XML Descriptor based approach for Real Time data messaging

Polly M.S. Poon[1], Tharam S Dillon[1], Elizabeth Chang[2], Ling Feng[3]
[1]*Faculty of Information Technology, UTS, Sydney, Australia*
{polly, tharam}@it.uts.edu.au

[2]*School of Information Systems,Curtin University of Technology,Australia*
change@cbs.curtin.edu.au

[3] *Database Group, Department of Computer Science, Univerisity of Twente, Enschede, The Netherlands,*
ling@cs.utwente.nl

Abstract

This paper presents an overview of the Real Time Markup Language (RTML). RTML is a XML profile which provides the syntactic representation for describing the semantics of real time data for exchange over distributed networked real time systems. For the basis of interoperability, this profile is described in the XML Schema language. This paper describes the background of this work and shows how the vocabularies are developed, and how it derives the extensibility of XML Schema in aiding the definition of data in real time systems in order to achieve the goal of interoperability.

1. Introduction

Traditionally, most of the developments of real time embedded systems are driven by technology. System requirements of this type are conventionally overwhelmed by the complex technicality. But today, the alignment of operation and development decisions has gradually moved to the business and industrial perspective and technology becomes a tool which drives these businesses or industries. This has contributed to the advent of enterprise service based system development methods, where technology allows the acquisition, consolidation and transformation of data or process instruction between discrete system domains. For instance, the work flow collaboration between logistics system and large production plants where manufacturing machineries are located in dispersed locations.

To enable the collaboration between the remote systems, it requires intelligent use of current technologies. Despite the wide spread use of the emerging distributed networking paradigm (E.g. Java RMI, IIOP, SOAP), there are still many applications that interface with the external environment using a fundamental form of networking protocol. (E.g. Socket, RPC). To consider the ever changing web architecture faced by enterprises, there are demands for system integration at the application level, where system components could collaborate with the support of interoperability mechanism, especially in the context of real time systems.

2. Challenge of developing distributed real time systems

[1] addresses the complexity involved with distributed real time systems when they are realized by the numerous middleware standards. The challenge is larger when the system is responsible for a wide range of tasks must be handled by the application including safety critical and the pressure for increased productivity [2]. Some of the challenges are listed as follow:

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 [4], one requires integration of real time data, or historic data to make intelligent analysis and allow the system to run predictably. This requires the

Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing
0-7695-2561-X/06 $20.00 © 2006 IEEE

integration between archived data and operating real time data. Another example would be the implementation of communication between segments of networks with the use of management agent, where information is exchanged by agent on behalf of network stations.

3. Representing the synchronization and concurrency of parallel activities of active objects in a textual representation

4. To map the meaning used by individual systems to a mutually understandable vocabulary that enable system integration.

5. Many of the available middleware solutions for distributed application integration are targeted at the typical business model, complex services like real time systems is not generally available

The key of success for the collaboration between distributed computing architecture is relied on consistent definition of interface and data definition/access, in particular of how the data is formatted, exchanged and stored. To achieve the above one requires a consistent specification of the data definitions to ensure data integrity and enhance the interoperability among heterogeneous data sources, as depicted in Fig. 1. This ensures a smooth implementation of service collaboration, since data contains the abstract from the logic of the domain [3]. This needs following consideration:

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.

In this paper we propose a XML based messaging method for the interoperability between real time systems. We are looking in terms of application perspective, by providing a clearly defined XML schema, this enable real time data exchange across multiple platforms, which individual systems could map their internal context to a mutually understandable vocabulary. To achieve this we describe a method of incorporating real-time semantics mapping using XML descriptor based model, for representing knowledge in real-time systems enabling XML based messaging. In the reminder of this paper, we will discuss the related work, design goal of RTML, and the complete specification of RTML follow by a summary to conclude this paper.

3. Related Work

Interoperability plays an important role in distributed system communication. There are existing works that targets the use of XML in representing quantitative measures of resources usage in distributed applications.

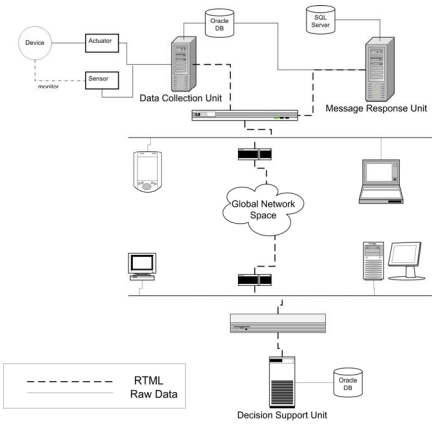

Figure 1 Distributed Communication using XML

RT-XML [12] is an XML based modeling language which describes a profile for XML and multi-media systems and models a few features of real time systems such as time ordering, time constraints, probabilistic behavior and qualitative description of a multimedia stream. It describes the application level control over the delivery of multimedia services. Whilst it provides a descriptive model for multimedia systems, it is not sufficient for appropriate for real time systems in more general applications. In addition, RT-XML does not provide adequate support in schema level representation which XML document cannot be automatically validated with defined constraints.

