require a high consumption of resources covers a very high percentage of the same, not allowing the execution of another type of task. On the contrary SCODA distributes the computational load so that puts this computational load on external services. The existing burden on the machine where it is running SCODA is independent of whether the required tasks require an excessive use of resources, since the implementation of the same is done in a distributed manner and independent.

4.1. Results in the Case of Study

To realize a study of the efficiency of both systems are made a set of requests simultaneously on SCODA requesting the services of prediction provided and then perform the same operations on the MAS implemented in this case of study. In addition, it discusses several points on these requests by performing the following tests:

- Good requests.
- Requests with faults in the input parameters.
- Requests by eliminating one of the responsible agents.
- QoS at the level of control and correction of incidents in the system.

Nº Requests	Response Time MAS (ms)	Response Time SCODA (ms)	Time by Request MAS (ms)	Time by Request SCODA (ms)
1	5571	4958	5571	4958
5	26542	18125	5308,4	3625
10	50021	37521	5002,1	3752,10
21	ERROR	97327	ERROR	4634,62
36	ERROR	154025	ERROR	4278,47

Table 4. Response Time (ms) for simultaneous requests in a MAS and in SCODA

The Table 4 shows the response times of the MAS implemented against those of SCODA for requests of services of prediction for products of different typology. As noted, the time for a petition is around the 5000 ms in both cases, however as you increase the simultaneous requests SCODA responds more quickly due to the execution of requests are made through different threads, allowing you to manage more quickly the results of the requests. Something important is that from 21 simultaneous requests the MAS generates an error in the JADEX platform, which implies that the accumulation of such requests saturates the capacity of the agent. The Figure 4 and 5 graphically show the above.

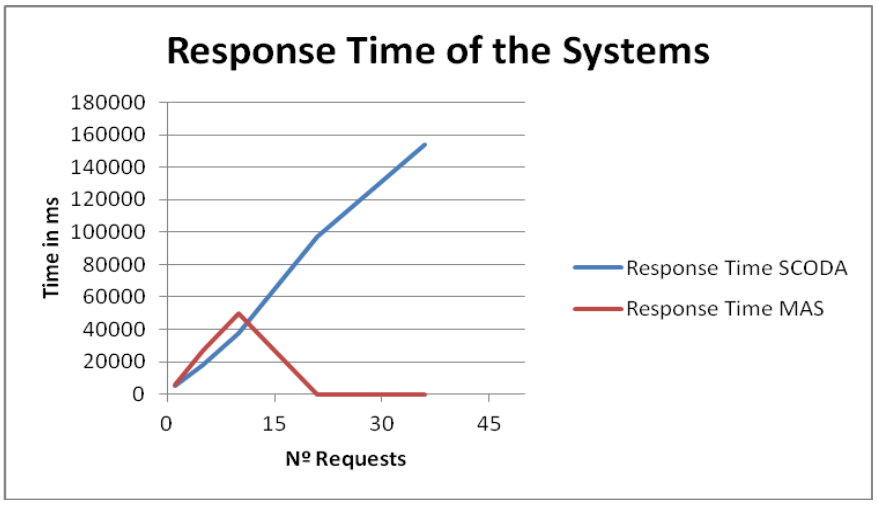


Figure 4. Graphic Detail of response (ms) on simultaneous requests for a MAS and SCODA

