UML Profile for DDS

a tutorial for OMG's Workshop on Real-Time, Embedded and Enterprise-Scale Time-Critical Systems

May 24, 2010

Sam Mancarella
CTO - Sparx Systems
sam.mancarella@sparxsystems.com

Overview

- Part 1 - Introduction
  - DDS Design Challenges
  - Language Architecture & Overview
- Part 2 – Worked Example
  - DCPS
  - DLRL
  - Application Targets
  - MDA, PIM → PSM
- Part 3 - Conclusion
  - Other Applications
  - Concluding Remarks
  - Discussion

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Introduction

- A UML Profile designed for the analysis and design of object-oriented systems using Data Distribution Service technology.

- Provides DDS designers, architects and practitioners with a standard, domain-specific modeling language to design DDS-based distributed information systems in a manner not specific to the underlying implementation of that design.

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Introduction

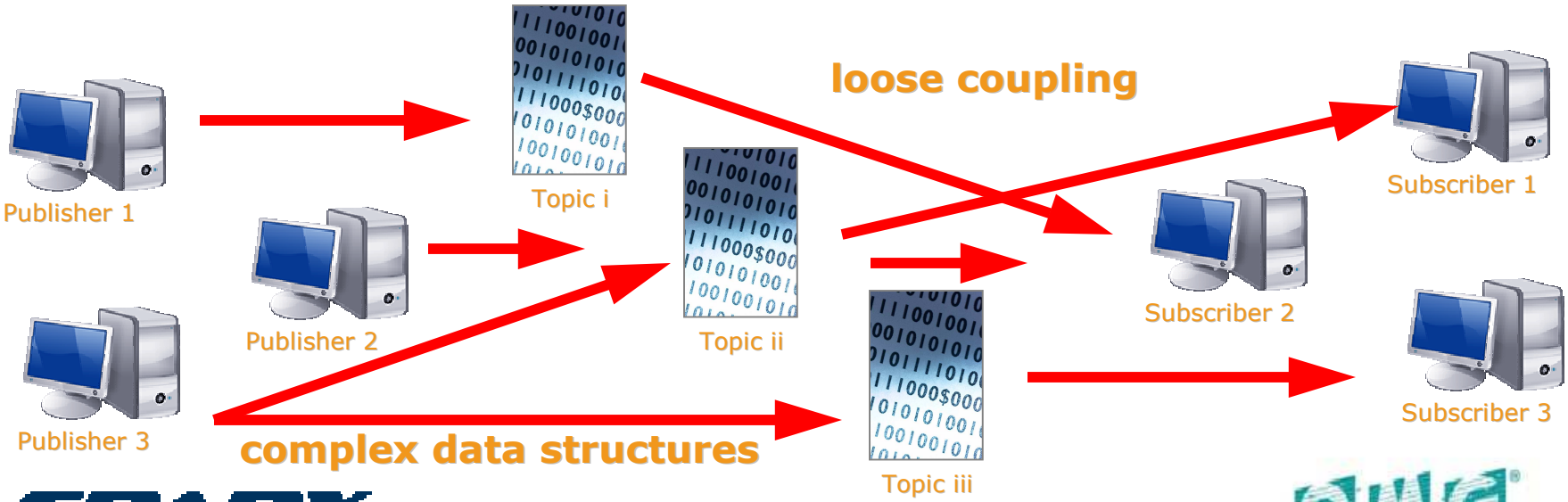
- Beta Specification: mars/2008-06-18
- Joint Submission by:
  - PrismTech
  - Real Time Innovations Inc
  - Sparx Systems
- Request For Proposal: mars/2006-09-40

