Descriptor based QoS Profile in XML

Polly M.S. Poon[1], Tharam S. Dillon[2], Fellow, IEEE, Ling Feng[3], Member, IEEE and Elizabeth Chang[4], Member, IEEE

[1,2] Faculty of Information Technology, Univerisity of Technology Sydney, Sydney, Australia, e-mail : (polly, tharam)@it.uts.edu.au
[3] Database Group, Department of Computer Science, Univerisity of Twente, Enschede, The Netherlands, e-mail:ling@cs.utwente.nl
[4] School of Information Systems, Curtin Univerisity of Technology, Perth, Australia, e-mail:change@cbs.curtin.edu.au

Abstract—**Many manufacturing processes involve remote control and command process to manage the industrial processes that situated in a disperse locations. For systems which operate in a hectic and distributed environment often involve frequent exchange of information from heterogeneous sources, particularly for those operates on the IP network, it requires the notion of resource indication – Quality of Service (QoS). Many has proposed the use of XML in such context, however many of the applied approach remains in an ad-hoc fashion and automate integration is not enabled. To provide a better interoperability between distributed industrial informatics systems, we introduce a descriptor based XML definition to feature resources description in distribute control systems with the notion of Quality of Service. We will discuss our XML methodologies used in developing the quality of service for distribute system, and demonstrate an example of telemetry system feature with QoS classes.**

Index Terms—**Quality of Service, XML, UML, model transformation.**

I. Introduction

Today, a lot of control systems used embedded commercial off the shelf software (COTS) to reduce cost of development. These systems employ different distributed technologies or different system architectures. For systems that operate in a complex, high data rate, distributed environments usually require frequent exchange of information from heterogeneous sources is an important requirement from services such as machine plant, sensor networks, etc. Many of these share the common feature of requiring constant collection of data from the external environment. Example include dealing room software for share trading, network management software or industrial process control systems. To build systems that operate in such contexts, it is important to precisely specify a consistent approach to delivering the intended QoS properties. The foundation of such a framework is reliant upon the construction of a set of consistent models. This is particularly important for mission critical software, as the consequence could be catastrophic if it is not tolerant to faults and errors heading to graceful degradation.

With the advent of enterprise software technologies and evolving software architectures, a number of emerging distributed real-time applications have been utilizing the Internet in the recent years. Many of IP network based industrial informatics applications, for example, e-Commerce systems, mission critical systems or personal communication software, require stringent timeliness on the arrival of data. Although IETF has been investigating the provision of QoS on the IP network [1], currently the Internet provides only best-effort service which has no indication of the transmission performance. For application which operates on the IP network which requires a guaranteed service level, they could suffer from unpredictable delays, jitters and bounding of bandwidth. To provide an end-to-end QoS over the IP network, one requires a Service Level Agreement (SLA) between the each independent service provider and user which can provide QoS measures at application level.

We are particularly concerned with a number of important features that characterize information dissemination in distributed systems and these include:

1. Collection and transmission of data from sensors to data concentrators
2. Storage of data at several levels and different databases that must interoperate with each other. Some systems, for example advanced systems which run in power plants [2], one requires integration of real time data, or historic data to make intelligent analysis and allow the system to run predictably. This could require the integration between archive data and operating real time data.

To fulfil these it requires a consistent specification of the data definitions to ensure data integrity and enhance the interoperability among heterogeneous data sources. This need consideration of the following:

a) A format that is widely accepted, platform independent which facilitate data exchange among different data sources
b) A suitable mechanism for storing 'annotated' data that has clearly defined semantics.

The focus of this paper is to discuss the methodology; we have developed for efficient data transformation using Object Oriented models to derive the set of XML Schema enabling data instance generation. The content of this paper is a constituent of our work on the Real Time Markup Language [3, 4] (RTML) which describes the semantics of operations in the context of real time systems. Part II of this we discuss the motivating reasons why we agree XML is the ideal medium for representing data for distributed industrial informatics systems, in Part III we discuss the existing related literature; in Part IV we introduce the QoS Framework we have adopted; in Part V we will discuss the use of a Descriptor based approach in organizing the XML instances and in the last section, we will give an example of using our proposed method by applying it to data exchange in Telemetry systems.

II. XML AND SEMANTICS

1. The Capability of UML in representing classes and relationships between classes

UML [5] is one of the most popular business process and organizational structure [6] modelling approaches with the advantage that it associates its modelling components to real world objects and processes. UML has been proven to be useful [7] in modelling large-scale software systems. However, many studies have pointed out the lack of formality in UML models. For instance in an object model, the representation of a relationship between classes is shown as a link attached with an AssociationEnd that has attributes including multiplicity(E.g. 1 to many or 0 to 1), navigation (unidirectional, or bidirectional) and/or aggregation (share aggregation or composition). These are expressed with annotations to describe their semantics. Each end of a link can be attached with a name in natural language to describe its meaning to the Classes that the relationship is connecting to. Though UML is powerful in visualizing model design, there is room for different interpretations in natural language. Unlike other modelling language, such as Specification and Description Language (SDL) or Estelle, which can be read into a hardware-software co-design tool, UML class diagrams lacks formal representation of semantics and is imprecise in the definition of constraints.

