Performance Analysis and Monitoring in Information Systems

J. P. Germano
Vodafone Portugal,
AV. D. João II, LT. 1.04.01 Parque Nacionais
1998-017, Lisboa, Portugal
Email: joao.germano@vodafone.com

Alberto R. Silva
INESC-ID
R. Alves Redol, 9
1000-029, Lisboa, Portugal
Email: alberto.silva@acm.org

Fernando M. Silva
INESC-ID
R. Alves Redol, 9
1000-029, Lisboa, Portugal
Email: fernando.silva@ist.utl.pt

Abstract—This work presents the BlackBird system, which is an analysis and monitoring service for data-intensive enterprise applications, without restrictions on the targeted architecture or employed technologies. Monitoring systems are an essential tool for the effective management of Enterprise Applications and the attainment of the demanding service level agreements imposed to these applications. However, due to the increasing complexity and diversity of these applications, adequate monitoring systems are rarely available. The BlackBird monitoring system is able to interact with these applications through different technologies employed by the Monitored Application, and able to produce Metrics regarding the application service level goals. The BlackBird architecture is composed by several Application Interface Modules, and by a central component responsible for Metrics calculation and presentation. Application Interface Modules interact with the target monitored Application in order to get performance data in a common format. These data are stored in a common repository and used for Metrics calculation and presentation. The BlackBird system can be specified through a set of pre-defined Configuration Objects, allowing it to be extensible and adaptable for applications with different architectures.

I - INTRODUCTION

The telecommunications business, in which Vodafone is included, is one of the most competitive markets. As a provider of high technology services there is a constant pressure to implement new technologies that will allow the diversification of the provided services and the improvement of existing services. Like in most large scale and technology based business, the Information Technologies (IT) infrastructure of telecommunications companies has become the main base of support to business processes, and many Enterprise Applications are now considered mission-critical, reaching a point where the performance of these applications has a direct relation to the performance goals of the entire company.

The complexity and diversity of the business rules and provided services, together with the pressure for fast implementation demand a vast portfolio of different applications in the organization. These applications can be extremely diverse, in terms of complexity, architecture, base technologies and application provider. And, as result of fierce competition environment, all these applications are required to constantly evolve and adapt, in order to implement new business requirements and support new services. In organizations such as Vodafone the teams responsible for the operation and management of these applications are faced with the challenge of assuring the best possible quality of service and the attainment of the negotiated Service Level Agreement (SLA). For this task it is essential to have monitoring systems capable of providing a comprehensive view of the application status and the most critical components, in order to anticipate performance problems and act before there is any impact on the quality of service. The currently available monitoring systems can provide efficient monitoring on the network and device level, however, due to the complexity and diversity of the applications, these systems are unable to provide the desired monitoring on the application level. Most of the available monitoring systems specifically target applications or technologies that have a large user base, Database servers, WEB servers, Unix or Windows Servers. In these cases the monitoring systems is adapted to a fixed architecture, monitors pre-determined system parameters and expected system components. This kind of monitoring ignores all the functionality that is developed over the base application. Even though, it is this added functionality that implements the business logic and produces the most relevant contribution to the delivered quality of service. The adaptation capabilities of this kind of systems are usually limited to predefined components and working parameters, which may not be the most relevant to the specific use of the application. The need for adequate monitoring applications is even more serious for applications developed in house or when the Monitored Application results from an extensive customization of a base application. In this case the only solution is to develop, also in house, the necessary monitoring systems. This extra development effort will certainly increase project cost and complexity and cause development delays.

The objective of the BlackBird monitoring system is assisting in the effective management of the extremely diversified set of applications from the Information & System Technologies department (DTSI) of Vodafone Portugal. Providing continuous monitoring of these applications and being actively used to optimize performance and diagnose problems. It has two main features: i) can monitor an extremely diversified range of applications such as the one found at Vodafone Portugal; ii) provides complex Metrics that relate to the main application goals, complementing the monitoring systems currently deployed at Vodafone Portugal. The diagram in fig. 1
represents a general view of the monitoring system, it obtains data from a number of servers using different technologies, and generates visual representations of the system status and performance that are presented to the application operators.