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

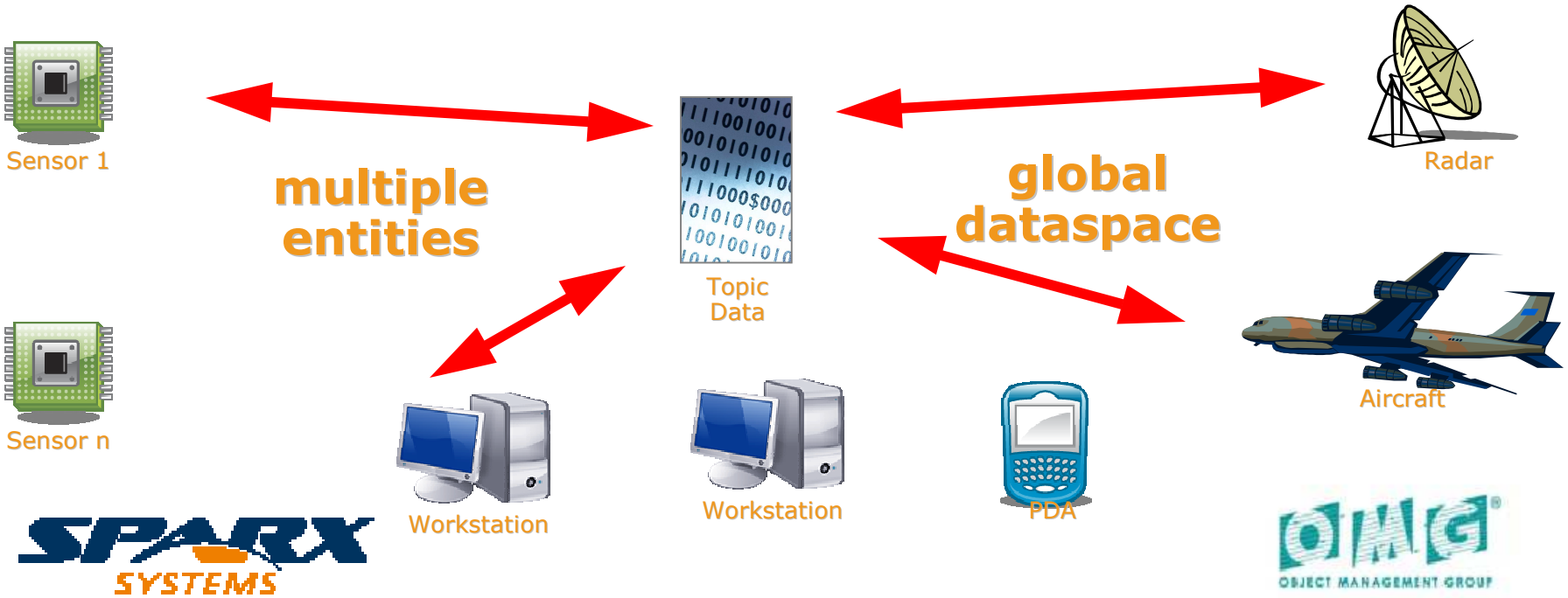
Overcoming the Challenges of DDS Design

- DDS Middleware Specification
  - Provides loosely-coupled data-centric publish/subscribe communications in real-time systems
  - Information producers (publishers)
  - Information consumers (subscribers)
- Quality of Service (QoS)
  - Diverse Quality of Service (QOS) requirements
  - Publishers, Subscribers, Readers, Writers, Topics, Participants
  - Balance predictable real-time behavior with implementation efficiency and performance



Overcoming the Challenges of DDS Design

- DDS is a PIM
  - Provides a platform independent model of entities, roles and QoS Policies
  - PIM is mapped to specific implementations, or platform specific models (PSM)
    - Variety of software languages
    - Variety of runtime platforms
    - Variety of vendors



Overcoming the Challenges of DDS Design

- Manage Complexity
  - Complex information models with QoS data
- Heterogeneous Design
  - Different implementations, same information model
- Reuse
  - Repository, Patterns
- Change Management
  - One change in model → 00's changes in code

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Model - Driven Architecture

- Domain-Specific Modeling
  - Taxonomy of constructs, relationships, constraints
  - Notation, presentation, diagrams
  - MOF or UML mappings (UML Profiles)

- PIM → PSM Transformation
  - Platform Independent Model transformed to Platform specific model automatically
  - One domain-specific model to another

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Timeline

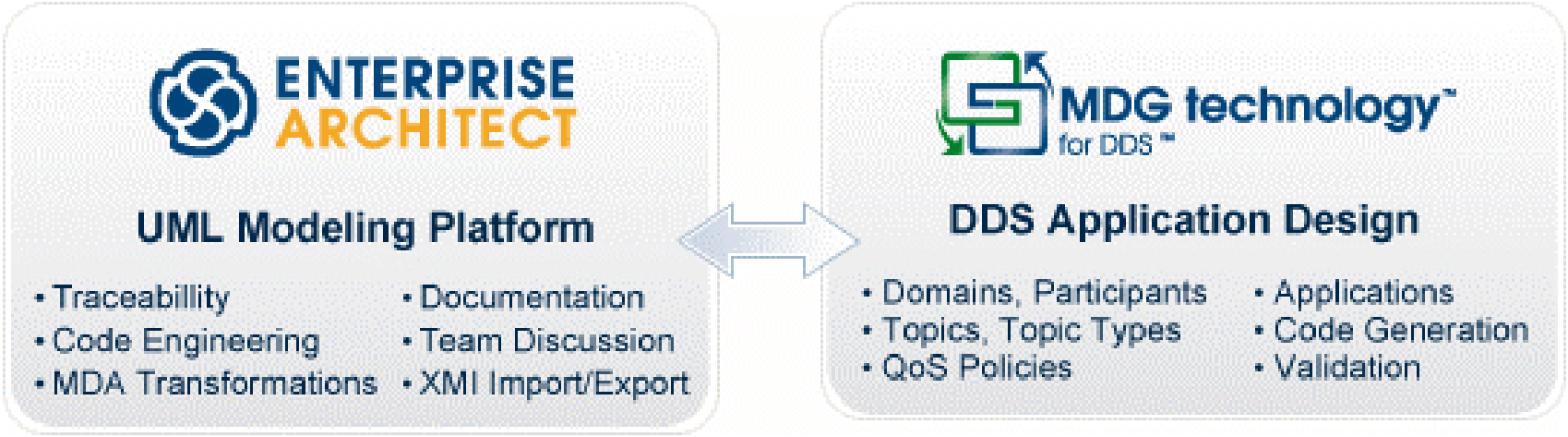
– RFP Issued	September, 2006
– First LOI	November, 2006
– First Initial Submission	March, 2007
– First Revised Submission	September, 2007
– Second LOI	October, 2007
– Second Initial Sub	December, 2007
– Second Revised Sub	February, 2008
– BoD Adoption	June, 2008
– FTF Charter	June, 2008
– FTF Charter 2	July, 2009
– FTF Report Due	June 2010