Although these issues have been taken into account in the extension in UML 2.0[8] in formalizing the Meta-Object Facilitys (MOF) for object model generation, but the rules defined in MOF is not strongly enforced. Transforming the object model into a formal representation has proven useful in a number of different contexts [9, 10]. As discussed in the previous section, XML is self-documenting in that it allows customized structure and tag names as long as it is validated against the strictly defined syntax (E.g. XML Schema). Along with the XPath capability, an attribute defined in XML Schema can be addressed across its current document or other namespace by quoting the location path. There are different approaches in modelling relationships in XML Schema using the XPath expression. This will be further discussed in a later section of this paper.

2. Direct transformation from conceptual model to XML

As mentioned previously many have adopted XML particularly for its advantage in providing. Many XML vocabularies have been created separately from the design without a systematic approach [11]. With a proper data model transformation method, the OO conceptual model can be directly translated into an XML Schema. Software designers can create the XML Schema directly and automatically given a set of class models, which will contain the exact semantics captured by the models. Although there are previous works on using an XML schema profile [12] in UML modelling, this approach would limit the verbosity of models with its stereotype definition and hence could limit the abstraction of models. From the point of view of application development, having a XML Schema automatically generated from a class model, allows developer to obtain the XML Schema directly and bind it to the implementation, e.g. Java Architecture for XML Binding (JAXB), XMLBeans. The seamless transformation has several advantages in different phases in a software development process.

3. Representing QoS in XML

A wealth of research from industry, business and academia seeks to provide quantitative measures for characterizing the performance of inter-domain applications that operate on IP network. Some solutions suggest the use of middleware which hardwires interfaces across multiple domains to provide a bridge between designated locations. This dedicated connection between selected domains can exchange information with a predefined specification. This solution would be good as long as communication is limited to systems that have similar (HW/SW) configurations and the number of tasks involved is low. For systems, which require automated integration or communication that occur between multiple systems with distinct and different architectural settings or hardware configurations, XML will be a better solution for a variety of reasons [13] including the cost of maintenance for ensuring the consistent agreement of protocols or interface structures. With the advent of XML and its related technology, communication between isolated domains, querying and disseminating of data using Web Services can be automated.

III. RELATED WORK

There has been research investigating the issue of improving the estimation of resource arrival on an IP network through the introduction of QoS. Some of this work use XML to represent QoS for particular applications. The QoS constraint is designed into an XML Schema.

1. Quality Modelling Language (QML)

Quality Modelling Language [14] is developed by HP. It is a general purpose QoS specification which is not tied to any domain. Instances are generated with its abstract mechanism with three fundamental concepts: contract type, contract and policy. The quality values defined in a specification are manipulated using the QoS fabric. Type checking is available with its QRR tool. Although it provides a flexible approach to describe QoS values, it lacks of common strong type checking mechanisms for data exchange over multiple applications, which is a crucial factor for interoperability between distributed applications. However, currently it is given in the form of Extended BNF notation without any specific syntax.

2. Hierarchical QoS Markup Language (HQML)

Hierarchical QoS Markup Language [15] is a XML representation of a distributed multimedia application to be delivered across the Internet with QoS capability. During runtime the HQML Executor translates the HQML instance into a user defined data structure and works with QoS Proxies which enable QoS related operations such as end-to-end QoS negotiation, setup or enforcement. This work is one of the earlier studies which investigate the use of QoS on an IP network using DTD as the data constraint language. However this work is specialized to the multimedia application context, and we believe there should be a more general-purpose description of QoS data representation for real time system.

3. SLAng

SLAng [16] is a XML based language for defining Service Level Agreements particularly functional integration for inter-organization service at storage, network, middleware and application layer. It proposes a contract agreement framework to capture the non-functional properties of an end-to-end communication of a service provider and client specific to the use via web service. SLAng encapsulates the QoS target with Application Service Provider architectural components. QoS negotiation proceeds through communication between an application and a component in the suggested service provision reference model. SLAng consists of Endpoint description (information about the parties involved in the contract), Contractual statements (conditions of a contract) and Service Level Specification (actual QoS description). It enforces the horizontal SLA, which governs the interaction between the coordinated peers, and the vertical SLA to subordinate pairs within the service provision architecture stack. SLAng has clearly represented the abstract level SLA details in a XML specification, however, the structure it has defined in the paper is limited in that it lacks flexibility in extending the element instance, e.g. the parameters defined in a particular QoS dimension (E.g. maximum throughput) of a QoS Characteristics (E.g. Performance, which can be described in different QoS dimension) is fixed, such as Performance is set to be described by response time, peak time latency, etc. In our proposed approach, we allow multiple QoS dimensional attribute.

