Improving the Distribution of Services in MAS

Jesús A. Román[1], Sara Rodríguez[2(✉)], and Fernando de la Prieta[2]

[1] Department of Computer Science and Automatic, EPS of Zamora, University of Salamanca, Avda. Cardenal Cisneros, 34, 49022 Zamora, Spain
zjarg@usal.es

[2] Department of Computer Science and Automatic, University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain
{srg,fer}@usal.es

Abstract. One way to reduce the computational load of the agents is the distribution of their services. To achieve this goal, the functionality of a MAS (multi-agent system) should not reside in the agents themselves, but ubiquitously be distributed so that allows the system to perform tasks in parallel avoiding an additional computational cost. The distribution of services that offers SCODA (Distributed 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 improvement 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.

Keywords: Multi-agent systems · Distributed services · Specialized communities · SOA

1 Introduction

The service-oriented computing (SOC) uses the services that allow the development of distributed applications quickly, operable, scalable and massive [15]. 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) [15]. 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 [16]. 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 [17]. The implementation of SOA is strongly linked to Web Services, these being a way to develop these architectures, but not the only [24, 32]. The development of multi-agent Systems (MAS) based on service-oriented computing[21], 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 [5, 26, 31]. SCODA

© Springer International Publishing Switzerland 2016
J. Bajo et al. (Eds.): PAAMS 2016 Workshops, CCIS 616, pp. 37–46, 2016.
DOI: 10.1007/978-3-319-39387-2_4

(Specialized and Distributed Agent COmmunities) [22] 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. 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. The paper is structured as follow: the next section shows a background of services distribution in specialized intelligent communities; in Sect. 3 a case of study is detailed and the conclusions are described in Sect. 4.

2 Background

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 [6]. 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 [22, 23].

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 [22, 23], 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.

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 [2, 9, 27]. 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 [15] 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 [11, 17], 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 [12].

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.

3 Case of Study

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 [30]. The prediction of values belonging to time series of various kinds is a widely discussed problem in the literature [1, 7, 8, 13, 14].

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 Fig. 1 presents the series of demand for three different products that are sold in the period between the 01/01/2009 until 30/09/2011.

As we can see in Fig. 1 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. In this way the prediction is realized by Support Vector Machines (SVM) [3, 4, 25, 28 – 30], due to its high capacity of generalization among other characteristics.

A comparison is presented 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 [18, 19] and on the same machine where the

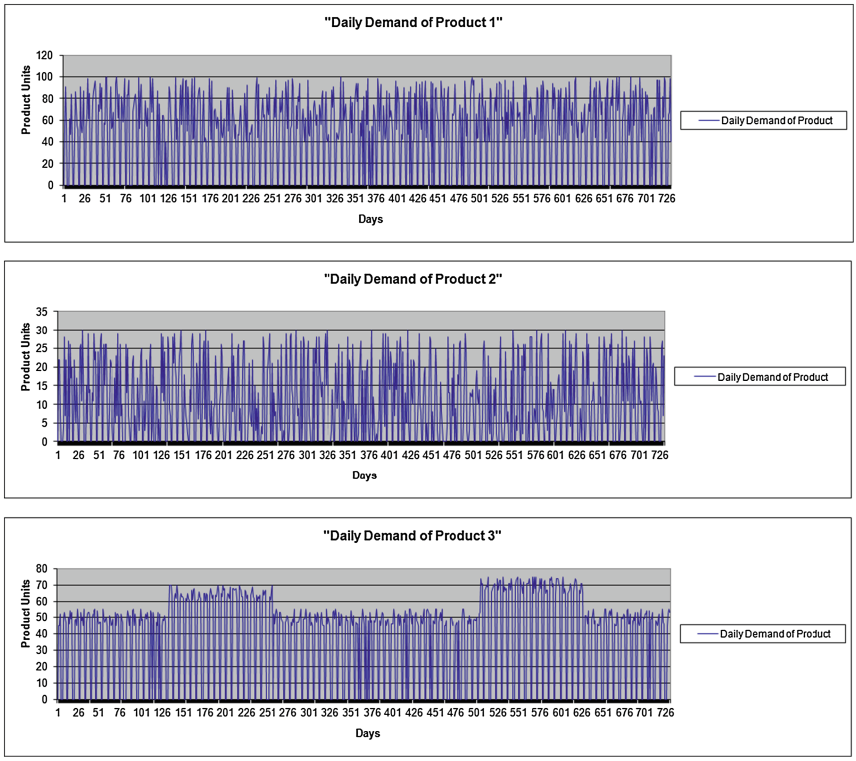

Fig. 1. Series of demand for different products

system is deployed with SCODA. Table 1 shows the differences at the structural level between the selected architectures for the resolution of the problem.

Analyzing the Table 1 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 [20] as external communication and ACL [10] 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 that it is possible to monitor the operation of the system and manage the replication capability of services.

Table 1. Comparative of SCODA with a MAS system at the structural level

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

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.

3.1 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: (i) good requests; (ii) requests with faults in the input parameters; (iii) requests by eliminating one of the responsible agents; (iv) QoS at the level of control and correction of incidents in the system.

The Table 2 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.

Table 2. Response time (ms) for simultaneous requests in a MAS and in SCODA

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

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 3 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.

Table 3. Response time (ms) for simultaneous requests in a MAS and in SCODA with errors

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

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 cannot 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.