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

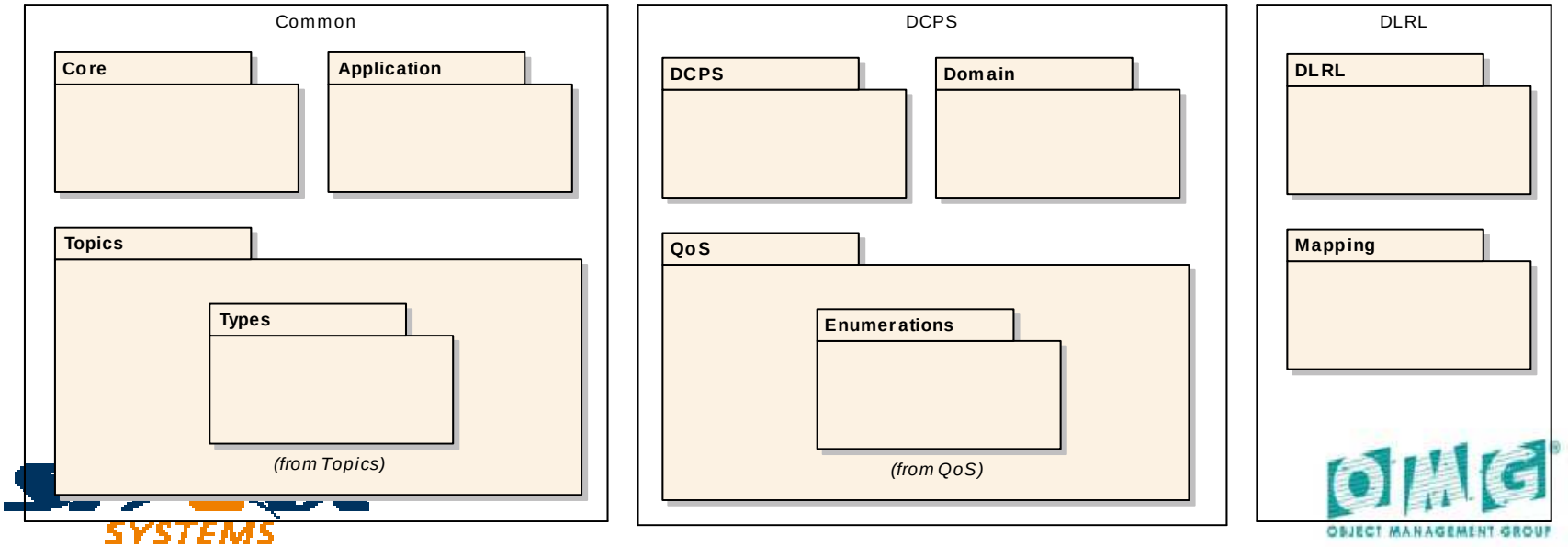
Vendor Support

- Sparx Systems – MDG Technology for DDS
  - Language Addin for Enterprise Architect
  - DDS-specific Toolboxes, Constructs, Diagrams
  - Automatically generates PSM code for OpenSplice & RTI DDS
    - Other DDS platform targets coming soon!