Hierarchical QoS Markup Language [13] is a XML representation of a distributed multimedia application to be delivered across the Internet with QoS capability. This work is one of the earlier studies which investigate the use of QoS on an IP network using DTDs as the data constraint language. HQML provide a solid model for describing the multimedia systems, its primary purpose as a XML profile is directed towards multimedia and network. Therefore, for real time applications, it requires further investigation for a more general XML profile.

4. Relationship with other standards: ITU X.641 model and OMG Profiles

In order to develop a consistent set of descriptive languages for DRE Systems it requires a solid framework that can provide the abstraction of the properties and behavior of the DRE systems. RTML is mainly derived based on the specification from Object

Management Group, UML™ Profile for Schedulability, Performance, and Time Specification from Object Management Group (SPT hereafter) [14] and the UML™ Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms from Object Management Group[15] (QoS Profile hereafter). The core resources model (See Fig. 2) provides the fundamental abstraction of the relationship between descriptor and instances of the modeling element within the SPT and QoS Profiles. It provides a high level viewpoint of the relationship between Resources, the ResourceService and QoSCharacteristics. In this paper we will discuss on the Resource section of the SPT profile. Other parts have been considered in [17, 18, 19] by the present authors.

Apart from OMG SPT, RTML has also adopted the specification developed by ISO and ITU-T document number X.641 Recommendation (ISO/IEC IS 13236) [10]. This specification provides a standard approach to describe QoS for different purposes, and how can QoS be described from different levels or viewpoints and precision. Along with the ISO/IEC specification, we have also adopt QoS Profile. The meta-model in QoS Profile provides a lightweight modeling component for conceptual design customization.The above standards have provided fundamental constructs for conceptual model transformation into RTML constraints. This design time construct can be complemented by realizing additional details to enrich the description to the design elements defined in those specifications. RTML provides a reference model for the realization of abstract details [17, 20].

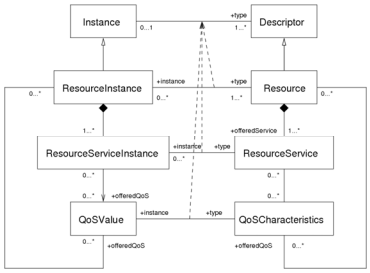

Figure 2 General Resource Model (Extracted from SPT)

5. Design Goal

The goal of developing RTML, is to provide interoperability between real time systems in distributed network, in particular among applications in which time (and a description of resources) is a strong feature in the data representation that determines the validity of process or data. We propose a design time description profile for a model based driven methodology enabling data integration between distributed domains. It is intended that, a XML based knowledge representation should allow the description of time dependent heuristics and dynamic properties into the XML document. RTML provides the notion of time, which is missing in many existing XML profiles. Intended applications of RTML could be network management systems, distributed process control center, teleconferencing software, e-Commerce systems such as an online ticket purchase application, in which time plays a crucial role in determining the validity of the user session.

Among the diversity of concerns for real time systems, RTML must capture and represent of 4 major aspects. These include:

1) **Time**. It can be represent as an instant or interval in RTML. Each time element is attributed with the Clock Reference. Clock Reference is an element which describe the ClockRef (As a IDREF of a global clock which this time element is synchronized to) and the reference current time instant. This ClockRef act as a guarding condition which implies that the execution of process should not exceed this reference time, based on the deadline validity.

RTML uses real-valued and time based representation. A real-valued time is annotated with unit (E.g. microseconds or minute), which can be can be generated in a list or as a recurring time.

2) **Resource**. RTML provides a collection of primitives for resource description including physical resources such as physical device (E.g. CPU or communication devices) and resource service such as application or services description. Resources have a complementary relationship with QoS in real time systems. In RTML, both elements of QoS and Resources can be represented statically as internal description is attached to the sub-element or be linked dynamically within RTML instance which is referenced using xPath with the aid of ID and IDREF.

3) **QoS**. In [18, 19] we have presented the XML Profile for QoS. At this stage we have provided a profile for representing QoS of Resource Instances description.

4) **Action Execution**. The description of action execution and resource service instance are described using the CausalityModel. Essentially the CausalityModel is the container which specifies the actions involved in a systems interaction instance between application domains. The details of relevant components will be discussed in a later section.

