



<http://www.rti.com>

Tools and techniques for monitoring real-time distributed applications

Gerardo Pardo-Castellote, CTO

Jens Pillgram-Larsen, Tools Lead

**The Real-Time
Middleware Experts**

OMG Real-time, Embedded and Enterprise-Scale
Time-Critical Systems workshop,
May 2010

Agenda

- **The Challenge**
- Infrastructure Monitoring
- Application Monitoring
- Future directions

Challenge

- Distributed systems are becoming larger...
 - composed of many independently-developed applications
 - by nature deployed on a network
- Understanding and management is a significant challenge
 - During System Integration/Testing
 - At deployment time
- Instrumentation and Monitoring are the key technologies that can address this:
 - What is running? Where?
 - How is it performing?
 - What is the application doing?
- Heisenberg challenge:
 - Do this without impacting the application

Example: DDS Systems

- DDS systems are decoupled and distributed
 - Dynamic
 - Highly scalable
 - Peer-to-peer
 - Real-Time

There is no central truth

Dimensions

Two dimensions to monitoring:

- Infrastructure
 - Middleware (e.g. DDS), Network, Operating system, Processor
- Application
 - Health
 - Internal state
 - Data
- These are not independent! System understanding requires correlation of all this data...
 - My tracking system is failing because is getting delayed data because network packets are dropped because the system is busy, because ...

Agenda

- The Challenge
- **Infrastructure Monitoring**
- Application Monitoring
- Future directions