![Monitoring System Overview](image-link)

**Fig. 1.** Monitoring System Overview.

The need for monitoring systems was born from the need to assure high availability of the first enterprise level systems. These first monitoring systems were developed by the system providers as part of services packages. With the appearance of large scale enterprise networks it became also necessary to develop monitoring systems capable of monitoring these vast collections of computers. Thereby, it was network monitoring that originated the first vendor independent monitoring systems, such as HP Openview [1]. As the systems and networks evolved and matured so did the monitoring systems, Simple Network Management Protocol (SNMP) [2] became the most widely used management protocol and is currently the base to most network and device management systems, HP Openview [1] and Nagios [3].

However, due to the numerous programming languages and the almost infinite number of architectures and purposes, application level monitoring remains an extremely diversified field with no predominant protocols or methodologies. For application level monitoring it remains the developer’s responsibility to provide the application with the interfaces that will facilitate its monitoring. Due to development costs, time constraints or technology constraints, the monitoring capabilities provided by Enterprise Applications can vary greatly: i) application specific monitoring systems provided by the developer, Microsoft Operations Manager [4], Oracle Enterprise Manager [5]; ii) proprietary APIs to be used by general purpose monitoring systems, Manage Engine [6]; iii) integration modules for standard monitoring protocols, SNMP [2]; iv) implementation of technology specific monitoring protocols, Java Management Extensions (JMX) [7], Windows Management Instrumentation (WMI) [8].

### 1.1 - Existing Systems

The currently available monitoring systems are as diverse as the application they target, and the differences between systems can range all aspects of the monitoring process: i) general architecture; ii) application instrumentation; ii) targeted applications.

In terms of architecture, most monitoring systems employ a Manager-Agent architecture, there is usually a central management system and several agents each one dedicated to a target resource or application. A complete monitoring system will have several types of agents specifically developed for the interfaces and architectures of the various types of monitored resources. The agent will usually reside on the same host as the monitored resource, it will communicate with the monitored resource and relay the obtained data to the central management system using a protocol such as SNMP [2]. This is the architecture that emerged from the first network management systems and is currently used by all monitoring systems based on the SNMP protocol. It is especially adequate for monitoring of vast numbers of distributed resources such as computer networks Openview [1], Nagios [3] or grid computing systems MonALISA [9]. As more applications evolved from centralized to distributed and from raw processing to providing services, a new monitoring architecture became possible, Agentless Monitoring. In this case there is no need to deploy an agent on the application host, all data gathering is accomplished by remote access using standard protocols and logins. There is only a central management system that executes calls to the services provided by the Monitored Application, and by remotely accessing applications resources like databases, servers and file systems. Agentless Monitoring typically provides lightweight monitoring, but with limited depth of data gathering or monitoring capabilities. However, it is less intrusive, easier to deploy and does not require continuous development. Agentless Monitoring is especially adequate for services based applications and is the only option for proprietary and closed source applications. One example of agentless monitoring system is Longitude [10] from Heroix.

In the first monitoring systems one key element of the monitoring process was application instrumentation, which consists in modifying the existing applications in order to collect additional data during run-time. The basic instrumentation technique is to insert instrumentation code at key points of interest in the program. During execution, the instrumentation code is then executed together with the original program code, producing events that will give the main monitoring system information about the application’s current status. The importance of application instrumentation for the management of complex distributed application resulted in several technology level standards like JMX for Java and Java2 Platform Enterprise Edition (J2EE) and WMI for Microsoft Products and the Microsoft .NET framework.

