



# ENTERPRISE ARCHITECT

SPARX  
SYSTEMS

# Realizing Data Distribution Services through MDA

Sam Mancarella  
Chief Technical Officer  
Sparx Systems



# Overview

- **Model-Driven Architecture**
- **UML Extensibility**
- **Example DDS Realization**
- **Other Applications**
- **Concluding Remarks**

# MDA – What is it?

## ● **Model-Driven Architecture**

- Models to ‘define’ architecture  
Use Case models, Interaction Overviews, Requirements
- Models to ‘describe’ architecture  
Class models, Activity models, Deployment models
- Models to ‘drive’ architecture  
One model to influence another (automated)

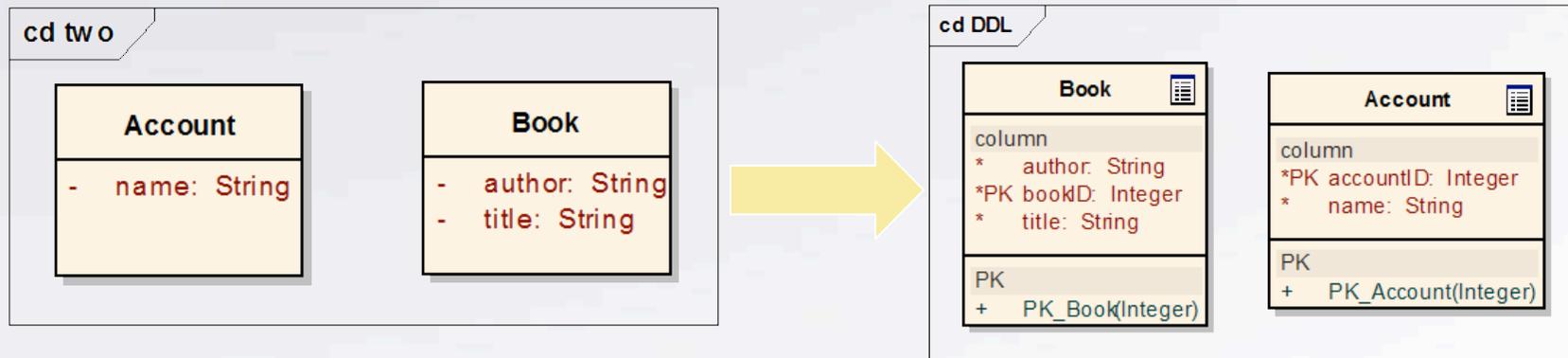
## ● **Longitudinal versus Latitudinal MDA**

- ‘Across’ modeling languages  
BPMN, UML, SysML, CWM
- ‘Within’ a modeling language  
UML Use Case, UML Class

**Apply Model  
Transformation**

# MDA – What is it?

- Design abstract services/design model, independent of the underlying technology
  - Platform Independent Model (PIM)
- Generate concrete services/design model, specific to the underlying technology
  - Platform Specific Model (PSM)



Automated  
Transformation

# MDA – Why is this important to DDS?

- **Describe a DDS system in an abstract design (PIM)**
  - DDS Domain
  - DDS Subscribers & Publishers
  - DDS Topics
  - Application, or system logic
- **Automatically generate the implementation (PSM)**
  - C++, Java, C Class models
  - Vendor-specific FreeDDS, RTI DDS, Thales DDS

# MDA – Why is this important to DDS?

## ● Benefits

- Reduces Complexity  
Application logic separate from implementation
- Reduces Design cycle time  
DDS design change can more easily be propagated down to implementation by retransforming PSM
- Increases Design Maintainability  
Segregation of logic design and implementation easier to troubleshoot

# UML Extensibility

## UML Profiles

- Provide a domain-specific taxonomy to UML
- Define stereotypes that apply to various UML constructs
- Define domain-specific semantics to those constructs
  - Properties (Tagged Values)
  - Constraints (human, machine)
- Define the appearance of constructs

## We can define a UML Profile for DDS

# UML Extensibility

## ● Benefits

- **Standardizes Nomenclature**  
Domain, Topic, Subscriber as DDS profile constructs
- **Standardizes Semantics**  
A DataReader can only read one topic
- **Promotes Industry Standardization**  
Throughout RealTime-embedded community. DDS designers, consultants across vendors, implementations, geographies all share a common domain model
- **Maximizes Communication**  
A DataReader can only read one topic