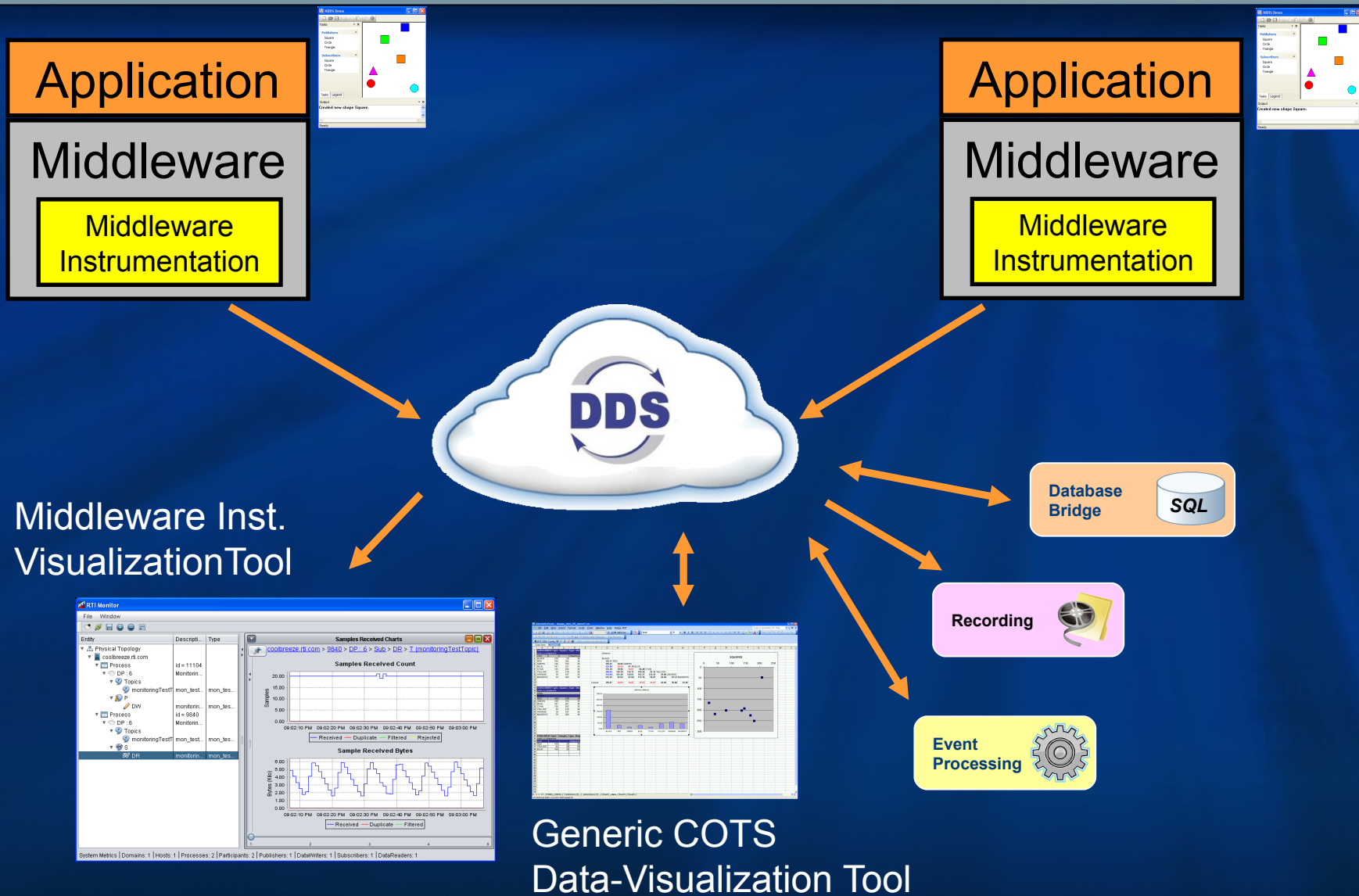
Monitoring Distributed Systems

- Faults come in many flavors
 - Hard faults (crash, hang etc.)
 - Violation of real-time constraints
- Hard to establish causality and chain of events
 - Can't single step a distributed systems
 - Changing timing often changes event
 - Fault is often not reproducible
- Monitoring requires centralized access to distributed system state
 - For visualization
 - For regression against a baseline

Non-functional requirements

- Robust to network partitioning
- Minimally intrusive in resources (CPU, Memory, Network)
- Scalable to many applications
 - Traditional techniques such as SNMP not scalable
- Support for multiple consumers of the data
 - Recording
 - Logging
 - Visualization
 - Algorithmic supervision

Solution: Data-Centric Monitoring: Use DDS for Infrastructure Instrumentation



Benefits

- Leverage DDS performance and scalability to distribute collected data
- Leverage Discovery and Pub-Sub
 - Information available to all consumers
- Use Data-Centric model for Collected Data
 - Leverage DDS Data-Centric Qos
 - History to keep recent-values if each metric
 - Time filters to enabled sub-sampling
 - Deadline/Liveliness to monitor availability

Example: Monitoring DDS Middleware

DDS itself is an example infrastructure which can be monitored

- Data-Model based on DDS Status model
- Additions for:
 - DDS Entity resource usage
 - DDS-RTPS Wire Protocol entity model
 - RTPS Writer & Reader Statistics
 - RTPS Statistics
- Each DDS or RTPS entity becomes a separate data-object in the model

Example: Monitoring DDS Status

RTI Data Distribution Service Monitor

File Window

Expand All Collapse All

Entity	Descrip...	Type
Physical Topology		
coolbreeze.rti.com		
Process	id = 8016	
DP : 0	RTI Sh...	
Topics		
Square	ShapeT...	ShapeType
Circle	ShapeT...	ShapeType
Triangle	ShapeT...	ShapeType
P		
DW	Square	ShapeType
DW	Circle	ShapeType
DW	Triangle	ShapeType
Process	id = 7456	
DP : 0	RTI Sh...	
Topics		
Square	ShapeT...	ShapeType
Circle	ShapeT...	ShapeType
Triangle	ShapeT...	ShapeType
S		
DR	Square	ShapeType
DR	Circle	ShapeType
DR	Triangle	ShapeType
Process	id = 7300	
DP : 0	RTI Sh...	
Topics		
Square	ShapeT...	ShapeType
Circle	ShapeT...	ShapeType
Triangle	ShapeT...	ShapeType
P		
DW	Square	ShapeType
DW	Circle	ShapeType
DW	Triangle	ShapeType
Process	id = 7072	
DP : 0	RTI Sh...	
Topics		
Square	ShapeT...	ShapeType
Circle	ShapeT...	ShapeType
Triangle	ShapeT...	ShapeType
P		
DW	Square	ShapeType
DW	Circle	ShapeType
DW	Triangle	ShapeType