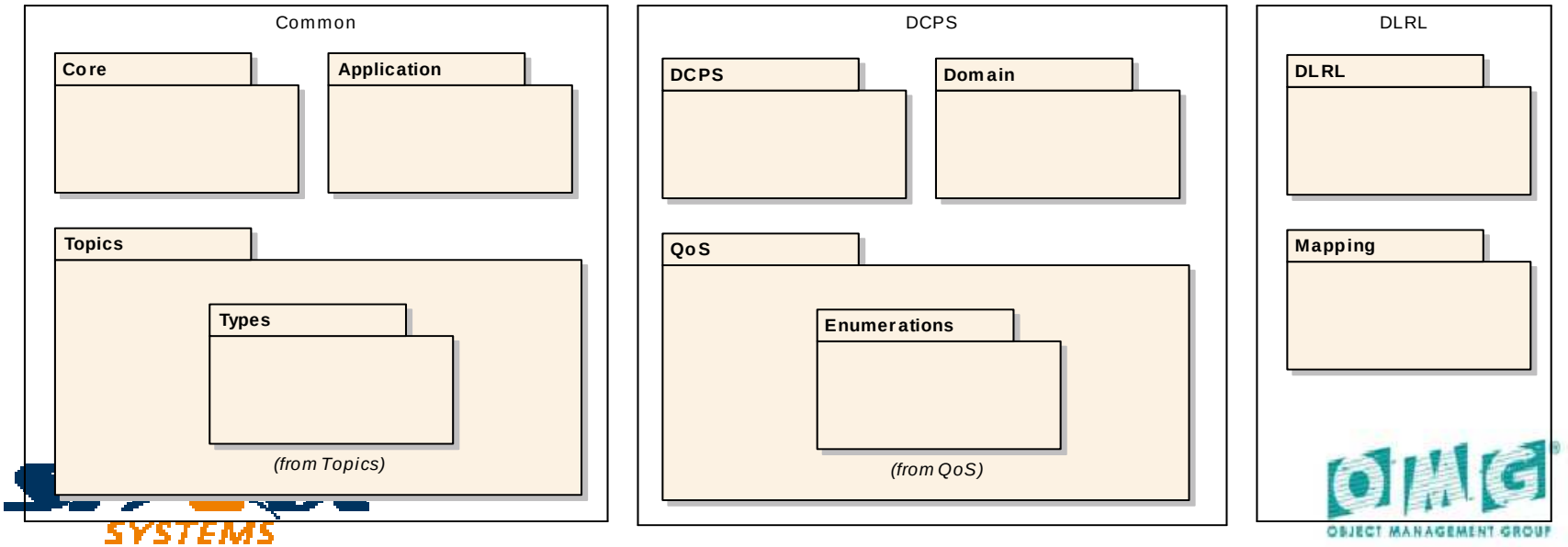
Language Architecture

- Part 1 - UML Profile
- Defines a collection of constructs that represent:
  - Data Centric Publish Subscribe Entites (+ QoS)
  - Data Local Reconstruction Layer
- Defined a collection of common constructs to define:
  - PSM Application Targets
  - Topic Data Types (IDL)



Language Architecture

- Part 2 – Metamodel
- Defines meta-level artifacts for XMI serialization



Worked Example - NetChat

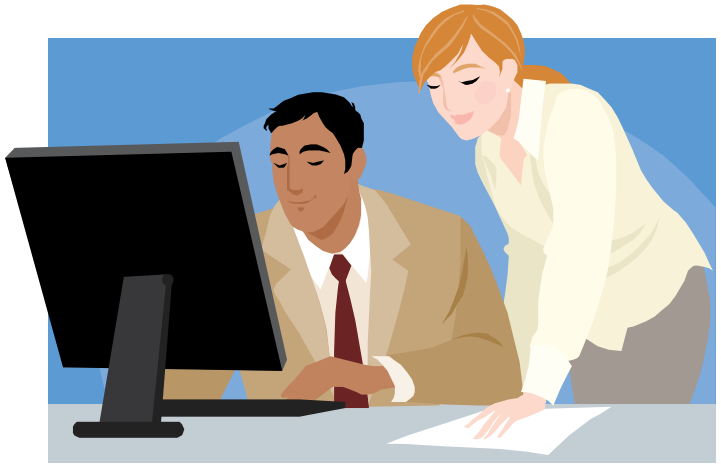
Stage 1 – DCPS-only Application

Stage 2 – DLRL-Enabled

NetChat Overview

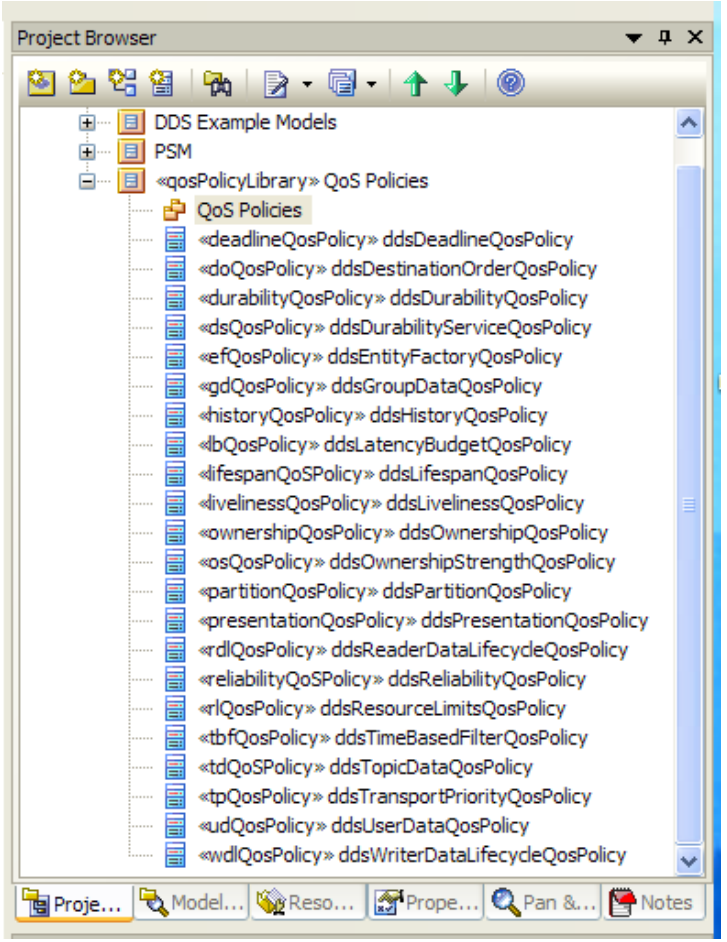
- Hypothetical, peer-to-peer network chat application
- Two Components:
  - “ChatRoom” DDS Dataspace containing conversation threads amongst users
  - “Directory Server” Application to maintain a collection of active NetChat users

