A SOFTWARE FRAMEWORK FOR OPEN STANDARD SELF-MANAGING SENSOR OVERLAY FOR WEB SERVICES

Wail M. Omar [1,2], Bassam A. Ahmad [2], A. Taleb-Bendiab [1], Yasir Karm[1]

[1] *School of Computing and Mathematical Sciences*
Liverpool John Moores University
Byrom Street
Liverpool, L3 3AF, UK

[2] *Faculty of Applied Sciences*
Sohar University
Sohar, P.C. 311
Sultanate of Oman

Keywords: QoS, Sensors Framework, Sensors for web services.

Abstract: To improve the usability and reliability of grid-based applications, instrumentation middleware services are now proposed and widely accepted as a means to monitor, control and manage grid users' applications. A plethora of research works now exist focusing on the design and implementation of a range of software instrumentation techniques (Lee et al. 2003, Reilly and Taleb 2002) to enhance general systems' management including; QoS, fault-tolerance, systems recovery and load-balancing. However, management, assurance and fidelity concerns related to sensors and actuators (effectors) support for grid and web services environment received little to no attention. This paper presents a lightweight framework for the generation, deployment and discovery of different types of sensors and actuators together with two associated description languages namely; monitor session description language and sensor and actuation description langue. These are used respectively to describe the set of deployed sensors and actuators in a given self-managing grid infrastructure, and to define monitoring properties and policies of a given target service/application. Moreover, negotiation process is considered between different units of the grid environment. In addition, the paper presents a developed sensor-based systems awareness fabric layer for self-managing decentralised web services. The paper concludes with a case study illustrating the use of the sensor framework for job monitoring.

1 INTRODUCTION

Recent advancements in networking, hardware, and middleware technologies have been a major catalyst for the recent popularity of grid-based applications (Foster et al. 2001), which are characterized by their very high computing and resource requirements. Thus, it needs powerful and active instrumentation[1] strategy. Originally, instrumentation was used to debug and test applications that run on single processor machines and for analyzing the performance of real-time systems. The parallel computing community later adopted instrumentation to debug, evaluate and visualize parallel applications. More recently distributed application developers have recognized the requirements of instrumentation, used in a dynamic regime, to monitor and manage distributed applications (Reilly and Taleb 2002).

It is now generally accepted that systems' introspection and general runtime monitoring for instance; for inconsistency and faults detection requires software instrumentation (introspection and sensing) services. However, the quality of service of sensor/actuation service received little or no attention. There is a lack of focus on management

[1] Software Instrumentation is the process of putting probes into software to record systems' operation state data (Foster and Kesselman 2003).

M. Omar W., A. Ahmad B., Taleb-Bendiab A. and Karm Y. (2005).
A SOFTWARE FRAMEWORK FOR OPEN STANDARD SELF-MANAGING SENSOR OVERLAY FOR WEB SERVICES.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 72-81
DOI: 10.5220/0002514100720081
Copyright © SciTePress

and control issues related to sensors and actuation (effectors) for grid and web services environment.

This paper presents a lightweight framework for the generation, deployment and discovery of different types of sensors and actuators together with two associated description languages namely; the Sensors and Actuator Description Language (SADL) and the Monitor Session Description Languages (MSDL). These are used respectively to describe the set of deployed sensors and actuators in a given self-managing grid infrastructure, and to define monitoring properties and policies of a given target including application services and environment.

This framework leverages the quality of services (QoS) for sensors to achieve an enhanced fidelity (accuracy), performance and offer standard method for exchange information based on mark-up language (XML). Each sensor cluster works with the others to form a sensor farm referred to here as cloud. Each of which has a zoning construct and cloud manager agent responsible for general management, access of deployed sensors and actuators and exchange of sensing information with other agents (clouds).

The remainder of this paper is structured as follow: Section two outlines related works followed by a software sensor and actuator overlay. Section 4 presents the main structure of the SADL followed by an illustrative example of using it. Model for Monitoring Sessions Description Language (MSDL) is demonstrated in Section 5. Case study for using on-fly instrumentation with the SADL framework is presented in Section 6. Finally, the paper concludes with general summary and statement of future work.

2 BACKGROUND

Over the coming years, many are anticipating grid computing infrastructure, utilities and services to grow dramatically in size and functions, over heterogeneous system to become an integral part of future socio-economical fabric (Omar *et al.* 2004). This vision is predicated on that such grid-based services, infrastructures and applications have to provide a high-degree of assurance, dependability, and agility to changes and cost effectiveness. This warrant for new models and supports for web services sensing and actuation infrastructures and middleware resources.

Much work related to systems monitoring for Grid computing is now widely published (Lee et al. 2003), describing numerous monitoring models including; visPerf and NetSolve (Satoshi *et al.* 1999), Heart Beat Monitor (HBM) and Enterprise Instrumentation Framework (EIF). Other approaches for instance presented by Reilly and Taleb-Bendiab (Reilly and Taleb-Bendiab 2002), describes a dynamic instrumentation framework, which provides support to monitor and manage Jini applications. The framework adopts a service-oriented programming model and the software factory pattern to dynamically generate specific instrument types, which are deployed and interfaced to client services via Java's dynamic proxy API and Jini's remote event. This enables on-demand insertion and removal of instrumentation services.