DataReader Status

DataReader Builtin Key: c0a80104.1d20.1.80000807

Requested Incompatible QoS Status

Total Count: 2 (Δ 0)

Last Policy Id: 4

Policies: id = Deadline, count = 2

Requested Deadline Missed Status

Total Count: 33 (Δ 1)

Last Instance Handle: 76 cc d5 cd 13 41 d7 74 98 28 d4 3e c1 43 2a ca

Liveliness Changed Status

Alive Count: 1 (Δ 0)

Not Alive Count: 0 (Δ 0)

Last Publication Handle: c0 a8 1 4 0 0 1f50 0 0 0 1 80 0 40 2

Subscription Matched Status

Total Count: 2 (Δ 0)

Current Count: 1 (Δ 0)

Last Publication Handle: c0 a8 1 4 0 0 1f50 0 0 0 1 80 0 40 2

DataReader Cache

Sample Count: 2

Sample Count Peak: 2

Cache Samples Mean: 0.0

Cache Samples Minimum: 0

Cache Samples Maximum: 0

Cache Samples Variance: 0.0

Sample Lost Status

Total Count: 0 (Δ 0)

Sample Rejected Status

Total Count: 0 (Δ 0)

Last Reason: NOT_REJECTED

Last Instance Handle: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Sample Statistics

Received Count: 1,135 (Δ 0)

Received Bytes: 122,580 (Δ 0)

Duplicates Count: 0 (Δ 0)

Duplicates Bytes: 0 (Δ 0)

Filtered Count: 0 (Δ 0)

Filtered Bytes: 0 (Δ 0)

Rejected Count: 0 (Δ 0)

Heartbeat Statistics

Heartbeat Count: 37 (Δ 0)

Heartbeat Bytes: 1,184 (Δ 0)

Gap Count: 600 (Δ 16)

Gap Bytes: 19,200 (Δ 512)

Acks Sent Count: 36 (Δ 0)

Acks Sent Bytes: 1,008 (Δ 0)

Nacks Sent Count: 2 (Δ 0)

Nacks Sent Bytes: 88 (Δ 0)

System Metrics | Domains: 1 | Participants: 4 | Publishers: 3 | DataWriters: 21 | Subscribers: 1 | DataReaders: 3

DDS Topics used to monitor DDS itself

- **EntityDescription** (QoS and other static information, published on creation, deletion and when QoS changes)
 - DomainParticipant
 - Publisher
 - Subscriber
 - Topic
 - DataReader
 - DataWriter
- **EntityStatistic** (Status, aggregated status and statistics, published periodically)
 - DomainParticipant
 - Topic
 - DataReader
 - DataWriter
 - DataReader with matched DataWriter
 - DataWriter with matched DataReader
 - DataWriter with matched locator

