

# *AQoS Management Functional Description*

## *Version 1.0*

Prepared by:  
**Mike Tsykin**  
**Systems Engineering Research Center**  
**Fujitsu Australia Limited**

Contributions acknowledged:

- 1. T. Bishop**  
**VIEO**
- 2. Dr. J. Bouhana**  
**Performance International, Inc.**
- 3. Dr. N. Davies**  
**Predictable Network Solutions**
- 4. J. Hammond**  
**JPH Associates**
- 5. C. Langshaw**  
**Fujitsu Australia Limited**
- 6. S. Marcie**  
**Tonic Software, Inc.**
- 7. L. Wittorff**  
**The Boeing Company**

Table of Contents

- 1. Summary and objectives..... 4
- 2. Vision and User Requirements ..... 4
  - 2.1 AQRM’s vision for the future..... 4
  - 2.2 Survey..... 5
  - 2.3 Netforecast’s classification..... 6
- 3. Application Quality / Resource Management..... 7
  - 3.1 Call To Action ..... 7
    - 3.1.2 Current situation ..... 7
    - 3.1.3 Solution & Benefits ..... 8
  - 3.2 Structure ..... 9
- 4. Overview Requirements and General Considerations ..... 10
  - 4.1 Measurement of Quality of Service ..... 10
    - 4.1.1 Stochastic vs. Deterministic Approach ..... 10
    - 4.1.2 Resolution of Measurement vs. Process Time Scale ..... 11
  - 4.2 Reactive and Predictive Approaches ..... 12
    - 4.2.1 Reactive Approach ..... 12
    - 4.2.2 Predictive Approach ..... 12
  - 4.3 Basic Technical Principle: Near-Real-Time, Asynchronous Operation ..... 12
  - 4.4 Role of Automation ..... 13
  - 4.5 Hierarchy of Predictive Approaches..... 14
- 5. AQoS Management Process ..... 15
  - 5.1 Overview ..... 15
  - 5.2 Process..... 15
  - 5.3 Capabilities of the system..... 17
  - 5.4 Implementation considerations..... 18
- Appendix 1. Descriptions of Work Groups ..... 19
  - A1.1 General ..... 19
  - A1.2 Work Group Definitions ..... 19
- Appendix 2. Definitions. .... 23
- Appendix 3. AQoS Implementation Considerations ..... 25
  - A3.1 General ..... 25
  - A3.2 Data Capture ..... 25
    - A3.2.1 General ..... 25
    - A3.2.2 Data capture and standards ..... 25
    - A3.2.3 Data items ..... 25
  - A3.3 Data Storage ..... 27
  - A3.4 Data Transformation..... 28
    - A3.4.1 General..... 28
    - A3.4.2 Overview ..... 28
    - A3.4.3 Advantages and Disadvantages of the Classes ..... 28
    - A3.4.4 Examples of metrics by class..... 29
  - A3.5 Information Delivery ..... 30
    - A3.5.1 General..... 30
    - A3.5.2 Trigger ..... 30
    - A3.5.3 Context..... 30
    - A3.5.4 Method..... 30
  - A3.6 Consumption of Information ..... 31
- Appendix 4. References..... 32

**Table of Figures**

Figure 1. Survey of QoS requirements in recent Outsourcing tenders ..... 6  
Figure 2. Netforecast Application Performance Classification Framework..... 7  
Figure 3. AQRM Levels and Interactions..... 9  
Figure 4. Management Architecture..... 10  
Figure 5. Measurement Resolution vs. Process Time Scale ..... 11  
Figure 6. Four *asynchronous* loops ..... 13  
Figure 7. Predictive Approach Hierarchy ..... 15  
Figure 8. QoS Management process..... 16  
Figure 9. Structure of Work Groups..... 19  
Figure 10. Compound metrics ..... 29

# 1. Summary and objectives

The Enterprise Management Forum of The Open Group, within their Application Quality and Resource Management (AQRM) initiative embarked on a project to develop an AQRM Industry Standard. Brief versions of the AQRM Vision Statement and the 'Call to Action' are presented below. For the full text of the call to action, please see [References, 1].

There is no formal, comprehensive AQRM structure in existence today that can be used as a starting point for the proposed standard. Therefore, it was decided to develop a standard set of requirements and an architecture that can be used as a basis for further work and whenever possible share this approach with other standards bodies and consortia. This paper represents a Functional Description of a generic Application Quality of Service (AQoS) system. Specifically, it contains:

- Customer requirements that drive this need
- Basic definitions relevant to AQRM
- General principles and considerations involved in AQoS Management
- Process description of AQoS Management
- Functional description of key elements of the required infrastructure

This information will be used to formulate the architecture and detailed functionality of a generic AQoS system and:

- Identify existing standards (e.g. SNMP/MIB)
- Identify existing standards that could be extended (e.g. CIM/WBEM)
- Define new standards that should be developed
- Provide solutions for the required new standards and/or challenge other organizations working in closely related fields to develop these (DMTF, OASIS, GGF, SNIA, IETF and others).
- Capture the views of users and customers and achieve their active participation to ensure that major vendors follow through with their support

This paper addresses two primary subjects. Section 2 – Section 4, with Appendix A focus on defining the problem, the AQRM mandate and vision for the future, as well as the proposed structure of the group to address the issues. Section 5 and Appendices outline functional requirements for an AQRM solution, including definitions, behaviors and aspects of the architecture framework that can meet the needs defined in the earlier sections.

This document is a part of the work of the AQRM activity of the Enterprise Management Forum of The Open Group. The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow™ will enable access to integrated information within and between enterprises based on open standards and global interoperability.

## 2. Vision and User Requirements

### 2.1 AQRM's vision for the future

Our vision for future IT environments is one where business managers can set the priority of various business functions and high level applications, and the data center and network infrastructure can respond by allocating resources on an end-to-end basis to match the user's quality of service needs. We believe the next-generation data center/network management solutions need to support diverse, highly distributed applications on dense, highly distributed, modular systems.

## AQoS Management Functional Requirements

Our objective is to define an open framework into which application and hardware component providers can plug their product, and have it work together with all other components to deliver priority tasks and adjust to meet critical workloads based on policy from a business viewpoint. We believe that various vendors will achieve in a range of ways but all will leverage standards-based instrumentation and management-oriented data collection.

If you are a user of IT, then you should participate in the AQRM effort in order to help define and the standard and to gain an understanding of what can be accomplished when applications and components from different realms can understand the same Quality of Service language. If you are a provider of IT, then you should participate in the AQRM effort to ensure that its work will provide services relevant to the broadest set of technology providers, thereby directly benefiting your customers.

### 2.2 Survey

A survey of QoS requirements in 23 recent tenders for Outsourcing of System Management services is presented in Figure 1 below. It shows:

- QoS Management is requested in 100% of cases
- QoS Management comprises three well-understood disciplines:
  - Capacity Planning: 22 cases (95%)
  - Performance Management: 23 cases (100%)
  - Service Level Reporting: 23 cases (100%)
- User-based Measurement of QoS is requested as follows:
  - Availability: 23 cases (100%)
  - Help Desk: 23 cases (100%)
  - End-To-End Response Time: 9 cases (45%)
  - End-To-End Service Management: 9 cases (45%)

Maturing Open Systems, geographically dispersed client-server and web services-based applications, and ubiquitous Web-based access are all driving the demand for QoS Measurement and Management. Increasing penetration of SANs, popularity of outsourcing and the emerging disciplines – Grid, On-Demand, Utility and Autonomic Computing – are expected to increase the importance of these requirements in the future.

In fact, End-To-End Service Management involves the following:

- Measuring end-to-end levels of service for User-based units of service (e.g., Application Transactions, Service Level Agreements, etc.)
- Breaking such units down into identifiable, measurable components (e.g. SQL requests for Application Transactions)
- Attributing end-to-end service levels and resource consumption to such units and their components. This involves tracing them through multi-domain (geographical, technological, application, supplier, etc.) infrastructures
- Identifying and predicting current problems and future requirements in User terms

The requirements above are not new, except for the emphasis on cross-domain tracing and service unit (e.g., transaction) breakdown. However, notable exceptions notwithstanding, consistent End-To-End Service Management is not currently possible due to lack of cross-domain standards; industry-wide standard development is required.

Another view of User Requirements is expressed in a white paper developed by the Open Group in Spring 2002 titled “Service Level Agreement – A White Paper” [References, 6].