IV. The application of 'service'

In speaking of 'service' in the context of this paper, the purpose of this work is to deliver a mechanism that describes a general purpose QoS at a detailed level. The service we are pinpointing in this paper is not particular service, and we intend our work to apply to the widest scope. There are, however, a number of subjects we are particularly interested in the course of this discussion. This includes the description of networked resources, for example, the CPU execution time of a particular function in a critical system; interactions between objects, entities that are featured with time-related elements, e.g. the rate on file retrieval speed of the web server. In view of usage of QoS in this paper, the disposition of application is dependent on the domain or usage. In some situations, a catalog-based description will be best suited for the use of resource acquirement for allocation, particularly for sizing and acquisition, whilst for a runtime environment where the communication is dynamic, a service level agreement would be more essential.

Adding QoS in an Internet based application can improve the predictability of resources. There is a common misunderstanding that the application of time critical system is not pragmatically efficient based on IP network. Though the nature of internet network is vulnerable to jitters and interruption by a collection of factors, with the use of QoS, a process that has a greater time duration interval can benefit by gaining a better economy of resources with perceived values.

Representing QoS in XML

We have used a Descriptor-based approach, to collectively assign each entity in our model into a descriptor unit. The use of XML-based descriptors allows one to describe the following: (1) The non-functional abstraction of QoS data for reuse while capturing the important properties that are relevant to QoS; (2) Mapping the QoS element from the QoS framework to a XML formalism; (3) Allows integration to other XML namespaces. The descriptors are classified into Concept Descriptor (CD) type or Relationship Descriptor (RD) type. A CD is like metadata which describes the quality of service or resources. An example is the Performance (QoSCharacteristics as a type of CD) of the work load of a web server (Resources from GRM). A RD is used to specify the structure and semantics among CDs or RDs. Having defined the collections of CDs, the final step is to transform the details into a XML Schema specification and assign with proper data types, which allows XML document instances to be generated. This is depicted in Fig. 1.

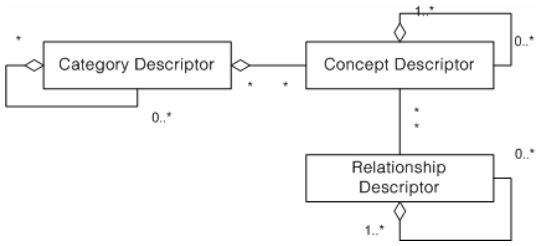

Fig.1 Descriptors structure

As the current point-to-point based Internet only provides a 'best-effort' service, it is important to provide a mechanism to allow quality values of resources that belong to a particular service domain to be discovered. One can do this in an autonomous fashion, where the service seeker and service provider can observe, analyze, predict, calculate, evaluate and validate the quality values that are relevant to their service needs. A detailed and flexible QoS Characteristics description along with an efficient QoS Negotiation mechanism can help eliminate the ambiguity. In addition to this, for successful collaboration between a service user and a provider, it should also include the special conditions for possible future changes in service levels or modes for graceful degradation or modification.

In this part we will present the intended application of this work, how it will effectively integrate with the Resource description framework. In previous work [3, 4] we have introduced the Real Time Markup Language. The work of RTML is based on the 'UML Profile for Schedulability, Performance, and Time Specification' [17] (SPT hereafter). SPT provide a unifying framework to encompass the specific facilities for modelling real time systems. As depicted in Figure 2, the core resource model is extracted from the OMG SPT profile; this shows the relationship of resource service instances to QoSCharacteristics. Resource is represented as an entity of resources which provide support for different tasks and QoS Characteristics can provide quantification for this, ResourceService is the entity which utilizes the Resources and perform activities in a real time system. For example, an instance of Resource could be the computing disk of a web server or documents that are located in a file server which is open to the public for download and an example of ResourceService could be the task that gathers real time data from the sensor to data collector. These entities are enhanced with a QoSCharacteristics description. The addition of QoS to Resource and ResourceService could be of different levels of complexity depending on the level of granularity and the nature of the entity. It could be represented as a primitive data type and it could also be represented as complex data structure. These information can be clearly represented in XML Schema.

The structure of RTML is partitioned into 6 components. It includes (1) Resources; (2) Time; (3) QoS; (4) Concurrency; (5) Schedulability; (6) Performance. A QoS Profile in RTML is defined as a separated namespace. It is intended to be utilized by other part of RTML, where one wants to include a QoS aspect.