Other major difference between the various available systems is the range of applications they target, monitoring systems can be application specific or general purpose. Ap-
application specific systems are usually developed by companies that wish to provide a high level of monitoring to its product range, the monitoring system will then be included in a support package or as an additional component. Naturally, application specific systems can provide the best monitoring of any application. However, it becomes impossible to combine the monitoring of applications that work together. Most of all, developing dedicated monitoring systems is a costly process that can only be supported by large companies with a significant application portfolio. Examples of such systems are Microsoft Operations Manager [4] and Oracle Enterprise Manager [5]. Third party companies will only risk developing application specific systems for applications that can guarantee a large user base. Quest Software provides versions of the Spotlight [11] monitoring system for BEA WebLogic Server, Oracle, etc. The option taken by most companies for adding monitoring capabilities to their applications is to implement a standard management technology such as SNMP, JMX or WMI, the organization employing the application will then use a general purpose monitoring system. General purpose monitoring systems aim to provide monitoring services to a range of applications as wide as possible. For this they will implement support for standard monitoring technologies and protocols, and for proprietary protocols used in applications with a large user base. These systems will also include a wide variety of agentless monitoring capabilities. One example is ManageEngine [6] from AdventNet. Due to the development costs and the limited market, application specific systems tend to be more expensive and limited to only a few applications. Organizations will employ application specific systems only for the most critical applications, or for widely used applications such as servers and databases, where general purpose systems will be used for the remaining low user base applications.

1.2 - Future Trends

Extensive research has been aimed at improving the monitoring support of Enterprise Applications, with the main focus on improving the monitoring extensions provided by the monitored application. Most of this work is focused on the JMX [12] technology which is part of J2SE platform. JMX defines an architecture in which client applications can provide a remotely accessible management interface that exposes their internal structures and resources. Although, it is only applicable to Java and the J2SE platform. It has a close resemblance to the WMI for the .NET framework, and the main concepts are applicable to other programming languages and architectures, including legacy applications [13]. The use of the standard management architectures and instrumentation techniques on enterprise applications opens the way to automated application management and self-managed applications [14].

All this research presents a common characteristic, it targets component based applications [15], and because of the component organization and the management interfaces provided by the base technologies it is possible to obtain a good detail of application monitoring. Even applications that do not employ a strict component based architecture can usually be modeled as a set of distinct programs. Whatever the purpose or technology used the complexity of Enterprise Applications dictates that the full task to be executed must be split between simpler tasks, that will be performed by different program modules. By considering a definition of Component less restrictive than the one usually associated with Component Based Software it should be possible to model any application as component based, where each component may have a number of parameters that can be used as indicators of the general application health and performance. And, by modeling the application, it should be possible to capture a more abstract level of application functionality, which is closer to the business logic and to the main application goals of quality of service.

1.3 - Comparative Analysis

The Monitoring Systems referenced in the previous section are some of the most widely used and present a representative sample of the existing monitoring solutions. The table summarizes the main features of these systems.

The objectives for this work clearly requires a general purpose monitoring application, and considering the context of Vodafone Portugal, the agentless architecture seems the most adequate, as it the less intrusive on the monitored systems. Also, in order to minimize the impact on the Monitored Applications, adding instrumentation code to existing applications is not an option.

Most of these systems allow user defined data gathering, however, the user is usually required to supply an extensive set scripts for obtaining data from the Monitored Application in a format accepted by the Monitoring System. The BlackBird System requires only a minimum of information for executing the same command, handles all data validation and conversion. Also, the BlackBird System allows the simultaneous execution of any commands regarded as necessary. Although most of these system provide some form of support to user defined Metrics, Metrics based on different data sources are usually limited to reporting purposes. Without combining data sources, Metrics will always provide a fragmented view of the Monitored Application. The BlackBird System is able to provide Metrics based on any combination of data sources. From these systems, the ones that provide a Component Based Monitoring, support only applications developed using the frameworks J2EE or .NET. Component based monitoring provides the best view of the high level functions of an application, and allows Metrics related to the application main goals. The BlackBird System introduces a simplified Component definition, for extending the concept of component based monitoring to applications that were not developed as Component Base Applications.