## AQoS Management Functional Requirements

No	Scope of management requested						QoS Services			QoS measurement requirements			
	Server	Network	Storage	App.	ETE	Other	CapPlan	PerfMgr	SLR	Availability	ETE RT	Help Desk	Resource
1	y	?	y	?	?	?	y	y	y	y	?	y	y
2	y	n	y	y	n	y	y	y	y	y	n	y	y
3	y	y	y	y	y	y	y	y	y	y	y	y	y
4	y	n	n	y	y	y	y	y	y	y	y	y	y
5	y	y	y	y	y	y	y	y	y	y	y	y	y
6	y	y	y	y	y	y	y	y	y	y	y	y	y
7	y	n	n	y	n	n	y	y	y	y	n	y	y
8	y	y	y	y	n	n	y	y	y	y	n	y	y
9	y	?	y	?	?	?	y	y	y	y	?	y	y
10	y	y	y	n	n	y	y	y	y	y	n	y	y
11	y	y	y	n	n	y	y	y	y	y	n	y	y
12	y	?	y	?	?	?	y	y	y	y	?	y	y
13	y	n	y	y	y	y	y	y	y	y	y	y	y
14	y	y	y	n	y	y	y	y	y	y	y	y	y
15	y	y	y	n	n	y	y	y	y	y	n	y	y
16	y	n	y	n	n	n	y	y	y	y	n	y	y
17	y	y	y	y	y	y	y	y	y	y	y	y	y
18	y	?	y	?	?	?	y	y	y	y	?	y	y
19	y	y	y	y	y	y	y	y	y	y	y	y	y
20	y	y	y	y	n	y	y	y	y	y	n	y	y
21	y	y	n	y	n	y	n	y	y	y	n	y	y
22	y	n	n	n	n	n	y	y	y	y	n	y	y
23	y	n	y	y	y	y	y	y	y	y	y	y	y

Figure 1. Survey of QoS requirements in recent Outsourcing tenders

### 2.3 Netforecast's classification

Peter Sevcik of Netforecast Inc. proposed a framework for classification of Application Performance requirements (Figure 2), and he included *Quality* as one of the required categories. In his own words: “Quality (is) the quality of the technical aspects of the user's experience with the system”. This places Quality squarely amongst the six most important Application metrics. Given Peter Sevcik’s extensive network design experience, his classification fairly represents the user requirements. Please refer to Reference 5 for the full text of the paper.

## AQoS Management Functional Requirements

		Asset Management			Experience Management		
		Provisioning	Efficiency	Protection	Accessibility	Quality	Safety
Real Time	Voice over IP						
	Video Conference						
Transactional	Terminal-Host						
	Client-Server						
	Web						
	Web Services						
Data Feed	Streaming Audio						
	Streaming Video						
	Telemetry						
Bulk Data	Email						
	Peer-Peer						
	File Transfer						

Figure 2. Netforecast Application Performance Classification Framework

### 3. Application Quality / Resource Management

#### 3.1 Call To Action

The summary of AQRM's Call to Action is presented below. For the full text, please refer to [References, 1].

The IT industry is starting a major technology shift to modular computing and component-based, web services application architectures. In addition, businesses are driving toward the real-time enterprise model. Taken together, these driving forces represent a fundamental change in the way systems will be implemented and, even more importantly, managed, thus requiring a new management paradigm. If the IT industry is to respond to these changes, we will need an appropriate set of Application Quality/Resource Management standards to allow the integration of applications and management systems. However, at present there is no single, cohesive industry standards effort focused on addressing these two issues.

This represents a strategic opportunity to form an influential standards initiative that will:

- Establish industry acceptance of an open architecture for managing such environments
- Accelerate the availability of application and infrastructure instrumentation
- Provide a vehicle for all relevant constituencies of the IT industry to cooperate
- Solve the applications management problem today and tomorrow

The key challenges to making this happen are to recruit a critical mass of key players in a timely fashion and to commit sufficient resources to lead the group and accomplish the technical work.

#### 3.1.2 Current situation

The IT industry movement towards modular computing and component-based application architectures will be accompanied by an equally significant evolution in the way data center applications, and even the data centers themselves, are designed in the coming years.

This data center evolution is driven by the removal of constraints and bottlenecks that are being designed out of modular equipment. This results in scalability and flexibility of both the applications and the

## AQoS Management Functional Requirements

equipment on which they run. The computing, network and storage resources required to deliver applications to users are following the trend towards modularity, enabling increasing levels of disaggregation – i.e., partitioning functionality onto modular, granular systems components which are then re-aggregated to provide high density, low power, and high performance scalable systems.

We can already see this trend unfolding in the server space, where modular blade servers are rapidly being deployed as replacements to large, monolithic standalone servers. To complicate matters further, web services technologies enable components of a single end-user service or function to reside in multiple data centers. The next -generation data center management solution needs to support diverse, highly distributed applications on dense, modular systems.

The storage networking industry is years ahead of server and software industries in management / virtualization technologies and standards convergence. However, their solutions do not provide a complete, end-to-end, application-oriented, solution for the data center. They will contribute components of the complete solution that needs to be developed.

The convergence of cluster/modular/blade architecture with autonomic/self-managing/utility computing requires a new management paradigm. The new management paradigm must define and embrace certain critical standards to span all aspects of the data center, including networks, servers, storage platforms, and distributed applications software. As no single company rules the data center, the achievement of this ambitious goal will require a cross-industry effort.

There are currently no standards activities that are addressing the totality of this problem. There are some components available that can be used to form part of the solution, but there is a need for both major integration work and the creation of new solutions in order to achieve a comprehensive and practical solution.

### 3.1.3 Solution & Benefits

In order to deliver suitable interoperable, plug -and-play solutions, there is a need for an industry standard architectural framework, together with a set of appropriate standards profiles, which will incorporate both existing standards and any additional ones that are required to be developed.

These standards will help manage the behavior of both new applications and legacy applications as they share the infrastructure within the data centers and the networks connecting them. They will be designed to ultimately support all phases of the application life cycle. Furthermore, the resulting framework will ensure that each layer can evolve (and that the vendors in each layer can innovate), while at the same time ensuring that the system can be managed as a whole, and that the individual pieces can interoperate.

With new equipment and management systems based on the new standards, the following benefits can be realized:

1. Application developers can develop software that interacts with the resources it uses, allowing better control over the resources needed to deliver the level and granularity of Quality of Service required by application users.
2. Businesses can more fully understand the Quality of Service they are delivering to their application users, which helps them to protect against loss of revenue and enables them to prioritize better their technology investments.
3. Data center managers can deploy new applications faster and run the ones they have more efficiently, increasing the return on investment (ROI) of their deployed systems.
4. Data center managers no longer need to use different, disparate, management systems for all the technologies in their environment, reducing the Total Cost of Ownership (TCO) associated with resource management.
5. The equipment vendors that support this transition will be part of a new way of managing the applications that run global commerce, ensuring participation in a rapid-growth market in years to come.

## AQoS Management Functional Requirements

This means that as new applications are rolled out for the ever-hungry enterprises that run large data centers, vendors can look forward to enterprises purchasing a new generation of equipment to deploy. This is important because the annual spend on IT totals over US\$1.7 trillion annually. Of this amount, over US\$850 billion is spent on equipment, software, facilities and communications and another US\$800 billion is spent on external services and staff.

### 3.2 Structure

The following high-level view illustrates the interactions that the AQRM initiative will address.

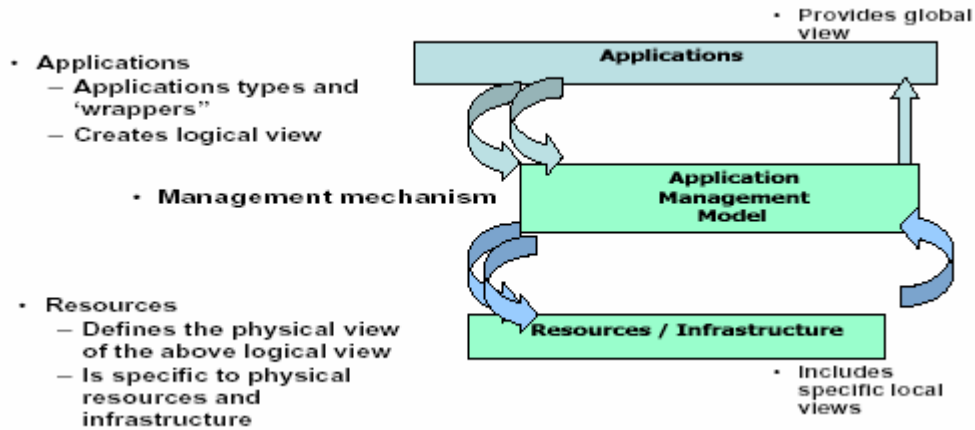


Figure 3. AQRM Levels and Interactions

The AQ/RM initiative consists of the following topic areas:

1. High-level Management Architecture
2. Application Quality of Service Specifications
3. Monitoring and Control Instrumentation
  - i. Application
  - ii. Application Services (“middleware”)
  - iii. Operating System (Server and Client)
  - iv. Network
  - v. Storage
  - vi. Internet/WAN
4. Management
  - i. Distributed Application Management
  - ii. Standard Behaviors
  - iii. Tool interaction processes/protocols
5. Human-Computer Interface
  - i. Standard policy templates
  - ii. Standard business dashboards

A number of Work Groups were formed in order to develop the necessary details (please refer to the Appendix 1). The work of the Architecture Work Group resulted in the QoS Management Architecture (Figure 4). AQoS Specification Standards Work Group produced this paper. Work of other groups is in progress.