- Real-world application of DDS DCPS and DLRL in distributed application designs

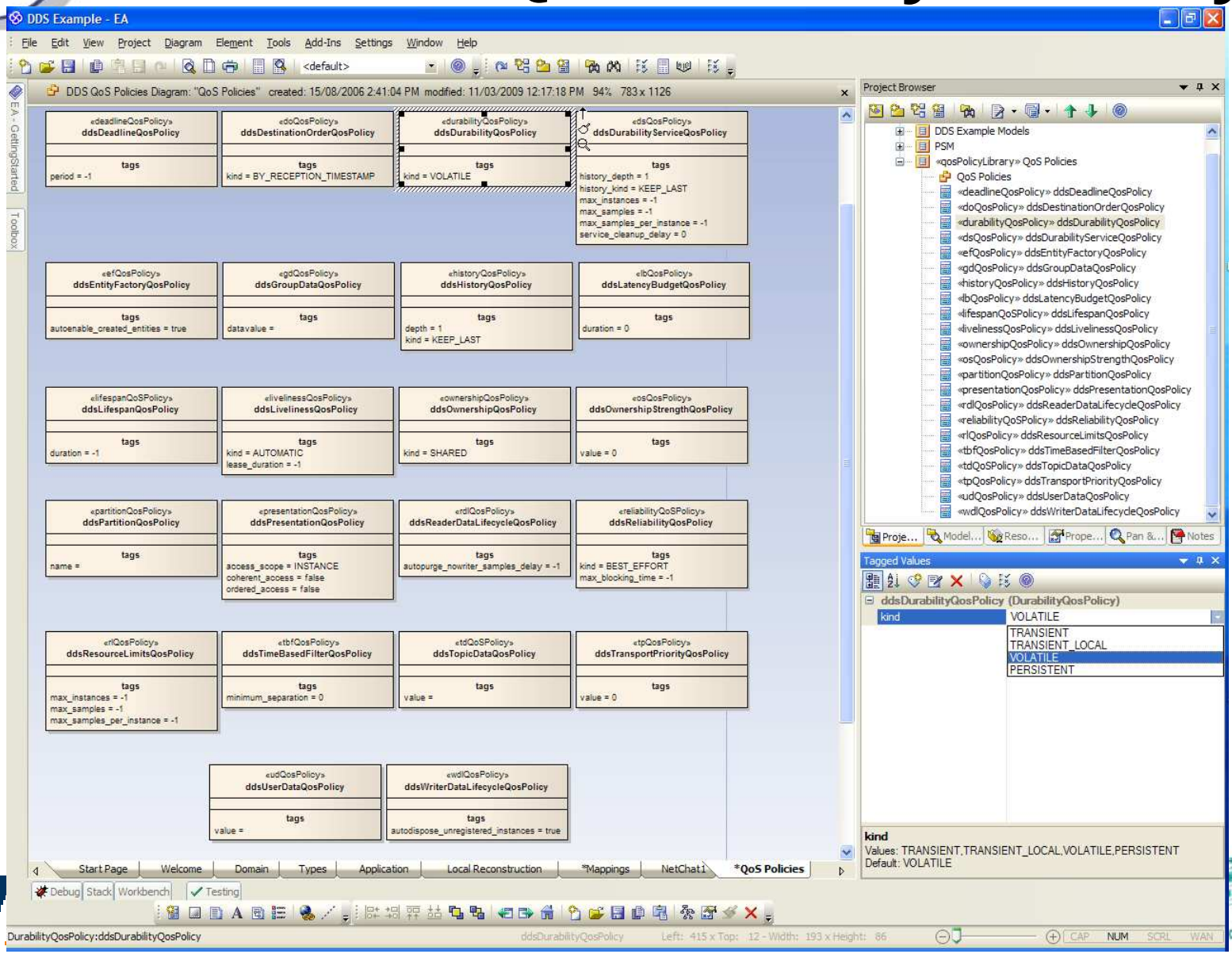


DCPS – QoS Policy Library

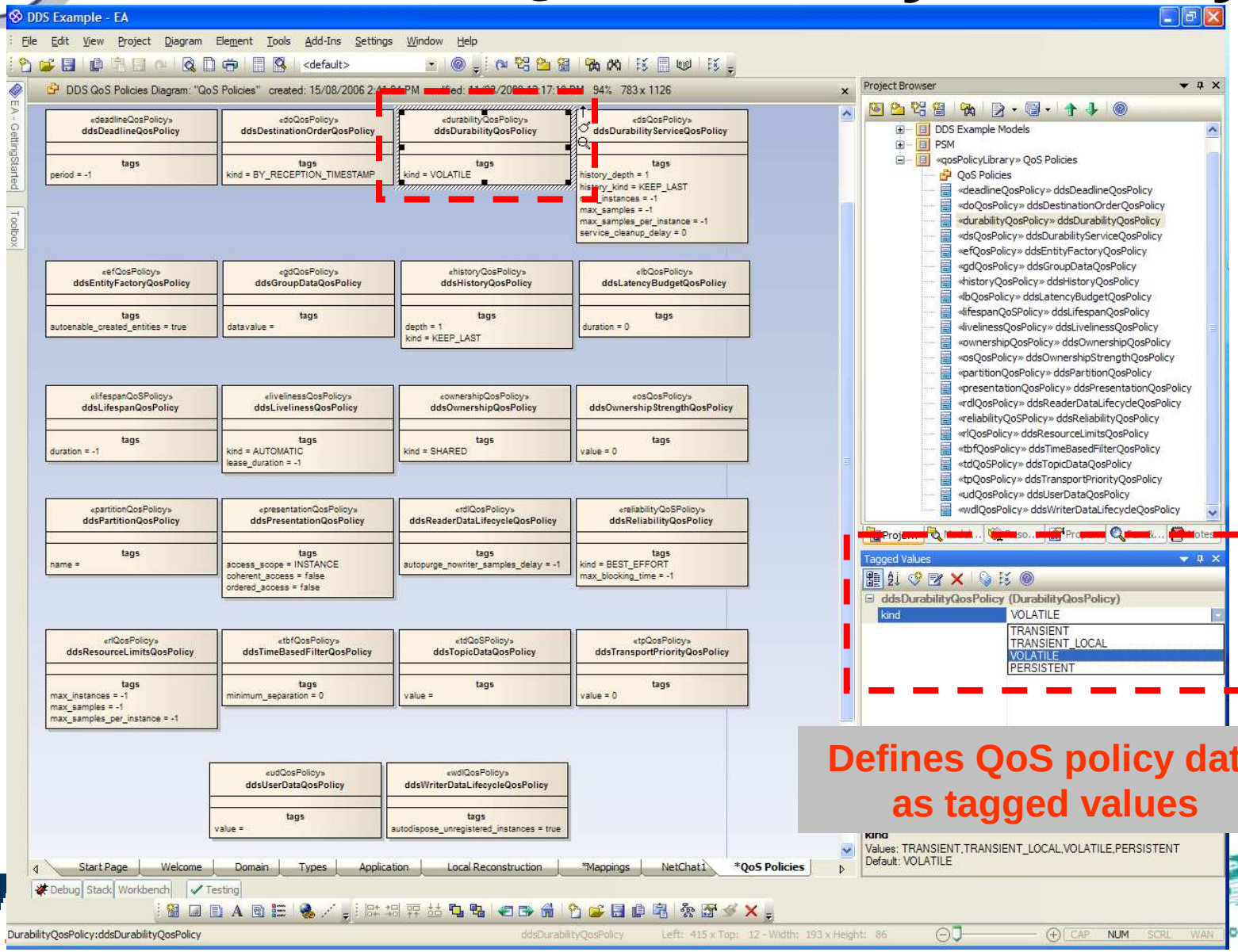
- qosPolicyLibrary Package
  - Top-Level Classifiers defining ‘default’ QoS Policies
  - Define sets of qosPolicyLibraries for domain-specific applications
  - Template of reusable QoS assets for multiple projects