6. Abstract elements of RTML – A descriptor based approach

In previous papers [19] we have shown how RTML uses the conceptual OO model transformation [20] into RTML XSD constraints. In this section we will provide an overview on overall the specification of RTML. We will discuss the fundamental constructs in this paper.

RTML is a XML descriptor based profile. Elements in RTML can be classified into three corresponding descriptor types: Descriptor (D), DescriptorMapper (DM) or DescriptorCategory. The use of descriptors allows multi level abstraction. Ds and DMs are further defined into corresponding spectrums. We will discuss these concepts in detail in this section.

Descriptor is a high level construct in RTML. It provides the syntactic and semantic descriptor of metamodel elements from class and transformed into XML Schema using [20]. Example of instantiation for Descriptor is Device, Scenario, Latency, or Throughput. At an abstract level, Descriptors are categorized into QoS Descriptor and Resource Descriptor. These primarily describe the concepts involve in real time system modeling.

DescriptorMapper: On the other hand DMs provide the details of the relationships between Descriptors, DescriptorMappers, or a combination of both. DMs provide the syntactic description of RTML in XML Schema, and it also provides the semantics of relationship between Descriptor elements. Example of a DM is ConstraintContext in the aspect of QoS, in a nutshell, it describe how the constraints are applied in the context of QoS. Both Constraint and Context are Descriptor elements, putting these together give the description of the scenario within which the constraints can be interpreted.

DescriptorCategory: It is used in RTML to provide a logical grouping for Descriptor and DescriptorMapper. DescriptorCategory is transformed in RTML using <xs:group>. An example of usage is in the general QoS descriptor package, which includes sub-packages such as Performance, Throughput and Latency. Each of the sub-packages is represented as a group in RTML.

7. Global Timing Device and Time

The timing mechanism is defined in RTML as a global invariant. The Timing Mechanism in RTML provides the reference to the location of a physical clock, and some of the offered QoS attributes for the characterization of the timing device. This allows the elements that are defined locally within the RTML structure to reference it through the entire hierarchy. As discussed earlier, the purpose of global timing device provides a mechanism as a point of reference to ensure the validity of data or process.

7.1. Global timing device and synchronization to the timing constraints of local elements

In [7] the concept of timed based automata was discussed. It models the discrete systems by state-transition graphs with the support of global clock and synchronization of clock constraints specified in the local 'locations' (vertices). The elapse of time is expressed through real-value clocks. In RTML timing constraint is enforced through the use of referencing of clock through the linking with the use of URI referencing to the physical location of a clock, or through the use of a relative duration. In RTML, the global timing mechanism is specified in the header unit and allows it to be referenced throughout the entire RTML document structure.

```
   <GlobalTimingDevice id="ID000107">
     <Clock id="ID000108">
       <CurrentValue>
         <TimeInstant  ClockRef="/@AA">2005-12-
11T09:30:47-05:00</TimeInstant>
       </CurrentValue>
       <ReferenceClock>

<ExternalClockURI>http://localhost/rtml/time</
ExternalClockURI>
       </ReferenceClock>
       <Resolution unit="ns">1</Resolution>
       <Drift unit='ns'>0.00001 </Drift>
     </Clock>
   </GlobalTimingDevice>
```

Figure 3 Global Timing Device in RTML

7.2. Extended time types

As discussed previously, the representation of time is different for real time systems. In some cases, time needs to be described in microseconds and in a recurring order. In RTML, we extend the definition of time in XML Schema which allows time to be facet in different dimension.

Generally the timing properties of RTML can be classified as absolute time (Time Instant) and relative time (duration). Time can be described as a real-valued with metric based measurement to specify its granularity. Each time value is associated with a Clock reference, as specified by the Global Clock defined in the header of the RTML instance.

```
<xs:complexType name="TimeInstantType">
  <xs:choice>
```

```
    <xs:element name="TimeInstant">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:dateTime">
           <xs:attribute name="ClockRef"
type="rtml:xPathReferenceType"/>
         </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  <xs:element name="MetricTimeInstant"
type="rtml:MetricTimeInstantType"/>
  </xs:choice>
  <xs:attribute name="ClockRef"
type="rtml:xPathReferenceType"/>
</xs:complexType>
```

Figure 4 Time Instant Type

```
<xs:complexType name="MetricTimeInstantType">
  <xs:simpleContent>
    <xs:extension base="xs:double">
      <xs:attribute                name="unit"
type="rtml:TimeUnitType"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Figure 5 Metric Time Instant Time Type

Time duration can be represented in primitive XML Schema duration time or using a metric based recurring time interval.