2 - REQUIREMENTS

The BlackBird monitoring system must provide the five key features: i) monitor a wide range of applications, being adaptable to the architecture and technologies of the Monitored
2.1 - Monitoring Requirement

The Blackbird system aims to provide a monitoring service to an application without imposing any limitations on the target architecture, therefore, the operations to be performed must be specified by a user with detailed knowledge of the Monitored Application. The Application Owner has the knowledge of the application goals and performance requirements and will be responsible for defining the Monitoring Requirement detailing the Monitoring Objects necessary to specify the required monitoring service.

A small set of Monitoring Objects is enough to specify the required monitoring service: Commands to be executed using the interfaces provided by the monitored application where the result will be stored and used to calculate Metrics; Metrics defined by a formula to be executed on the stored data to produce a result that is related to the application’s performance indicators; Alerts for evaluating thresholds on Metrics and send notifications; Graphics that use the Metric as a data source and plot the data according to the type and format; Pages for containing graphics relative to the same application component or type of component.

2.2 - Use Cases

The use of the Blackbird Monitoring System involves one Application Owner which is the manager of an Enterprise Application that will be monitored using the Blackbird system, the Blackbird Administrator which is the responsible for the maintenance and administration of the Blackbird system, and multiple Operators that accesses the monitoring pages generated by the Blackbird system and respond to alarms.

The initial deployment of a Monitoring Requirement, fig. 2, involves the configuration of all Monitoring Objects necessary to gather the required data, compute Metrics and alarms, format graphics and pages.

Once the Monitoring Requirement is deployed and the BlackBird System is started, it will initiate data gathering and produce the graphics and pages that compose the final result of the monitoring process. Now, application operators will access these pages and visualize the graphics, fig. 3, in order to determine the status of the application.
The dataStore in the BlackBird architecture isolates the BlackBird architecture from any technology details, they handle all technology specific logic like establishing a connection, authentication, formatting the command, obtaining and validating the response. Finally, they convert the command result to a normalized Extensible Markup Language (XML) document and deliver that document to the dataStore. For simplifying Metrics definition and calculation, the dataStore is designed to be accessed as a relational entity. The Metric class provides the data processing and aggregation functionalities of the BlackBird System by computing the formula specified in the Monitoring Requirement. The Graphic class provides the visual presentation to the Metric objects, it will use the output of the Metric as a data source and apply the type of graphic and the format request in the Monitoring Requirement. A Graphic may be a table of values or various chart formats. The Page class provides the base for generating the monitoring interface that will be accessed and navigated by the operators. By combining the information from the Page and Graphic objects, the BlackBird system generates the required interface as an WEB Applacation containing the requested charts and tables. The Alert class provide automatic notification of performance problems. It is defined as boolean condition to be evaluated.

### 3.1 - BlackBird Components

The Blackbird system uses an agentless architecture, it is composed of an Adaptation Layer itself composed by a variable number local Interface Modules designed for specific protocols, an Aggregation Layer that handles data storage and Metrics calculation, and a Presentation Layer for generating the monitoring pages and graphics. The Component Diagram of the BlackBird System is represented in fig. 6.

#### 3.1.1 - Adaptation Layer

Provides one of the main features of the BlackBird System, adaptation to the technologies of the Monitored Application. All technology and protocol specific processing is performed by interface modules, where each type of interface module handles a specific technology or protocol, performs all the tasks necessary for executing the requested command, converts the command output to the normalized format and delivers it to the dataStore. The Adaptation Layer also performs the first step for providing a monitoring service adapted to the architecture of the monitored system. By allowing multiple Module to execute independently it becomes possible...
to specify as many data sources as required for compiling a complete repository of performance data that will allow the calculation of any relevant Metrics. The ModuleManager component controls execution of the Interface Modules and manages the dataStore of those objects. For this work only two modules where developed, an SQL module that uses JDBC to connect to the monitored Database for executing an SQL statement and an WEB Services module that implements the Apache Axis Framework for dynamically invoking Web services. These technologies are base to most of Vodafone applications, and allow the demonstration of the BlackBird capabilities in combining data obtained using different technologies.

