Journal of Computer Science 2 (6): 521-527, 2006
ISSN 1549-3636
© 2006 Science Publications

Autonomic Middleware Services for Just-In-Time Grid Services Provisioning

[1,2]Wail M. Omar, [1]A. Taleb-Bendiab and [1]Yasir Karam
[1]Liverpool John Moores University, Liverpool, UK
[2]Sohar University, Sohar, Oman

Abstract: The advent of widespread distributed computing environment, such as information systems and computational grids computing has enabled a new generation of applications that are based on seamless access, aggregation and interaction. The dramatic side of the story is a strong presence of the plea that those decentralized Grids are potentially affected by a number of primitives derived from their anatomy, in that, they are inherently large, complex, heterogeneous and dynamic, globally aggregating a large number of independent computing and communication resources. This has clearly exposed an essential exigency for a vital change to how these applications are developed and managed, which has motivated researchers to consider other techniques used by biological systems to deal with such problems. This is referred to as autonomic computing, which is defined by Ganek and Corbi[1] "… *as a collection and integration of technologies that enable the creation of an information technology computing infrastructure for the next era of computing—e-business on demand* …". This study presents a computational model to support just-in-time and on-demand services for autonomic computing. Service reservation and job schedule systems are employed in this model to estimate the required services in advanced. Intelligent classification is utilized to cluster consumers into groups sharing the same behaviour and hence offer the required services for each consumer in advance, according to the group's usage pattern of application services. To this end, a machine learning middleware service based on Self-Organizing Maps (SOM) is designed, developed and implemented to carry out the intelligent classification for the autonomic computing. A case-study scenario of intelligent connected homes is demonstrated in this study to show the usability of such system.

Key words: Autonomic computing, intelligent middleware, Self-Organizing Map (SOM), Grid Computing

INTRODUCTION

Over the coming years, many are anticipating grid-computing infrastructures, utilities and services to become an integral part of the future socio-economical fabric. Though, the realization of such a vision will be very much affected by a host of factors including; cost of access and ownership, reliability, dependability, interpretability, the ubiquitous nature and security of grid services. Many commentators contend that the autonomic computing vision[1] will offer a crucial paradigm shift to delegating vital functions of systems' self-management including: configuration, healing, tuning and protecting to the software itself, along with curbing the ever increasing complexity.

In particular, autonomic computing research is exploring and developing models of bio-inspired taxonomies to support distributed systems' lifetime management and unpredictability by delegating many of the system's management and maintenance tasks to the software itself including: resource management, job scheduling, services failure prediction, load-balancing, QoS, services reservations and resources discovery and availability[1-4].

A prevailing design model of autonomic computing systems is one of a model-based architecture with roots in applied control theory, for which control and autonomy are encoded as rules. Such rules are generally embedded in meta-software systems (agents)[5] exhibiting the advocated autonomic systems' capabilities such as; self-management, self-optimization, self-tuning, self-organizing, self-configuration, self-tuning and/or self-healing.

To this end, autonomic computing is expected to perform the intelligent behavior of predicting the patterns for the embedded environment, taking into consideration the boundaries (rules) and nature of the environment applications. Therefore, unsupervised machine learning techniques are here proposed to be used to develop and refine a given target system's operational regulation (rules). Hence, this study will introduce the motivations for a machine learning utility and associated middleware to capture and evolve knowledge models (sources) from the infrastructure operating systems. Such knowledge models can then be

Corresponding Author: Wail M. Omar, Faculty of Applied Sciences, Sohar University, Oman

used by our developed autonomic middleware control services to regulate (govern) a target software system. In particular, the study will focus on the presentation of an architectural model and machine learning middleware services. As an example, the Self-Organising Maps (SOM) for feature extraction, classification and clustering is applied to the on-demand home-networked appliances scenario, in which the SOM service is used to classify types of consumers in accordance to each of their respective usage models, which are expressed as classes/features and services dependencies. The generated models are invoked by self-managing infrastructure middleware services in support of an on-demand and Just-in-Time deployment and activation of required services. This is conducted in line with learnt/extracted usage models and the baseline architecture of specified services federations (assemblies) and discovering and activating additional services on-demand.