Agenda

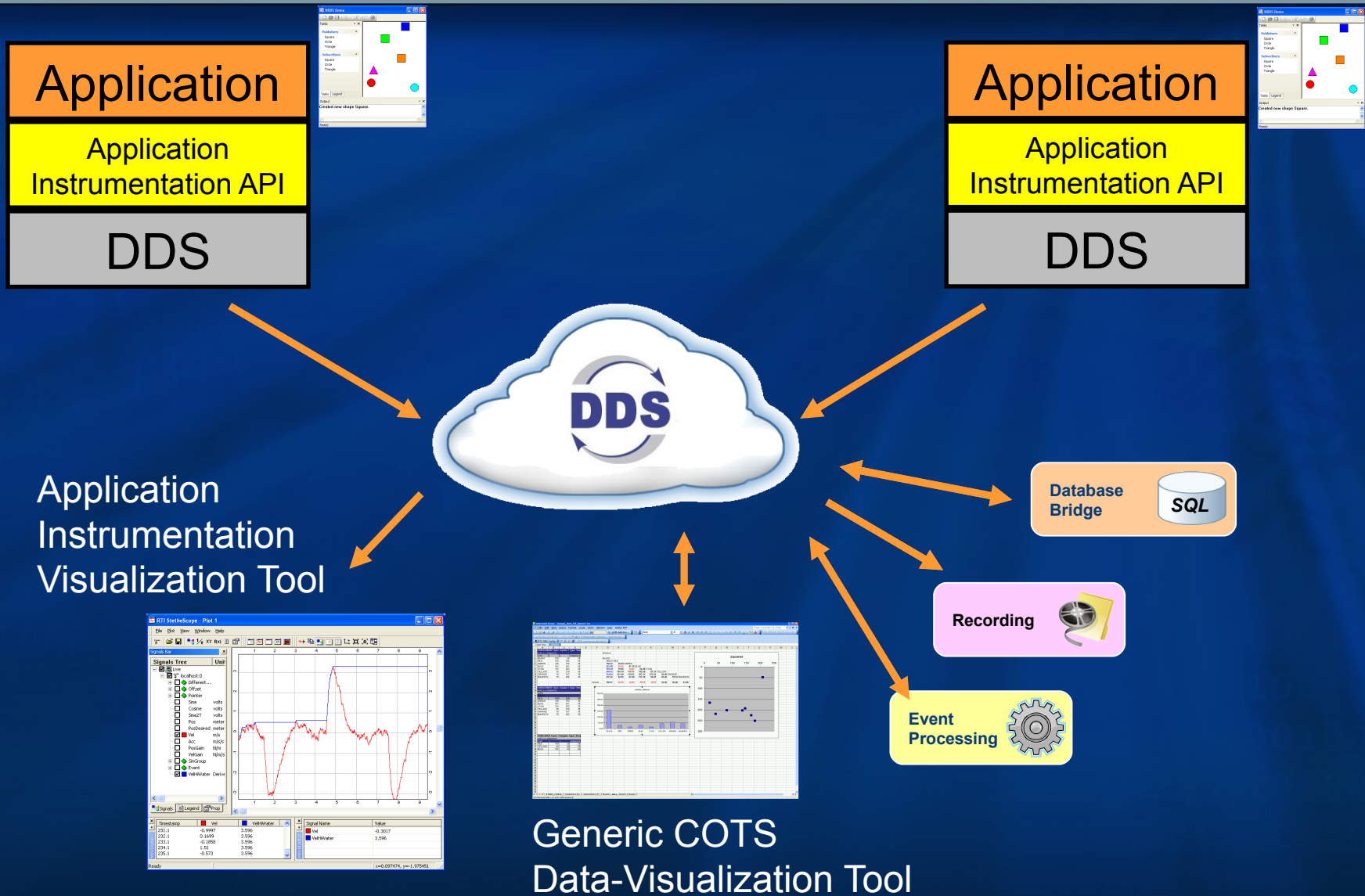
- The Challenge
- Infrastructure Monitoring
- **Application Monitoring**
- Future directions

Challenges of Application Instrumentation

- Access to application-specific data from remote locations
- Minimal intrusiveness
 - CPU, Memory, Network
- Access from many points
 - Live monitoring / HMIs
 - Logging
 - Real-Time analysis...

Same non-functional requirements than Infrastructure Instrumentation!