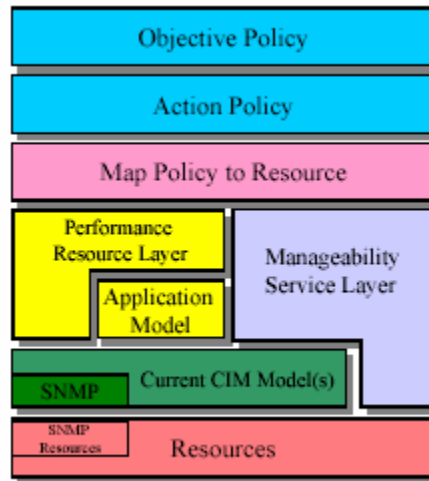


Figure 4. Management Architecture

## 4. Overview Requirements and General Considerations

### 4.1 Measurement of Quality of Service

By definition, this occurs after the service has been delivered by *Supplier* and used by *Customer* (please refer to the Appendix 2 for the definitions of the underlined terms). Examples are:

- *SLA* report is delivered after the end of a *Service Period*
- End-To-End Response Time is calculated once a transaction is completed
- Packet loss and similar Telecommunication metrics are calculated over agreed periods of time

Such disparate activities require different approaches to measurement. In principle, one of the three approaches can be used: Real-Time, Near-Real-Time or Non-Real-Time. In selecting an appropriate one, two major issues should be considered:

- Stochastic vs. Deterministic approach
- Resolution of Measurement vs. Process Time Scale

#### 4.1.1 Stochastic vs. Deterministic Approach

The following 'Decision Tree' is proposed:

- Is the measured process repetitive? If **YES**:
  - Is instrumentation available? If **YES**:
    - Are the overheads of measurement insignificant, compared with the scope of the measured process? If **YES**:
      - ❖ Is it possible to process the data volume the measurement will generate? If **YES**:
        - ◆ Are the costs justified? If **YES**:
          - Choose the Deterministic approach
- If the answer is **NO** at any stage:
  - Is it possible to obtain a representative quantity of data? If **YES**, choose the Stochastic approach
  - If **NO**, reconsider approach to measurement

## AQoS Management Functional Requirements

In real life, the Stochastic approach is selected most of the time. However, in doing so, one must be careful. The Stochastic approach relies on building a relatively small set of numbers to represent the composite, time aggregated, characterization of an entire process. The underlying mathematics of such a measurement process has an assumption that the samples are from a specific random process. Care should be taken in the interpretation of the resulting measures if this is not the case.

### 4.1.2 Resolution of Measurement vs. Process Time Scale

The Deterministic approach to measurement is, by definition, Real-Time. If every instance of a process is to be measured, then that is Real-Time indeed.

That is not so for the Stochastic approach. The following considerations apply:

- Real-Time measurement is appropriate if measurement resolution is high – that is, many measurements are available for each instance of a process
- Near-Real-Time measurement applies if measurement resolution is relatively low, but still comparable with the time scale of the process (for example, every other instance is measurable). It also applies, however, for homogenous processes – where all process instances are the same for the purposes of measurement
- Non-Real-Time measurement applies when resolutions are low.

This is illustrated by the Figure 5 below.

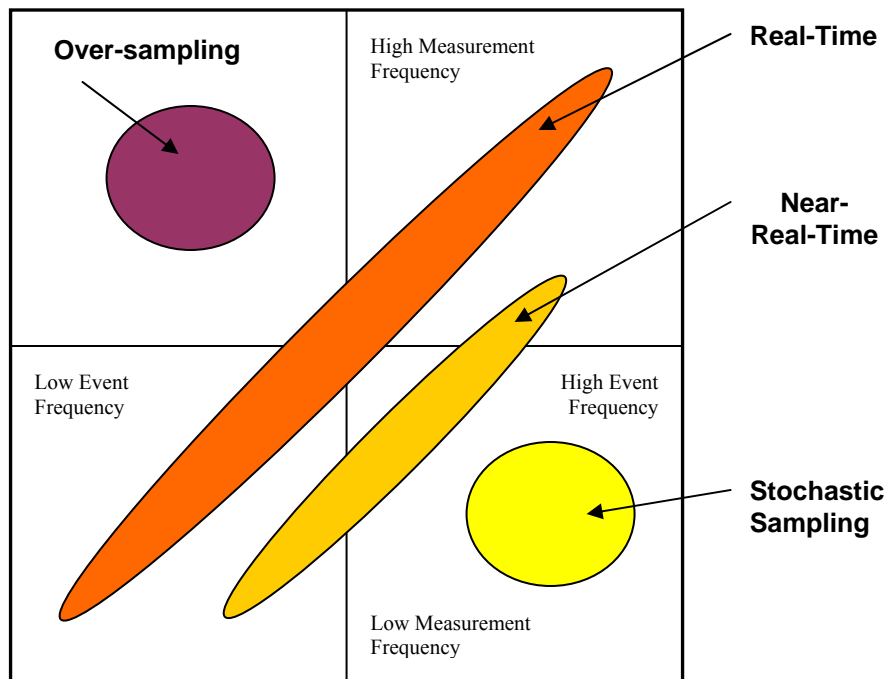


Figure 5. Measurement Resolution vs. Process Time Scale

## 4.2 Reactive and Predictive Approaches

### 4.2.1 Reactive Approach

The conventional way of managing service is to measure QoS and determine whether the requirements were met. This means that a problem (if any) is detected and reacted to after the event.

This approach presents problems for everybody, for instance

- For a *Customer*: Deterioration of service cannot be prevented. Problems MUST happen, to trigger a corrective action. In the worst cases, on-going conduct of business may suffer
- For a *Supplier*: Penalties (frequently specified in modern SLAs) cannot be avoided. In the worst cases, business continuation becomes problematic

### 4.2.2 Predictive Approach

This relies on predicting the result of service in advance, in order to assess the likelihood of an SLA violation. Therefore, it is frequently possible to take a corrective action before a problem actually occurs, thus eliminating it or, at least, minimizing its impact.

This approach avoids the problems associated with the reactive approach:

- For a *Customer*: Deterioration of service may be prevented and business conduct optimized
- For a *Supplier*: SLA Penalties may be avoided and prospects for the business continuation improved

## 4.3 Basic Technical Principle: Near-Real-Time, Asynchronous Operation

Any information acquisition and delivery process may be reduced to four basic tasks:

- Capture: acquiring raw data,
- Transformation: transforming data into information,
- Delivery: delivering information to a recipient,
- Consumption: use of information by a recipient.

A process incorporating these tasks is usually considered to be synchronous: component tasks follow each other in sequence. This is the principle of the so-called Real-Time approach: data is captured and delivered to users 'as is'. This represents a major problem in Systems Management, as it leads to either overloading of networks due to the high volume data movements (sub-second cycle) or poor quality of data due to long sampling cycles. That, however, need not be the case.

An alternative is the so-called Near-Real-Time approach: in this case, timing of information delivery is dictated by user needs. In other words, information is delivered in time to make a difference. For instance, if a user manages a typical IT environment, it is pointless to capture sub-second data on the performance of a server: network delays will ensure that such data is outdated anyway. Once-a-minute data delivery is often sufficient. However, data can be captured frequently, be buffered up, processed and made available for delivery only when needed. This keeps data movements low while preserving the quality of data. This is accomplished by keeping the three basic tasks asynchronous, or *time-independent*.

This principle is illustrated by Figure 6.