Motivations and related work: Many applications of the autonomic computing model in grid computing have been widely reported by major IT players including; Sun Microsystems, Hewlett-Packard and IBM as a way forward for a highly automated computing systems[1]. This takes the form of creating both hardware and software that can diagnose and solve network problems, thus improving high-availability, while reducing IT operation costs[1].

In particular, autonomic computing research is exploring and developing models to hide distributed systems lifetime management. This includes resource management, job scheduling, services failure prediction, load-balancing, QoS and services reservation and discovery. A prevailing design model of autonomic computing systems is one of a model and goal-based approach, where the rules are developed through a given domain analysis, data mining and/or heuristics.

An autonomic computing standard model has been well-defined by many researchers[2-5], which consists of four units inside the knowledge system and two units (sensors and effectors) outside the knowledge system responsible for collecting data and execute an actions, as shown in Fig. 1. The four units inside the knowledge system are:

* The monitor function provides the mechanisms that collect, aggregate, filter and report details collected from a managed resource.
* The analyze function provides the mechanisms that correlate and model complex situations.
* The plan function provides the mechanisms that construct the actions needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
* The execute function provides the mechanisms that control the execution of a plan with considerations for dynamic updates.

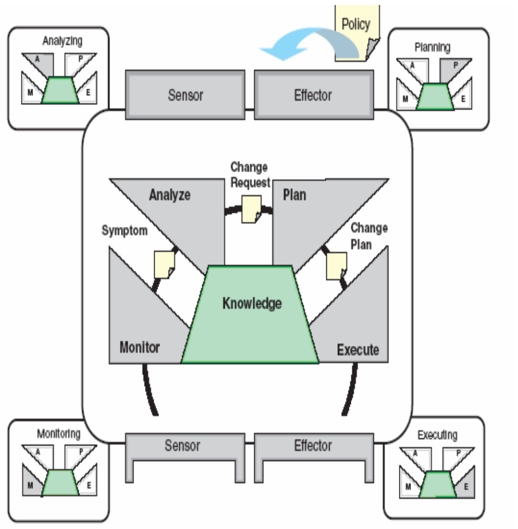

Fig. 1: Autonomic computing model[2-5]

Much research is now underway, adopting the use of machine learning to support a range of autonomic behavior including other general QoS improvements. For instance, Chen *et al.*[6] report on their application of the C4.5 decision tree algorithm and data mining to categorize causes of failure in large Internet sites such as eBay. Candea, *et al.*[7] present an Automatic Failure-Path Inference (AFPI) as an application-generic and automatic technique for dynamically discovering the failure dependency graphs of componentized Internet applications. Chen *et al.*[8] demonstrate a new approach in using machine learning to manage the failures and evolution in large, complex distributed systems using runtime paths.

As will be discussed in the following sections, our study uses a Self-Organizing Maps (SOM) algorithm to underpin autonomic middleware service for on-demand service reservation and provision. This SOM-based classification maps users' to networked services requirements, which is then used by the middleware to anticipate service requests and starts reservation of required services including their required dependencies.

Just-in-time and on-demand grid services provisioning model: Grid services and applications necessitate a range of management processes for bridging the interaction between the consumers and services/infrastructures. In addition, such management processes are vital for hiding some of the system complexity from the consumers. Therefore, management processes manage the QoS in order to improve response time, fidelity, interpretability and reliability. Just-in-time and on-demand services are another way that management process can be utilized to reduce the interaction between the users and the system's resources by adjusting the resources to request and prepare services prior to the users' requests. For such systems, the use of autonomic computing services is essential in order to perform the automated management processes. Autonomic services are proposed to be run inside the middleware as one of its core functions for the Open Grid Services Architecture (OGSA).

In this study, we focus on a practical scenario of using machine learning middleware services to support autonomic computing in order to anticipate users' requirements for networked services and appliances. Intuitively, this expected to reduce the unnecessary and redundant invocation of processes and hence reduces the response time. A service reservation is much more useful for the management of an advanced on-demand service. Therefore the middleware needs to be more specific in the provision of the services required. To achieve this goal, clustering and classifying users according to their usage patterns are suggested. An unsupervised learning technique is adopted by this approach in order to perform the intelligent classification. This method of classification is much applicable in our case, where the target of hypothesis is unknown until the final trained model is built. Next sections describe the On-Demand Services Model.