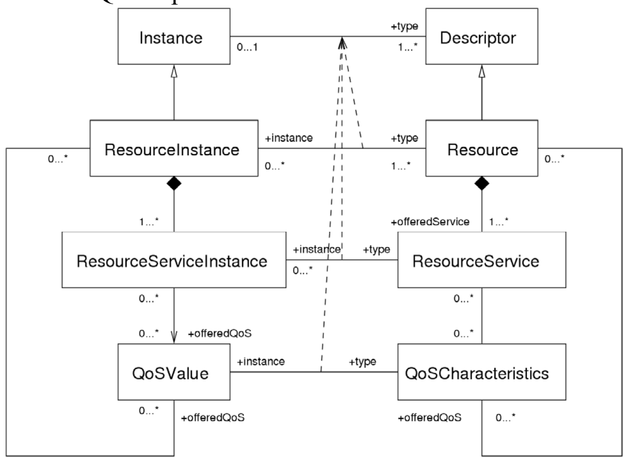

Fig.2 General Resource Model (Extracted from SPT)

ITU X.641 model and OMG QoS Profile

We adopt the specification of QoS developed by ISO and ITU-T document number X.641 Recommendation (ISO/IEC IS 13236) [18]. This specification provides a standard approach to describe QoS when used for different purposes, and also how QoS can be described from different viewpoint and to different level of precision. This specification has generally been refined to suit for the application in open distributed processing. Along with the ISO/IEC specification, we have also adopted the interpretation of QoS from the UML™ Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms from Object Management Group[19] (QoS Profile hereafter). This profile has described conceptual meta-models for Quality of Service which are based on the semantics of CQML, and the models were transformed into a UML Profile for QoS. Literally, the UML Profile is an extension to the standard UML specification. The extensions are usually described by lightweight modeling components such as tagged values or stereotypes. By adapting this approach our QoS XML Profile will be integrated with the mainstream standards on QoS and ensure the comprehensive representation of QoS. In the following section we will discuss each constituent of our adopted QoS Framework. As depicted in Fig. 3.

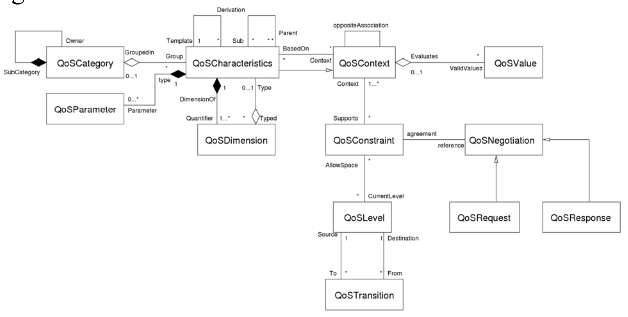

Fig.3 QoS Framework

QoS Framework

In the following section, the collection of QoS elements in the QoS Framework is to be explained. The foundation of this framework is based on QoS Profile.

QoSCharacteristic: It represents a quantifiable metric for services that have QoS support. It is expressed as a constructor which is composed of specific quality value descriptions such as latency, throughput or security. It is defined independently from the entity it is intend to qualify. It also can be derived based on other QoSCharacteristics to compose a specialized form of QoSCharacteristics.

QoSDimension: A QoSDimension is a facet of QoSCharacteristics and often, a QoSCharacterisitics can be viewed from different aspects. An individual QoSCharacteristics is partially faceted by the QoSDimension quantifier which gives a direct measure with a fixed value. For instance, one of the QoSCharacterisitics of end-to-end communication is responsiveness. To give a pragmatic measure one can describe its maximum time to response. The representation as QoSDimension can include the statistical quantifier; the direction (whether this value is preferred to grow) and the unit of measurement (such as nanoseconds or milliseconds).

QoSValue: QoSValue is represented as an instance of the QoSCharacteristics. It directly quantifies a QoSCharacteristics with assigned values described by the composed QoSDimension/s.

QoSCategory: A QoSCategory is literally a non-empty set that consists of QoSCharacteristics. A QoSCategory is an entity in QoSFramework which contains a group of QoSCharacteristics and other QoSCategory. The use of QoSCategory help organize the QoSCharacteristics by keeping related ones together. A directory which belongs to a QoSCategory is a sub category. Often, when the number of QoSCharacteristics defined grows, one needs a structure to consistently organize the QoSCharacteristics. For instance, performance-related QoSCharacteristics such as throughput, latency can be collected together under the category of Performance.

QoSContext: A QoSContext element provide an extensible semantic description for a QoSCharacteristics. It provides the meaning of QoSCharacteristic(s), the perception which may be associated to another entity. The description of a QoSContext is based on QoSCharacteristic(s). The oppositeAssociation relationship indicate whether or not this QoSContext is based on another QoSContext.

QoSConstraint: It is expressed as an abstract element which provides a general description for the specialized constraint types. QoSConstraint is provided as a check point for elements which are associated with QoSCharacteristics. Its association to QoSContext implies the QoSCharacteristics, or other elements such as function element or user requirement involve in this particular constraint. Through the association with QoSContext, a QoSConstraint is expressed in a QoSValue. The constraint and association with the related entity is depicted in Figure 4.