Other models such as the Globus Heart Beat Monitor (HBM) is provided at container level to provide health-check monitoring service for instance for faults detection of grid resource, that is, checking the status of a target machine and reports it to a higher-level collector machine (Globus 2003). Others such as the GridMonitor provides access to Grid information and server status for all sites including Globus Meta-computing Directory Service (MDS). This in combination with JAMM an agent-based monitoring system for Grid environments it automate the execution of monitoring sensors and the collection of event data (Globus 2003).

Operating systems specific instrumentation frameworks include Windows Management Instrumentation (WMI) consists of three parts described below (Travis B. 2003):

- *Management Infrastructure*: providing object manager called Common Information Model (CIM). Users use CIM Object Manager (CIMOM) to handle communications between management applications and providers.
- *Managed Objects*: provide management services that access managed objects using the CIM Object Manager.
- *WMI Providers*: provider components that supply dynamic management data about managed objects, handle object-specific requests, or generate WMI events.

Enterprise Instrumentation Framework (EIF) is another technology for monitoring and troubleshooting high-volume, distributed environments. EIF is a technology for Visual Studio.NET applications. It works hand-in-hand with Application Centre (AC) and Microsoft Operations Manager (MOM), providing a uniform data for event management, tracing and logs.

Other works focused on sensors discovery mechanisms to support fault-tolerance of heterogeneous distributed systems. For instance, Karuppiah *et al.* (Karuppiah2001) discussed the design of a distributed vision system that enables several heterogeneous sensors with different processing rates to exchange information in a timely manner to support the tracking of multiple human subjects and mobile robots in an indoor smart environment.

Moreover, many more concerns related to enterprise systems self-awareness and monitoring remain to be addressed including; the control and management of sensors, the accuracy of selecting sensors, robustness, assurance and scalability of the sensor system. In this paper, we will discuss some of these problems.

3 SOFTWARE SENSOR AND ACTUATOR OVERLAY

In this work, a sensor framework is developed to support sensor and actuator generation[1] (Reilly and Taleb 2002), deployment, discovery and general management providing high-availability, control and management for the deploying sensors. As illustrated in Figure 1, each deployed sensor cloud (cluster) in the sensor overlay has at least an associated sensor manager agent, which provide on-demand sensor generation, deployment, lookup and/or publish subscribe services for instance to provide intelligent matching between the available sensors and consumer requirements[2]. The consumers can select different types of deploying sensors with a framework and inject them to the targets (monitored services). The consumers can select to access (or subscribe) to more than one target instrumentation data in accordance with a given the Service Level Agreement (SLA) and/or a given contract between the sensor consumers and providers. A third party agent referred to here as “SLA administrator”, is responsible for establish the SLA between the consumer and services provider, which involves the negotiation process.

In view of the scalability concerns of massively decentralised systems of sensor networks and grid computing a zoning/clouds abstraction is here introduced, where each zone/cloud has a sensor manager agent (Figure 1). Such an agent is also responsible for the interaction with other agents in other zones where it can act as a gateway sensor node to its sensor node, as in sensor networks the zone agent can be hosted by an edge sensor node. For example as shown in Figure 1, an agent in *zone A* is responsible for offering services for the sensor provider and consumers register with *zone A*. In addition, sensor manager agents implement various zone-based policies for instance for; information exchange, access control and self-healing monitoring and actuation[3]. To this end, a Sensor and Actuator Description Language (SADL) was designed and developed to provide an open standard description markup language for lightweight access to deployed sensor and actuators (effectors) metamodel in any given zone (Sec. 4). Thus, SADL provides a *ubiquitous* interaction mechanism with a given sensors overlay (Figure 1).

[1] This is based on the software factory pattern.

[2] In addition the framework offers the monitoring, management, control and advertisement for the deployed sensors

3.1 Fidelity of Sensors

Sensor fidelity, robustness and assurance are some of the major concerns considered by the proposed and developed sensor framework, which borrows concepts developed by the intelligent systems engineering community including; self-convergence, self-optimization, self healing and self-adaptive.

Many critical concerns in the sensor and actuations overlay are the on-demand selection and access to the correct sensor type for a given monitoring task. Thus, the sensor framework is designed to assistant to the consumer in selecting the most appropriate type of senor according to consumer’s requirements.

[3] The full description of the zoning and clouds abstraction and sensor manager agent is out of the scope of this paper and will be described in future paper.

