XML Profile for distributed real time systems

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

[2]*Digital Ecosystems and Business Intelligence Institute,*
Curtin University of Technology,
Perth, Australia
{Tharam.Dillon, Elizabeth.Chang}@cbs.curtin.edu.au

[3] *Database Group,*
Dept. of Computer Science & Technology,
Tsinghua University,
Beijing 100084, China,
fengling@tsinghua.edu.cn

Abstract

In this paper, we describe a XML based profile for modeling the semantics of real time systems. We aim to use Real Time Markup Language (RTML) to provide a comprehensive description of temporal properties for the use in distributed systems communication. RTML is derived from a number of specifications including OMG UML Profile in Schedulability, Performance, and Time. In this paper, we discuss the technique that was used to develop RTML semantic model and the important concepts in RTML.

1. Introduction

XML has been generally accepted as a suitable medium for data representation format for distributed systems. Currently, there are existing standards that models some of the real time properties. However, to effectively utilize the benefits of XML in distributed systems, there must be a consistent schema to capture the important concepts of real time systems.

In order to represent the semantics of real time properties, the schema must meet the following requirements:

1. Ensure the uniformity of the constraints understood by applications which adopts it.
2. Address the properties that are crucial to each systems within the communication channel (e.g. Timing properties, system description, QoS)
3. Effective method that converts data into a form that can be globally readable and understandable (E.g. Handling and descriptions of special data types)

The semantic markup for the real time systems also addresses the following issues:

- Timeliness requirements;
- Resources descriptions (e.g. hardware or software applications);
- Quality of service;
- Schedule of tasks;
- Other functional issues that must be addressed.

In the next section, we will describe the interoperability requirements for data messaging and data storage for XML messaging.

2. Interoperability requirements

One of the key factors of data exchange in the distributed network is consistency. This includes definition of interface and data definition, message format and data storage mechanism. It is important to provide a consistent standard to enable the integration of services [3]. We are particularly concerned with the following features that characterize information dissemination in distributed systems:

1. Collection and transmission of data from sensors to data concentrators

978-0-7695-3132-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ISORC.2008.62

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 [1], 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. Another example would be the implementation of communication between segments of networks with the use of management agent, where messages are exchanged by agent on behalf of network stations.

To achieve this one requires a consistent specification of the data definitions to ensure data integrity and enhance the interoperability among heterogeneous data sources, as depicted in Figure 1. This requires the followings:

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.

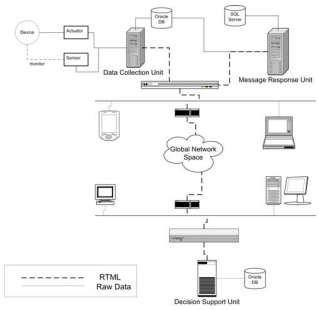

Figure 1 Distributed Communication using XML

3. Related work

A significant number of standards have been developed targets the service integration between distributed real time systems. These can be generally classified into distributed component technologies and protocol design. The former includes the advances with J2EE Web Service support or .NET web service framework.

Earlier work carried out the study of formalism for distributed real time embedded system processes which were based on Algebraic and set based schemes such as Z notation [2], and graphical representations such as state based automata [3], state charts [4, 5], activity diagrams [6, 7] and tabular based representations [3]. These formalisms provide a discipline for software practitioners to construct a description based representation of system analysis and design and also provide a very formal, solid groundwork for formalizing the requirements for DRE systems. Yet, most of these FDL are created for requirement analysis. The specification is not accompanied by the concern for interoperability between different system nor directly executable.

3.1Real Time XML (RT-XML)

RT-XML [8] is an XML based modeling language which describes the distributed real-time multi-media systems. It defines the time ordering, time constraints, probabilistic behavior and qualitative description of a multimedia stream with use of XML. It uses temporal predicates to describe the condition for sets of object based on defined conditions. It describes the application level control over the delivery of multimedia services. RT-XML provides a very descriptive model in defining the temporal elements of multimedia systems; however, we believe there should be a more general purpose profile for use in real time systems.

3.2 Hierarchical QoS Markup Language (HQML)

Hierarchical QoS Markup Language [9] 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 DTDs as the data constraint language.

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

In order to provide such a semantic model, it requires expert knowledge of real time systems and fully understands the constraints of the entities in real time system. In our approach, we adopt existing standards that describes the properties of real time systems.

RTML is derived from a opened specification from Object Management Group, UML™ Profile for Schedulability, Performance, and Time Specification from Object Management Group (SPT) [10] and the UML™ Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms from Object Management Group[11] (QoS Profile). We also investigate the specification developed by ISO and ITU-T document number X.641 Recommendation (ISO/IEC IS 13236) [6].

These standards have provided fundamental constructs for RTML. 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.

4. RTML semantic components

RTML is divided into four main semantic components:

Time—It uses a metric based and time based representation. To model the a sequence of time instant, a metric based time instant or duration can be described in a list of integer or double. Apart from the time granularity extension from the XML Schema, we have also included the illustration of timing mechanisms in our schema which was represented in [12].

Resources—In order to illustrate the resource representation, RTML provides a range of elements for resource description including physical resources such as device, CPU or communication devices; as well as 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 at the local level where the description is attached to the sub-element, or be linked dynamically through the description contained within ResourceDescription and QoSDescription as a header unit associated to the root RTML instance. They are referenced using xPath within the sub-elements with its unique identifier within the message.