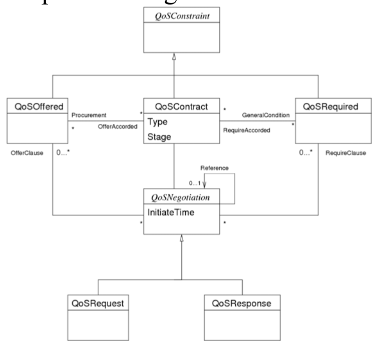

Fig.4 QoS Constraint and Negotiation (Adopted from SPT)

QoSRequired: A QoSRequired concept can be viewed in the perspective of a client or a server. A client who is seeking service may state the obliged quality. This means where a service provider is willing to provide the service to the client of this particular constraint of QoSCharacteristics, the service provider must be able adjust the service quality level to fulfill the client's request specified by the QoSRequired in the client's specification. Service client and provider will use QoSRequired to evaluate the required service quality level.

QoSOffered: A QoSOffered element is used to advertise the quality level a service is available to facilitate. This element is described with QoSContext, and accords to the space limited by the instantiation of a generalized QoSConstraint. The advertised value is dependant on the existing resources. When a client, or provider presents the QoSOffered, it must reach the quality level advertised.

QoSContract: A QoSContract establishs a service level agreement between a service provider and the client. The quality level stated by the provider and client is described as a QoSCharacteristics. Each participant of the contract is given a role as provider or client. Each will specify the QoSRequired and QoSOffered. In general, each participant will have to ensure they can provide the advertised quality values and be able to fulfill the requested quality level. In other words:

Client-QoSRequired ≤ Provider-QoSOffered or
Provider-QoSRequired ≤ Client-QoSOffered

QoSNegotiation: A QoSNegotiation provides an abstract template structure for QoSEnquiry and QoSResponse. An InitiateTime as a TimeValue remark when a QoSNegotiation is triggered.

QoSEnquiry: Often, an anonymous domain may need to seek collaboration with another resource service domain without prior agreement. A service seeker may wish to make an enquiry to the service provider to collect the historical information on the QoS measure to analyze the future possible QoS that can be acquired. This may involve issues of access control or admission policy, however are outside the scope of this paper. The anonymous domain will look up the registry (This can be achieved through information retrieved from UDDI, for example) hence making an enquiry based on given quality values that it requires. The fairness of the final service level agreement is reached through the negotiation process between the service seeker and provider. When a QoSEnquiry is being made, it will include the information of the enquirer which is of concern for the service provider. This may include the contact details, trust level or the ability to provide QoSOffered promised.

QoSEnquiry is basically for gathering QoS information of a domain or resources. There are three cases of how a resource service enquirer will make use of the QoSEnquiry facility. 1) A resource service seeker may send a QoSRequired with the QoSEnquiry and expect a reply with QoSOffered. 2) The service seeker may gather the value by providing the QoSOffered information and in return collects the value of QoSOffered proffer by the service provider. 3) A resource service enquirer will send both the QoSRequired and QoSOffered to the service provider to evaluate what the service value that can be proffered to the enquirer. When the service provider receives such an enquiry, it will evaluate such a measure based on its current resource availability.

QoSResponse: The QoSEnquiry may not get processed immediately, and could be put in a queue for bulk processing. Whenever a service provider receives a QoSEnquiry, it should generate a unique identifier to the specific QoSEnquiry made. An initial response will be made to the resource service seeker to acknowledge the enquiry.

The resource service provider will to evaluate the appropriate operating target base on the given requirements and/or offerings. A QoSResponse is sent back to the service enquirer including a collection of resources featured with the QoSCharacteristics. Sometimes, a QoSOffered may include bespoke conditions, this will be an optional constituent to the QoSResponse. Special care should be taken in developing the bespoke condition as this will prevail over the QoSOffered. The bespoke condition can be described in QoSLevel and QoSTransition, which states when the provider fails to supply the service at the assigned level and the result of triggering a QoSTransition to another QoSLevel.

QoSLevel: A QoSLevel represents the working mode of QoS that a service/sub-system can support. A working mode is driven from the current service quality level where the 'level' of this service is gathered from the performance value partition into a numbers of intervals. The QoSLevel can be derived based on a function of a collection of services or resources. For example the total amount of free storage space together with the bandwidth or time during the day. One can use these factors to indicate the quality level that is available for services. This may be defined with more than one QoS level attribute, which may mean that an elevation or decline in level may lead to a transition in the quality values from the standard that is currently being provided from one to other. This could require renegotiation or adjustment.

QoSTransition: A QoSTransition is the result of a change that happened to the QoSLevel. A QoSTransition connects a source level and a destination level. A Boolean value AllowSpace indicates whether this current QoSLevel is valid for a particular QoSConstraint. When the state of the resource or service indicated by the current source level drops and the AllowSpace value become false, a QoSTransition fires and changes the associated level to the new destination for level which its AllowSpace is indicated true. As depicted in Figure 5.

