Representing real time semantics for distributed application integration

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

Traditional real time system design and development are driven by technological requirements. With the ever growing complexity of requirements and the advances in software design, the alignment of focus has gradually been shifted to the perspective of business and industrial needs. This paper discusses a new paradigm in developing distributed real time system with the emphasis of knowledge representation and interoperability using XML. Real Time Markup Language (RTML) aims to provide a basis for interoperability.

1. Introduction

In the past, most of the real time applications run in specialized devices or hardware as a part of engineering solution to fulfill complex computational task. Over time, the application domain of real time systems has been extended to a wider spectrum. Today, soft real time system can be found running in enterprise server. Both of these types of systems share a common feature, timeliness. The only differences between these two are the way how the system responds to the overdue deadline. Hard real time systems, as the former type are most likely to be found, needs to face the absolutely correct timeliness, the past of deadline is not an option or the consequences could be catastrophic; whilst a soft real time system might accept the occasional misses of deadlines.

Today, businesses strive to become a winner in a competitive market. The urge of providing reliable, timely service is one of the main priorities, so timeliness plays an important role. For instances, an online ticketing systems are required to serve hundreds of customers in every seconds when a major event comes start the sales. During the peak time, the online ticket purchase system would need to handle the user loads as much as tens of thousands per seconds. Systems will need to provide a robust service for 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.

2. Problem

The advances in software design technologies allow a more general level of abstraction. Software design models can be directly translated into implementation using computer-based automation. However, in many cases, the application of these tools are only limited at application code transformation. The knowledge capture during design fails to translate into useful data representation format, or in other words, there are no direct benefit resulted from the initial design to data semantics. The broken link between design and data definition could cause the loss of semantics. This could cause the problem in the long run as the change in design could not eventually be represented to the data layer until the application code is changed.

Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)
0-7695-2511-3/06 $20.00 © 2006 IEEE

To address this, it requires a framework that can transform the semantics from the conceptual design and apply to data. In this paper we present Real Time Markup Language (RTML), a XML based solution for transforming the design time class model into XML instances. In this paper, we will discuss the importance of interoperability between real time systems, our model driven design and the details of RTML.

3. Interoperability between real time systems

The diversity in system architecture and component has resulted in heterogeneity. 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. It is important to provide a consistent standard to ensure a smooth implementation of service collaboration, since data contains the abstract from the logic of the domain [1]. 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. 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.

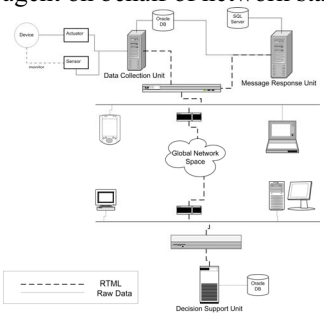

Figure 1 Distributed Communication using XML

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 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.

XML is ideal in this case. XML documents representing the application logic is clearly defined by XML schema, this enable real time data exchange across multiple platforms, which individual systems could map the data defined by the context within the internal application to a mutually understandable vocabulary.

4. Approach

In previous paper [3, 4, 5] we discussed the use of model transformation in obtaining data instances. In this section we will discuss a new approach in mapping conceptual model into RTML instances. As depicted in Figure 2.