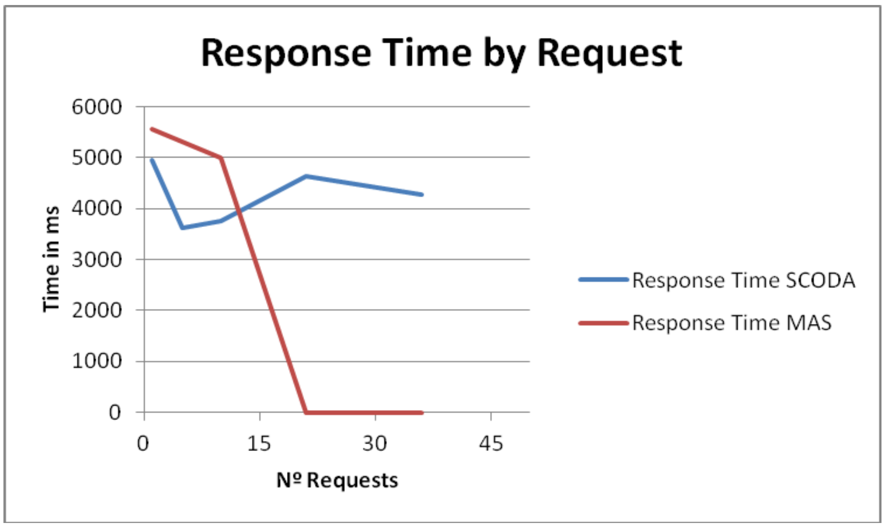


Figure 5. Graphic detail response (ms) by a request, on simultaneous requests for a MAS and SCODA

As specified at the beginning of this point, the next test that is being performed is the perform of requests on the same terms as the previous test, but introducing various faults in the required parameters, services which do not exist, etc., randomly, to check the reaction of both systems. Table 5 shows that the results obtained follow the same line as in the previous case, which implies that at both systems the existence of errors does not vary virtually the response times. In the Figure 6 and 7, shows the foregoing of a graphical way.

Nº Requests	Response Time MAS (ms)	Response Time SCODA (ms)	Time by Request MAS (ms)	Time by Request SCODA (ms)
1	5471	5058	5471	5058
5	27534	18534	5506,8	3706,8
10	49929	37965	4992,9	3796,50
21	ERROR	97278	ERROR	4632,29
36	ERROR	155025	ERROR	4306,25

Table 5. Response Time (ms) for simultaneous requests in a MAS and in SCODA with errors

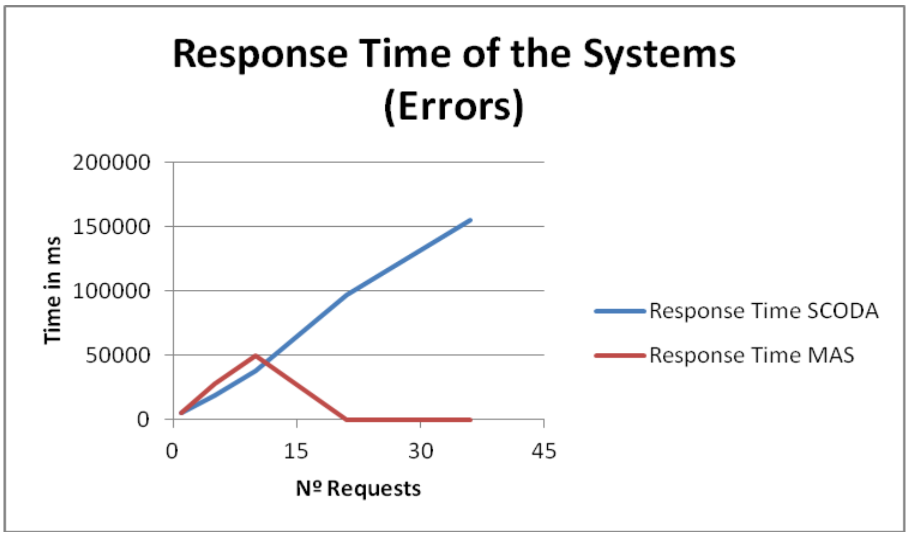


Figure 6. Graphic Detail of response (ms) on simultaneous requests for a MAS and SCODA with errors