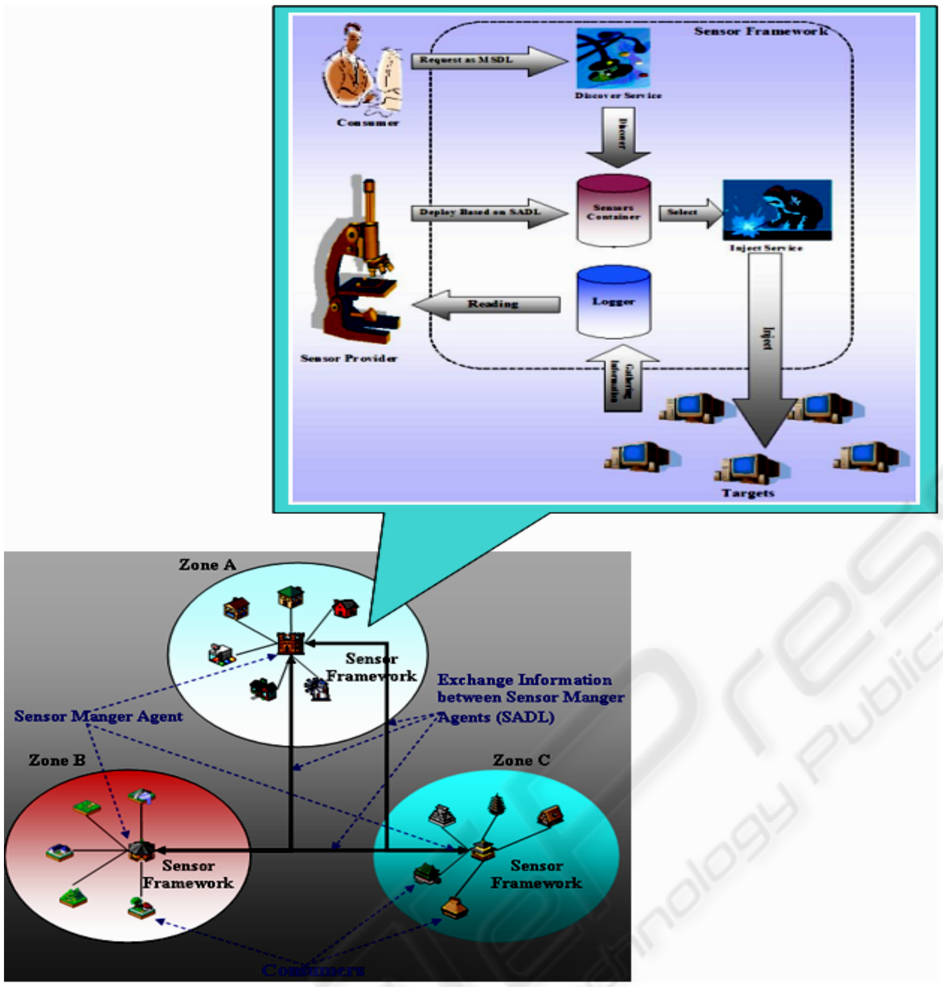


Figure 1: Sensor Framework Scenario including Sensor zoning Framework

For this reason, Sensor and Actuator Description Language is proposed to provide a ubiquitous access to sensor meta-model, which assist in the discovery and selection of required types of sensors. SADL will feed the intelligent system (sensor manger agent) with the data that is required to do the intelligent matching between consumer request and available sensors.

4 SENSING AND ACTUATION DESCRIPTION LANGUAGE

Awareness and governance model exposes the infrastructure states and associate conditional triggers (actuations) to detected events and systems events of interest. In particular, deployed sensors (software instruments) publish monitoring data to the infrastructure self-governance middleware service (Kephart and Chess 2003), which provides the actuation model triggering and enactment. For example in the case of services/infrastructures overload problem, different methods are used to solve this problem; one of them is the replication. The sensor is used here to get information from the original node and from replication services after creation. Many of researchers and developers intended to generate different types of on-the-fly sensors; the management system should have well known information regarding each one of these sensors. A prototype of the Sensing and Actuation Description Language (SADL) is designed for the purpose of deploying, discovering and managing the sensors in an open-standard format.

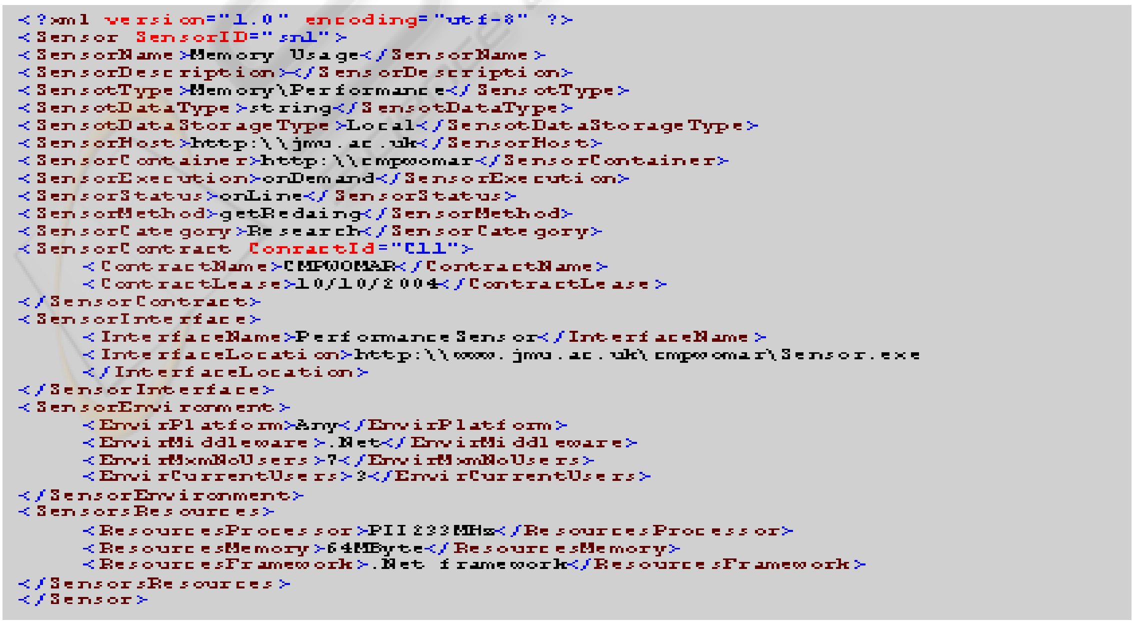
SADL is used to deploy and discover different types of sensors (processor, memory, web services, etc.) in open standard format. This information assists the consumer and the analysis system to select the required sensor from the discover list.

