

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

Wail M. Omar<sup>1,2</sup>  
[w.omar@soharuni.edu.om](mailto:w.omar@soharuni.edu.om)

Bassam A. Ahmad<sup>2</sup>  
[b.abass@soharuni.edu.om](mailto:b.abass@soharuni.edu.om)

A. Taleb-Bendiab<sup>1</sup>  
[a.talebendiab@livjm.ac.uk](mailto:a.talebendiab@livjm.ac.uk)

Yasir Karm<sup>1</sup>  
[cmphykara@livjm.ac.uk](mailto:cmphykara@livjm.ac.uk)

<sup>1</sup> Liverpool John Moores University  
Byrom Street  
Liverpool, L3 3AF, UK

<sup>2</sup> 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 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 and assurance concerns of related to sensors and actuation (effectors) 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 language. 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. In addition, the paper presents a developed sensor framework to provide the basic systems awareness fabric layer for managing decentralised web services. The paper concludes with a case study illustrating the use of the sensor framework and monitoring job request to manage and schedule the sensor's operation.

## 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<sup>1</sup> 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 potentials of instrumentation, used in a dynamic regime, to monitor and manage today's 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/sensors services. However, the quality of service of sensor/actuation service received little or no attention including; there is a lack of focus on management 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

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

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 for sensors to achieve an enhanced fidelity (accuracy), performance and offer standard method for exchange information based on XML. Each sensor cluster works with the others to form a sensor farm or what we call zones. Each zone has sensor manager agent responsible to do the tasks of control, governance the deployed sensors and exchange information with other agents. The remainder of this paper is structured as follow: Section two outlines related works followed by a software sensor and actuator overlay. In Section 4.1 an illustrative example of using SADL. 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 mean any services, infrastructures and applications work under the grid computing framework should be reliable enough to achieve this future vision. This indicates the needs for significant tools of sensors to convergence the reliability of the system.

A list of monitoring systems for Grid computing are available nowadays (Lee *et al.* 2003), each one of them handles the monitoring in specific way like visPerf and NetSolve, Heart Beat Monitor (HBM) and Enterprise Instrumentation Framework (EIF). To design Instrumentation architecture of monitoring service for Grid computing, it is better to give an overview to those monitoring systems to get the benefits from the experience of other scientists to serve the same solution:

A dynamic instrumentation framework is presented by Reilly and Taleb-Bendiab (Reilly and Taleb-Bendiab 2002), which provides support to monitor and manage Jini applications. The framework adopts a service-oriented approach that employs Jini's support for code mobility, Java's dynamic proxy API and Jini's remote event model to enable runtime insertion and removal of instrumentation services. visPerf is a kind of monitoring system for Grid computing in which multiple computing entities are

involved to solve a computational problem with parallel and distributed computing tools. Originally, this work was performed as a simple monitoring facility for a specific system, NetSolve (Satoshi *et al.* 1999). However, this work can be extended to provide a monitor for Globus, Condor, Legion, Ninf and so forth, on a UNIX system (Lee *et al.* 2003). There are other types of monitor that used to read reading with grid environment such as Globus Heart Beat Monitor (HBM). Globus Heart Beat Monitor (HBM) is a monitor facility to detect faults of a computing resource involved in Globus. It checks the status of a target machine and reports it to a higher-level collector machine (Globus 2003). Another monitor is the GridMonitor Java Applet which is a kind of monitor for Globus system. It works by displaying the Grid information and server status for all sites including Globus Meta-computing Directory Service (MDS) and JAMM is an agent-based monitoring system for Grid environments that can automate the execution of monitoring sensors and the collection of event data (Globus 2003).

Astrolabe is a robust and scalable technology for distributed system monitoring, management and data mining, each host runs an Astrolabe agent. Such an agent runs Astrolabe's gossip protocol with other agents, and also supports clients that want to access the Astrolabe service (Renesse *et al.* 2002).

Windows Management Instrumentation (WMI) that consist of three parts which are as listed by (Travis B. 2003):

- *Management Infrastructure*: There is an object manager called Common Information Model (CIM). Users use CIM Object Manager (CIMOM) to handle communications between management applications and providers.
- *Managed Objects*: Management applications access managed objects using the CIM Object Manager.
- *WMI Providers*: WMI providers are 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.