The BlackBird architecture is expandable to other types of protocols. Adding support for aditional protocols to the BlackBird system requires no changes to existing components only the development of: i) a new interface module to handle the required protocol; ii) two new configuration tables and associated views; iii) new stored procedures for the edit operations on the new type of module.

3.1.2 - Aggregation Layer

Stores all performance data in an organized an easily accessible form, performs Metrics calculation and alerts verification. The Aggregation Layer contains the dataStore attributes from all existing Module objects implemented as database tables and views, the combination of all these objects constitutes a complete repository of all performance data gathered from the Monitored Application. And, since all this data is accessible through relational queries, it should be possible to implement any Metric required by the Application Owner. The Aggregation Layer is responsible for other of the main features of the BlackBird System, Component Based Monitoring. Since all performance data gathered from the application can be used as input for the Metric objects, it is possible aggregate the data collected by different Modules using the component identifier to produce Metrics that provide a complete view of all aspects for that Application Component. The MetricManager component is responsible for creating and updating the implementation of Metric formulas, and actively evaluating alert conditions.

3.1.3 - Presentation Layer

Is responsible for the final output of the BlackBird Monitoring System, which are monitoring pages containing visualizations of the status and performance of the Monitored Application. The monitoring pages are generated from the Graphic and Page objects and deployed on an Application Server as a WEB Application. However, the automated generation of the WEB Application was considered to be out the scope of this work, so the Webcockpit [16] application is used to generate the monitoring pages based on the configuration file created by the BlackBird System from the existing Page and Graphic objects. The PageManager component is responsible for managing the generation and deployment of the WEB Application that provides the monitoring interface.

3.2 - Result Format

In order to isolate the BlackBird architecture from any technology details it is necessary to define a normalized format for the results data, this format must be: i) capable of containing any result; ii) easily convertible to a relational format. According to the application model defined at the beginning of this section (fig. 4), and to support component level monitoring, the output of a command executed on the monitored system may be defined as a tree of components each containing any number of working parameters, including a component identifier. According to this definition, any result may be formatted as a XML document with the schema from fig. 7.

4 - IMPLEMENTATION AND DESIGN

The BlackBird Database serves two purposes, storage and processing of Performance Data from the Monitored Applications, and support to the configuration and control of Monitoring Objects. The BlackBird System is intended for an enterprise environment where security and role separation is always a major concern, because of that for each Monitored Application there is a dedicated database schema for contain the implementations of dataStore Metric and Alert that exist for that application. The main BlackBird schema contains the definitions of all Monitoring Objects in use.

4.0.1 - Main Schema

The BlackBird main schema presented in fig. 8 is used for controlling the monitoring process, contains Monitoring
Objects definitions, and additional entities for implementing application separation and access control.

Fig. 8. Main Schema.

4.0.2 - Monitored Application Schema

For each application there is a separate schema that contains the result data obtained from the Monitored Application. This schema contains the database objects that implements the dataStore and Metric. The creation and modification of all database objects in the Client Schema is performed automatically by the BlackBird System as a result of changes to the Configuration Objects. Figure 9 presents an example schema containing the implementation of two dataStore and two Metric objects.

Fig. 9. Monitored Application Schema.

The dataStore provides a vital function of the BlackBird System, it receives data in the form of XML documents and then provides access to that same data through a relational interface. The final implementation of dataStore is composed of a Results Table and a Translation View. result data is stored in XML format in the Results Table, then based on the format of the XML result the BlackBird System creates a view for selecting all XML elements as columns of result set. The complex implementation of the dataStore allows the implementation of the Metric object simply as a database view created from the SQL formula defined in the Metric.