The current prototype of a SADL has been developed using a.Net-based sensor software factory and environment to support remote monitoring, logging and analysis of a range of web service properties including; structural, functional and operational aspects.

Table 1 describes the most important parameters of SADL.

Table 1: SADL parameters.

Elements Name	Comments
SensorID	This ID should be unique for each Infrastructure. The ID is generated automatically by the system.
SensorName	This should be give by the deplorer.
SensorDescription	The deplorer can describe anything for the infrastructure
SensotType	This element is used to describe the parameter that this sensor is used to read it. Like performance, security, etc….
SensotDataType	This can be string, integer, object, etc….
SensotDataStorageType	This element is used to describe the location of storage data, if it is stored locally, centre, or in a host computer.
SensorHost	The host that hold the sensor software.
SensorContainer	This can hold different type of sensors.
SensorExecution	This may be control flow, on demand or event driven
SensorStatus	If the sensor is online or offline.
SensorMethod	Method that used by the sensor
SensorCategory	The category of the sensor. This may be research, free, commercial, military, etc…
SensorContract	This element describes the sensor to whom it belongs and what is lease time for the sensor.
SensorInterface	Sensor interface is used to connect to the sensor
InterfaceName	Interface name
InterfaceLocation	The path to the sensor interface
SensorEnvironment	Sensor environment is used to describe all the information about the required environment for sensor to work
EnvirPlatform	The required platform
EnvirMiddleware	The required middleware
EnvirMxmNoUsers	This describes the maximum number of users those can use the sensor at the same time
EnvirCurrentUsers	This describes the current users those use the sensor at the request time. This may be changed dynamically by the system according to the current users.
SensorsResources	Describes the minimum required resources for the sensor to work.
ResourcesProcessor	The minimum speed for the processor
ResourcesMemory	The minimum size for the memory
ResourcesFramework	The required framework.



```
<?xml version="1.0" encoding="utf-8" ?>
<Sensor SensorID="xml">
<SensorName>Memory Usage</SensorName>
<SensorDescription></SensorDescription>
<SensotType>Memory\Performance</SensotType>
<SensotDataType>string</SensotDataType>
<SensotDataStorageType>Local</SensotDataStorageType>
<SensorHost>http:\\jmu.ac.uk</SensorHost>
<SensorContainer>http:\\cmpwomar</SensorContainer>
<SensorExecution>onDemand</SensorExecution>
<SensorStatus>onLine</SensorStatus>
<SensorMethod>getRedaing</SensorMethod>
<SensorCategory>Research</SensorCategory>
<SensorContract ConractId="C11">
      <ContractName>CMPWOMAR</ContractName>
      <ContractLease>10/10/2004</ContractLease>
</SensorContract>
<SensorInterface>
      <InterfaceName>PerformanceSensor</InterfaceName>
      <InterfaceLocation>http:\\www.jmu.ac.uk\cmpwomar\Sensor.exe
      </InterfaceLocation>
</SensorInterface>
<SensorEnvironment>
      <EnvirPlatform>Any</EnvirPlatform>
      <EnvirMiddleware>.Net</EnvirMiddleware>
      <EnvirMxmNoUsers>7</EnvirMxmNoUsers>
      <EnvirCurrentUsers>3</EnvirCurrentUsers>
</SensorEnvironment>
<SensorsResources>
      <ResourcesProcessor>PII 233MHz</ResourcesProcessor>
      <ResourcesMemory>64MByte</ResourcesMemory>
      <ResourcesFramework>.Net framework</ResourcesFramework>
</SensorsResources>
</Sensor>
```

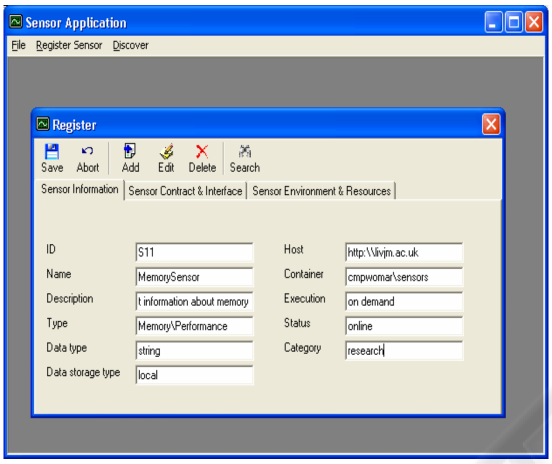
Figure 2: A Simplified example of SADL

4.1 Illustrative Example

Figure 2, provides an example of sensor and actuators service deployment using the SADL framework. This is for instance, to deploy and activate a memory sensor in a given container (node). Where the sensor in accordance with an associated monitoring contract it reads at a specified frequency memory measurement data, which is made available to the monitoring layer through either logging or streaming.

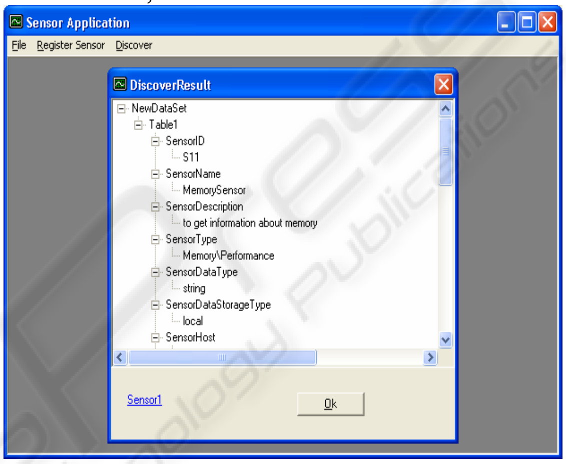
Interfaces information of the sensor is exposed via the SADL.