On-demand service components: On-Demand Service (ODS) requires integration of diversity of components to perform the demanded jobs. These components are shown in Fig. 2, which can be summarized in the following points:

* Consumer: This represents software or human agent requesting a given set of the resources.
* Request services agent: This manages and handles all consumers' requests. A semantic support for meta-model sharing between consumer and the system is used to support open standard interoperation.
* Discovery service: This supports services discovery within a given (virtual or physical container (host). This provides service for request services container and job schedule service.
* Monitoring service: This monitors the consumer behavior, activities and usage. The collected information is provided to the autonomic computing unit for intelligence processes.
* Autonomic computing: This unit offers an intelligent layer to the ODS system. This can provide self-organizing, self-configuration, self-protection and self-configuration capabilities.
* Group: This defines a number of consumers or members which share the same behavior or patterns.
* Service reservation service: This queue and handles all the required services for a given consumers as soon as they are login prior to services requests. The recorded information includes list of required services, time of operation and contract information. This unit forwards this information to job schedule service.
* Job schedule service: This unit schedules the request of services, according to the received information from services reservation unit. The requested services or resources are passed to the discovery unit at the required time of invocation.
* Resource container: This contains all the deployed resources from services and infrastructures by the providers.

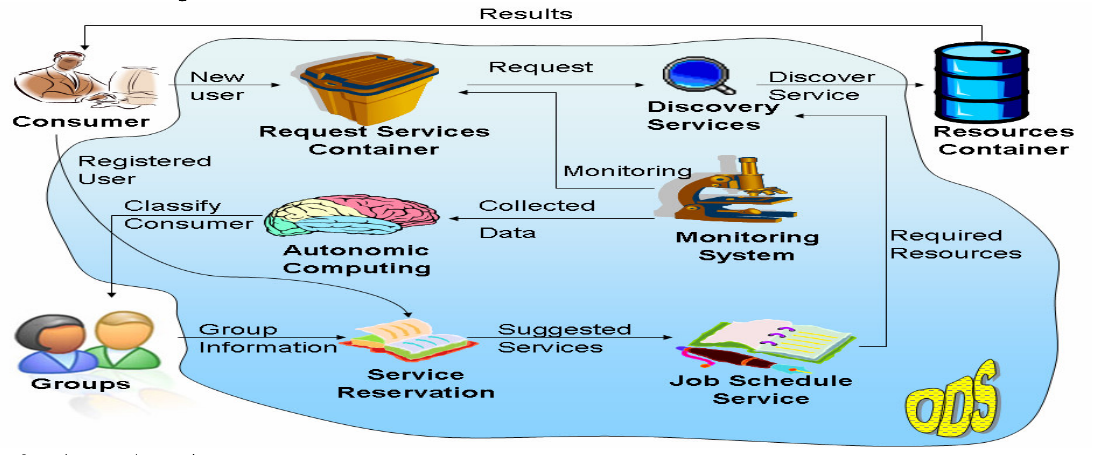

Fig. 2: On-demand service

On-demand service scenario: An advanced on-demand services scenario starts by sending a request from the consumer, as shown in the UML activity diagram in Fig. 3. The middleware scrutinizes the status of the consumer, which can be defined by the ODS as a new consumer or registered consumer. The ODS system is considered as one of the middleware services. If the consumer status is new, then the process can be described in the following points:

* System analyzes the request and sends it to the discovery service.
* A discovery service in turn begins the discovery process within target resources container.

* After finding the demanded services, the necessary services' invocation processes will be initiated.
* The result of the requested task will be presented to the consumer as results or actions.

At the same time the monitoring service is working to record all consumers' demands and activities in order to save it in the consumer profile, inside the logger as shown in Fig. 3. This information is passed to the autonomic computing service inside the middleware system. The autonomic computing here is responsible for performing an intelligent classification process. The classification service depends on the consumer activity, behavior, pattern, and service usage to classify them to one of the classified groups. The classified group is clustered at the first run of the system according to the common parameters between the consumers and then an adjustment process starts to adjust the groups. The group indicates the shared behavior or activities between the consumers in the same group. After the classification process for the new consumer completes, it is then considered a registered consumer.