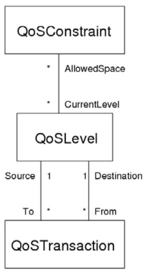

Fig.5 QoS Level, QoS Transition (Reproduced from SPT Profile)

V. MODEL TRANSFORMATION INTO XML

In this following section, we are going to discuss the transformation of the QoS Framework from the OMG QoS Profile. Here we will demonstrate using the transformation method proposed in [LING FENG PAPER] to convert the conceptual model into XML. In previous papers [], we have discussed the method of transforming a conceptual class model into XML Schema using clearly defined rules. Here we are combining this together with the XML-based descriptor approach. Firstly we have defined the high level descriptor structure. This is depicted in Fig. 6. The highest level of element in the XML schema is the generic descriptor DType which is defined as an 'anyType' of the existing XML dataype, which can be typecast to any complexType definition. The ConceptDescriptor, RelationshipDescriptor as well as the CategoryDescriptor are the derived to be complexType element of DType, as depicted in Fig.6.

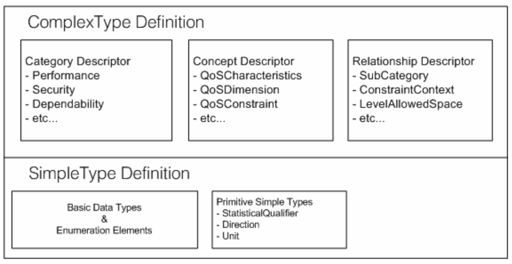

Fig.6 Telemetry example scenario (Reproduced from the QoS Profile)

VI. TELEMETRY EXAMPLE

This section includes a telemetry design example extracted from the OMG QoS Profile. In this example we have used our XML methodology to illustrate the communication between the QoS featured classes. Here we focus on the portion in the penned outline for simplicity. The expression of schedulability for this subsystem is shown in Fig.7. The QoS Contract represents the agreement between the different caller and operator non-functional requirements such as performance, security. In this paper we have implemented a reference model of QoSCatalog fea-tured with a collection of reference quality parameters and organized into a number of QoSCatagory. A small example of Scheduling Analysis XML Schema was also defined to describe the example used in the OMG QoS Profile, and lastly the telemetry example which uses these three name-spaces to describe the quality values used in the telemetry example scenario.

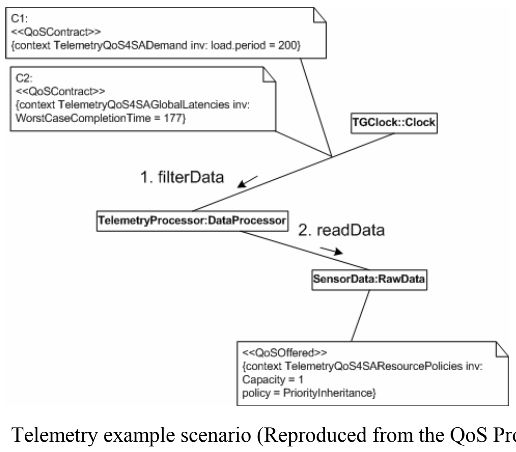
C1:
<<QoSContract>>
{context TelemetryQoS4SADemand inv: load.period = 200}
C2:
<<QoSContract>>
{context TelemetryQoS4SAGlobalLatencies inv:
WorstCaseCompletionTime = 177}
TGClock::Clock
1. filterData
TelemetryProcessor:DataProcessor
2. readData
SensorData:RawData
<<QoSOffered>>
{context TelemetryQoS4SAResourcePolicies inv:
Capacity = 1
policy = PriorityInheritance}

Fig.7 Telemetry example scenario (Reproduced from the QoS Profile)

The QoSContracts in Fig.7 features the communication between Clock, Data Processor and the Raw Data. Given the QoS Contract applied on the filterData function between Clock and Data Processor, we come up with the XML Schema definition in Fig.8.

```
<element name="ClockTelemetryContract">
 <complexType>
  <complexContent>
   <restriction base="qos:ContractCDType">
    <sequence>
     <element name="ContractClause" maxOccurs="unbounded">
      <complexType>
       <complexContent>
        <restriction base="qos:CharacteristicsCDType">
         <sequence>
          <element name="TelemetryQoS4SAGlobalLatencies">
           <complexType>
            <complexContent>
```

```
             <restriction
base="telemetry:TelemetryQoS4SAGlobalLatencies">
              <sequence>
               <element name="WorstCaseCompletionTime" maxOc-
curs="unbounded">
                <complexType>
                 <complexContent>
                  <restriction base="qos:DimensionCDType">
                   <choice>
                    <element name="ValidValue" type="double" minOc-
curs="0" maxOccurs="unbounded"/>
                   </choice>
                  <attribute name="Direction" fixed="decreasing"/>
                 <attribute name="Unit" fixed="ms"/>
                </restriction>
               </complexContent>
              </complexType>
             </element>
            </sequence>
           </restriction>
          </complexContent>
         </complexType>
        </element>
       <element name="TelemetryQoS4SADemand">
        <complexType>
         <complexContent>
          <restriction base="telemetry:TelemetryQoS4SADemand">
           <sequence>
            <element name="Load" minOccurs="0">
             <complexType>
              <complexContent>
               <restriction base="q_catalog:ArrivalPatternType">
                <sequence>
                 <element name="period" minOccurs="0" maxOc-
curs="unbounded">
                  <complexType>
                   <complexContent>
                    <restriction base="qos:DimensionCDType">
                     <choice>
                      <element name="ValidValue" type="string" mi-
nOccurs="0" maxOccurs="unbounded"/>
         <!-- Details omit here for simplicity -->
```