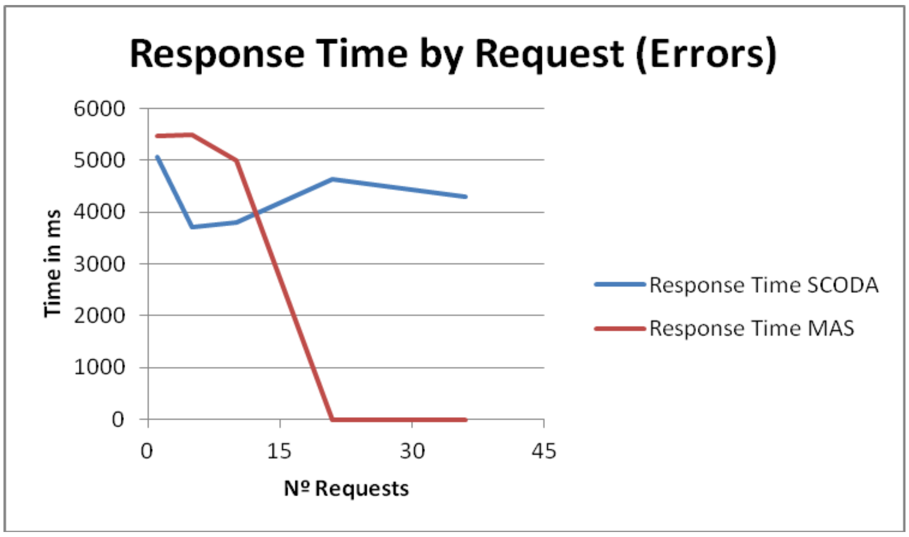


Figure 7. Graphic detail response (ms) by a request, on simultaneous requests for a MAS and SCODA with errors

The following test that is performed on both systems is to eliminate one of the agents responsible for servicing the requests and compare the reaction of the systems. As soon as the MAS, directly removes the agent that implements the functionality of execute the predictions, implies that the request is not resolved and is returned an error because the agent can not resolve the request.

In SCODA is deleted the *CommunityController* responsible for predictions and is the own architecture which has the ability to detect problems in the agents that comprise it and restart those agents. Therefore, the *CommunityController* is restarted and the request of the service runs correctly.

Finally the comparison between the two systems is by the management of the QoS to incident monitoring level and statistics of successes and failures in the system. In the second type of tests that were conducted by introducing faults in the input parameters, response times were similar, however the MAS does not give any information on the number of requests serviced properly, the type of failure that has occurred, etc., and however SCODA carries a complete control of the incidents that occurred in the system through the *QualityAgent*.

The tests performed and the proposed results above lead to demonstrate the existing improvement in the distribution of services that offers SCODA. With regard to its behavior in situations where it is required for an effective response, SCODA has obtained better results than the system MAS developed.

5. Conclusions

The distribution of services in multi-agent systems (MAS) allows to execute functionality that provide these systems more efficiently due to that the computational load required to perform this functionality does not reside in the agents as such, but in different locations in a distributed manner.

The SCODA architecture provides its functionality in a distributed way by the Specialized Intelligent Communities (SIC). These functionalities are planned internally by the SIC and executed remotely by them. For each request, from the beginning of the process creates a thread so that these requests can be answered asynchronously and do not generate an overload to the agents.

On the basis of the comparative carried out between a MAS with ability to predict time series of food products and a system based on SCODA for the same work has been carried out a comparison on aspects of computational efficiency, as well as in the QoS in regard to the control and monitoring of communications and of the services offered by the both systems. Additionally, the architecture provides a mechanism for dynamically calculate the number of replicas of service necessary to respond to all the requests that are carried out in an optimal way.

SCODA attends correctly all requests for an efficient way improving the times that are reflected in the MAS. In addition SCODA there is no memory overflow error as in the MAS where from the 21 concurrent requests generates an error in JADEX as deployment platform. In addition SCODA provides a monitoring of communications between agents and manages the incidences in the services and on their own agents of the system.

6. References

Atiya, A., El-Shoura, S., Shaheen, S., & El-Sherif, M. (1999). A comparison between neural network forecasting techniques-case study: river flow forecasting, Neural Networks. *IEEE Transactions on Neural Networks , 2* (10), 402-409.

Balaji, P., Bhagvat, S., Jin, H.-W., & Panda, D. K. (2006). Asynchronous Zero-copy Communication for Synchronous Sockets in the Sockets Direct Protocol (SDP) over InfiniBand. *International Parallel and Distributed Processing Symposium (IPDPS 2006).*

Belousov, A., Verzakov, S., & Von Frese, J. (2002). A flexible classification approach with optimal generalisation performance: support vector machines. *Chemometrics and Intelligent Laboratory Systems* (64), 15-25.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery , 2* (2), 121-167.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2007). A Comprehensive Modeling Framework for Collaborative Networked Organizations. *Journal of Intelligent Manufacturing , 5* (18), 529-542.

Cerami, E. (2002). *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL (1st ed.).* O'Reilly & Associates, Inc.