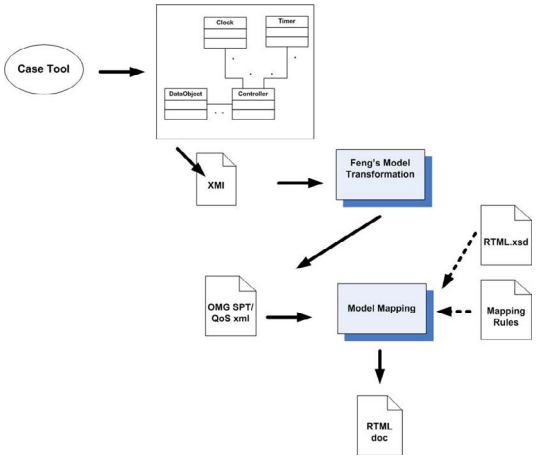

Figure 2 Model driven data generation

4.1 OMG SPT and XMI

RTML is derived based on a number of specification including specification from Object Management Group, UML™ Profile for Schedulability, Performance, and Time Specification from Object Management Group (SPT hereafter) [5] and the UML™ Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms from Object Management Group[7] (QoS Profile hereafter). Our approach targets design that is based on the use of class models using these profiles. The widespread use of UML provides a solid ground for this approach to be practical. In particular, modeling in class diagram is extremely useful in modeling Object Oriented systems as the modeling notation can clearly visualize the relationship between entities. This importance is even highlighted when it applies to real time systems [8]

Models developed in SPT/QoSProfile can be exported in to XML Metadata Interchange (XMI) [9, 10], which represents metadata information in XML. It can be used for any metadata whose metamodel can be expressed in

Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)
0-7695-2511-3/06 $20.00 © 2006 IEEE

Meta-Object Facility (MOF). MOF-based models (such as Class Diagram in UML) can be transformed in XMI format. Once models are exported from diagram format to XMI, we need to extract the knowledge from the XMI document. Previous work [11, 12] has shown the model transformation from UML class model into XML data instances. This transformation will be the first step of RTML model transformation.

The next step will be transforming a SPT XML document into a RTML document. SPT models describe the design time details involve in a component. Since the details in a SPT model are too abstract to be directly usable at in the application, it requires further details to be included (E.g. identifier for an instance) for data exchange over distributed applications. This process involves the mapping from SPT syntax to RTML syntax. In the following section, we will describe some of the important concepts in RTML.

5. RTML Structure

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

Time --- RTML provides a detailed definition of time. It provides a metric based and time based representation. A metric based time instant or duration representation 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 [3]

Resources --- In order to illustrate the resource representation, RTML provides a wide range of primitives 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.

QoS --- In [4, 5] 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 acts as a container to illustrate the actions involved in a message interaction instance between two application domains.

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 3 Global Timing Mechanism

6.3 Extended time types

As mentioned previously, many of the existing XML profiles do not cater for general timing properties. The primitive built-in types from XML Schema are normally adequate to handle most business transactions. However, the timing constraints could be different for real time systems depending on the application domain. 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 collection of timing standards in order to provide an extension that captures a more specific characterization of timing description for XML elements.

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 4 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 5Recurring 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 6Time 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:

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

Which 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 $t_{end.}$

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

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:

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

$$\wedge (t_{start} \le t' \wedge t' \le t_{end}) : P \rightarrow Q$$

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.

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 7 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).
SPT has provided detailed description on the modeling elements of causality 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 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.

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 [7, 4, 5], 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 [13, 14].
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 8Acquire 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 9Exclusive Service XML example

Conclusion

In this paper, we discuss the use of XML in representing the semantics of soft real time application in distributed network. We proposed a XML based solution, which

allows models develop in UML CASE based tools be directly translated into RTML based solution using model transformation method and rule mapping mechanism. We have also explained the structure of RTML in this paper. RTML is derived from OMG SPT since UML is widespread and it has become one of the common standards in software design. RTML aims to provide interoperability for real time systems. By providing such a XML profile, it establishes a knowledge based for organizations to exchange data using XML messaging.

Reference

[1] 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

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

[3] 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)

[4] 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)

[5] 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)

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

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

[8] Douglass B. P., Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns. Addison-Wesley Professional, 1999

[9] Lopes, S., Silva C., Tavares, A., Monteiro J.: Extending ArgoUML for Real Time UML, in Proceedings of the IASTED International Conference Advances in Computer Science and Technology, November 2004, US Virgin Island

[10] Artisan Real-Time Studio. http://www.artisansw.com/

[11] 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)

[12] Kudrass T., Krumbein, T.: Rule-Based Generation of XML DTDs from UML Class Diagrams: Advances in Databases and Information Systems, LNCS 2798, 2003

[13] 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.

[14] 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.