Solution: Use DDS for distribution of instrumented data

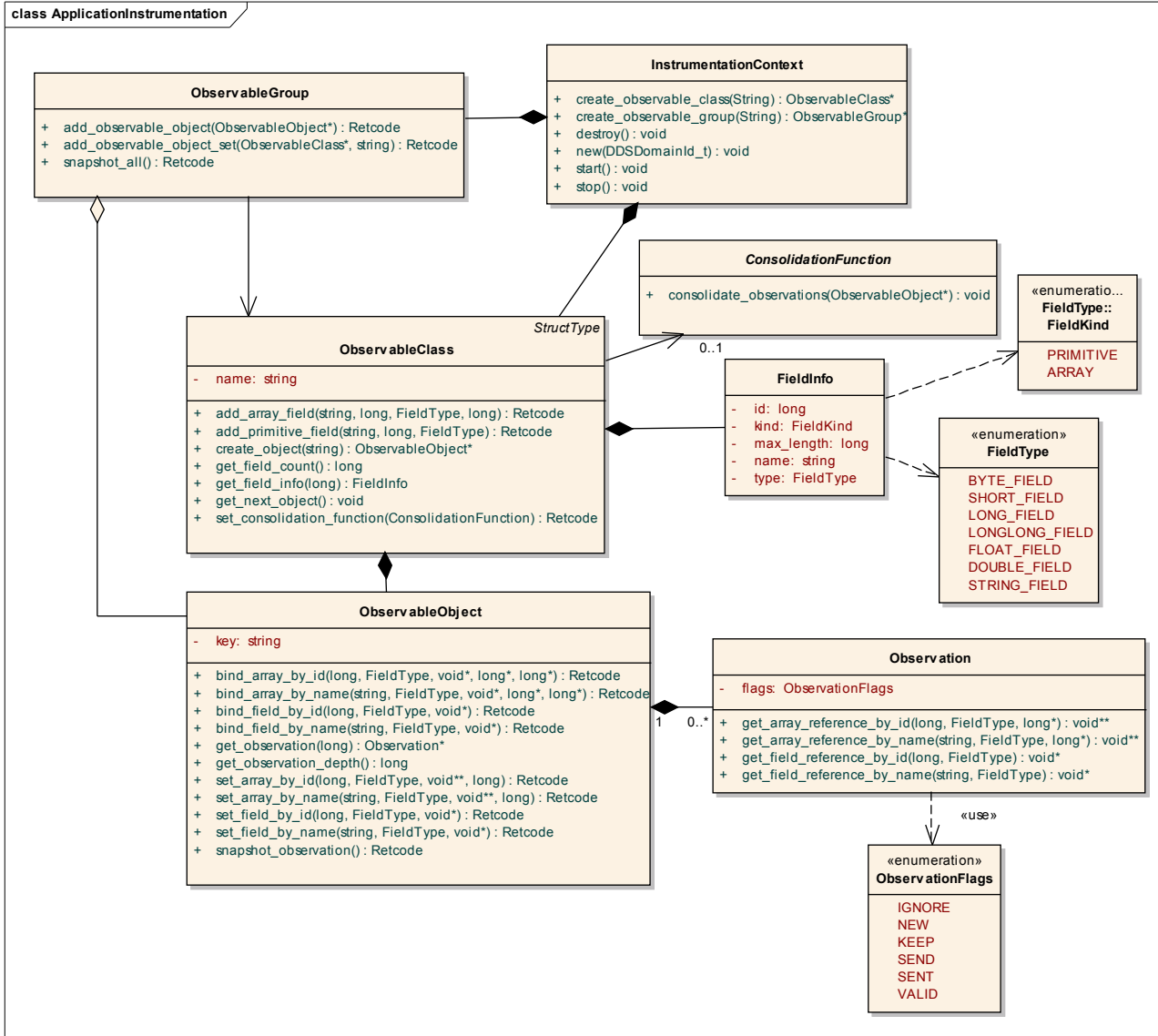


Two aspects

- Instrumentation API
 - API used by developers to instrument their application
 - “the super printf()”

- Instrumentation Data-Model
 - How is the instrumented data modeled in DDS?