```
<xs:complexType
name="RecurringMetricDurationType">
  <xs:sequence>
    <xs:element name="Interval"
maxOccurs="unbounded">
      <xs:simpleType>
        <xs:list
itemType="rtml:DurationIncrementType"/>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="unit"
type="rtml:TimeUnitType"/>
</xs:complexType>

<xs:simpleType
name="RecurringTimeDurationType">
  <xs:list itemType="xs:time"/>
</xs:simpleType>
```

Figure 6 Recurring Metric Duration Type in XML Schema

8. Resources and the services offered

Resource, in SPT, is defined as a run-time entity that can offer services which is characterized with the notion of QoS. For instance, one can express the latency of the network connection or to reflect the dimension of the inter-arrival time of the packet to reflect is available capacity. The purpose of this is to quantify the variable consumption and existence of the resources physical underpinnings.

As described above some resources offer services. In RTML, resource services are expressed as a functionality offer by the resource. This could be an event generated from an instance of Resource (E.g. Sensor, A monitoring unit in the data collection server). A service could be described as an event or an action execution, which consists of a number of operation statements. In normal cases, an event should be bounded by the time constraints which are validated at runtime to ensure the time it was executed does not exceed the allowed deadline. Once this time instant or allowed duration is passed, the given condition for execution will be changed based on the fact of whether it is a hard or soft deadline. To describe the causality of the deadline and action, we can represent it as follows:

Using Pre-condition P which specifies that if the current time tc is less than the deadline t, then it implies that module Q occurs.

$$\forall P, Q : P \rightarrow Q$$

8.1. Execution Time

Execution time specifies the time that is allocated to an action to complete its course of execution. This is represented as a type of Time Interval in RTML. The constraint of Execution time is specified as follows:

8.1.1. Delay. Delay can be represented as an Event with a given period of time to elapse from the time it is initiated to when the event can be executed. For a given event predicate P, clock N, delay is described as propositional statements:

$$\forall P, t \in N : P \xrightarrow{t} P$$

Which t is the firing delay that must elapse for P to hold true.

8.1.2. Execution time range. In some cases, one cannot predict the exact firing time of an event but merely a time zone when it will be executed. To model this, one can use a range specified by its starting instant and ending instant. Predicate *P* is valid as long as *t* is within the range of t_{start} and $t_{end.}$:

$$\forall t \in N, P, Q \exists t' \, t_{start} \le t' \wedge t' \le t_{end} : P \rightarrow Q$$

8.1.3. Timeout. Particularly in real time systems, often events are controlled by some timing device. For instance, a timer is set to expire at a particular period of time and notifies the corresponding object about the total elapse of time. The purpose of a timeout event is to monitor some continuously running process by periodically checking some of the correctness of some parameters, to ensure its normal working mode.

As specified in OMG SPT profile, a Timeout event is associated to a Timer. It is assumed that a duration ttimeout is the value which indicates the timeout, which is the time instant when an event is fired and tcur is the current time. We represent timeout in logical equivalence below:

$$\forall P, Q, t \in N \exists t', t_{timeout}, t_{cur}, (t_{cur} \le (t' \wedge t_{timeout}))$$
$$\wedge (t_{start} \le t' \wedge t' \le t_{end}) : P \rightarrow Q$$

In other words, the concept of timeout is incorporated into this statement such that if all the conditions of *P* are satisfied and if the timing constraints which the current time is less then or equal to the assigned firing time and the duration of timeout, then it is legal to execute module Q.

8.1.4. Polling. Polling is a process which checks the value of a particular resource instances at the frequency of defined time interval. This could be for example a monitoring module sample the buffer in a memory space. A polling event could also sample the external resource instance. There are two types of polling process:

1. An event which interrupts the currently running process. This interrupt could result in some routine service actions.
2. An event is execute when the particular resource instances has completed all its process. In some cases, for example a DB transaction, one would want to ensure the atomicity of the process and therefore the poll only executes when the transaction is completed:

This could be also represents in RTML as follows:

```
<MemPoll id="ID002345"
name="RegularSpaceCheck" isInterrupt="1">
       <FiringInterval
ClockRef="//GlobalClock">
              <RecurringMetricDuration
unit="s">
                     <Interval>0 2 4 5 6
11</Interval>
              </RecurringMetricDuration>
       </FiringInterval>
       <ResourceDescriptor>//DKSMem</Resource
Descriptor>
</MemPoll>
```

Figure 7 Polling in RTML

9. Causality Model

Causality is an important concept in RTML as it describe the relationship between the individual application domains through the use of various modeling elements in SPT. The Causality Model in RTML, on the other hand, presents a logical sequence of events in an ordered fashion. The lifecycle of interactions between application domains are described in object-oriented representation which can clearly describe the relationship, and through careful model transformation, models are converted into XML instance. The relationship between events is not only represents by the order in time but can be augmented with the constraint of time. Special events require the control of time validity can be described using timed based event elements (with optional support of QoS).

9.1. Components in Causality model

Causality model in RTML is derived from SPT. SPT has provided detailed description on the modeling elements of this model. In RTML terms, these elements are transformed into XML Schema constraints. Note here, the importance is how these model elements are transformed and then structured in RTML. The important concepts are discussed in the following section.

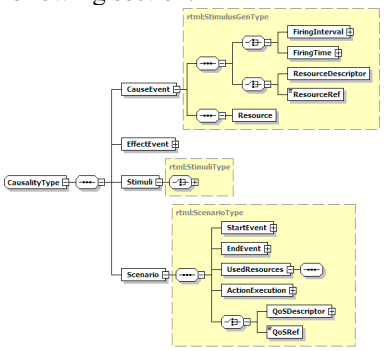

Figure 8 Causality Model in RTML

9**.1.1. Stimulus.** A Stimulus is the result of the trigger of an event or command generated from a remote application domain. The description of Stimulus in RTML is described as a element related to some Resources Descriptors. The trigger of an event is described as a Stimulus Generation Event which generates this Stimulus at a point in time. In the context of real time systems, this could be the case where a sensor network generates an alarm signal (as a Stimulus) to notify that an unusual surge in temperature has occurred. This signal is delivered to its target application such as the monitoring unit, this Stimulus Generation event would specifically address the receiver by pointing to the Resources and interface methods offered by the target component.

9.1.2. Scenario. A scenario is the result of the delivery of the stimulus from the remote application domain to its target application domain. A scenario can be described as a consequence upon the creation of the generation of particular Stimulus. An example of a

scenario occurs when a receipt of a rapid surge in temperature (Stimulus) from the sensor network; a collection of actions is executed such as some checking process against some attributes from the environment or the working order of devices. This Checking routine can be seen as a Scenario, which is associated to the start and event which signifies the start of action execution (E.g. InitializeCheckingRoutine). Similarly, a Scenario can be characterized by the QoS, either statically or dynamically, and a scenario could relate to the required resources described using usedResource primitive.

9.1.3. Exclusive Service Type. Exclusive service is a type of service provided by the Resource Instance (E.g. Physical resources or application resources). It is guarded by the control of access policy defined by a collection of primitives. These primitives, as a type of QoSCharacteristic [15, 18, 19], give a quantitative control on the usage of resources. The details of access control policy are outside the scope of this paper. Further details can be found in [22, 23].

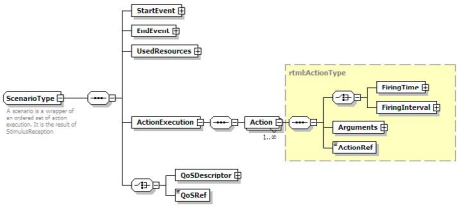

Figure 9 Scenario in RTML

To get access to an Exclusive Service one needs to first trigger an Acquire Service Event. If one successfully gets hold of the Exclusive Service, The Exclusive Service would be marked with the following attribute:

```
<AcquireServiceEvent                id="ID000034"
name="GetBuyTicketSession" isBlocking="false">

<UsedResources>/ExclusiveService/@ID045345</Us
edResources>
    <PerformanceDescriptor>
      <ResourceCommunciation      id="ID000067"
isQoSObservation="1">
        <WorseCaseRequest
name="WC_Request_Waiting"         id="ID000067"
StatisticalQualifier="MAX"
Direction="decreasing" Unit="min">
          <Value>120</Value>
        </WorseCaseRequest>
        <MeanCaseRequest
name="MC_Request_Waiting"         id="ID000067"
StatisticalQualifier="MAX"
Direction="decreasing" Unit="min">
          <Value>35</Value>
        </MeanCaseRequest>
      </ResourceCommunciation>
      <Demand         name="Session_Valid_Time"
id="ID000075" isQoSObservation="1">
        <Load                       id="ID000088"
isQoSObservation="1">
          <period id="ID000088" Unit="min">
            <Value>12</Value>
          </period>
        </Load>
      </Demand>
    </PerformanceDescriptor>
  </QoSDescriptor>
</AcquireServiceEvent>
```

Figure 10 Acquire Service Event XML example