Information Acquisition and delivery process

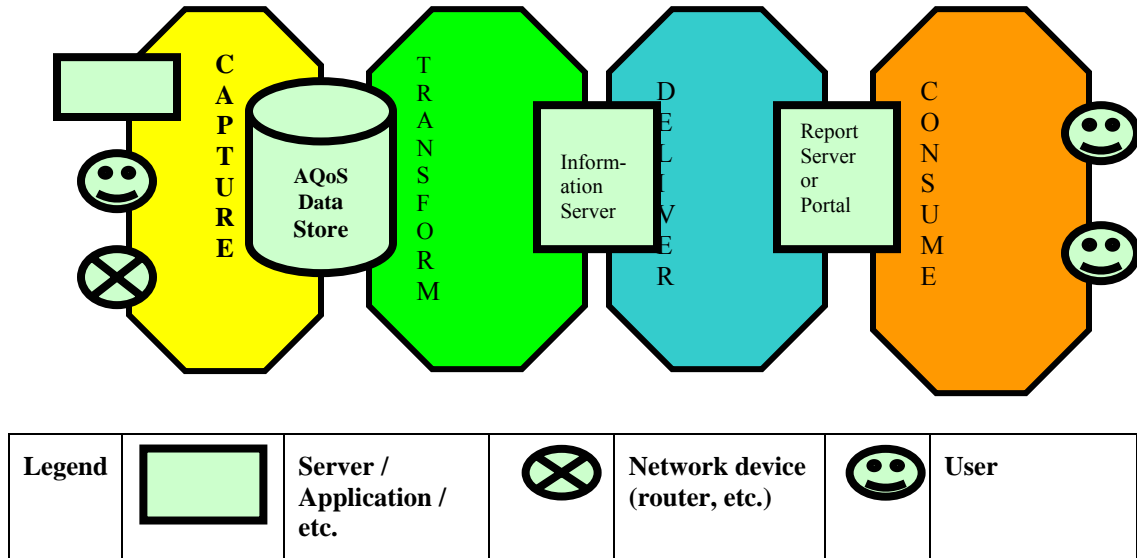


Figure 6. Four asynchronous loops

#### 4.4 Role of Automation

Obviously, automation of the process plays a critical role: without it, no meaningful management can be possible. However, it is useful to consider the extent of its applicability. In other words: is it sensible to attempt to automate the entire process? Let us analyze the situation.

AQoS Management Systems specifically target modern distributed Open environments. Two of the inherent characteristics of such environments (and major causes of system management chaos) are:

- Rapid rate of change
- Ever-increasing complexity

These are the very reason why AQoS is required and there is no doubt that, in order to be cost-effective, the AQoS management process has to be automated. However, to be effective, an automated management regime is fundamentally dependent upon the knowledge and data:

- Knowledge of basic principles or ‘laws’ of the managed entity
- Data or ‘measurement results’ reflecting the latest behavior of an entity

Availability of both knowledge and data are gravely affected by the rate of change and growing complexity – by the time the vast quantities of measurements are collected, analyzed and transformed into knowledge, the situation would have changed. Therefore, it is evident that AQoS management systems have to operate in an environment with incomplete information. This means that, at least for automation purposes, AQoS must heavily rely on heuristics. Modern technology does support heuristics, but not very well. Indeed, computers in general are not very good at it. However, we do have the best possible heuristic engine available to us – the human brain.

At this point it is useful to compare the inherent strengths and weaknesses of humans and computers:

- Computers are fast, accurate, reliable, precise – and quite dead. They are only able to operate as programmed – that is, given the current state-of-the-art, in a totally known environment. All

## AQoS Management Functional Requirements

uncertainties must be taken care of by programmers, on the basis of the pre-existing knowledge. At the moment and for the foreseeable future, this applies to knowledge-based systems, AI and all the other computerized approaches to decision-making. Most of the time, this translates into high efficiency with repeatable operations and ineptitude in unique situations.

- Humans are slow, not particularly accurate, unreliable and, most of the time, imprecise – yet alive. They are quite incapable of sifting through the data oceans that computers handle in their stride, but, given surprisingly small quantities of relevant data, can have surprising insights. They handle unique, complex decisions in their stride but flag and make mistakes when asked to perform the same operation repeatedly.

This entire discussion is well captured by a saying attributed to Pablo Picasso: “Computers are useless. They can only give you answers”.

This is hardly surprising: after all, humans created computers for an express purpose of supplementing human capabilities in their areas of weakness. This is exactly what should be done in this case:

- Computerized systems should capture, store, and process data. They should make the routine, repeatable (thus, foreseeable and programmable) decisions and identify the unique situations. For these, computerized systems should present the relevant, summarized information to humans.
- Humans should make heuristic decisions in unique situations on the basis of the pre-processed information

### **4.5 Hierarchy of Predictive Approaches**

As noted in the previous section, an automated approach to AQoS management depends upon knowledge of the principles of the managed entity and availability of measurements of the entity’s behavior. When this information is combined with well-defined SLA’s, predictive approaches become possible. In a sense, a predictive approach combines the best capabilities of humans and computers. That is, the tabulation and storage capabilities of computers are combined with our reasoning ability in order to gain knowledge about what future outcomes are possible with regard to achieving SLA’s.

There is a hierarchy of prediction that answers increasingly complex questions, which in turn can provide increasingly valuable information for managing applications and systems. The following list identifies several levels of prediction that could be implemented. The predictive levels are described in terms of the alerts that might be issued, their triggering conditions, and the information provided that would assist in application or systems management (see Figure 7).

## AQoS Management Functional Requirements

Predictive Level	Triggering Condition	Questions Answered
Threshold Alerts	Measured metric exceeds threshold value.	Has an alert threshold been breecched?
Sequential Alerts	Repeated exceeding of threshold value.	Is the alert condition persistent?
Correlated Alerts	Tandem threshold alerts for multiple resources.	How pervasive is the alert condition?
Pattern Alerts	Current trend fits initial segment of previous alert pattern.	Is a future alert likely?
SLA Predictive Alerts	Current trend indicates that achieving an SLA is in jeopardy.	Can an SLA target be met?
Advisory Alerts	(Applies to all alert levels)	What can be done to decrease the likelihood of future alerts?

Figure 7. Predictive Approach Hierarchy

## 5. AQoS Management Process

### 5.1 Overview

In real terms, AQoS Management equates to the conventional discipline of Capacity Management, but with an overriding emphasis on achieving Quality of Service objectives (please refer to Definitions in Section 1.2). Thus, it may be safely said that AQoS Management comprises the following, integrated disciplines:

- Capacity Planning
- Performance Management
- Service Level Reporting

With this in mind, we define the objectives of AQoS Management as follows:

- Continuous collection of Performance, Configuration and User data, and storage of that data remote to their systems, to support the data needs and automation of subsequent processes
- Short-term monitoring of system and application resources to effect:
  - Problem detection
  - Identification of potential problems
  - Prediction of problems
  - Initiation of problem alerts
- Regular Service Level Reporting, including the prediction of results for the next Service Level interval, in accordance with a Service Level Agreement (SLA)
- Prediction of potential problems in the medium and long-term
- On-demand Capacity Planning, sourcing pre-collected, fit-for-purpose data from remote storage
- On-demand Performance Tuning, sourcing pre-collected, fit-for-purpose data from remote storage
- On-demand Web-enabled Reporting

### 5.2 Process

The process of QoS Management is presented in Figure 8.