One of the most important functions of the BlackBird System is the management of the dataStore. The creation of a new Module requires the creation of a new Results Table and the associated Decoding View. The Results Table has a fixed definition so it’s creation posses no problems. The Decoding View definition however, performs the mapping of XML elements in relational columns, thus it depends on the results format. For creating the Decoding View, the ModuleManager requests the Module a XML Schema Definition (XSD) which describes the XML document that it will be used for delivering command results. To determine the result format the Module execute the command on the Monitored Application for the first time and use the first command response to produce the XSD schema that will describe the result XML document. The management process of Metric is much simpler, upon changes to the formula, the existing view is dropped and recreated according the new SQL select statement.

4.1 - User Interface

The BlackBird interface supports two main functions, visualization of the final output of the monitoring process, and configuration of the Monitoring Objects. The interface is implemented as a WEB application running on an Apache Tomcat application server. Figure 10 represents the page structures of the BlackBird interface.

Fig. 10. BlackBird Interface.

The starting page for both monitoring and management is the Application Owner Page, it presents a list of all applications currently being monitored and provides links to the Application Pages and to the configuration page. The
Communications between these systems is provided by middleware of WEB services from WebMethods. The PPB application was developed at Vodafone Portugal, it contains an Oracle database and was developed using C and PL/SQL. There is already a dedicated monitoring system for PPB which was developed together with the main application, and provides performance information and alerts. Although it delivers extremely valuable information, it implements a static set of configuration objects and is not applicable to any other applications. This example will add monitoring of recent functionalities of the PPB application that have not yet been included in the dedicated monitoring system, currently only basic unavailability alerts are provided.

One of the tasks performed by the PPB application is credits to the customer balance in the IN platform. Recent Business Requirements "Vita Recarga" and "SOS Extra" have introduced new scenarios on which it is necessary to apply credits to the customer balance. When the condition defined in the Business Requirement is verified, the PPB system generates a request for balance credit, this request is then added to a queue for processing. The processing queue exists as a database table containing requests to be processed, and the processing of each request requires a call to a WebService for confirming current balance. The main performance criteria for this task are the average processing time and maximum processing time for these requests, the current monitoring includes alarms if thresholds on these parameters are exceeded.

5.0.2 - PPB Monitoring Example

For monitoring this aspect of the PPB application, problems diagnosis and performance tuning, it would be extremely helpful to have a line chart presenting the average request execution time, the average Web Service response time, and the difference between the two times, which represents the contribution of the PPB System to the total processing time. This is a relatively simple example, however, the presented parameters have direct relation to the application performance goal. Furthermore, by splitting the application goal into sub parameters, it becomes easier to identify the contribution of each application component to the final result. The output of this graphic is also helpful for performance tuning, since it illustrates the impact of configuration changes on each of the parameters that contribute to the application performance goal, it should be easier to establish a compromise that produces the best overall results.

5.0.3 - Mediation Device

The second example applies to the MD system also from Vodafone Portugal. The Mediation Device processes billing records of all traffic types, GSM, 3G, GPRS, SMS, etc. Collects billing records from all Network Elements, validates and selects billable records, pre-rates and sends billable records to the billing systems (ARBOR) and PPB. The Mediation Device was also developed at Vodafone Portugal, it contains an Sybase database and was developed using C and Transact SQL. Currently all monitoring and alerts on the Mediation
Device Application are provided by scripts and programs integrated in the Openview Operations System. This example will use some of the same scripts as input and use the data processing capabilities of the BlackBird System for providing Metrics directly related to the application performance goals. The Mediation Device processes various types of records referred to as Data Stream, and for each Data Stream, there are various Network Elements. Thereby, the Mediation Device architecture can easily be modeled into components. The basic components are the Network Elements that have parameters such as current delay and records processing rate. These elements can be grouped into the main application components the Data Streams. The most critical SLA defined for the Mediation Device are all related to the delay between record generation on the network element and its delivery to the destination billing systems. Given these, the fundamental indicator of performance for the Mediation Device is the number of records processed per unit of time. This example uses the existing scripts as data source for creating charts on the delay associated with network elements and records processing rate according to type of traffic.