# Example DDS Realization

## ● “Hello World” Example

- Simple example of 1 Topic, 1 Publisher, 1 Subscriber
- Modeled using a UML Profile for DDS

## ● DDS Application Design

- How DDS Entities are defined and bound to each other
- How QoS Policies are allocated to entities

## ● Implementation

- How the DDS design is used as a PIM to generate a PSM
- C++ Domain Class model
  - Data Dictionary
  - Factory
  - QoS
- Generate to executable code

# Example DDS Realization

Class Diagram: "DDS" created: 30/03/2006 9:05:53 AM modified: 30/03/2006 9:06:23 AM 100% 827 x 1169

Project View

- DDS
- DDS
- DDS
- «DDS\_Domain» HelloWorldDomain
- HelloWorldDomain

Properties

Package Settings

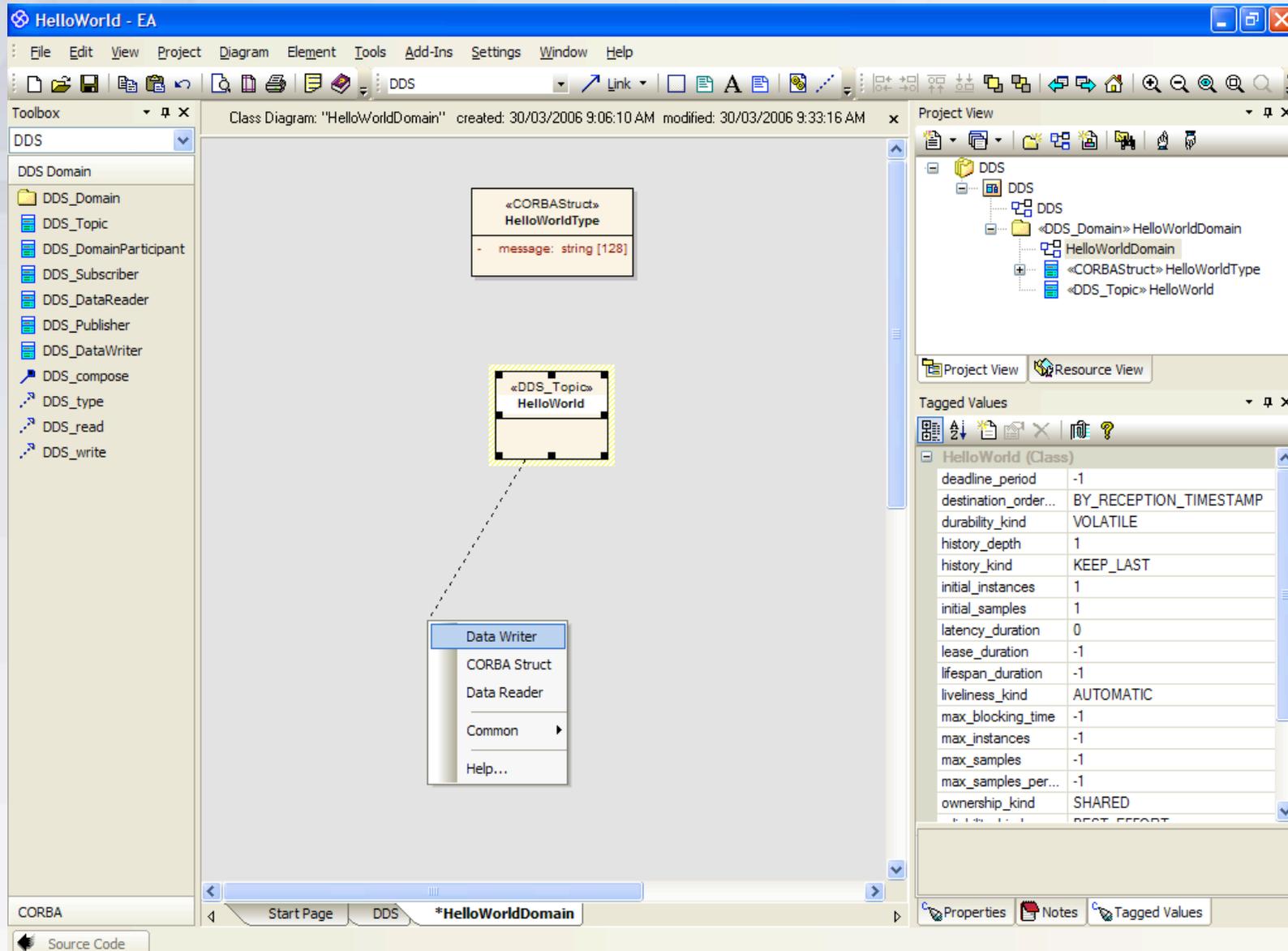
Name	HelloWorldDomain
Scope	Public
Type	Package
Stereotype	DDS_Domain
Alias	
Complexity	Easy
Version	1.0
Phase	1.0
Language	Java
Filename	