## AQoS Management Functional Requirements

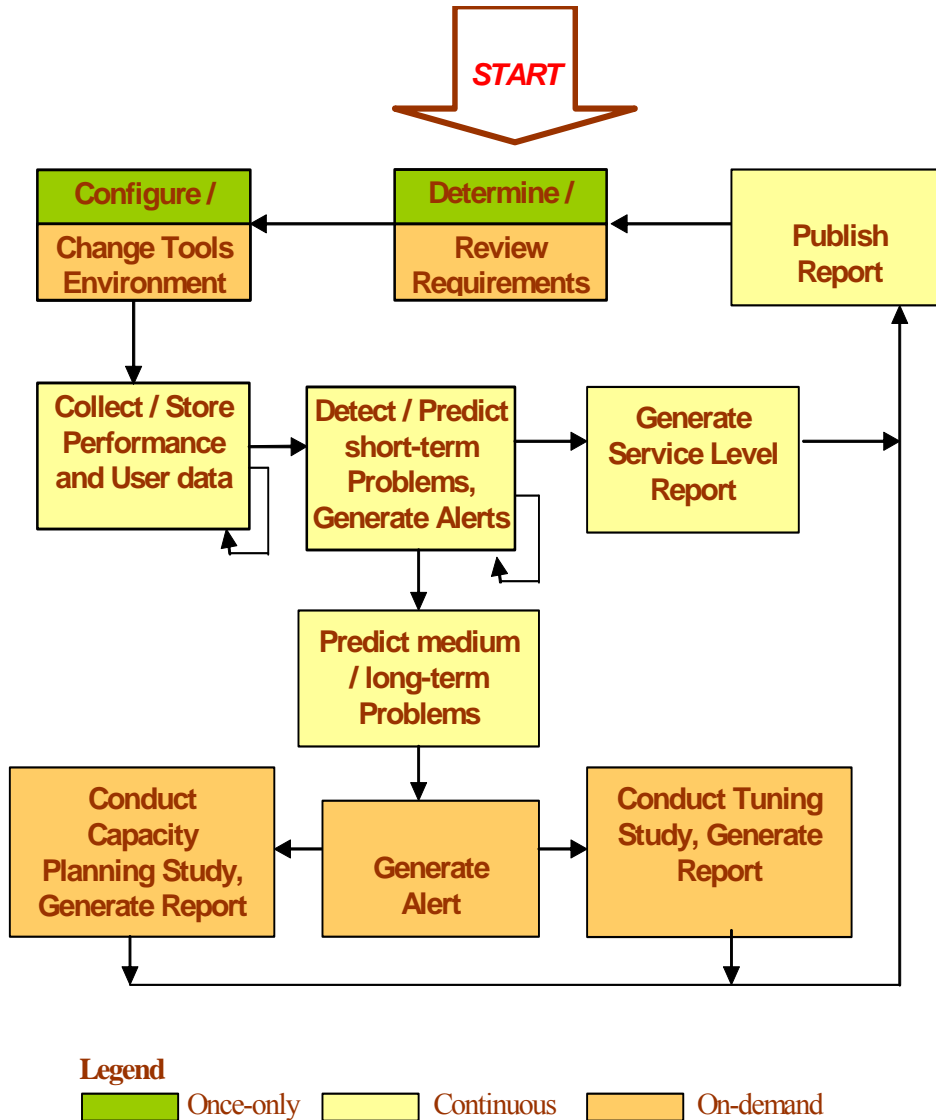


Figure 8. QoS Management process.

Separate stages are briefly described below.

❖ Once-only

- **Determine requirements.** This stage necessarily occurs before the Capacity Planning commences, then as required. Subjects to be addressed are:
  - Frequency of reporting. Regular, usually monthly for Service Level Reporting.
  - Reported items and metrics. Determined by contractual requirements and availability of data. Usually, the following metrics are used:
    - Availability (calculated as absence of reported problems)
    - Help Desk statistics (incidents by class, resolution times, etc.)
    - Response time
    - Resource utilization (CPU, disk, memory, communications lines, etc.)
    - User data (usage projections, major changes, etc.)
  - Format of reports. For preference Web-based, or at least electronically delivered on a CD-ROM.

## AQoS Management Functional Requirements

- Distribution list
- **Create / modify tools environment.** The entire process relies heavily on automation, therefore on the usage of tools. The tools environment must be always kept up-to-date. For instance, every new system placed under management must have the necessary tools installed, as a condition for the service delivery. Equally, certain events (operating system upgrades, etc.) may result in a tools upgrade requirement.
- ❖ Continuous
  - **Collect / store data.** The objective of this process is to collect the necessary data for Service Level Reporting and other related tasks (e.g. Capacity Planning). Varies from fully automated for all (or, at least most) platforms to manual for user forecast data. Storage of all data is in a set of automated Performance Databases.
  - **Predict results of the next Service Level Report.** This is an automated process which forecasts the probability of Service Levels being met for the next reporting cycle.
  - **Predict future problem.** This process compares the predicted Service Levels with required ones, in order to forecast the service level problems before they occur. This is an automated process.
- ❖ Cyclical (monthly)
  - **Generate Service Level Report.** The process consists of two components:
    - Fully automated: generate graphs for all the necessary metrics and insert into the report template (Word or Web-based).
    - Manual: write the report
- ❖ Ad-hoc and related processes
  - **Generate Alert.** This occurs when a problem was forecast. It is an automated process
  - **Generate Capacity Planning Report.** This is a manual process, but it relies on the availability of all the necessary data from the performance databases and use of advanced tools. Thus, manpower requirements are dramatically reduced.
  - **Produce and Publish Reports.** This is an automated process, consisting of:
    - Web publishing, or
    - Automated delivery of a Word document (e.g. E-mail) or ‘burning’ of a CD-ROM.

### 5.3 Capabilities of the system

The capabilities required to support this complex, integrated process are listed below.

- Pervasive measurement and data capture.
  - We must capture all the relevant data for:
    - Servers
    - Storage
    - Network devices
    - Applications
    - QoS
      - ❖ Help Desk
      - ❖ End-To-End-Response
      - ❖ Etc.
    - Etc.
  - We must do so at relatively high resolution and with maximum accuracy (to support the Near-Real-Time approach), yet without disrupting operations or imposing undue load upon resources. In other words, our collectors must be efficient.
  - We must keep in mind that most operating systems, network entities and applications these days are instrumented. In order to minimize the overhead of implementing our system, we must be able to utilize such existing instrumentation where it exists, instead of replacing it. This means that we must provide ‘agent-less’ data capture, using MIBs, native system capabilities and similar approaches.
  - We must be able to relate the disparate items of data to each other, to derive end-to-end characteristics of processes.

## AQoS Management Functional Requirements

- Data Management capability. For large, distributed systems, volumes of data that are generated by this process may be quite large. Yet, to support historical analysis, they must be available on-line, on demand. Therefore, we must be able to manage that data: compress, consolidate, archive, back up, restore, etc.
- Maximum flexibility. Organizations mature enough and large enough to be interested in QoS are likely to have complex environments, with many suppliers and/or customers, and a variety of software and hardware platforms, applications and SLAs. In order to cater to all these requirements, utmost flexibility is necessary.
- Full scalability, no bottlenecks. Systems that our customers are likely to have are frequently very large. Such systems grow and evolve over extended periods of time and may comprise many thousands of nodes. Therefore, our software must be able to scale almost without limit in all respects, no internal bottlenecks are allowed.
- Reliability and recoverability. IT systems are business-critical already and will become even more so in time. It follows, therefore, that our software must be reliable and recoverable, to support this mode of operation.
- Maximum automation. Systems on that scale demand automation. We must provide automation at all stages (data capture, management, reporting, problem detection, etc.). Indeed, all the repetitive aspects of operation for our system must be automated.
- Utilization of the best heuristic engine available: the human brain. We must handle routine problems, but leave important decision-making to humans. For instance our system should handle threshold monitoring and alerting automatically, but should not take automatic action to resolve problems that cause thresholds to be exceeded.
- Open solution. Our solution should be fully open. Whenever possible, we should support standards (both actual and de-facto). We should publish all the interfaces.
- Best practice. We should implement solutions that either are or widely considered to be 'best practice'.

### **5.4 Implementation considerations**

Please refer to the Appendix 3. AQoS Implementation Considerations for this information.

## Appendix 1. Descriptions of Work Groups

### A1.1 General

Substantial work is involved in identifying and defining the relevant enhancements to existing standards and developing the new standards. Therefore, it was decided to split the work into multiple logical units and create appropriate Work Groups to address each one.

The diagram of the relationship between these Work Groups (WGs) is presented below in Figure 9.