Monitoring system for Grid computing problems have been addressed by (Lee *et al.* 2003), as following:

- *Heterogeneity in a Grid computing environment*
- *Access Network Due to the various network access policies including firewall, Network Address Translation (NAT), and so forth*

- *Size of Information and the speed of transferring monitored information to an appropriate location have a restriction according to the capacity of the remote system and the communication channel*
- *Interoperability Monitored information can be shared with other applications*
- *Scalability in the Grid network.*

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

#### 3.1. Framework

A sensor framework is developed to support sensor actuator generation<sup>2</sup> (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 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<sup>3</sup>. The consumers can select different types of deploying sensors with a framework and attached them to 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 of agreement and/or a given contract between the sensor consumers and providers. The readings store in a logger which can be accessed by the consumer. In view of the scalability concerns of massively decentralised systems including grid computing a zoning abstraction are introduced, where each zone has a sensor manager agent (Fig. 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<sup>4</sup>. 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).

#### 3.2. Fidelity of Sensors

Sensor fidelity, robustness and assurance are some of the major concerns considered by the developed sensor framework, which borrows many 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 selection of the correct sensor type for each monitoring request session, depending on the request information. This problem can cause an overload on the network by gathering wrong information, and selecting sensor that doesn't specify the consumer required sensor. Even if the consumer finds the required sensor, the difficulty of using such sensor still exist by the needing to know the required infrastructure and resources to run such sensor. For example, there are many types of memory sensors like the base memory , extended memory, cache bytes memory, available KBytes is the amount of physical memory, and other types of memory sensors. In this case, the consumer will face the selecting of the correct type of memory sensor difficulty. Therefore sensor framework system will be responsible in providing assistant to the consumer

<sup>2</sup> This is based on the software factory pattern.

<sup>3</sup> In addition the framework will offer the monitoring, management, control and advertisement for the deployed sensors

<sup>4</sup> The full description of the zoning abstraction and sensor manager agent is out of the scope of this paper and will be described in future paper.

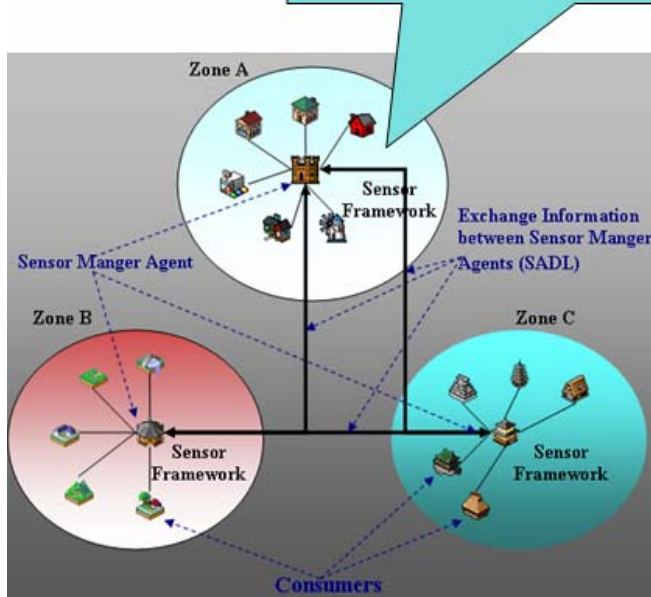
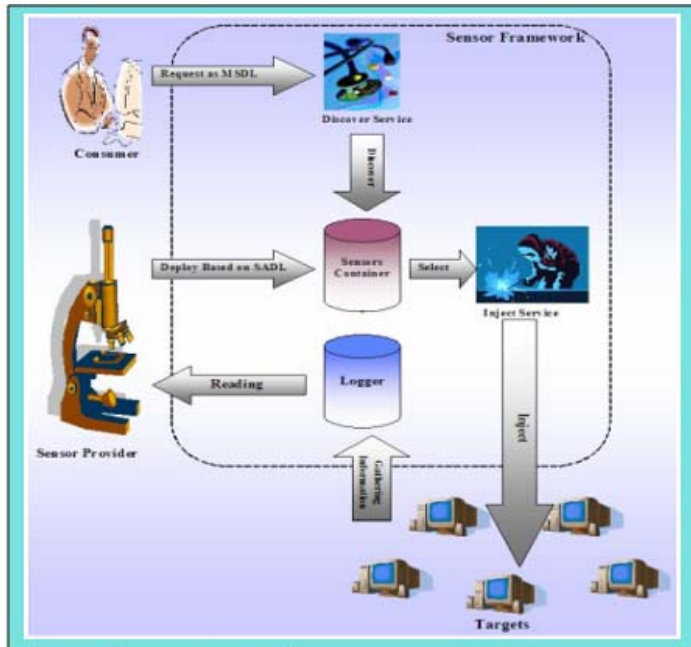