Project

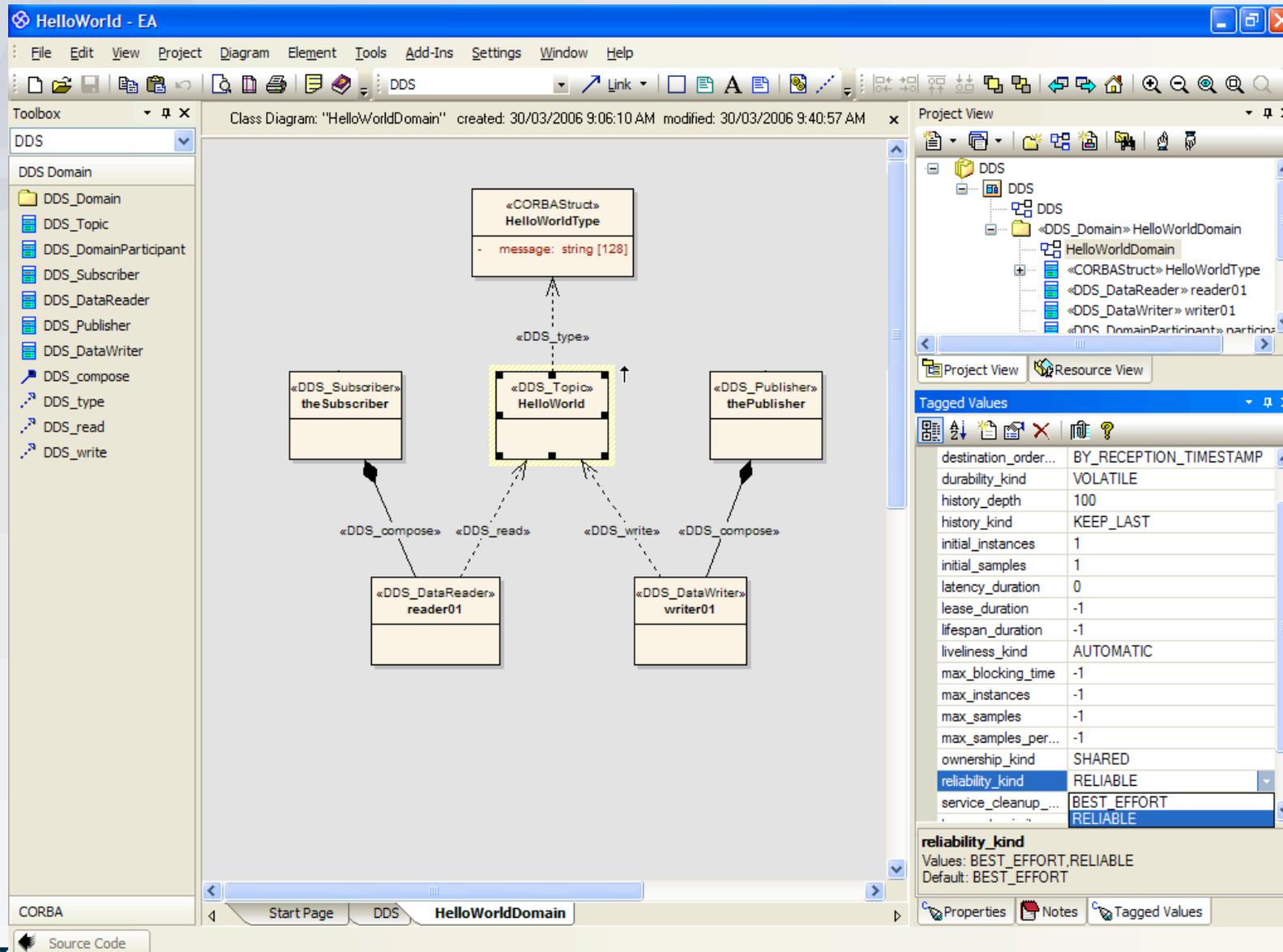
Advanced & Settings

Properties Notes Tagged Values

# Example DDS Realization



# Example DDS Realization



# Example DDS Realization

The screenshot displays the Sparx Systems Enterprise Architect (EA) interface. The main window shows a Class Diagram for "HelloWorldDomain" with elements like «DDS\_DomainParticipant», «CORBAStruct», and «DDS\_DomainParticipant». A "Model Transformation" dialog is open, showing a list of elements to be transformed and a list of target packages.