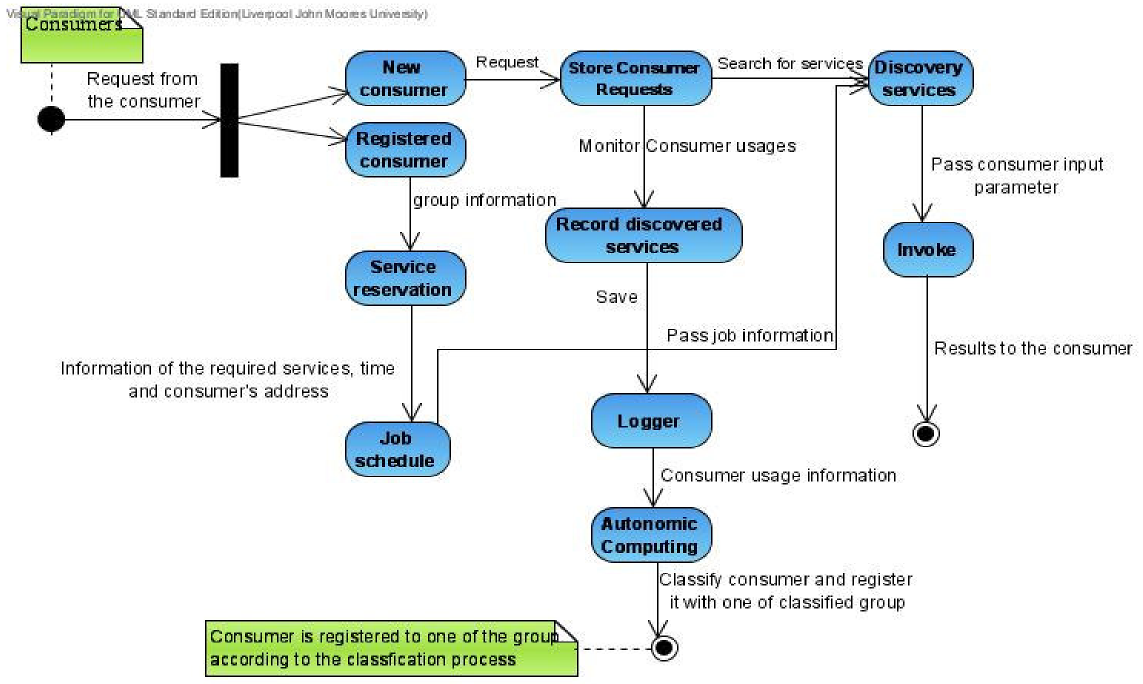
Visual Paradigm for UML Standard Edition(Liverpool John Moores University)
Consumers
Request from the consumer
New consumer
Registered consumer
Request
Store Consumer Requests
Search for services
Discovery services
Monitor Consumer usages
Pass consumer input parameter
group information
Record discovered services
Invoke
Service reservation
Save
Pass job information
Results to the consumer
Information of the required services, time and consumer's address
Job schedule
Logger
Consumer usage information
Autonomic Computing
Classify consumer and register it with one of classified group
Consumer is registered to one of the group according to the classfication process

Fig. 3: ODS-UML activity diagram

On the other hand, the system deals with the registered consumer as describe in the following points:

* The middleware starts gathering the group information of the requested consumer. The consumer sends his group ID with his demands of the required services to help service reservation function to arrange the demanded services in advance.
* The service reservation unit sends a request with all demanded services to the job schedule unit. This request consists of information regarding the time for executing each service, contract, service information and consumers' information.
* The job schedule service manages the execution of the consumers' demanded tasks. This is done by scheduling the execution of each service. The job schedule service sends the demanded service to the discovery service to find and invoke the aimed process. On the other hand, the job schedule service provides the system with advanced information regarding the expected load on each service. This information is used to anticipate the load on the services. In this way, the system has the time to establish its plans, strategies and policies to overcome fault tolerance problems according to the overload. Load balancing, replication, mirroring and other methods can be used in solving such problems of fault tolerance depending on the information that is provided by the job schedule service.