Figure 1: Sensor Framework Scenario including Sensor zoning Framework

in selecting the most appropriate type of sensor according to consumer's requirements. For this reason, SADL is suggested to be used by the sensor framework and sensor manager agent to facilitate the discovery and selection of required/correct types of sensors taking into account the consumer's requested data, target environment, and sensor required resources. SADL will feed the intelligent system (sensor manager 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 (actuators) 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) was 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 (processor, memory, web services, etc.) of sensors in

open standard format. These 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="srl">
  <SensorName>Memory Usage</SensorName>
  <SensorDescription</SensorDescription>
  <SensorType>Memory\Performance</SensorType>
  <SensorDataType>string</SensorDataType>
  <SensorDataStorageType>Local</SensorDataStorageType>
  <SensorHost>http://jmu.ac.uk</SensorHost>
  <SensorContainer>http://cmpoomar</SensorContainer>
  <SensorExecution>onDemand</SensorExecution>
  <SensorStatus>online</SensorStatus>
  <SensorMethod>getRedding</SensorMethod>
  <SensorCategory>Research</SensorCategory>
  <SensorContract ContractId="C11">
    <ContractName>CMPWOM&R</ContractName>
    <ContractLease>10/10/2004</ContractLease>
  </SensorContract>
  <SensorInterface>
    <InterfaceName>PerformanceSensor</InterfaceName>
    <InterfaceLocation>http://www.jmu.ac.uk/cmpoomar/Sensor.exe
  </SensorInterface>
  <SensorEnvironment>
    <EnvirPlatform>Any</EnvirPlatform>
    <EnvirMiddleware>.Net</EnvirMiddleware>
    <EnvirMinMoUsers>7</EnvirMinMoUsers>