**Model Transformation Dialog:**

Element Name	Element Type
HelloWorld	Class
HelloWorldType	Class
participant01	Class
participant02	Class
reader01	Class
thePublisher	Class
theSubscriber	Class
writer01	Class

Name	Target Package
<input type="checkbox"/> C#	
<input type="checkbox"/> DDL	
<input type="checkbox"/> EJB Entity	
<input type="checkbox"/> EJB Session	
<input type="checkbox"/> Java	
<input type="checkbox"/> JUnit	
<input type="checkbox"/> NUnit	
<input checked="" type="checkbox"/> RTIDDS_C++	DDS
<input type="checkbox"/> WSDL	
<input type="checkbox"/> XSD	

Buttons: All, None, Include Child Packages, Generate Code on result, Perform Transformations on result, Intermediary File (optional for debugging only), Do Transform, Close, Help, Write Always, Write Now.

# Example DDS Realization

The screenshot displays an IDE window titled "HelloWorld - EA" showing a class diagram for "HelloWorldDomain\_PSM". The diagram includes three main classes: "theSubscriber", "thePublisher", and "HelloWorldDomainFactory".

- theSubscriber**:
  - Methods: `shutdown(DDSDomainParticipant, DDSubscriber, DDSTopic, DDSDataReader) : int`, `run(int, int, int) : int`
- thePublisher**:
  - Methods: `shutdown(DDSDomainParticipant, DDSPublisher, DDSTopic, DDSDataWriter) : int`, `run(int, int, int) : int`
- HelloWorldDomainFactory**:
  - Attributes: `- _dataDictionary: HelloWorldDomainDataDictionary*`
  - Methods: `HelloWorldDomainFactory()`, `create_topic(DDSDomainParticipant*, char*, DDSTopicListener*, int) : DDSTopic*`, `create_reader(DDSDomainParticipant*, DDSubscriber, DDSTopic, char*, DDSDataReaderListener*, int) : DDSDataReader*`, `create_writer(DDSDomainParticipant*, DDSPublisher, DDSTopic, char*, DDSDataWriterListener*, int) : DDSDataWriter*`, `create_publisher(DDSDomainParticipant*, char*, DDSPublisherListener*, int) : DDSPublisher*`, `create_subscriber(DDSDomainParticipant*, char*, DDSubscriberListener*, int) : DDSubscriber*`, `create_participant(int, int, char*, DDSDomainParticipantListener*, int) : DDSDomainParticipant*`

Below these is the **HelloWorldDomainDataDictionary** class:

- Attributes: `- topiclist[NUM_TOPICS]: TopicInformation*`, `- readerlist[NUM_READERS]: DataReaderInformation*`, `- writerlist[NUM_WRITERS]: DataWriterInformation*`, `- publisherlist[NUM_PUBLISHERS]: PublisherInformation*`, `- subscriberlist[NUM_SUBSCRIBERS]: SubscriberInformation*`, `- participantlist[NUM_PARTICIPANTS]: DomainParticipantInformation*`
- Methods: `HelloWorldDomainDataDictionary()`, `~HelloWorldDomainDataDictionary()`, `getTopicQosAndType(char*) : QosTypePair*`, `getDataReaderQos(char*) : QosDataReaderPair*`, `getDataWriterQos(char*) : QosDataWriterPair*`, `getPublisherQos(char*) : QosPublisherPair*`, `getSubscriberQos(char*) : QosSubscriberPair*`, `getDomainParticipantQos(char*) : QosDomainParticipantPair*`, `CreateLists() : void`, `DeleteLists() : void`, `getTopicQosAndType(char*, int, int) : QosTypePair*`

The Project View on the right shows a tree structure for the "DDS" project, with "HelloWorldDomain\_PSM" selected. The tree includes sub-projects like "QoS", "Facility", and various classes such as "CommonListener", "DataReaderInformation", "DataWriterInformation", "DomainParticipantInformation", "HelloWorldDomainDataDictionary", "HelloWorldDomainFactory", "HelloWorldTypeListener", "PublisherInformation", "QosDataReaderPair", "QosDataWriterPair", "QosDomainParticipantPair", "QosPublisherPair", "QosSubscriberPair", "QosTypePair", "SubscriberInformation", and "TopicInformation".