DCPS – QoS Policy Library



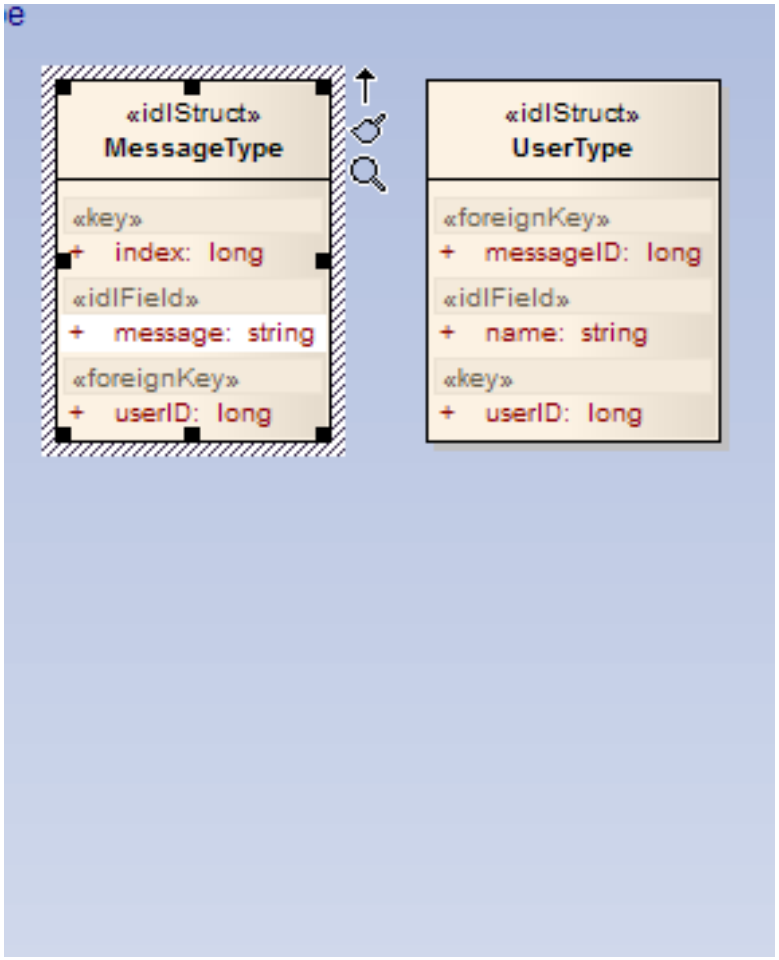
DCPS – QoS Policy Library



Defines QoS policy data as tagged values

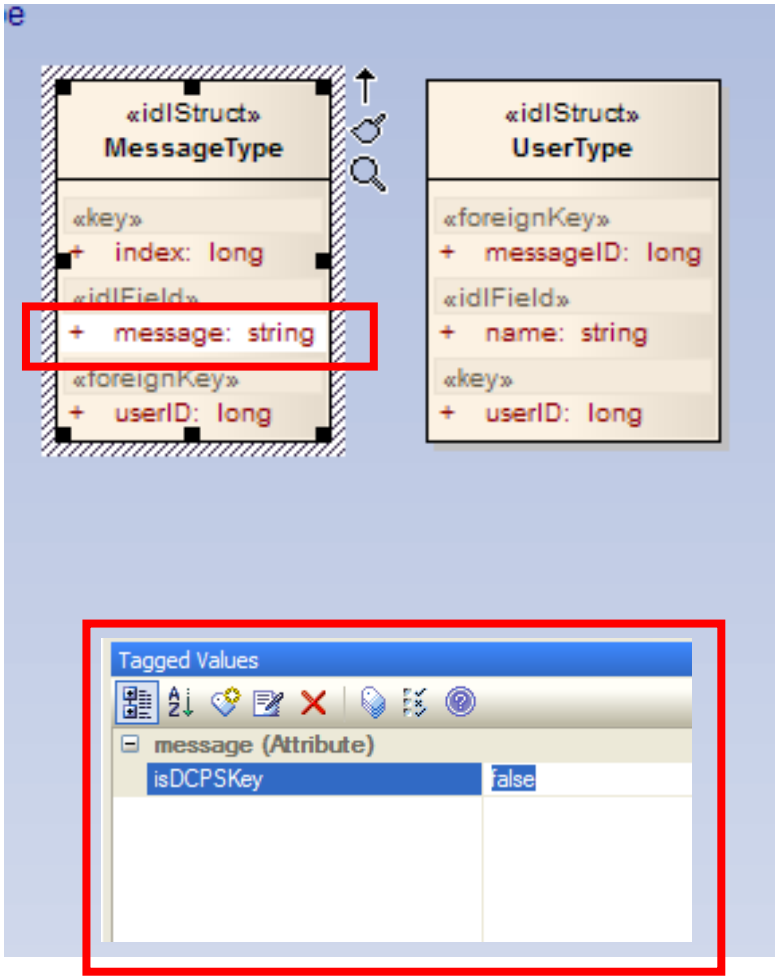
DCPS Topics & Data Types

- Data Types
  - Describes the data payloads for DCPS topics
  - IDL-based library
  - structs, unions, arrays