5.1 - Configuration Process

Once the required output is defined, the Application Owner must elaborate the Monitoring Requirement in order to specify the Configuration Objects necessary for producing the desired monitoring output. The Monitoring Requirement for the PPB example contains the following Configuration Objects: i) Login details for the PPB database; ii) Login details for WEB Service calls; iii) SQL command for obtaining the average processing time; iv) WEB command for obtaining the Web Service response time; v) Metric for combining data from both commands; vi) A line chart that uses that Metric; vii) A page for containing the the Graphic.

The configuration of the Monitoring Requirement can be performed either by the BlackBird Manager or the Application Owner using the Configuration Interface described in the previous section. Starting from the Configuration Page the user selects and action to be applied on an existing Configuration Object or the creation of a new object. Then the user is presented with a form for providing or updating the configuration parameters of the object. Upon edit confirmation the user is returned to the Configuration Page and the BlackBird Manager proceeds to execute all the necessary actions such as creating and updating the objects in the Application Schema.

Figure 11 shows the monitoring configuration page for the PPB application listing the Monitoring Objects configured for this example.

5.2 - Monitoring Output

The final output of the PPB monitoring example is shown in figure 12 This chart shows the evolution of the requests processing time and the contributions of the WEB Service call and PPB processing. This capture shows normal conditions, where the contribution of WEB Service call to final processing time is negligible.

In this example the BlackBird System was used for fast implementation of a detailed monitoring service, and to generate Metrics that aggregate data obtained using different technologies, Database access and WEB Services call.

The chart from figure 13 shows the evolution of the delay in Call Records processing for the Network Element components of the MD application, only the ones that present a delay above a defined threshold are shown. Since this is batch process the delay tends to evolve in steps. The figure was captured during a peak hour so there is a tendency for an increase in processing delay. On the bottom there is a table that shows current processing status of Network Elements.

Figure 14 shows a chart presenting the evolution of number of Call Records processed per hour for the Stream components of the MD. This figure was captured during a maintenance operation that required stopping the processing of some types of traffic. In this example the BlackBird System was used to improve the existing monitoring to detail the information on the component level, and provide real time visualization of Metrics directly related to the SLA an performance indicators.

These examples are not enough explore the full capabilities of the BlackBird System, all aspects the architecture are aimed at flexibility, so that an Application Owner may implement any monitoring he regards as necessary. Even though these simple examples are enough to show that the BlackBird System can
be effectively used for providing an advanced monitoring service to real world enterprise application.

6 - CONCLUSIONS AND FUTURE WORK

The BlackBird System application examples on production Enterprise Applications have demonstrated that the BlackBird System can be used for monitoring enterprise applications with non standard complex architectures, and that it can provide valuable metrics close to the main application goals. More specifically, these examples show that the BlackBird System can be used for:

- fast implementation of a detailed monitoring service;
- generating Metrics that aggregate data obtained using different technologies;
- improving the existing monitoring to detail the information on the component level;
- providing real time visualization of Metrics directly related to the SLAs as performance indicators.

In terms of data collection and interface to the Monitored Applications, the natural improvement is the development of additional Interface Modules for additional protocols: Remote Shell for access to unix systems, JMX for J2EE applications, WMI for .NET application. Another possible improvement would be usage of dynamically generated arguments for the commands executed on the Monitored Applications.

Although the data repository provides easily accessible source of data, the definition of Metrics is still the most demanding configuration task. This task can be greatly simplified by the addition of a query building interface, either as an applet on the Edit Page or as a separate application.

For visualization purposes, it could be possible to have better drill down capabilities and root cause analysis by automatically generating a hierarchical view of the Monitored Application, such as the one provided by the Spotlight [11] monitoring system. This would require an improved definition of application modeling that also includes the relations between Applications Components.

REFERENCES