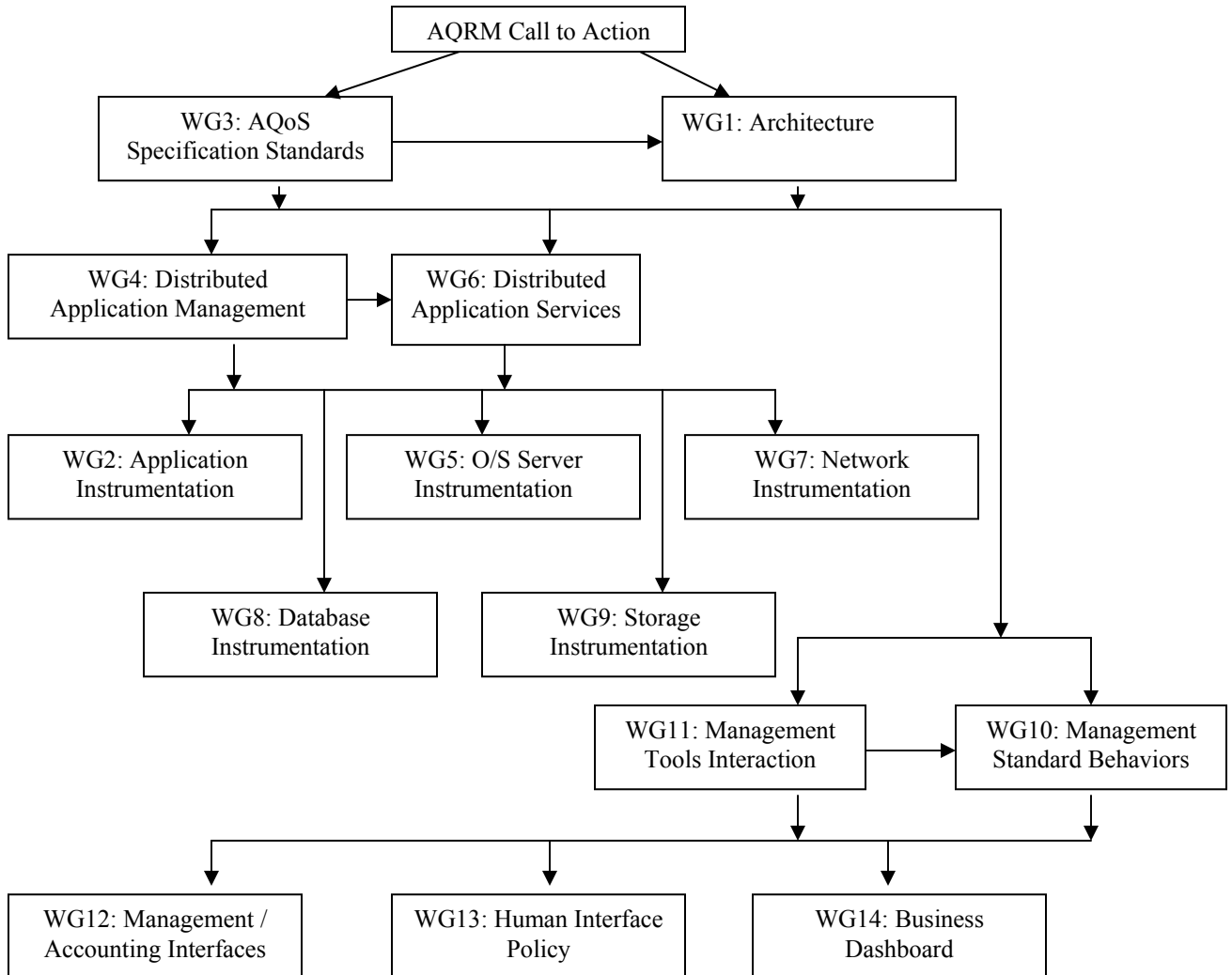


Figure 9. Structure of Work Groups

### A1.2 Work Group Definitions

#### WG1. Architecture

With reference to:

## AQoS Management Functional Requirements

- Functional Requirements document produced by WG3

Produce:

- Comprehensive architectural framework for AQRM

### **WG2. Application Instrumentation**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Distributed Application Services Requirements defined by WG6

Produce:

- Definition of standard requirements for application instrumentation
- List of existing standards to be modified
- List of required new standards

### **WG3 AQoS Specification Standards**

With reference to:

- AQRM Call to Action

Produce:

- Functional Requirements document, containing:
  - Objectives of AQRM 'Forum'
  - Basic definitions
  - Functionality requirements
  - Implementation consider

### **WG4. Distributed Application Management**

Produce:

- AQoS Functional Description produced by WG3 and
- Architecture produced by WG1

Produce:

- Requirements for the management of distributed applications

### **WG5. OS/Server Instrumentation**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Distributed Application Services Requirements defined by WG6

Produce:

- Definition of standard requirements for O/S server instrumentation
- List of existing standards to be modified
- List of required new standards

### **WG6. Distributed Application Services**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Architecture produced by WG1

Produce:

- Standard requirements for distributed application services

### **WG7. Network Instrumentation**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Distributed Application Services Requirements defined by WG6

Produce:

- Definition of standard requirements for Network instrumentation
- List of existing standards to be modified
- List of required new standards

### **WG8. Database Instrumentation**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Distributed Application Services Requirements defined by WG6

Produce:

## AQoS Management Functional Requirements

- Definition of standard requirements for Database instrumentation
- List of existing standards to be modified
- List of required new standards

### **WG9. Storage Instrumentation**

With reference to:

- Distributed Application Management Requirements defined by WG4 and
- Distributed Application Services Requirements defined by WG6

Produce:

- Definition of standard requirements for Storage Sub-system instrumentation
- List of existing standards to be modified
- List of required new standards

### **WG10. Management Standard behaviors**

With reference to:

- AQoS Functional Requirements produced by WG3 and

Produce:

- AQRM process description
- Structured list of roles involved in AQoS Management process
- For each role:
  - Label (name)
  - Function
  - Responsibilities
  - Requirements
    - Information
      - ❖ Notification (alerting)
      - ❖ Standard (periodical) reporting
      - ❖ Queries and/or data investigation
    - Action support

### **WG11. Management Tool Interaction processes/protocols**

With reference to:

- Output of the WG10

Produce:

- Combined AQRM and Management Tools data / control flow diagram
- Structured list of interaction points between Management and AQRM tools
- For each point:
  - Label (name)
  - Required for: reference to a generic sub-process or role (refer to WG10)
  - Functionality
  - Input / output requirements
  - Interaction with other tools
- List of existing standards to be modified
- List of required new standards

### **WG12. Management/Accounting Interfaces with business systems**

With reference to:

- Output of the WG10

Produce:

- Combined AQRM and Management / Accounting tools data / control flow diagram
- Structured list of interaction points between AQRM and Management / Accounting tools
- For each point:
  - Label (name)
  - Required for: reference to a generic sub-process or role (refer to WG10)
  - Information requirements
  - Data requirements (data details and origins as per generic tools defined by WG11)
- List of existing standards to be modified

## AQoS Management Functional Requirements

- List of required new standards

### **WG13. Human Interface Policy template standards**

With reference to:

- Output of the WG10

Produce:

- Structured list of SLAs
- For each SLA (all or some of the items below)
  - Preceding and succeeding SLA
  - Type (static / dynamic / auction / other?)
  - SLA Objective
  - AQoS parameters (e.g. metrics and targets)
  - Duration and service period
  - Rewards and Penalties
  - Administration
    - Participants
    - Responsibilities
    - Disputes and resolution
    - Reporting
- List of existing standards to be modified
- List of required new standards

### **WG14. Human Interface Business Dashboard standards**

With reference to:

- Output of the WG10

Produce:

- List of AQRM dashboards
- For each dashboard:
  - Label (name)
  - Required for: reference to a generic sub-processes and / or roles (refer to WG10)
  - Function
  - Front-end for: list of generic tools defined by WG11
  - Mock-up (if possible)
- List of existing standards to be modified
- List of required new standards