```
<ExclusiveService                   id="ID045345"
name="TicketPurchase" isAcquired="1">
  <AccessAcquiredTime
ClockRef="/GlobalTimingDevice/Clock ">
    <TimeInstant
ClockRef="/GlobalTimingDevice/Clock">2004-11-
17T09:30:47</TimeInstant>
  </AccessAcquiredTime>
  <AllowAccessTime
ClockRef="/GlobalTimingDevice/Clock">
    <Duration><Value>PT12M</Value></Duration>
  </AllowAccessTime>
</ExclusiveService>
```

Figure 11 Exclusive Service in RTML

10. RTML applied: Online ticket purchase

In this section we demonstrate the use of RTML used in a hectic heterogeneous environment. Many ticket sales occupy a significant proportion of overall sales today. Online ticket sales provide a convenient way of ticket purchase for customers as this saves time in lining up outside the ticket office. However, whenever there are tickets of major events going on sale, during critical peak, the online ticket purchase system would need to handle the user loads with volumes of connection as much as tens of thousands per seconds. Systems will need to provide a robust service for all elements involved in the ticket purchase transaction. In such a chaotic situation, the system needs to ensure the atomicity of a transaction (Ticket is booked and paid by customer) and the fairness of resource access (E.g. First in first out).

The sample scenario is shown in Fig. 12. Here we are setting out an online ticketing system allowing collaboration with different ticketing agents. These agents could be the ticket reseller, corporate bodies or credit card points redemption systems which require access to the ticket. Though the number of entities in the scenario is quite small, this could reflect what happens in many different cases where resources are limited and being competed for by a number of sources of demands.

We have created our scenario based on a number of parameters. In this scenario, there are only two customers trying to purchase tickets through agents. Agents directly communicate with the Ticketing Manager of the Ticket System, which controls the

availability of tickets in the database. Agent has no direct access to tickets. Agent can initiate a session with the Ticketing Manager. This session is limited based on duration, during which the agent can search for available seats, reserve the seats, make a booking and make a payment. To secure a ticket, agent must make the payment before the session is ended. Here we set out the rules that, as long as there still are available tickets, Ticketing Manager allows new session to be initiated by agents. In this case, when the number of agents is greater than the number of available tickets, agents will need to compete for the available ticket. If an agent cannot get the available ticket during when the reservation is being checked, then the agent can wait and re-check until the session is finished.

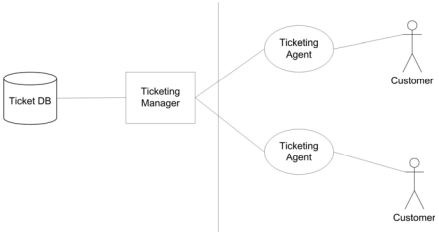

Figure 12 Modeled Ticket System

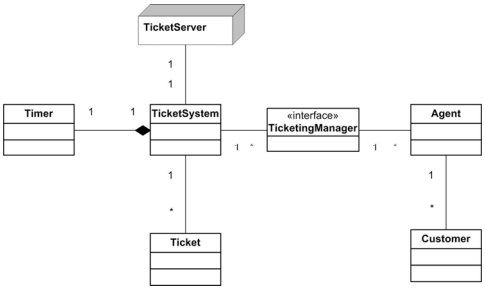

Figure 13 Online ticketing class diagram

10.1. Representing real time semantics in XML

RTML is an ideal candidate to represent the complex data structure in this type of situation. The central notion in this case is interoperability and the management of resources and ensuring the correct timeliness. In this scenario, the control of timeliness will be the duration of session that is allocated to the ticket purchase allows time.

Assume the agent has requested to initiate a ticket purchase session from the Ticketing Manager. The following XML document describes how the Ticketing system returns the details of the current session to the client. Note here the Ticket Purchase schema file has utilized the RMTL schema namespace in characterizing the description of the timing properties and the server description. The server here is characterized with QoS Descriptor which here can specify the access control policy to the Ticket Server.

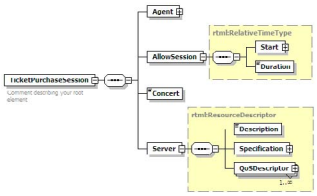

Figure 14 XML Specification of Ticket Purchase Session from Ticketing Manager

```
<GlobalClock id="ID0001">
  <rtml:Clock>
    <rtml:Origin ClockRef="/LocalSysClock">
      <rtml:TimeInstant
ClockRef="ID0001">2001-11-
11T09:30:47</rtml:TimeInstant>
    </rtml:Origin>
    <rtml:Resolution
unit="s">1</rtml:Resolution>
    <rtml:Drift>0.0001</rtml:Drift>
  </rtml:Clock>
</GlobalClock>
```

Figure 15 XML instances of TicketPurchaseSession utilizing RTML

The above representation shows the description of global clock that defines the time reference for the rest of document. Those elements which locally defined within the GlobalClock element are to characterize the QoS of the Clock.