4 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.

Acknowledgements. This work has been carried out by the project EKRUCAmI: Europe-Korea Research on Ubiquitous Computing and Ambient Intelligence. Ref. 318878. FP7-PEOPLE-2012-IRSES.

References

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

2. Balaji, P., Bhagvat, S., Jin, H.-W., Panda, D.K.: Asynchronous zero-copy communication for synchronous sockets in the sockets direct protocol (SDP) over InfiniBand. In: International Parallel and Distributed Processing Symposium (IPDPS 2006) (2006)
3. Belousov, A., Verzakov, S., Von Frese, J.: A flexible classification approach with optimal generalisation performance: support vector machines. Chemometr. Intell. Lab. Syst. **64**, 15–25 (2002)
4. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2), 121–167 (1998)
5. Camarinha-Matos, L.M., Afsarmanesh, H.: A comprehensive modeling framework for collaborative networked organizations. J. Intell. Manuf. **5**(18), 529–542 (2007)
6. Cerami, E.: Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, 1st edn. O'Reilly & Associates Inc., Sebastopol (2002)
7. Corchado, J., Aiken, J.: Hybrid artificial intelligence methods in oceanographic forecasting models. IEEE SMC Trans. Part C **32**, 307–313 (2002)
8. Corchado, J., Lees, B.: A hybrid case-based model for forecasting. Appl. Artif. Intell. Int. J. **15**, 105–127 (2001)
9. Douglas, C., Pai, V.: Seekable sockets: a mechanism to reduce copy overheads in TCP-based messaging. In: International Parallel and Distributed Processing Symposium (IPDPS 2006) (2006)
10. FIPA. Foundation for Intelligent Physical Agents. http://www.fipa.org (2005). Accessed 14 Aug 2006
11. Leymann, F., Roller, D., Schmidt, M.-T.: Web services and business process management. IBM Syst. J. **2**(41), 198–211 (2002)
12. López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. Comput. Math. Organ. Theory **12**, 227–250 (2006)
13. Martín-Merino, M., Román, J.: A new SOM algorithm for electricity load forecasting. In: King, I., Wang, J., Chan, L.-W., Wang, D.L. (eds.) ICONIP 2006. LNCS, vol. 4232, pp. 995–1003. Springer, Heidelberg (2006)
14. Martín-Merino, M., Román, J.: Electricity load forecasting using self organizing maps. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4132, pp. 709–716. Springer, Heidelberg (2006)
15. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. Computer **11**(40), 38–45 (2007)
16. Papazoglou, M., Georgakapoulos, G.: Introduction to the special issue about service-oriented computing. Commun. ACM **46**(10), 24–29 (2003)
17. Papazoglou, M., van den Heuvel, W.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. **2**(4), 412–442 (2006)
18. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: Implementing a BDI-infrastructure for JADE agents. In: EXP - In Search of Innovation (Special Issue on JADE), pp. 76–85 (2003)
19. Pokahr, A., Braubach, L., Walczak, A., Lamersdorf, W.: Jadex - engineering goal-oriented agents. In: Developing Multi-agent Systems with JADE, pp. 254–258. Wiley, Hoboken (2007)
20. RFC 2616. Hypertext Transfer Protocol – HTTP/1.1. The Internet Society (1999)
21. Ricci, A., Buda, C., Zaghini, N.: An agent-oriented programming model for SOA & web services. In: 5th IEEE International Conference on Industrial Informatics (INDIN 2007), pp. 1059–1064, Viena (2007)
22. Román, J., Rodríguez, S., Corchado, J.: Distributed and specialized agent communities. In: Pérez, J.B., et al. (eds.) Trends in Practical Applications of Agents and Multiagent Systems. AISC, vol. 221, pp. 33–40. Springer, Berlin (2013)

23. Román, J., Tapia, D., Corchado, J.: SCODA para el Desarrollo de Sistemas Multiagente. Revista Ibérica de Sistemas y Tecnologías de Información **8**, 25–38 (2011)
24. Rosen, M., Lublinsky, B., Smith, K., Balcer, M.: Applied SOA: Service-Oriented Architecture and Design Strategies. Wiley, Hoboken (2008)
25. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
26. Shen, W., Norrie, D.H.: An agent-based approach for distributed manufacturing and supply chain management. In: Globalization of Manufacturing in the Digital Communications Era of the 21st Century, pp. 579–590 (1998)
27. Sunwook, K., Chanho, P., Seongwoon, K., Yongwha, C.: The offloading of socket information for TCP/IP offload engine. In: 11th International Conference on Advanced Communication Technology (ICACT 2009), vol. 1, pp. 826–831 (2009)
28. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
29. Vapnik, V., Golowich, S., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. Adv. Neural Inf. Process. Syst. **9**, 281–287 (1996)
30. Velásquez, J., Olaya, Y., Franco, C.: Predicción de Series Temporales usando Máquinas de Vectores Soporte. Ingeniare. Revista chilena de ingeniería **18**, 64–75 (2010)
31. Voos, H.: Agent-based distributed resource allocation in technical dynamic systems. In: Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications, pp. 157–162. IEEE Computer Society (2006)
32. Zimmermann, O., Schlimm, N., Waller, G., Pestel, M.: Analysis and design techniques for service-oriented development and integration. In: INFORMATIK, pp. 606–611 (2005)