SOM implementation and outline SOM algorithm: The concepts of the SOM are out of the scope of this study and there are many references on the using of SOM in different applications[9-12]. The SOM toolbox for Matlab is an effective software tool for the visualization of high-dimensional data. It converts complex, non-linear statistical relationships between high-dimensional data items into simple geometric relationships on a low-dimensional display[12].

The SOM is initialized using either random or linear initialization. To train the map, SOM uses sequential or batch algorithms by using *som_make* function. The resulting visual map shows the neighbourhood between the neurons and the input training samples updating the Best Matching Unit (BMU). The quantization error can be measured using *som_quality* function which supplies two measures: average quantization error and topographic error. The SOM model delivers logic decisions from the visual maps, benefiting from the labelling features in *som_autolabel* and *som_addlabels* functions, hence we can build a programming model using the SOM method and outputting decisions from calculating BMU for a given data vectors using *som_bmus* function and other related useful functions provided by the toolbox.

SOM classification comprises efficient, mostly accurate mean as a visual data analysis for the maps, While in our approach we have to reduce the role of administration to the minimum. Thus we opted to use other classification algorithms like K-Nearest Neighbourhood (KNN)[13], which is suitable for similar cases. A special MatLab function would implement KNN classifier using arbitrary distance matrix. Despite its sub optimality, the asymptotic performance of the NN rule is good and the rule is simple. Given a set of labelled training examples, *T* and an unseen test point *x*, computing the squared distance in the input space from *x* to each of the examples in *T*. Discarding all training cases, except for the K cases which are closest to the test point. The test point is assigned to the most numerous class amongst this K-NN. A probability distribution over classes is also easily constructed. The probability of class *i (p_i)* is simply specified by:

p_i = n_i / sum_j { n_j}

Where *n_j* is number of nearest neighbours in class *j*. KNN function is also supported in MatLab SOM Toolbox:

[C,P]=knn(d,Cp,[K])

Where *C:* is a matrix of size *N*K* of integers indicating the class decision for data items according to the KNN rule for each *K*.

P: is a matrix of size *N*k*K*, which represent the number of prototypes for each classifier

*d: is an N*P matrix which is pre-calculated dissimilarity (distance matrix)*

Cp: is a Px1 vector that contains integer class labels, in our case (OMAR, …, YASIR)

Case-study: just-in-time service provisioning for intelligent connected home scenario: To elucidate the idea of just-in-time and on-demand services based on user usage classification, a “connected home devices” scenario is adopted. The intelligent connected home machines or smart home devices are the next generation of home devices, which depend on local and remote services to be available on-time in order to generate a comfortable environment for users. In order to improve the usability and QoS of such technology, the system should provide services with a short response time. To achieve this, an ODS is suggested to be adopted in this scenario.

The scenario starts by training the machine learning service with users’ usage data. After the training phase, the machine learning service clusters the users into a number of groups depending on their usage. Then, the system is ready to receive information regarding the new users in order to classify him/her/it into one of the groups. In order to get such information, on-the-fly[14,15] sensors are utilized to collect information for the new users and store this data within the user profile inside the logger. The benefit after classifying the users to one of the existing groups is to anticipate the required services for each user according to the group behavior. These results are provided to the service reservation system with the required information as explained before. This system determines the required services, time of operation, required resources, dependency and other stuff, which is required to perform the demanded tasks. The expected services are sent to the job schedule service. The job schedule system in this stage is responsible for planning a timetable to execute the services, anticipating the load on each service and recover from fault conditions before occurrence by passing an alert to one of the fault tolerance services as shown in Fig. 4.

The Self-Organizing Map (SOM) is adopted in this scenario to carry out the classification process. SOM is selected since it is one of the unsupervised learning techniques. Such a technique is required in this case due to the absence of known target data. The data collected from the middleware repository is employed to feed the SOM services with the required data for classifying the user to one of the existing group.