The software for the generation, deployment and discovery of sensor services is implemented using VS.Net. Figure 3 shows an example that demonstrating the processes of sensor and actuator generation, deployment and discovery. In this example a sensor and actuator of type memory usage monitor and analyser are selected. In addition, the current prototype provides operational information of discovered sensors and actuators including; sensor properties, contract, interface, environment, and resources.



ID	S11	Host	http:\\livjm.ac.uk
Name	MemorySensor	Container	cmpwomar\sensors
Description	t information about memory	Execution	on demand
Type	Memory\Performance	Status	online
Data type	string	Category	research
Data storage type	local		

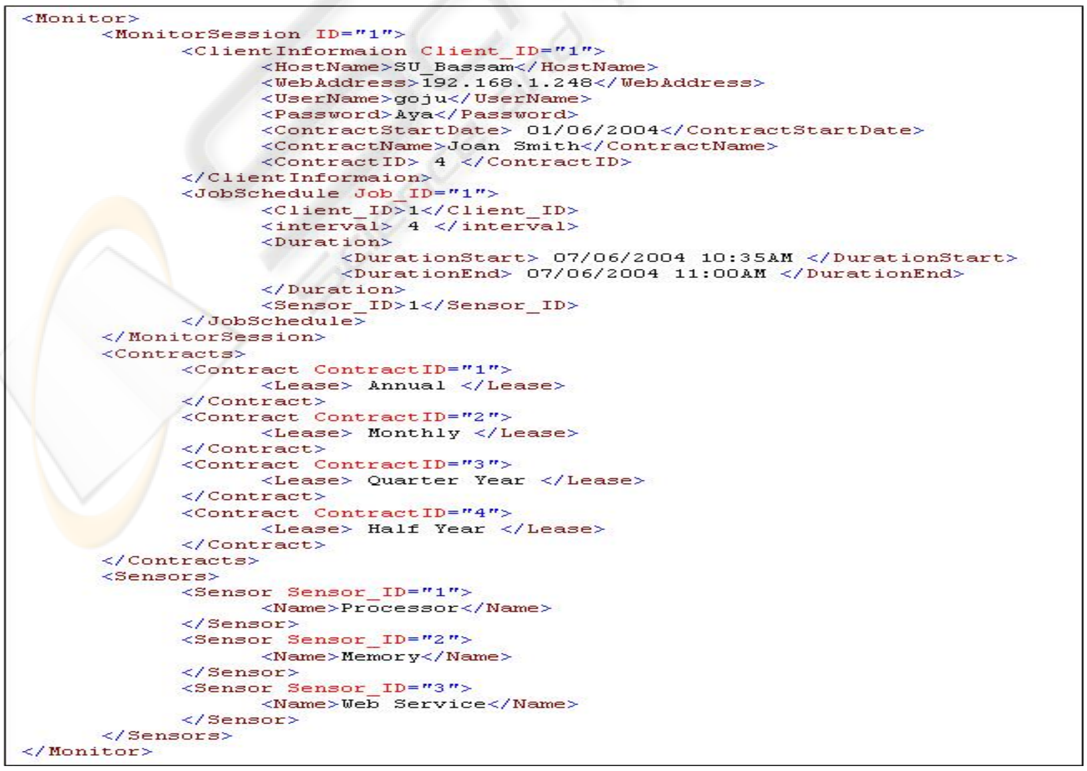
Figure 3.a: Sensor registration screen (sensor information).



```
NewDataSet
  Table1
    SensorID
      S11
    SensorName
      MemorySensor
    SensorDescription
      to get information about memory
    SensorType
      Memory\Performance
    SensorDataType
      string
    SensorDataStorageType
      local
    SensorHost
```

Figure 3.b: Discover results screen.

Figure 3: SADL Interface.



```
<Monitor>
      <MonitorSession ID="1">
            <ClientInformaion Client_ID="1">
                  <HostName>SU_Bassam</HostName>
                  <WebAddress>192.168.1.248</WebAddress>
                  <UserName>goju</UserName>
                  <Password>Aya</Password>
                  <ContractStartDate> 01/06/2004</ContractStartDate>
                  <ContractName>Joan Smith</ContractName>
                  <ContractID> 4 </ContractID>
            </ClientInformaion>
            <JobSchedule Job_ID="1">
                  <Client_ID>1</Client_ID>
                  <interval> 4 </interval>
                  <Duration>
                        <DurationStart> 07/06/2004 10:35AM </DurationStart>
                        <DurationEnd> 07/06/2004 11:00AM </DurationEnd>
                  </Duration>
                  <Sensor_ID>1</Sensor_ID>
            </JobSchedule>
      </MonitorSession>
      <Contracts>
            <Contract ContractID="1">
                  <Lease> Annual </Lease>
            </Contract>
            <Contract ContractID="2">
                  <Lease> Monthly </Lease>
            </Contract>
            <Contract ContractID="3">
                  <Lease> Quarter Year </Lease>
            </Contract>
            <Contract ContractID="4">
                  <Lease> Half Year </Lease>
            </Contract>
      </Contracts>
      <Sensors>
            <Sensor Sensor_ID="1">
                  <Name>Processor</Name>
            </Sensor>
            <Sensor Sensor_ID="2">
                  <Name>Memory</Name>
            </Sensor>
            <Sensor Sensor_ID="3">
                  <Name>Web Service</Name>
            </Sensor>
      </Sensors>
</Monitor>
```

Figure 4: Monitor Sessions Description Language (MSDL)

5 MONITOR SESSIONS DESCRIPTION LANGUAGE

Monitor Sessions Description Language (MSDL) is used to create a standard way for sending monitor tasks by the consumers based on using XML. The consumer sends a request to establish sensor task to the SADL framework. The SADL uses the passing information to find the suitable sensor from its available sensor services. MSDL categorized into three parts:

- the monitor session information,
- Service Level Agreement (SLA) (contract),
- sensors information description.