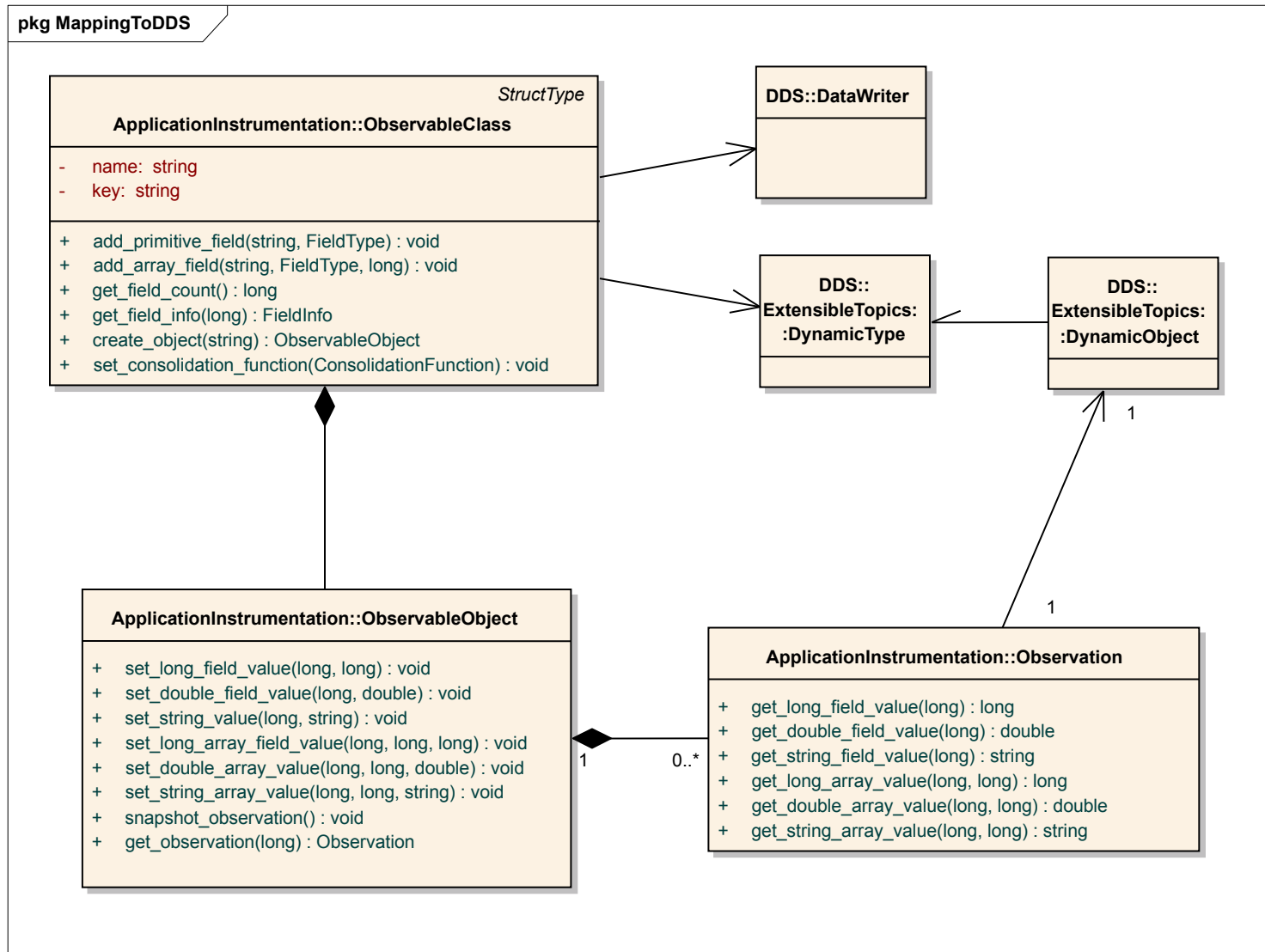
Instrumentation API



Why a different API and not just DDS?

- More focused on the instrumentation task
 - Lower learning curb
 - More easily adopted
- Enforces Instrumentation Data Model
- Enforces QoS sensible for instrumentation
- No need for compile-time type declarations or code-generation
 - Underneath it leverages DDS-X-TYPES spec.