Fig.8 XML Schema for ClockTelemetryContract

In the above XML Schema definition we have outlined the data definition of the communication between the Clock and Telemetry contract. This contract is used to characterize the load of filterData including the Telemetry-QoS4SAGlobalLatencies and TelemetrySADemand. A ClockTelemetryContract is created as an instance of CD, QoSContract from the 'qos' namespace which is restricted to the definition of `<restriction base="qos:ContractCDType">`. The `ClockTelemetryContract` features the communication between `Clock` and `DataProcessor` instances. The `ContractClause` states the telemetry specific to `TelemetryQoS4SAGlobalLatencies` of the `filterdata` function. It is a generalized form of QoSCharacteritiscs, which is composed of the QoSDimension constituent to quantify the different dimension of a QoSCharacteritics. The actual value of a QoSDimension is represented as `ValidValue`. In the example above, the `ValidValue` of the `WorstCaseCompletionTime` is given by the QoSDimension characterization with direction in favor of 'decreasing', and the Unit is typed to 'ms' (milliseconds).

Example 1: filterData

The OCL expression for contract C1 `<<QoSContract>> {context TelemetryQoS4SADemand inv: load.period = 200} and C2: <<QoSContract>> {context TelemetryQoS4SAGlobalLatencies inv: WorstCaseCompletionTime = 177}`. In the first QoSContract, the constraints were defined for the load of `filterData` is constrained to the period within 200. If this constraint is to be instantiated based on the above XML Schema definition, the result will be featured in Fig. 9.

```
  <ClockTelemetryContract>
     <ContractClause QCtx_ID="ID000089">
       <!-- The TelemetryQoS4SADemand as a type of QoSCharacteris-
tics -->
          <TelemetryQoS4SADemand QCtx_ID="ID000090">
             <period>
                <ValidValue>200</ValidValue>
             </period>
          </TelemetryQoS4SADemand>
       <!-- The TelemetryQoS4SAGlobalLatencies as a type of QoSChar-
acteristics -->
          <TelemetryQoS4SAGlobalLatencies QCtx_ID="ID000091">
             <WorstCaseCompletionTime>
                <ValidValue>177</ValidValue>
             </WorstCaseCompletionTime>
          </TelemetryQoS4SAGlobalLatencies>
          <Context/>
       </ContractClause>
     </ClockTelemetryContract>
```

Fig.9 XML Schema for Telemetry Example 1

Example 2: readData

In this example, a QoSOffered is used by RawData class to provides the QoS inforamtion for DataProcessor class. This time the QoS description is used to describe the Class itself. First of all, there needs to have a XML Schema defined for the telemetry policy requirements (As depicted in Fig. 10):

```
<complexType name="TelemetryQoS4SAResourcePolicies">
  <complexContent>
   <restriction base="sa:QoS4SAResourcePolicies">
    <sequence>
     <sequence>
      <element name="Base" type="qos:CharacteristicsCDType" minOc-
curs="0" maxOccurs="unbounded"/>
     </sequence>
     <sequence>
      <element name="Context" type="qos:ContextBaseType"/>
      <element name="Policy" maxOccurs="unbounded">
       <complexType>
        <complexContent>
         <restriction base="qos:DimensionCDType">
          <choice>
           <element name="TypedCharacteristics"
type="qos:CharacteristicsCDType" minOccurs="0"/>
           <element name="ValidValue" minOccurs="0" maxOc-
curs="unbounded">
            <complexType>
             <simpleContent>
              <extension base="anySimpleType">
               <attribute name="Type" type="string"
fixed="string"/>
              <!-- Closing tags omitted  -->
      <element name="Capacity">
       <complexType>
        <complexContent>
         <restriction base="qos:DimensionCDType">
          <choice>
           <element name="TypedCharacteristics"
type="qos:CharacteristicsCDType" minOccurs="0"/>
           <element name="ValidValue" type="integer" minOccurs="0"
maxOccurs="unbounded"/>
          </choice>
          <attribute name="Direction" fixed="increasing"/>
    <!—Details omitted for simplicity -->
```