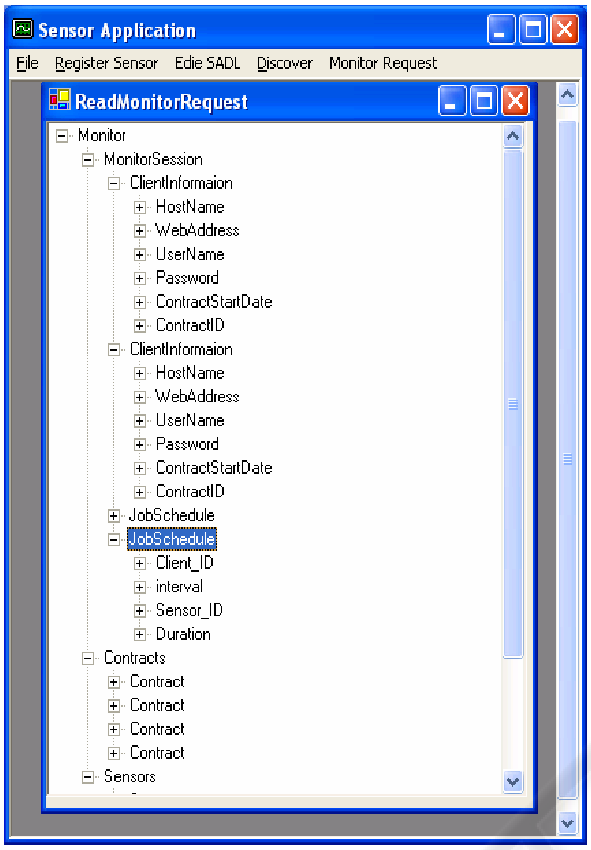
The monitor session information is divided into client information and job schedule information. The client information includes host name, SLA for the client and authentication information. The job schedule tags describe the duration and intervals of the task jobs, interval tags is used to indicate the time between reading and next one, this help to reduce the information that transfer from the target to the control and analysis system. The administrator (which uses the sensors to look after his clients) can select more than one type of sensors to get different types of reading such as; memory, process, processor. Also he can use the same sensors for more than one of its clients according to his SLA. Figure 4 presents an example of using MSDL.

6 CASE STUDY

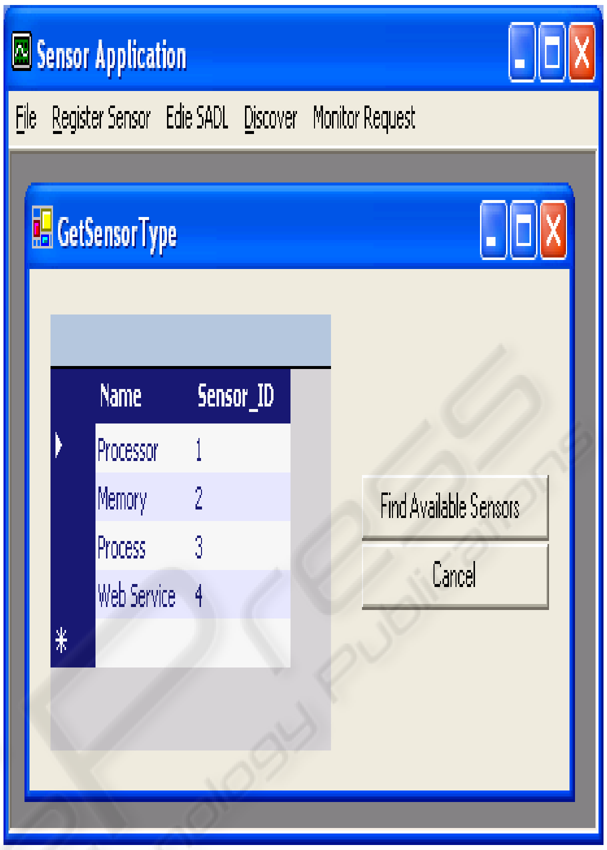
A case study is used to show how the sensor framework can be used to read in sensor requirements from users and discover and select suitable sensor from a sensor array. The experiment is conducted using a set of PIV 2.7 GHz with RAM 256 nodes running .Net framework 1.1 and a number of Web Services. The case study is divided into three phases:

- the creation of the MSDL for a given consumer and sending it to the SADL.
- the discovery of deployed sensors for a given monitoring requirement,
- the interface with the sensor and exchange measurement data.