Corchado, J., & Aiken, J. (2002). Hybrid Artificial Intelligence Methods in Oceanographic Forecasting Models. *IEEE SMC Transactions Part C* (32), 307-313.

Corchado, J., & Lees, B. (2001). A hybrid case-based model for forecasting. *Applied Artificial Intelligence: An International Journal* (15), 105-127.

Douglas, C., & Pai, V. (2006). Seekable sockets: a mechanism to reduce copy overheads in TCP-based messaging. *International Parallel and Distributed Processing Symposium (IPDPS 2006).*

FIPA. (2005). Foundation for Intelligent Physical Agents. Retrieved 7 14, 2006, from http://www.fipa.org

Leymann, F., Roller, D., & Schmidt, M.-T. (2002). Web services and business process management. *IBM Systems Journal , 2* (41), 198-211.

López, F., Luck, M., & d'Inverno, M. (2006). A normative framework for agent-based systems. *Computational and Mathematical Organization Theory* (12), 227-250.

Martín-Merino, M., & Román, J. (2006a). A New SOM Algorithm for Electricity Load Forecasting. *ICONIP*, (págs. 995-1003).

Martín-Merino, M., & Román, J. (2006b). Electricity Load Forecasting Using Self Organizing Maps. *ICANN. 4132*, págs. 709-716. LNCS.

Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges. *Computer , 11* (40), 38-45.

Papazoglou, M., & Georgakapoulos, G. (2003). Introduction to the Special Issue about Service-Oriented Computing. *10* (46), 24-29.

Papazoglou, M., & van den Heuvel, W. (2006). Service-Oriented Design and Development Methodology. *In Proc. Int. J. of Web Engineering and Technology (IJWET).*

Pokahr, A., Braubach, L., & Lamersdorf, W. (2003). Jadex: Implementing a BDI-Infrastructure for JADE Agents. En *In EXP - in search of innovation (Special Issue on JADE)* (págs. 76-85).

Pokahr, A., Braubach, L., Walczak, A., & Lamersdorf, W. (2007). Jadex - Engineering Goal-Oriented Agents. En *In Developing Multi-Agent Systems with JADE* (págs. 254-258). Wiley & Sons Eds.

RFC 2616. (1999). Hypertext Transfer Protocol -- HTTP/1.1.The Internet Society 1999.

Ricci, A., Buda, C., & Zaghini, N. (2007). An agent-oriented programming model for SOA & web services. *In 5th IEEE International Conference on Industrial Informatics (INDIN'07),* (págs. 1059-1064). Viena.

Román, J., Rodríguez, S., & Corchado, J. (2013). Distributed and Specialized Agent Communities. *Trends in Practical Applications of Agents and Multiagents Systems, 221*, págs. 33-40.

Román, J., Tapia, D., & Corchado, J. (2011). SCODA para el Desarrollo de Sistemas Multiagente. *Revista Ibérica de Sistemas y Tecnologías de Información* (8), 25-38.

Rosen, M., Lublinsky, B., Smith, K., & Balcer, M. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies.* Wiley.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels.* Cambridge, MA.: MIT Press.

Shen, W., & Norrie, D. H. (1998). An agent-based approach for distributed manufacturing and supply chain management. En *Globalization of manufacturing in the digital communications era of the 21st century* (págs. 579–590).

Sunwook, K., Chanho, P., Seongwoon, K., & Yongwha, C. (2009). The offloading of socket information for TCP/IP offload engine. *In 11th International Conference on Advanced Communication Technology (ICACT 2009), 1*, págs. 826-831.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* N.Y.: Springer.

Vapnik, V., Golowich, S., & Smola, A. (1996). Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems* (9), 281-287.

Velásquez, J., Olaya, Y., & Franco, C. (2010). Predicción de Series Temporales usando Máquinas de Vectores Soporte. *Ingeniare. Revista chilena de ingeniería* (18), 64-75.

Voos, H. (2006). Agent-Based Distributed Resource Allocation in Technical Dynamic Systems. *In Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications* (págs. 157-162). IEEE Computer Society.

Zimmermann, O., Schlimm, N., Waller, G., & Pestel, M. (2005). Analysis and Design Techniques for Service-Oriented Development and Integration. *INFORMATIK* , 606-611