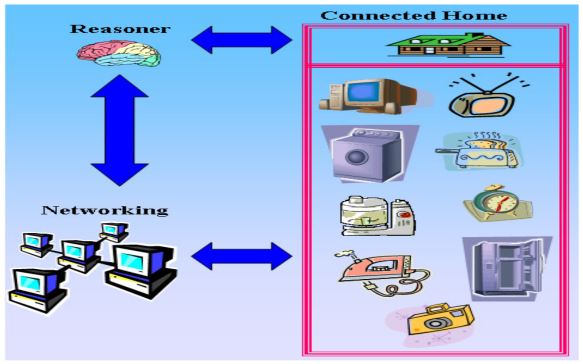

Fig. 4: Illustration of the connected home case study

Results of SOM classification for connected home machine: The experiment was conducted using a .Net prototype, machine learning middleware service using the Matlab SOM library[9]. As illustrated by Fig. 5-8, the results represent classifications for different types of users and devices. Figure 5 shows many correlations between devices, which are obtained after the training phase which included 1000 input sample data and 10 trainees. As shown in Fig. 5, a sample of these correlations can be summarized as follows:

* Lights and PlayStation II correlation
* Video and Coffee Machine correlation
* Video CD and Fans correlates
* Vacuum cleaner and Washing machine correlation

Figure 6, represents U-Matrix distribution of labels for the connected home devices. Figure 7 shows a shaped U-Matrix with coloured regions exhibiting three clear clusters of the map. Figure 8, demonstrates the Probability Distribution Function (PDF) of the input vectors. The critical analysis of this approach depends on selecting and scaling the correct data for training the system, because using un-normalised data might produce inaccurate user classification. Therefore, selecting the adequate training data is the vital to get right classified model.

At runtime, the machine learning middleware service, along with the training data (user and device classification) can classify logged users according to known users (one of the seven classified regions). These classes specify the user types and their usage model, such as the device usage order and time of usage. This is used in this case study to guide the autonomic middleware services for service reservation and provisioning.