Fig.10 XML Schema for Telemetry Example 2

Based on the OCL expression: `<<QoSOfered>>{context TelemetryQoS4SAResourcePolicies inv: Capacity = 1 policy = PriorityInheritance}`, a XML instance can be created based on the above schema definition:

```
<RawDataOffer>
    <Policy StatQuantifier="min" Direction="increasing"
Unit="bit/sec">
        <ValidValue>PriorityInheritance</ValidValue>
    </Policy>
    <Capacity StatQuantifier="min" Direction="increasing"
Unit="bit/sec">
        <ValidValue>1</ValidValue>
    </Capacity>
</RawDataOffer>
```

Fig.11 XML Instance of RawDataOffer

VII. Conclusion

In this paper, we outline a technique for generating an data format by using a XML Descriptor approach to describe QoS data which can feature the quality values between classes, especially in distributed process. QoS plays an important role in Real Time Systems as illustrated by the logical model of the telemetry system. We intend to expand this XML Descriptor based methodologies to describe real time systems. The benefit of our proposed XML methodologies is three folds: 1) Increasing flexibility for model design which enables flexible reconfiguration of the specification; 2) Improve the interoperability which facilitates data exchange over the distributed environment 3) minimizing the engineering gap. By having the XML Schema defined, one can generate XML document instances and exchange data corresponding to the common schema. Our intention for the next step is to apply the descriptor based XML methodology to the modelling of concurrency, scheduling, resources and performance. We wish to achieve a model that would allow distributed real time systems to exchange data by using RTML.

VIII. Reference

[1] S., Blake, D., Black, M., Carlson, E., Davies, Z., Wang and W., Weiss, "An Architecture for Differentiated Services:, IETF RFC 2475, December 1998

[2] A. deVos, C. T. Rowbotham, "Knowledge representation for power system modeling", 22nd IEEE Power Engineering Society International Conference, May 2001

[3] P.Poon , T. Dillon and E. Chang, "XML as a basis for interoperability in Real Time Distributed Systems", Workshop on Software Technologies for Future Embedded and Ubiquitous Computing Systems(WSTFEUS'04), May, 2004, Austria.

[4] P.Poon , T. Dillon and E. Chang, "Transformation of QoS data into XML characterising data communication in Real Time Distributed Systems", The 2nd IEEE International Conference on Industrial Informatics(INDIN'04), June 2004, Germany

[5] Object Management Group, OMG UML Specification 1.5. OMG document number ad/03-03-01 (March 2003)http://www.omg.org/technology/documents/formal/uml.htm

[6] S. Johnson, "Rational UML Profile for business modelling", http://www-128.ibm.com/developerworks/rational/library/5167.html. (Accessed on 1st April, 2005)

[7] B. Selic, "Unified Modeling Language version 2.0 - In support of model-driven development", http://www-128.ibm.com/developerworks/rational/library/05/321_uml/. (Accessed on 2nd April, 2004)

[8] Object Management Group, UML 2.0 Superstructure, 3rd Revision, OMG document number ad/03-04-01, (2003), www.omg.org/cgi-bin/doc?ad/03-04-01.

[9] L. Feng, E. Chang, T. Dillon, "Schemata Transformation of object-oriented conceptual models to XML" International Journal of Computer Systems Science & Engineering, vol 18, no. 1, January, 2003, P.46-60

[10] Rajugan R., Elizabeth Chang, Tharam S Dillon & Ling Feng; "A Three-Layered XML View Model: A Practical Approach"; 24th International Conference on Conceptual Modeling (ER '05), Klagenfurt, Austria, October 24-18, 2005

[11] J. Ryan, "Modeling One-To-Many Relationships With XML", http://www.developer.com/xml/article.php/10929_1575731_1 (Accessed on 19th April, 2005)

[12] M. Bernauer, G. Kappel and G. Kramler, "Representing XML Schema in UML - An UML Profile for XML Schema"

[13] G. Olsen, "An overview of B2B integration", http://www-3.ibm.com/software/webservers/pam/EAI_Journal_B2B_Integration_Overview.pdf (Accessed on 25th April, 2005)

[14] S. Frolund and J. Koistinen. QML: A Language for Quality of Service Specification. Technical Report HPL-98-10, HP Laboratories, 1998.

[15] X. Gu, K. Nahrstedt, W. Yuan, D. Wichadakul, and D. Xu. An XML-based Quality of Service Enabling Language for the Web. Technical report, Department of Computer Science, University of Illinois, April 2001.

[16] D. Lamanna, J. Skene and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements", 9th IEEE Workshop on Future Trends in Computing Systems, June 2003, San Juan, Puerto Rico.

[17] Object Management Group, UMLTM Profile for Schedulability,

[18] Performance, and Time Specification, Version 1.1, OMG document number formal/05-01-02, (2005), http://www.omg.org/cgi-bin/doc?formal/05-01-02.pdf.

[19] International Organization for Standardization, Final Text of X.641 for Acceptance at the SG7 December 1997 Plenary, ISO document ITU-Telecommunication Standari-zation Sector TD0115 (December 1997).