    <EnvirCurrentUsers>2</EnvirCurrentUsers>
  </SensorEnvironment>
  <SensorResources>
    <ResourcesProcessor>PII 233MHz</ResourcesProcessor>
    <ResourcesMemory>64MByte</ResourcesMemory>
    <ResourcesFramework>.Net framework</ResourcesFramework>
  </SensorResources>
</Sensor>

```

Figure 3: A Simplified example of SADL.

#### 4.1. Illustrative Example

Figure 2, provides an example for deployment of sensors and actuators services with SADL framework, in that, for example a memory sensor of a given container. In this example, a memory sensor is deployed with the SADL to be available within the framework. First, sensor general information is feed to the system, followed by contract information which will be use for negotiation with the consumers. Interfaces information is giving in this stage for user usage.

Environment information and required resources are giving in the last sections of the SADL.

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

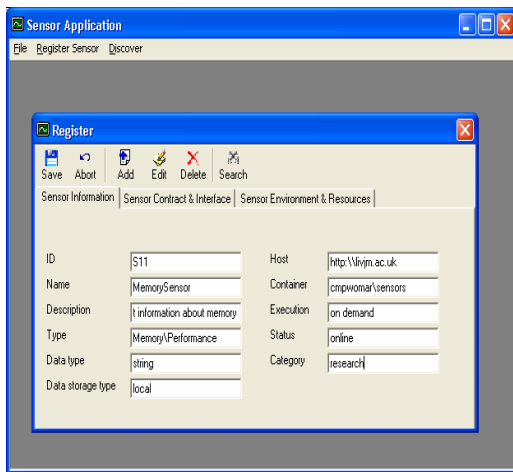


Figure 3.a: Sensor registration screen (sensor information).

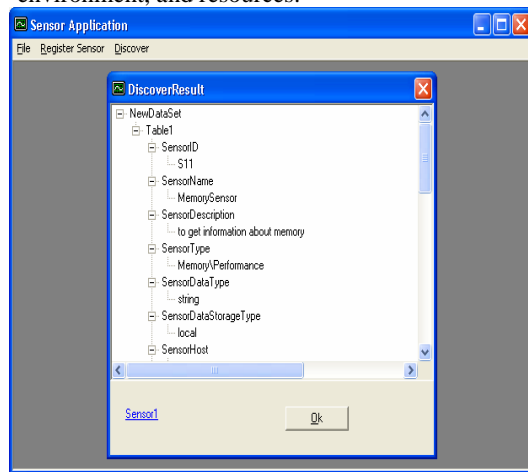


Figure 3.b: Discover results screen.

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 (MSDL)

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 first part concerns with the monitor session information, second worries about Service Level of Agreement (SLA) (contract), and the last part describes sensors information. 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. In the other hand, 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 consumer sensor requirements and finding the most suitable sensors. PIV 2.7 GHz with RAM 256 is used as a WSC (host) to include numbers of Web Services. The case study is divided into three phases: first one is creating MSDL from the consumer and sending it to the SADL, second phase is searching in SADL for the required instrumentations, and the last phase is to get reading from the clients.