QoS—In [13, 14] we have presented the XML Profile for QoS. At this stage we have provided a profile for illustration of QoS concepts for Resource Instances description.

Causality—The description of action execution and resource service instance are described using the CausalityModel. Essentially the CausalityModel is modeled as a container to illustrate the actions involved in a message interaction instance between two application domains. The details of causality components will be discussed in a later section.

5. Abstract elements

RTML uses a XML descriptor based approach.. The modeling constructs in RTML are classified into three corresponding descriptor types including Descriptor, DescriptorMapper or DescriptorCategory. The use of descriptors allows multi level abstraction. Ds and DMs are further defined into a specialized modeling constructs (E.g. QoS Descriptor, Resource Descriptor). This allows a specialization of modeling elements in XML.

Descriptor: Descriptor defines the primitive concept in RTML. It provides the syntactic and semantic descriptor of modeling elements from class and transformed into XML Schema using [15]. Example of Descriptors 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 provides the details of the relationships between Descriptors, or with other DescriptorMappers. A DescriptorMapper 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, which 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.

6. The anatomy of RTML

6.1 Global Timing Mechanism

The timing mechanism is defined in RTML as a global invariant in a RTML instance. 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.

```
<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 2 Global Timing Mechanism

6.2 Domain Type

A domain specifies the details of the parties that are involved in this interaction. It provides general details of the network node including the description, simple network specification and its location. As depict in Figure 3.

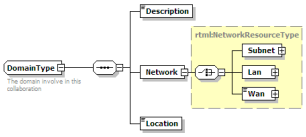

Figure 3 Domain in RTML

6.3 Extended time types

The built-in data and time types from XML Schema are adequate to handle non-real time systems need. However, the timing constraints are different for most real time systems. In some cases the time granularity could be as fine as nanoseconds whilst in others it could be minutes or hours. In RTML, we have proposed a set of timing data types to provide an extension that captures a more specific timing characterization.

Generally the timing properties of RTML can be classified as absolute time (Time Instant) and relative time (duration). Time can be described using metric based measurement with the augmentation of Time Unit to specify its granularity. Each of the Instant 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 3 Time Instant in RTML

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 4 Recurring Metric Duration Type RTML

```
<xs:complexType name="TimeIntervalType">
  <xs:choice>
    <xs:choice>
      <xs:element name="RecurringMetricTime"
type="rtml:RecurringMetricDurationType"/>
      <xs:element name="RecurringTime"
type="rtml:RecurringTimeDurationType"/>
      <xs:element name="RelativeDuration"
type="rtml:RelativeTimeType"/>
      <xs:element name="Duration" type="xs:duration"/>
    </xs:choice>
    <xs:choice>
     <xs:element name="MetricTime" type="rtml:MetricTimeType"/>
     <xs:element name="TimeInstant"
type="rtml:TimeInstantType"/>
    </xs:choice>
  </xs:choice>
</xs:complexType>
```

Figure 5 Time Interval Type

6.4 Resources and the services offered

Resource, in SPT, is defined as a run-time entity that can offer services which can be characterize with the notion of QoS. For instance, one can express the latency of the network connection. One can 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 the resource can offer services. In RTML, resource services are expressed as a functionality offer by the resource. This could be an event generated from a 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 is 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.

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

6.6 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:

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

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

6.7 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 $t_{end.}$

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

This provides the condition that for all the time instant of clock N, if predicate P satisfies the condition that the current time instant t' is within the range of t_{start} and t_{end}, then module Q can fire.

In RTML we do not explicitly describe how an event should behave or the detailed course of execution. We intend to allow flexibility here to be extended to an external source of description. The declaration of event elements can be referenced using an external namespace. Similarly, Resource Service can also be characterized with QoS in the same way as Resource. Here, we specify a number of special events that are interesting to discuss.

6.8 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 $t_{timeout}$ is the value which indicates the timeout, which t' is the time instant when an event is fired and t_{cur} is the current time. We represent timeout in logical equivalence below:

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

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.

6.9 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:

An event which interrupts the currently running process. This interrupt could result in some routine service actions.

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</ResourceDescriptor>
</MemPoll>
```

Figure 6 Polling in RTML

6.10 Causality Model

Causality is the most important concept in RTML as it portrays the relationship between the independent 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 an *interaction* between application domains is described in its corresponding semantic representation through careful modeled transformation into tagged elements. The relationship between these events, are not only represents 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).

The causality model is extracted from the SPT profile. 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. Important concepts are discussed in the following section.

6.11 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 an 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.

6.12 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.

6.13 Exclusive Service Type

Exclusive service is a type of service provided by the Resource Instance (E.g. Physical resources or application resources). It is guided by the control of access policy defined by a collection of primitives. These primitives, as a type of QoSCharacteristic [11, 13, 14], 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 [16, 17].

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</UsedResources>
    <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 7 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 8 Exclusive Service XML example

7 Conclusion

In this paper we demonstrated the XML descriptor based approach to describe the real time semantics in XML. XML is useful to describe real time properties for data communication[18]. In order to create a consistent schema that contains the correct semantics offer by conceptual design model, we have used the object oriented conceptual model transformation approach by [15]. We believe that, this work will bring 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.

Reference

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

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

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

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

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

[6] 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

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

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

[9] 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)

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

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

[12] 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)

[13] 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)
[14] 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)
[15] 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)
[16] 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.
[17] 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.
[18] der Vlist, E., Comparing XML Schema Languages,
http://www.xml.com/pub/a/2001/12/12/schemacompare.html (Access on 28th December 2007) (2001)