As illustrated in Figure 5a, the scenario starts by sending the monitor session request by the consumer to the sensors framework as MSDL (Figure 5.a). The consumer sends request for a number of sensors to be injected in the target. In this case study, the consumer demands processor, memory, process, and web services sensors. In the other hand, the sensor providers use the SADL to pass information regarding their sensors to the framework, which will be as an advertisement location for the sensors. The SADL framework reads this MSDL and tries to find the available sensors those are deployed with it, as shown in the Figures (5.b. and 5.c.). In many cases there is more than one sensor for each type of sensors, the SADL framework is responsible for selecting the most appropriate sensor for each target. SADL framework may depend on using one of the intelligent services, such as prediction or classification, to anticipate the most likely hood sensors with the consumer’s requirements. The SADL runs these sensors on the target and get the information and send it to the consumer (Figure 5.d).



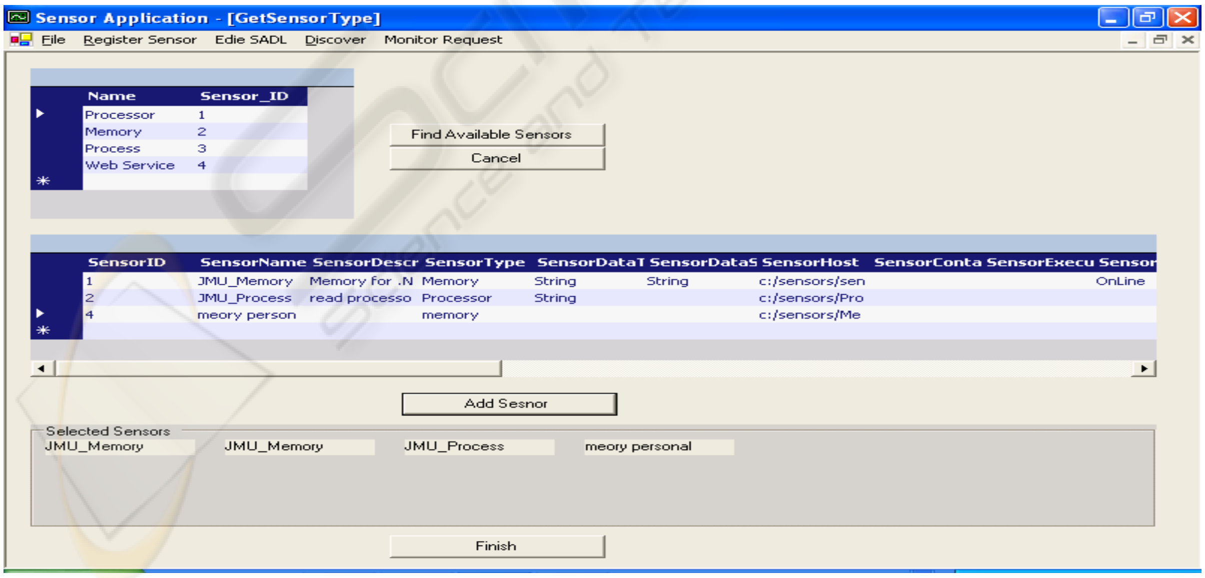
```
Monitor
  MonitorSession
    ClientInformaion
      HostName
      WebAddress
      UserName
      Password
      ContractStartDate
      ContractID
    ClientInformaion
      HostName
      WebAddress
      UserName
      Password
      ContractStartDate
      ContractID
    JobSchedule
    JobSchedule
      Client_ID
      interval
      Sensor_ID
      Duration
  Contracts
    Contract
    Contract
    Contract
    Contract
  Sensors
```



Name	Sensor_ID
Processor	1
Memory	2
Process	3
Web Service	4

Figure 5.a: Monitor request information

Figure 5.b: Sensors required


Sensor Application - [GetSensorType]
File Register Sensor Edie SADL Discover Monitor Request
Find Available Sensors
Cancel
Add Sesnor
Finish

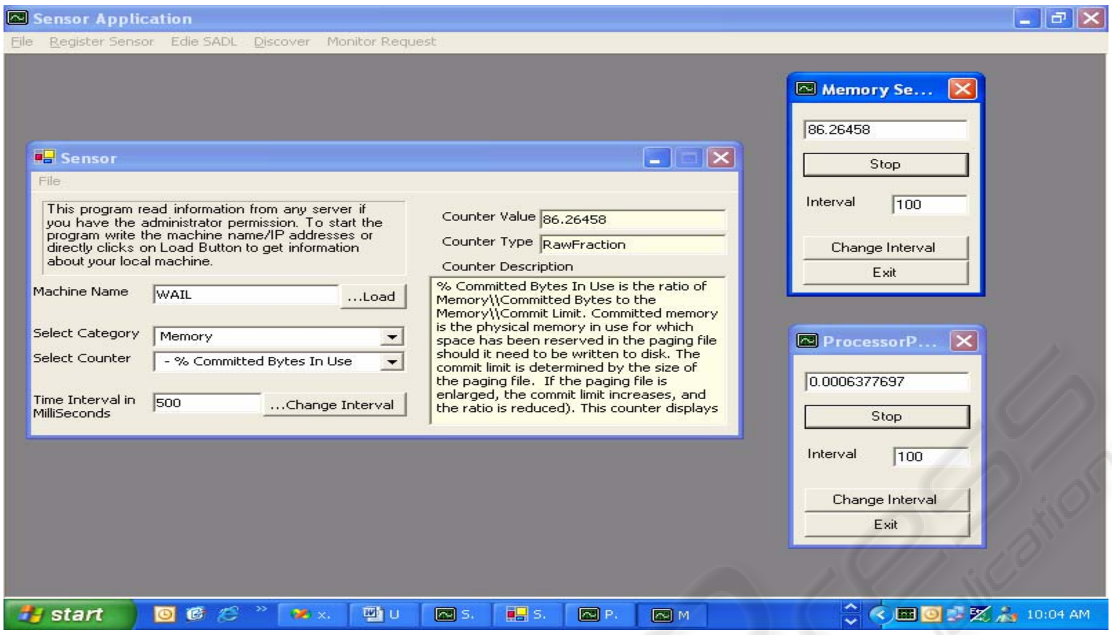
Name	Sensor_ID
Processor	1
Memory	2
Process	3
Web Service	4