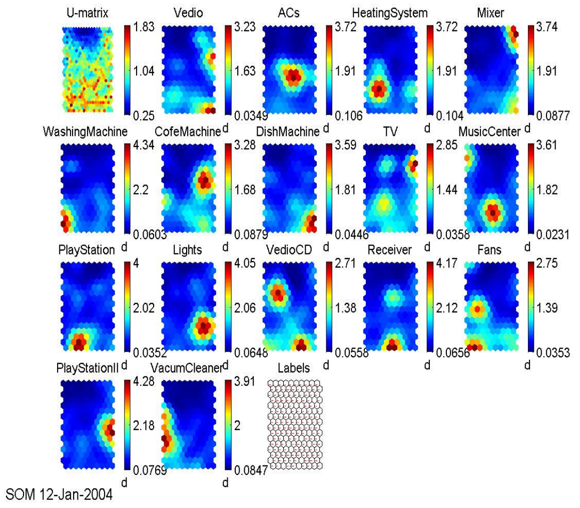

Fig. 5: SOM visual classification

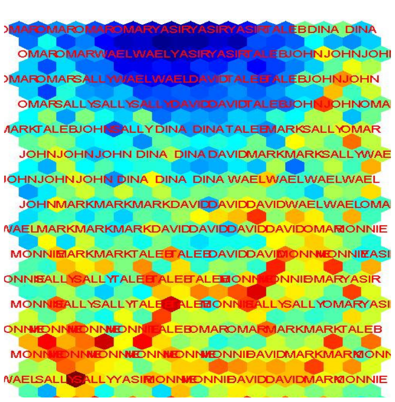

Fig. 6: U-Matrix distribution of labels

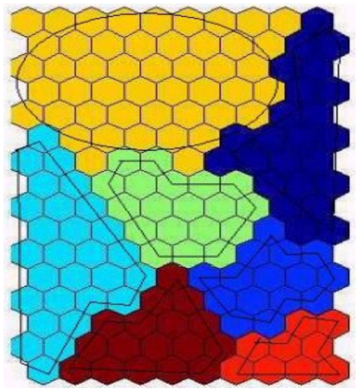

Fig. 7: U-map of SOM maps resulted data

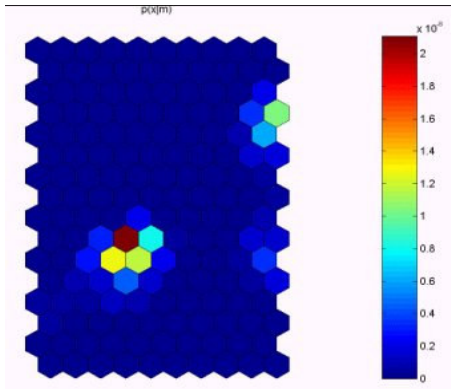

Fig. 8: Probability Distribution Function PDF of the input vectors

CONCLUSION AND FUTURE WORK

Just-in-time and on-demand services for grid and planetary scale environments are presented in this study. Such services are adopted to improve the QoS, reliability and fidelity of the systems, along with reducing the interaction of the consumers with the system. Autonomic computing services are utilized in this case to perform the intelligent stuff inside the middleware for on-time and on-demand services. On-Demand Services (ODS) model is described in this study depending on using service reservation, job schedule services and monitoring system.

Users' classifications according to their usage are selected in this scenario to improve the performance of job scheduling and services reservation. Machine learning is utilized to perform the automated classification tasks. Due to the absence of classification targets an unsupervised learning technique is used in this case. SOM, as an unsupervised learning technique, is adopted to be used as a web service integrated with autonomic computing service for the OGSA. VS.Net and Matlab functions are used to develop the framework for the user classification process. This model is employed to predict the type and pattern of the new user. The goal of predicting the new user is to determine the required services in advance. This assists in making the system plan, managing and controlling its resources to give better service and overcome the failure, due to overloading resources. The connected home scenario is adopted to demonstrate the idea of classifying users according to their device usage in order to show the potential of such system.

REFERENCES

1. Lamonica, M., 2003. IBM draws self-management blueprint. IBM.
2. IBM, 2004. An Architectural Blueprint for Autonomic Computing.
3. Studwell, T., K.S., J. Baekelmans, P. Brittenham, T. Deckers, C. Laet, E. Merenda, B. Miller, D. Ogle, B. Rajaraman, K. Sinclair and J. Sweitzer, 2003. Adaptive Services Framework. IBM.
4. Kephart, J.D.C., 2003. The vision of autonomic computing. IEEE Computer, 36: 41-50.
5. Miller, B., 2005. The sutonomic computing edge: The "Standard" way of autonomic computing. IBM.
6. Chen, M.A.X.Z., J. Lloyd, M.I. Jordan and E. Brewer, 2004. Failure diagnosis using decision trees. 1st Intl. Conf. Autonomic Computing (ICAC'04), New York, USA, pp: 36-43.
7. Candea, G., M.D., M. Chen and A. Fox, 2003. Automatic failure-path inference: A Generic Introspection Technique for Internet Applications.
8. Chen, M.A.A., E. Kiciman, J. Lloyd, D. Patterson, A. Fox and E. Brewer, 2003. Path-Based Failure and Evolution Management.
9. Vesanto, J.H., E. Alhoniemi and J. Parhankangas, 1999. Self-Organizing Map in Matlab: the SOM Toolbox. Proc. Matlab DSP Conf., Espoo, Finland, pp: 35-40.
10. White, R.H., 1992. Competitive Hebbian Learning 2: An Introduction.
11. Bingham, E.J.K. and K. Lagus. 2002. ICA and SOM in Text Document Analysis. 25th annual international ACM SIGIR conference on Research and development in information retrieval.
12. Kohonen, T., 1997. Self-organizing Maps. Springer Series In Information Sciences, 30: 426.
13. http://www.cs.toronto.edu/~delve/methods/knn-class-1/knn-class-1.html.
14. Omar, W., A. Taleb-Bendiab and Y. Karam, 2005. PlanetLab Overlay: Experimenting with Sensing and Actuation Support for Situated Autonomic Computing Services. 6th PG net2005 Conf., Liverpool, UK.
15. Omar, W., B. Ahmad, A. Taleb-Bendiab and Y. Karam, 2005. A software framework for open standard self-managing sensor overlay for web services. 7th Intl. Conf. Enterprise Information Systems (ICEIS2005), MIAMI BEACH-FLORIDA-USA., pp: 72-82.