```
<SessionTime ClockRef="ID0001">
  <rtml:Duration>
    <rtml:Interval>PT0H14M</rtml:Interval>
  </rtml:Duration>
</SessionTime>
```

Figure 16 Session time allocated to a Ticket Purchase period

In the above XML fragment, it shows the declaration of session time allow to a ticket purchase session allocate to the agent. Notice here the ClockRef is actually of Type `<xs:IDREF>` . The XSD mechanism controls the integrity between the `<xs:ID>` defined by the `<GlobalClock>` and `<SessionTime>`.

The following section describes how TicketPurchase schema utilize RTML in defining the resource of Server. Here the server is characterized with QoS Characteristics. In this example, the details of access control policy are specified;

```
<Server     id="ID003432"     name="TIXSERVER"
protected="1" isActive="1">
```

```
  <rtml:Description>Ticketing server only
allow access to registered
users</rtml:Description>

<rtml:Specification>AS/400</rtml:Specification
>
  <rtml:QoSDescriptor>
    <rtml:FunctionalityDescriptor>
      <rtml:AccessControl>
        <rtml:FirewallLocation>
<rtml:Value>123.54.23.22</rtml:Value>
      </rtml:FirewallLocation>
      <rtml:FirewallCapability>
        <rtml:Value>Emergency Alert
Logging</rtml:Value>
        <rtml:Value>User
authentication</rtml:Value>
        <rtml:Value>Resume
Connection</rtml:Value>
      </rtml:FirewallCapability>
      <rtml:ProxyLocation>
        <rtml:Value>123.54.234.23</rtml:Value>
      </rtml:ProxyLocation>
      <rtml:EncryptionMethod>
        <rtml:Value>RSA</rtml:Value>
      </rtml:EncryptionMethod>
    </rtml:AccessControl>
    </rtml:FunctionalityDescriptor>
  </rtml:QoSDescriptor>
</Server>
```

Figure 17 Server instance in RTML

10.2. Specifying process in RTML

The following example shows how processes can be specified in RTML. Here a TicketPurchaseProcess is described as a Scenario which is associate to a list of Actions. Note here a Scenario can be dynamically attributed with external argument as shown here the description of Ticket is sourced from external schema definition.

```
<rtml:Scenario id="ID000010"
name="TIXPURCHASEPROCESS">
  <rtml:ActionExecution>
    <rtml:Action id="ID000013"
name="CHECKTIXAVAILABILITY">
      <rtml:FiringTime ClockRef="ID0001">
        <rtml:TimeInstant>2005-12-
02T09:30:49</rtml:TimeInstant>
      </rtml:FiringTime>
      <rtml:Arguments>
        <rtml:Argument id="ARGU_01"
name="ACT_ARGU" mode="IN">
          <rtml:Complex
location="http://www.ticketnow.com/Ticket.xsd"
namespace="http://www.ticketnow.com">
            <Ticket>
              <Concert
type=”string”>U2April1Sydney</Concert>
              <Num_Ticket
type=”int”>2</Num_Ticket>
            </Ticket>
          </rtml:Complex>
        </rtml:Argument>
    </rtml:Action>
    <rtml:Action id="ID000013"
name="PURCHASETICKET">
      .
      .
    </rtml:Action>
  </rtml:ActionExecution>
  <rtml:QoSDescriptor>
    .
    .
  </rtml:QoSDescriptor>
</rtml:Scenario>
```

Figure 18 Scenario in RTML

The agent can directly utilize this XML instance by directly binding to the application module. Here we show an example of using JAXB API from Sun. This API allows dynamic binding of XML Schema into Java code and unmarshall data from XML document and maps to a Java instance.

```
   //TicketIFBean
   try {

     JAXBContext jc = JAXBContext.newInstance(
"ticketing.purchase" );
     Unmarshaller u = jc.createUnmarshaller();

     TicketPurchaseSession tps =
(TicketPurchaseSession)u.unmarshal( new
FileInputStream( "Tix_Pur_u2_032.xml" ) );

     //Unmarshalling XML documet and maps to
command stack
     ActionExecution ae =
tps.getActionExecution();
     List actions = new List();
     //Extract collection of actions from ae
and map to individual Action, code omitted
     for (int i = 0; i < aList.size(); i++)
     {
       ... //Mapping the details of action in
an XML document to Action instances
     }