SensorID	SensorName	SensorDescr	SensorType	SensorDataT	SensorDataS	SensorHost	SensorConta	SensorExecu	Sensor
1	JMU_Memory	Memory for .N	Memory	String	String	c:/sensors/sen			OnLine
2	JMU_Process	read processo	Processor	String		c:/sensors/Pro			
4	meory person		memory			c:/sensors/Me			

Selected Sensors

JMU_Memory	JMU_Memory	JMU_Process	meory personal

Figure 5.c: Finding sensors with in SADL framework



This program read information from any server if you have the administrator permission. To start the program write the machine name/IP addresses or directly clicks on Load Button to get information about your local machine.

Machine Name: WAIL
Select Category: Memory
Select Counter: - % Committed Bytes In Use
Time Interval in MilliSeconds: 500
Counter Value: 86.26458
Counter Type: RawFraction
Counter Description: % Committed Bytes In Use is the ratio of Memory\\Committed Bytes to the Memory\\Commit Limit. Committed memory is the physical memory in use for which space has been reserved in the paging file should it need to be written to disk. The commit limit is determined by the size of the paging file. If the paging file is enlarged, the commit limit increases, and the ratio is reduced). This counter displays

86.26458
Interval: 100

0.0006377697
Interval: 100

Figure 5.d: Running sensors

7 CONCLUSIONS AND FUTURE WORK

The paper presented a proposed and developed sensor framework that supports the deployment, discovery and management of sensors to support better quality of sensors. Sensor Manger Agent is argued to do the intelligent stuff of matching between user needs and available sensors inside the framework. Sensors framework zones are recommended to be used for overcoming the problem of scalability. Sensor manger agent is in charge to exchange information regarding available sensors between different zones. A prototype of the Sensing and Actuation Description Language (SADL) is designed for the purpose of deploying, discovering, and exchange information concerning sensors (Sec. 4). The SADL will be used to discover and invoke different types of sensors, which can be used with different types of platform (heterogeneous system). A prototype for Monitor Session Description Language (MSDL) (Sec. 5) has been developed to present standard method for sending monitor job schedule from the consumer to the sensor framework, which controls the huge numbers of requested sensors sessions that read the required information periodically through the Grid. A case study is developed to show the importance and usage of the sensor framework in finding the best sensor for the consumer.

So far the results are promising; it has been tested under LAN-Grid environment with different types of deployed sensors. The framework provides the consumer with most appropriate sensors for their request and injects them in the target.

Further work is under way to investigate the framework under WAN-Grid environment with larger number of deployed heterogeneous sensors. A storage system is suggested to be added to the system for providing complete framework for gathering information and provide it to the consumers. Moreover, sensor framework will be adopted to be used with different types of sensors for different applications, such as; e-health system, and intelligent connected home networks and self-management middleware.

REFERENCES

Omar, W., Taleb-Bendiab, A., and Yu, M. 2004. An Open Standard Description Language for Semantic Grid Services Assembly for Autonomic Computing Overlay. In *Proceedings of the Services Computing, 2004 IEEE International Conference on (SCC'04) - Volume 00*

Menkhaus, G., Pree, W., Baumeister, P., Deichsel, U. 2002. Interaction of Device-Independent User Interfaces with Web services.

Mridula, P., Chandler, J., Hatfield, B., Lassan, R., Macintyre, P., Wanta, D., 2002. ASP.NET. Publisher: Hungry Minds, ISBN: 0764548166.

Kirtland, M. 2001. A Platform for Web services.

Foster, I, Kesselman, C, 2003. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman, ISBN: 1-55860-933-4.

Foster, I., Kesselman, C. and Tuecke S. 2001, The Anatomy of the Grid Supercomputer Applications:

Diakov, N., 2002. Concepts of Software Monitoring: Activities, Instrumentation and Organization of Monitored Data – Data Flows.

Lee D., Dongarra J., J., and Ramakrishna R., S. 2003. visPerf: Monitoring Tool for Grid Computing.

Satoshi N., Mitsuhisa S. 1999. Design and implementations of nimf: towards a global computing infrastructure.

Globus 2003. Globus Heartbeat Monitor. URL: http://www.globus.org/hbm/heartbeat spec.html .

Travis B. 2003. FoodMovers: Building Distributed Applications using Visual Studio .NET.

Reilly D. and Taleb-Bendiab A. 2002. Dynamic Instrumentation for Jini Applications.

Kephart J. and Chess D. 2003. The Vision of Autonomic Computing.

Renesse, R., Birman, K., and Vogels, W. 2002. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining.

Karuppiah , D., Zhu , Z., Shenoy , P., and Riseman, E.,A Fault-Tolerant Distributed Vision System Architecture for Object Tracking in a Smart Room. Computer Vision Systems: Second International Workshop, ICVS 2001 Vancouver, Canada, July 7-8, 2001.