## Appendix 2. Definitions.

- **Availability:** a metric that shows whether an entity operates normally (available for business) or not, during a given interval. There are two main types:
  - **Measured:** an entity is available if there is evidence of the normal completion of standard activities. For instance, an application is available if transactions (user or robotic) complete normally
  - **Derived:** an entity is available if there is no evidence to the contrary. For example, an application is available if there are no user complaints that it is not available
- **Customer or Client:** Service Recipient. Similar to Suppliers, these may be internal or external
- **End-To-End:** description of a metric/process/entity that characterizes a complete route or a path. For instance, End-To-End Response Time (ETE RT) describes response time for a transaction across its entire path.
- **Measurement**
  - **Real-Time Management:** conducted at least at the time resolution of a process being measured / managed or better. For instance, both monthly and sub-second measurements are real-time, so long as they are appropriate for a given process. Good examples are management of Static SLAs and mainframe performance management, respectively.
  - **Near-Real-Time Management:** conducted on the time-scale of a process being measured, but at a coarser resolution than the process itself. One way to describe it is: in sufficient time to detect the desired phenomenon or affect the process as required
  - **Non-Real-Time Management:** conducted not on the time-scale of a process being measured, but rather at a significantly coarser resolution than a process itself
  - **Stochastic:** objective is to be able to determine characteristics of a measured process (its distribution or its parameters, e.g. average response time, and/or standard deviation / variance) without sampling every instance of that process. Stochastic measures may be derived by:
    - Sampling or statistically probing the process being measured
    - Performing statistical operations over the sample population available a-priori
 In either case, care must be taken to ensure that a representative sample of data (that adequately represent the process as a whole, rather than the sample itself) is obtained.
  - **Deterministic:** objective is to measure every instance of a process (e.g. every individual transaction, IP packet, etc.)
- **Quality of Experience or QoE:** any metric that shows whether the immediate experience of Customer with regard to Service, either documented in a Service Level Agreement (SLA) or implied (but measurable), was met over a Service Period
- **Quality of Service or QoS:** any metric that shows whether the requirements of a Service Level Agreement (SLA) or implied (but measurable) customer expectations were met over a Service Period.
- **Real-Time Enterprise** (or E-Enterprise, or...). An enterprise that has time constraints for the delivery of its services or products.
- **Service:** measurable result or outcome of a business or technical process involving a Supplier / Server and Customer / Client
- **Service Level Agreement or SLA:** a formal agreement (contract) between a Supplier and Customer, formalizing the details of a service (contents, price, delivery process, acceptance and quality criteria, penalties and so on). Informal SLAs exist also and may be as important and binding as the formal ones. However, informal SLAs cause misunderstandings, misinterpretation and disputes to a greater extent than the formal ones and are not recommended. There are three major types of SLA's:
  - **Static:** an SLA that generally remains unchanged for multiple Service Periods. Such SLAs are common in Outsourcing
  - **Dynamic:** an SLA that generally changes from Service Period to Service Period, to accommodate changes in provision of Service. Such SLAs are increasingly common in Telecommunications
- **Service Period:** a period for assessment of the Quality of Service. For a business process that is subject to an SLA, it is typically a calendar month. For other processes, it may be a transaction or any other measurable and relevant period of time.

## AQoS Management Functional Requirements

- **Supplier** or **Server**: Service Provider. These may be internal (e. g. IT Department) or external (Outsourcer)

## Appendix 3. AQoS Implementation Considerations

### A3.1 General

As was mentioned before in Section 4.3, this process is readily implementable as a sequence of asynchronous loops. The Implementation considerations for these loops are presented below.

### A3.2 Data Capture

#### A3.2.1 General

The following aspects are addressed:

- Standards that are likely to apply
- Items of information
- Storage and management of captured data

#### A3.2.2 Data capture and standards

In general, it is always preferable to use standard mechanisms. Thus, ARM [References, 2], CIM [References, 3] and SNMP / MIBs [References, 4] are likely to form the basis of data capture. However, it must be noted that, given the rate of change in the IT area, all business-critical systems within an enterprise cannot be expected to support these (or, indeed, any other) standards. Therefore, data capture mechanisms must be able to cope with and / or without CIM and similar standards.

#### A3.2.3 Data items

Five classes of information items are commonly used for QoS purposes:

- Help Desk / CRM
- Availability
- Response Time
- Resource Usage
- Other

##### A3.2.3.1 Help Desk / CRM

CIM will be one of the main sources of this information, provided that relevant Help Desk / CRM systems support that interface. Alternatively, direct access to CRM Databases is feasible. At least the following information should be captured:

- Timestamps
  - Call received
  - Call answered
  - Call transferred or altered. Multiple transfers are possible.
  - Call logged
  - Call-related activity. This refers to significant action relevant to a call, such as investigation, contact with a customer, etc. Multiple activities of this nature are likely over a lifetime of a call
  - Call priority altered. Multiple alterations are likely over a lifetime of a call.
  - Call closed
  - Call closure reported to a customer
  - Call closure authorized by a customer
- Priority. Multiple alterations of priority are possible over a lifetime of a call
- References (to other calls or any identifiable entity)
- Names (of customer, operator, etc.)
- Status. Multiple changes in status are to be expected

## AQoS Management Functional Requirements

- Free-format description. Multiple descriptions should be expected, routinely one per activity, status change, etc.

### A3.2.3.2 Availability

ARM and CIM are expected to be widely, but not exclusively used for this purpose. Two main types of availability can be identified:

- Measured availability.
  - Application: it is available during an interval if results of transaction-based measurements indicate normal completion. Such results may be based on:
    - ARM
    - Transaction logs (e.g. Web)
    - Measurement of transaction queues (e.g. CICS or MQSeries)
    - Robotic tools
- Derived availability
  - CRM-based. An entity (application, system, device, etc.) is available during an interval if there is no CRM-based information that it is not
- Uptime
  - Network node is up during an interval if a ping utility (or similar) responds normally
  - Operating system is up during an interval if OS-based measurements (e.g. CPU usage) are present
  - Device is up during an interval if device-based metrics (e.g. storage) are present
  - Application is up during an interval if a process / task representing this application is present
  - Etc.
- End-To-End Availability (for a path). It may be measured or derived. One way to derive it is to assume that a path is available if all the nodes that it consists of are available or at least up.

### A3.2.3.3 Response Time

ARM is expected to be widely, but not exclusively used for this purpose. The following types should be identified:

- Transaction-based. This type of response time is associated with client-server systems. It is usually applied to applications with multiple logical units of work performed during a session. Most frequently, these are the so-called on-line systems.
- Turn-around. This type is associated with so-called 'batch' systems (usually, applications with one or few logical units of work during a session). This also applies to packet-based telecommunications systems.
- End-To-End. This type of response time is usually associated with transaction-based client-server systems and describes a response time of a single transaction for its entire cycle, from client to server(s) and back. It may be measured or derived as a sum of Component response times.
- Component. This applies to a response time for a single component or segment of an End-To-End Response Time (for instance, network response time).

### A3.2.3.4 Resource Usage

CIM is expected to play a significant part in capturing this information. However, until that is widely adopted (and even thereafter, for 'legacy' systems) other approaches have to be used as well. Two types may be identified:

- Interval. This refers to resource usage over a set interval, for instance average daily CPU usage, number of reads or writes per hour, etc.
- Event. This refers to resource usage tied to an event, for instance CPU consumption by a process over its entire life.

Below are the examples of relevant resources:

- Physical. Data for such resources is measured.
  - Hardware (CPU, memory, disk, router, etc.)
  - Software (Operating System, Transaction Processor, Application, Network, SAN, etc.)
- Logical. Data for such resources is derived.
  - Application Group
  - Communications Network and/or Network Path

## AQoS Management Functional Requirements

- Storage Area Network (SAN)

### A3.2.3.5 Other data

The following data is required:

- SLA composition
- Configuration and assets
- Corporate forward projections (for Capacity Planning purposes)

Some of these data are measured, but most are derived.

## A3.3 Data Storage

Captured data must be stored. In order to minimize the network loads and maximize the data availability, the following storage model is suggested:

Two types of data are defined, according to their intended usage:

- Trouble-shooting data. This is detailed data, intended to predict, diagnose and resolve immediate problems. Their characteristics are:
  - High level of detail
  - High volume
  - Short life span
  - High localization (relevance to a specific location, entity, etc.)Storage requirements are explained below, using a server as an example. Similar rules apply to network devices and other resources:
  - Outside of the node (server, network sub-segment) that they describe, to maximize the availability at a time of a crash
  - On the network segment to which the relevant server or sub-segment belongs, to minimize data traffic and maximize availability
  - At the maximum resolution, i.e. that of capture
  - For a limited time, as dictated by trouble-shooting requirements
- Planning and management data. These are summarized and abstracted data, used for management and planning purposes. There may be multiple levels of such data, depending upon the degree of summarization. Their common characteristics are changed with increases in the level of summarization and abstraction
  - Relatively low level of detail
  - Relatively low volume
  - Relatively long life span
  - Relatively low localizationStorage requirements for such data are presented below. They also change with increases in the level of summarization and abstraction
  - In an increasingly centralized location / locations
  - At a consistently decreasing resolution
  - For an increasingly long time
  - With an increasing level of recoverability

The following functionality must be available as part of data storage (without loss of precision of the data, even though the loss of resolution cannot be avoided)

- Data consolidation (in time)
- Data aggregation (against arbitrary criteria, to achieve derived data)
- Data archival, deletion and reclamation of storage space
- Data recovery
- Metadata capability

### A3.4 Data Transformation

#### A3.4.1 General

The purpose of data transformation is to transform data into information – that is, to calculate and assess an appropriate summary metric. It is the fundamental part of the functionality of the AQoS system, particularly within the context of the predictive management of the Quality of Service: it is this process that is responsible for identifying and predicting outages. Therefore, the data transformation facility must have the following characteristics:

- Pervasive availability
- Flexibility
- Accuracy
- Low overheads
- Ability to produce repeatable results

Other Work Groups will create detailed definitions of these facilities. However, it is useful to present a discussion on the subject of summary metrics at this point.