   }
   catch( JAXBException je ) {
     je.printStackTrace();
     } catch( IOException ioe ) {
     ioe.printStackTrace();
   }
```

Figure 19 RTML Java binding using JAXB API

11. Conclusion

In this paper we have demonstrate a method which uses descriptor based approach to describe the real time semantics in distributed real time systems using XML. We have chosen XML Schema for a number of advantages [24]. In order to create a consistent schema that contains the correct semantics offer by conceptual design model, we have used the OO conceptual model transformation approach proposed by [20]. As discuss in this paper, very often that individual real time systems usually do not have knowledge of the application from each other. So attempting to set up

collaboration for data exchange will be a major challenge and especially when it is dealing with multiple heterogeneous sources. We introduced a descriptor based approach for describing the resources using three types of descriptors namely Concept Descriptor for illustrating the details of the features from a real times systems; a Category Descriptor for organizing the logical structure of Concept Descriptors whose share common similarities; and Relationship Descriptor which describe the structure of a collection of Concept Descriptors and Relationship Descriptors which illustrate the semantics, syntax and organization of such a structure. We believe that, this work will be one of the first steps towards defining an approach using XML to describe real time resources. By providing such a XML profile, it establishes a knowledge based for organizations to exchange data using XML messaging.

12. References

[1] ZHAO, W., Challenges in Design and Implementation of Middlewares for Real-Time Systems: Guest Editor's Introduction. Kluwer Academic Publishers Real-Time Systems, 2001. 20: p. 115–116.

[2] Sharp, D. Real-Time Distributed Object Computing: Ready For Mission-Critical Embedded System Applications in 3rd International Symposium on Distributed Objects & Applications. 2001. Rome, Italy.

[3] Messenheimer, S., Weiszmann, C.: The Impact of Service-Oriented Architectures On Data Access and Integration, http://www.sdtimes.com/article/special-20050501-01.html, Software Development Times, Last Access on May 2005

[4] deVos, A., Rowbotham, C. T.: Knowledge representation for power system modeling, 22nd IEEE Power Engineering Society International Conference, PICA 2001, (2001)

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

[6] Spivey, J.M., The Z Notation: A Reference Manual Second Edition, in Prentice Hall International (UK) Ltd. 1998: Oxford, London.

[7] Alur, R.: Timed Automata, In Proc 11th International Conference of Computer Aided Verification: CAV'99, (1999)

[8] Harel, D.: Statecharts: A Visual Formalism For Complex Systems, Science of Computer Programming, Vol 8, (1987), 231-274

[9] Shaw, A. C.: Real Time Systems and Software. John Wiley & Sons, Inc., (2001)

[10] Apvrille L., Courtiat J., Lohr C., Saqui-Sannes P., TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit, IEEE Transaction on Software Engineering, Vol 30, No. 7, (2004), 473-487

[11] Eshuis, R. and R. Wieringa, Tool Support for Verifying UML Activity Diagrams. IEEE Transactions on Software Engineering, 2004. 30(7): p. 437-447.

[12] Tsang, T.: An XML-Based Architecture for Distributed Real-Time Multimedia Systems, In Proc 1st Global Telecommunications Conference, GLOBECOM 01(2001)

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

[14] Object Management Group, UMLTM Profile for Schedulability, Performance, and Time Specification, Version 1.1, OMG document number formal/05-01-02, (2005)

[15] Object Management Group, UML Profile for QoS and FT Draft Adopted Specification, OMG document number ptc/04-01-05

[16] Gogniat G., Rouxel S., Diguet J., UML Profile for SDR Hardware/Software Adequacy Verification, OMG 1st Annual Software-Based Communications Workshop:From Mobile to Agile Communications, SBC04' (http://www.omg.org/news/meetings/workshops/SBC_2004_Manual/02-3_Goubard_etal.pdf) (Access on 1st February, 2005) (2004)

[17] Poon, P. M. S., Dillon, T. S., Chang, E.: XML as a basis for interoperability in Real Time Distributed Systems, In Proc 2nd Workshop on Software Technologies for Future Embedded and Ubiquitous Computing Systems, WSTFEUS 2004, (2004)

[18] Poon, P. M. S., Dillon, T. S. Chang, E.,: Transformation of QoS data into XML characterising data communication in Real Time Distributed Systems, In Proc 2nd IEEE International Conference on Industrial Informatics, INDIN 2004, (2004)

[19] Poon, P. M. S., Dillon, T. S, Feng, L., Chang, E..: Descriptor based QoS Profile in XML, In Proc 3rd IEEE International Conference on Industrial Informatics, INDIN 2005, (2005)

[20] Feng, L., Chang, E., Dillon, T. S.: Schemata Transformation of Object-Oriented Conceptual Models to XML: Int. Journal of Computer Systems Science & Engineering,Vol 18 No. 1 Jan 2003. (2003)

[21] W3C Recommendation. 1999, Available from: http://www.w3.org/TR/REC-xml-names/.

[22] Niz, D.d., L. Abeni, Saowanee, Saewong, and a.R. Rajkumar. Resource Sharing in Reservation-Based Systems. in IEEE Real-Time Systems Symposium. 2001. London, U.K.

[23] OASIS. eXtensible Access Control Markup Language (XACML) version 1.0. 2003 [cited 30 May 2004]; Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[24] der Vlist, E.,: Comparing XML Schema Languages,http://www.xml.com/pub/a/2001/12/12/schemacompare.html (Access on 23th May 2005) (2001)