Mapping to DDS



Two categories of Observations

- Periodic Observations
 - Sent at a specific, configured period
- Event-based Observations
 - Sent when “something interesting” happens
 - E.g. an observation exceeds a threshold
 - An observation changes “significantly” from the previously-reported value

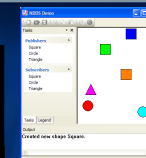
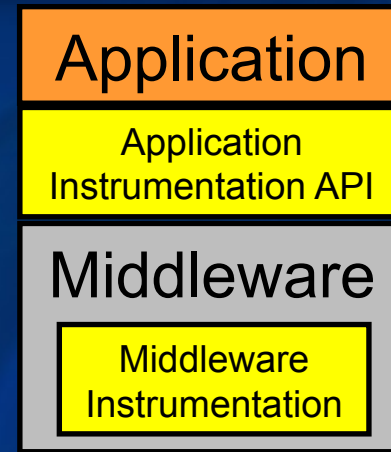
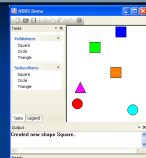
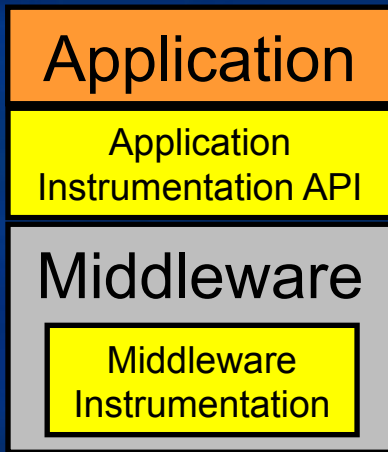
DDS QoS for Periodic Observations

<i>Qos Policy</i>	<i>Value</i>	<i>Reason</i>
<i>RELIABILITY</i>	<i>BEST_EFFORT</i>	<i>It is sufficient to send the collected information best efforts because it is being collected periodically.</i>
<i>HISTORY</i>	<i>KEEP_LAST</i>	<i>Since data is being continually produced the system only needs to keep the most recent values</i>
<i>OWNERSHIP</i>	<i>EXCLUSIVE</i>	<i>At any one time there can only be one application publishing the instrumentation data of any given ObservableObject</i>
<i>DEADLINE</i>	<i>3X periodic rate</i>	<i>DEADLINE can be used to detect that a specific ObservableObject is no longer being produced. It should be set to match the periodicity of the disseminated observations.</i>
<i>DURABILITY</i>	<i>VOLATILE</i>	<i>There is no need to save data for late joiners because the fresh data will soon be produced</i>

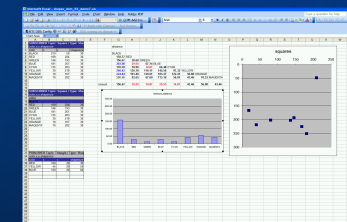
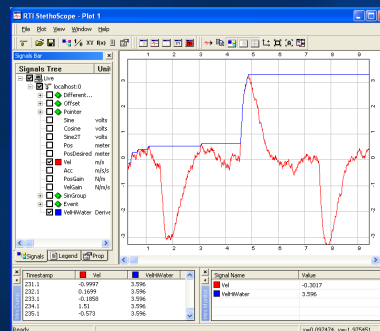
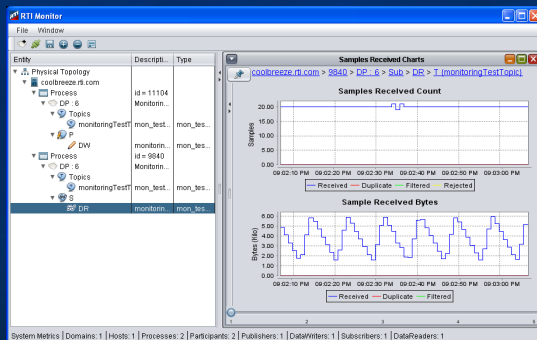
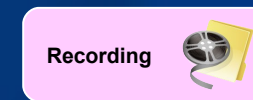
DDS QoS for Event-Based Observations