The case study starts by sending the monitor session request by the consumer to the sensors framework as MSDL (Fig. 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 SSDL to pass information regarding their sensors to the framework, which will be as an advertisement location for the sensors. The SSDL 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 SSDL framework is responsible for selecting the most appropriate sensor for each target. The SSDL runs these sensors on the target and get the information and send it to the consumer (Fig. 5.d).

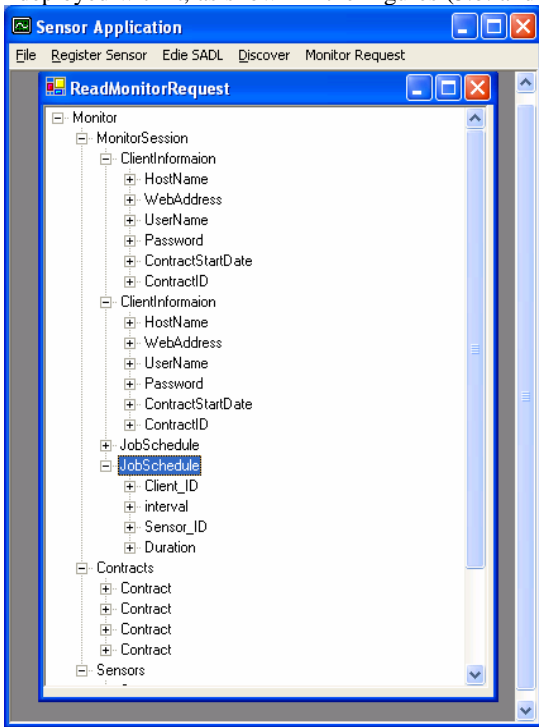


Figure 5.a: Monitor request information

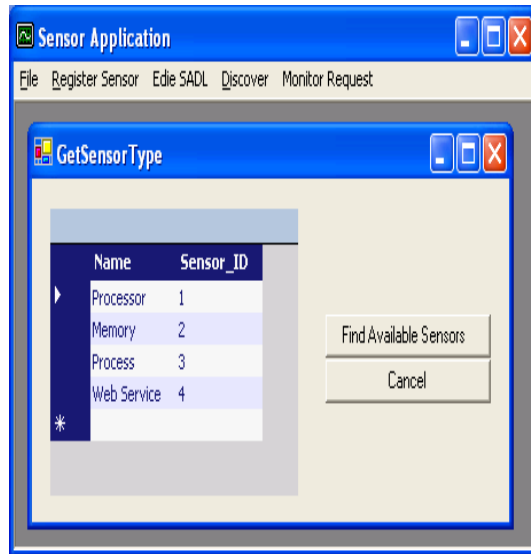


Figure 5.b: Sensors required

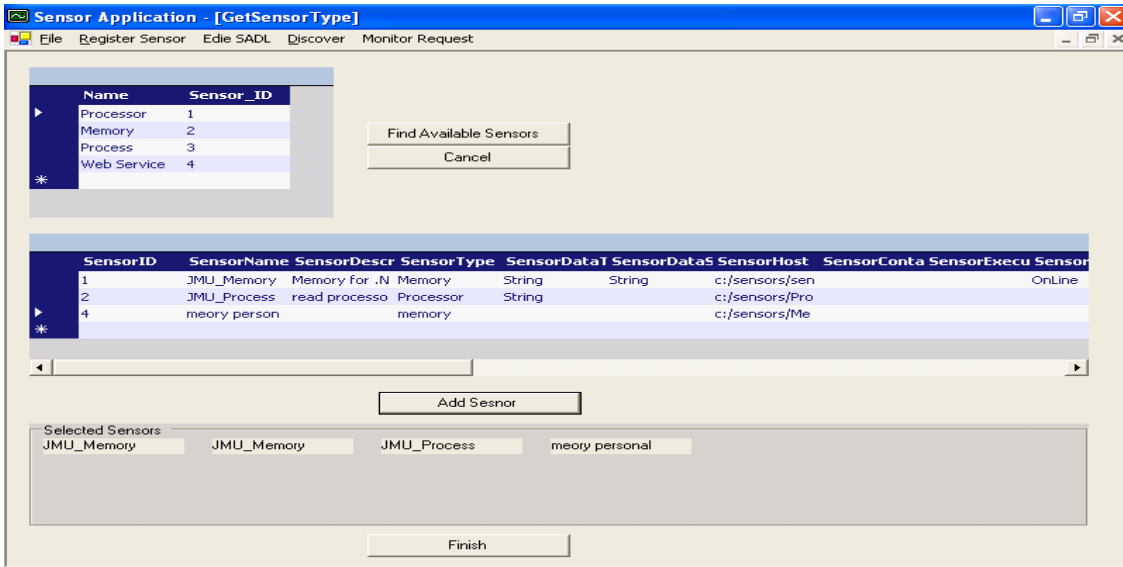


Figure 5.c: Finding sensors with in SSDL framework

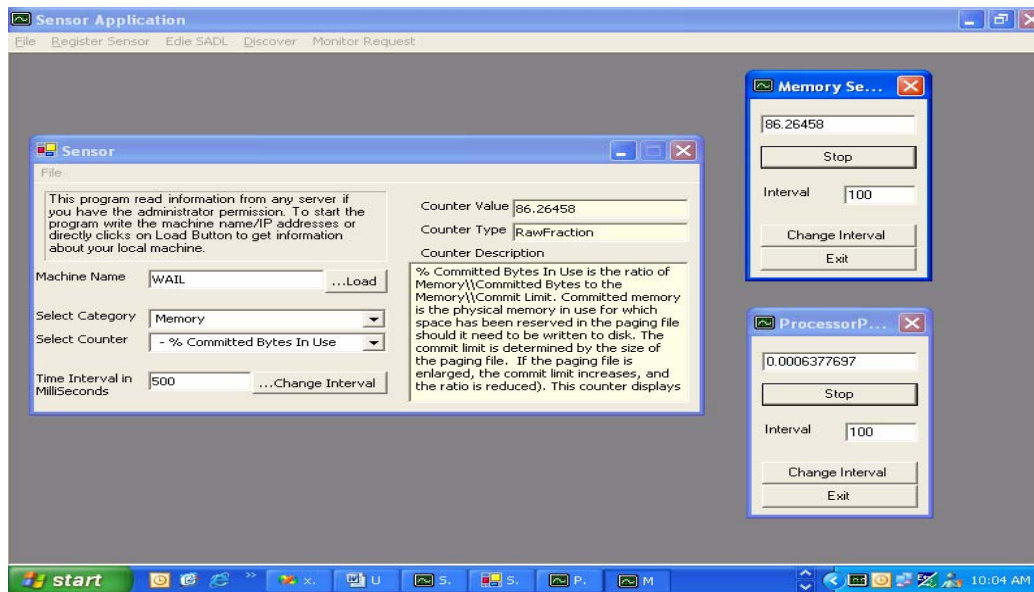


Figure 5.d: Running sensors

## 7. CONCLUSIONS AND FUTURE WORK

Sensor framework is suggested to be a broker for deploying, discovering and managing 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) was 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 more numbers of deployed sensors. A storage system is suggested to be added to the system for providing complete framework for gathering information and provided to the consumers.

## 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., Lissan, 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., Mitsuhsa 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](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.