#### A3.4.2 Overview

Potential summary metrics can be divided into three classes:

- Base metrics are simply a subset of the raw or detailed data. They are the important metrics that provide a broad view of system or application status, such as CPU busy, Available memory, etc.
- Compound metrics are quantitative or qualitative values derived from the analysis of a particular set of base metrics. For example, 3 or 4 key metrics could be selected and threshold values or rules and formulae could be developed to detect abnormal or anomalous conditions based on the values of the metric set. The rules might be qualitative or quantitative.
- In-context (for instance, Time-context) metrics are derived from base metrics and represent the current base metrics value in the context of its behavior against the behavior of another metric (e.g. over a specific time interval). Trends or “deltas” are examples of time-context metrics.

#### A3.4.3 Advantages and Disadvantages of the Classes

All base metrics are constrained within a theoretic range. For metrics like CPU busy, that theoretic range is also a real range, in that we can reasonably expect to see metrics values right across the range, and know from experience what sorts of values are likely to indicate a system with problems and what values indicate a system without problems. However, the theoretic range of something like “page faults” is far more difficult to establish (it **does** exist, but is dependent on many variables), and even if established, does not provide a useful practical range.

Even if an operating range for a given metric can be established, it is not always clear what constitutes “good” and “bad” within that range. For example, CPU busy has an operating range from 0 to 100%, but in some systems a value of 80% may indicate performance problems whereas in others 80% may be perfectly normal. This leads to the issue of hardware and software configurations, which can, of course, differ widely. The interpretation of performance metrics collected from an 8-way processor with 2Gb RAM, a RAID disk system running Solaris 2.9 and a large Oracle database and acting as a web server will be significantly different than the interpretation of metrics collected from a uniprocessor with 512Mb RAM, an IDE disk system running Windows and operating as a mail server. Because of this extraordinary range of hardware and software configurations to be found, deriving any compound metrics which have absolute meaning – i.e. do not require interpretation in an historical or technical context - is extremely difficult.

The fundamental problem with both base and compound metrics is that they lack meaningful context or a frame of reference. That is to say, a given metric value has little meaning on its own, and needs to be compared with other metric values before any judgments can be made as to whether it indicates a good or

## AQoS Management Functional Requirements

bad situation. Comparison with other metrics values can take two forms, either comparison with the values of other metrics for the same time interval, or comparison with other values of the same metric for different intervals. Time-context metrics represent an implementation of the latter approach.

Any overall “system health index”, or indeed any compound metric, is only really useful when it is seen in the light of historical data. No one base or raw metric can be used as an absolute good/bad indicator; neither can any simple combination of base metrics. Too much depends upon machine size and configuration, and application mix, and, of course, the type of metric.

### A3.4.4 Examples of metrics by class

#### A3.4.4.1 Base Metrics

The following summary metrics represent a good overall view of any system.

- Processor busy (average of all processors)
- Throughput (Kbytes per sec) & Average IO Response (averaged across all disks on a system)
- Throughput & Average Response on top  $n$  disks
- Processor usage, virtual bytes/virtual size, working set/resident set size, for top  $m$  processes
- Free memory, page faults & pages paged in. (These 3 metrics are available on Windows and Unix and are useful in establishing the efficiency of memory management.)
- Network packets in/out

#### A3.4.4.2 Compound Metrics

Despite the difficulties inherent in compound metrics, it can be useful to consider some simple indicators of system operations, such as the ratios of some base metrics. These are illustrated below in Figure 10.

	Proc busy	Throughput	Free mem
Proc busy			
Throughput	●		
Free mem			
Page faults	●		●
Pages in		●	●
Network packets	●		

Figure 10. Compound metrics

#### A3.4.4.3 In-context (Time-context) Metrics

Time-context metrics provide a frame of reference for selected base metrics in the context of their own “recent” history. To do this it is necessary to calculate three new time-context metrics for each base metric – one average figure and two measures of deviation.

The average, known as the Recent Historical Average or RHA is a geometric mean of the historical metrics values – i.e. it is a long term weighted average which gives more weight to recent values and less weight to older values. Thus the RHA “considers” all historical data for a given metric, but tends to reflect recent values more strongly than less recent ones.

The two measures of deviation are themselves geometric means of the difference between the current metrics value and the RHA calculated up to the previous metric value. One measure of deviation is the “below-mean deviation” or BMD, and the other is the “above mean deviation” or AMD. The BMD is a measure of the average degree of deviation from the RHA when the current metric value is less than the RHA; the AMD is a measure of the average degree of deviation from the RHA when the current metric value is more than the RHA.

Thus, the current metric value can be compared to “recent history”, either to the RHA directly, or to a range which represents normal operation in the recent past – i.e. [RHA-BMD, RHA+AMD]. For example, assume the current metric value is  $V_n$  where  $n$  indicates the current interval,  $RHA_{n-1}$  is the RHA calculated for the previous interval; similarly,  $BMD_{n-1}$  and  $AMD_{n-1}$  are the deviations calculated for the previous interval.

## AQoS Management Functional Requirements

$V_n$  can be compared directly to  $RHA_{n-1}$  or to the range [ $RHA_{n-1} - BMD_{n-1}$ ,  $RHA_{n-1} + AMD_{n-1}$ ]

The values are to be used for the interval  $n+1$  are calculated as follows:

$$RHA_n = \alpha * V_n + (1 - \alpha) * RHA_{n-1}$$

$$BMD_n = \alpha * (RHA_{n-1} - V_n) + (1 - \alpha) * BMD_{n-1} \dots\dots \text{if } V_n \leq RHA_{n-1}$$

$$AMD_n = \alpha * (V_n - RHA_{n-1}) + (1 - \alpha) * AMD_{n-1} \dots\dots \text{if } V_n \geq RHA_{n-1}$$

The closer the value of  $\alpha$  is to 1, the more heavily the recent values are weighted. So if  $\alpha = 0.8$  then the effect of historical values is quickly dissipated, whereas if  $\alpha = 0.2$  historical values have a more pronounced effect on the result. Regardless of the value of  $\alpha$ , more recent metric values always carry a higher weight than older ones.

### A3.5 Information Delivery

#### A3.5.1 General

Three aspects of information delivery must be considered:

- Trigger
- Context
- Method

Full details of this process will be determined by other Work Groups, but a short discussion is in order.

#### A3.5.2 Trigger

Trigger refers to the reason for information delivery. Three major triggers may be defined:

- Timing. This refers to periodic delivery of information, e.g. regular reporting
- Push. This refers to unsolicited delivery of information, e.g. alerts
- Pull. This refers to on-demand delivery of information, e.g. in response to a query

#### A3.5.3 Context

Information may only be understood 'in context'. 'Out of context' information, regardless of its innate quality, is misleading – a good example of that is news reporting. In the context of AQoS and bearing in mind the requirement for maximum automation, information must always be delivered 'in context'. Thus, a task of information assembly – reporting – becomes fundamental to information delivery.

Two classes of information may be thus defined:

Primary. This is the information, which triggered the delivery process in the first place – for instance, the level of CPU Busy that triggered an alert. This information is always delivered

Secondary. This is the information that is necessary for comprehensive understanding of the primary information – for instance, a time series of CPU Busy values. Such information may or may not be delivered: for instance, secondary information is not necessary in response to an explicit query.

#### A3.5.4 Method

This will be examined by other Work Groups.

### ***A3.6 Consumption of Information***

Consumption of information refers to the ultimate objective: decision-making. Such decision-making may be:

- Automated (computerized)
- Human

A number of other Work Groups will be examining this area in detail.

## Appendix 4. References

1. AQRM Call To Action  
<http://www.opengroup.org/aqrm/calltoaction.pdf>
2. ARM: Application Response Measurement  
<http://www.opengroup.org/tech/management/arm>
3. CIM: Common Information Model  
[http://www.dmtf.org/standards/standard\\_cim.php](http://www.dmtf.org/standards/standard_cim.php)
4. SNMP: Simple Network Management Protocol  
<http://www.snmpLink.org/>
5. A Framework for Enterprise Application Performance  
Peter J. Sevcik, BCR Volume 33, Number 11, November 2003
6. Service Level Agreement – A White Paper  
Jon Saperia, (Primary Author), Quality of Service Task Force, The Open Group, 2002