<i>Qos Policy</i>	<i>Value</i>	<i>Reason</i>
<i>RELIABILITY</i>	<i>RELIABLE</i>	<i>Reliable communications are needed because the Observations are not updated continuously and losing one could cause the receivers to not receive some important change or state update and potentially never find out the correct values.</i>
<i>HISTORY</i>	<i>KEEP_LAST</i>	<i>Ensure that the system retains the last “history depth” Observations for each ObservableObject</i>
<i>OWNERSHIP</i>	<i>EXCLUSIVE</i>	<i>At any one time there can only be one application publishing the instrumentation data of any given ObservableObject</i>
<i>DEADLINE</i>	<i>INFINITE</i>	<i>Since Observations are not being sent continually published we cannot guarantee any deadline.</i>
<i>DURABILITY</i>	<i>TRANSIENT_LOCAL</i>	<i>Observations should be retained for late joiners because otherwise they may not receive data until the next significant event that causes the Observation to be published occurs. Depending on the system and ConsolidationFunction this might take an unbounded time.</i>

Combined Solution: Middleware + Application Instrumentation



Middleware Inst. Visualization Tool



Application Inst. Viz Tool

Generic COTS

Agenda

- The Challenge
- Infrastructure Monitoring
- Application Monitoring
- **Future directions**

Proposal: Application Instrumentation RFP

Scope of the RFP

- Generic Instrumentation API
 - Allows applications to instrument themselves and define the relevant internal state / measures to monitoring
 - This is application dependant
- Monitoring Data Model for different middleware platforms: DDS, CORBA, ...
 - What internal state must the middleware report to understand its health
- Mapping of the monitoring data to standard middleware (e.g. DDS) so that the information can be remotely accessed by recording and HMI tools.

Proposed approach

- PIM on Application Instrumentation (AI)
- Three kinds of PSMs:
 - Application Instrumentation PSMs (AI-PSM)
 - Middleware Instrumentation PSMs (MI-PSM)
 - Middleware mapping of AI-PSM and MI-PSM
- AI-PSM
 - Concrete Platform/Language API to instrument applications
 - Mapping to middleware to access the collected data on a distributed system
- MI-PSM
 - What information should be collected on each middleware platform

AI-PSM Example:

- API that allows an application to define what data to collect and when:
 - Events
 - InstrEventDeclare("Event Name", EventType, ...)
 - InstrEventNotify("Event Name", EventValue)
 - Continuous data.
 - InstrMeasureDeclare("Measure Name", MeasureType, ...)
 - InstrMeasuresCollect("Measure Name", MeasureValue);
 - Triggers
 - InstrTriggerDeclare("Measure Name", TriggerExpression)

```
InstrTriggerDeclare("HostileTrackCount",  
                    "track_count > 1000");
```

MI-PSM Example:

- For DDS define relevant instrumentation data
 - Samples/sec received by a DataReader
 - Number of samples in a DataWriter cache
 - Number of NACKs received by a DataWriter
 - Number of Data samples that are un-acknowledged in a DataWriter Cache
 - ...

Middleware Mapping of AI-PSM and MI-PSM

- How is the Instrumentation API realized in terms of underlying data-model and how it is made available to other applications
 - E.g. map collected data to DDS Types and Topics
- How is the Data-Collected from the middleware modeled and distributed
 - E.g. map middleware data to other DDS Types and Topics
- NOTE: The middleware used for this PSM need not be the same as for the MI-PSM
 - E.g. it is possible to define a MI-PSM for CORBA and distributed the collected data via DDS.

Future Directions

- Monitor non-DDS system resources
 - e.g. CPU, memory, network
- Network topology with alerts
- Real-Time DDS Latency and Throughput statistics
- Integration with COTS tools
 - i.e. HP Openview

Thank You!

Q & A