# Example DDS Realization

The screenshot displays an IDE window titled "HelloWorld - EA" with the following components:

- Toolbox:** Lists various DDS Domain components such as DDS\_Domain, DDS\_Topic, DDS\_DomainParticipant, DDS\_Subscriber, DDS\_DataReader, DDS\_Publisher, DDS\_DataWriter, DDS\_compose, DDS\_type, DDS\_read, and DDS\_write.
- Code Editor:** Shows the implementation of `HelloWorldDomainDataDictionary::CreateLists()` in `HelloWorldDomainDataDictionary.cpp`. The code configures a `topicQos0` with various QoS parameters, including `DDS_RELIABLE_RELIABILITY_QOS` for reliability. It also creates a `TopicInformation` object and sets its name to "HelloWorld" and type to "HelloWorldType".
- Project View:** Shows a hierarchical tree structure of the project, including folders like `DDS`, `HelloWorldDomain_PSM`, and `QoS`, and various implementation files.
- Tagged Values:** Shows the `HelloWorldDomainDataDictionary (Class)` with its properties.

# Example DDS Realization

Microsoft Visual C++ [design] - theSubscriber.cpp

File Edit View Project Build Debug Tools MainWin Window Help

Debug MyDomainSubscriberQos

C:\DDSDesign\objs\i86Win32VS2003\HelloWorld\_publisher.exe

```
Writing myMyHello, count 0
Writing myMyHello, count 1
Writing myMyHello, count 2
Writing myMyHello, count 3
```

Solution Explorer - HelloWorld\_subscriber

- Solution 'HelloWorld' (2 projects)
- HelloWorld\_publisher
  - References
  - Source Files
  - Header Files
  - Resource Files
- HelloWorld\_subscriber
  - References
  - Source Files
    - \_subscriber\_main.cxx
    - CommonListener.cpp
    - DataReaderInformation.cpp
    - DataWriterInformation.cpp
    - DomainParticipantInformation.cpp
    - HelloWorldDomainDataDictionary.cpp
    - HelloWorldDomainDataReaderQos.cpp

C:\DDSDesign\objs\i86Win32VS2003\HelloWorld\_subscriber.exe

```
message [105]: '''
message [106]: '''
message [107]: '''
message [108]: '''
message [109]: '''
message [110]: '''
message [111]: '''
message [112]: '''
message [113]: '''
message [114]: '''
message [115]: '''
message [116]: '''
message [117]: '''
message [118]: '''
message [119]: '''
message [120]: '''
message [121]: '''
message [122]: '''
message [123]: '''
message [124]: '''
message [125]: '''
message [126]: '''
message [127]: '''
Subscriber sleeping for 4 sec...
```

Output

Build

```
HelloWorld_subscriber - up-to-date.

----- Done -----

Build: 2 succeeded, 0 failed, 0 sk:
```

Task List Command Window Output

Ln 16 Col 1 Ch 1 INS

# Other Applications

## ● Maintenance

- Using a UML Agent to interface to a DDS
- Agent can perform a DDS 'discovery'
- Publish the results to a DDS model visually
  - Common presentation of DDS discoveries
  - Visually inspect, or execute a 'model check' to flag faults

# Other Applications

## ● Prototyping

- Using a UML Agent to interface to a DDS as per Maintenance
- Automatically instantiate DDS Entities from a model
- Real-Time update of QoS policies
- Observe behavior
- Useful process in eliciting QoS policies for a deployment  
Which values work, or do not work

# Concluding Remarks

## Summary

- Understanding of how the MDA process functions between models  
PIM → PSM using automated transformation
- Extensibility of UML through Profiles  
UML Stereotypes for constructs to specify domain-specific modeling taxonomy
- Example of DDS Realization using MDA and UML Profile  
HelloWorld Example, how a simple PIM can generate a complex PSM without human intervention
- Other Applications  
Using a UML agent to visualize a 'live' DDS model

# Concluding Remarks

## ● Importance of Realizing DDS through MDA

- MDA Process to reduce complexity, reduce cycle times
- UML Profile to standardize nomenclature, and increase industry acceptance

## ● Current & Future Works

- Sparx Systems developing an integration application for RTI's DDS implementation – publicly available September 2006
- Sparx Systems plans to initiate an RFP for a UML Profile for DDS through the OMG



**SPARX**  
**SYSTEMS**