DCPS Topics & Data Types

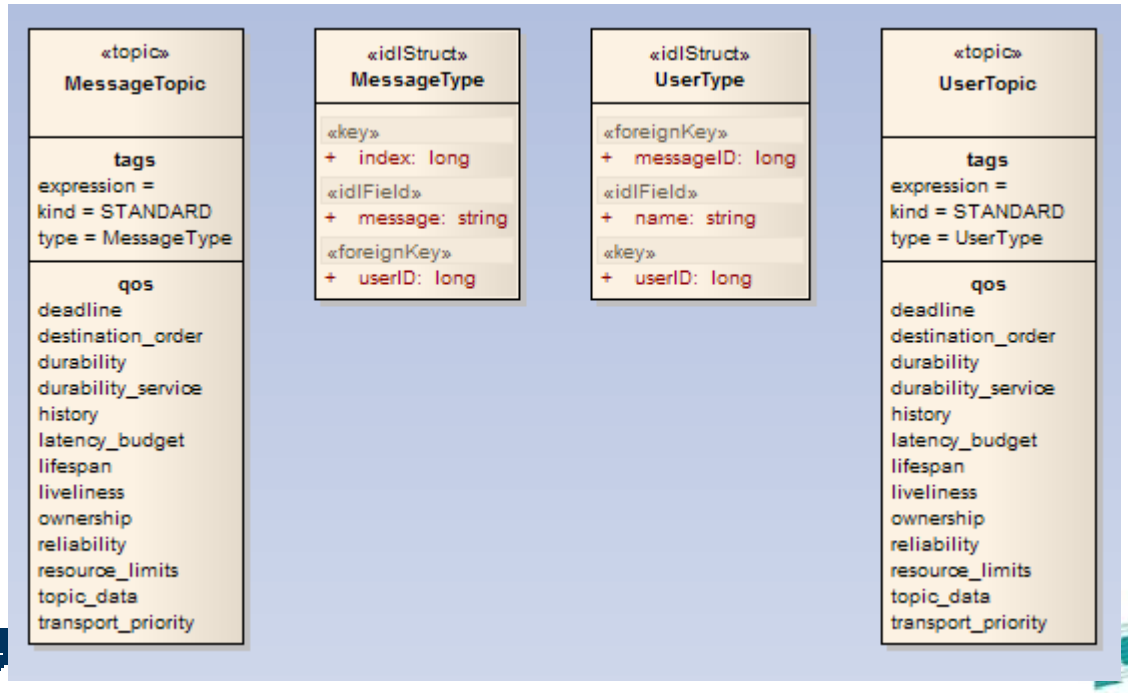
- Data Types
  - Describes the data payloads for DCPS topics
  - IDL-based library
  - structs, unions, arrays



Attributes can be nominated as DCPS Key fields

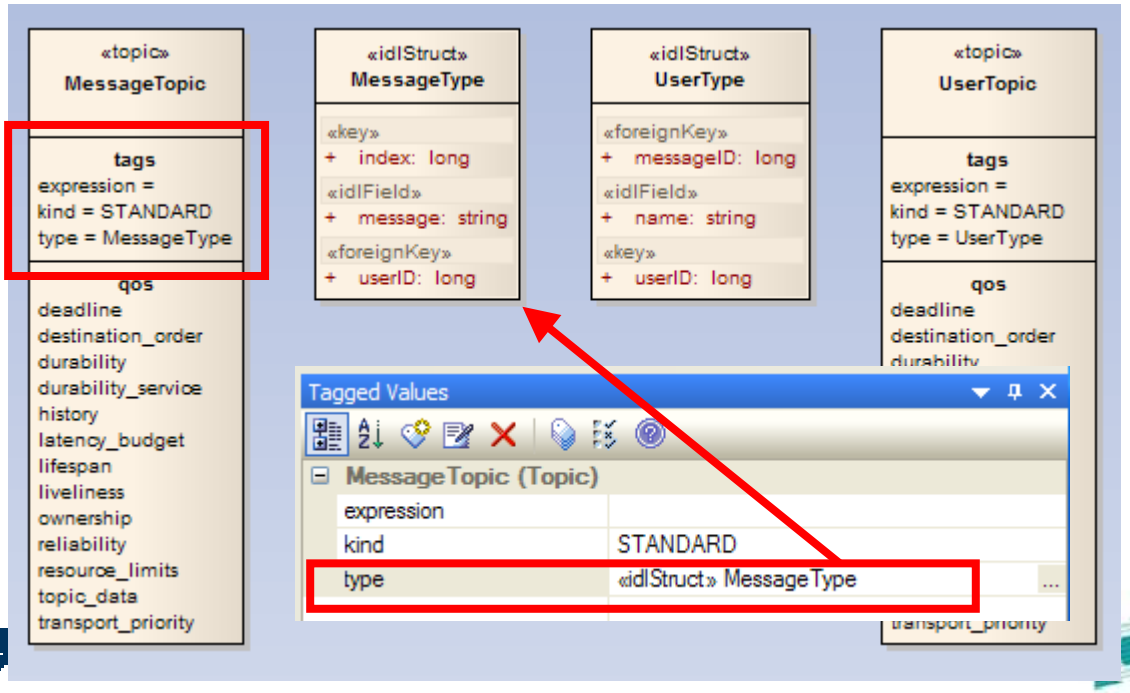
DCPS Topics & Data Types

- DDS Topics
  - Describes the DCPS characteristics of the published/subscribe data type, constrained to QoS Policy



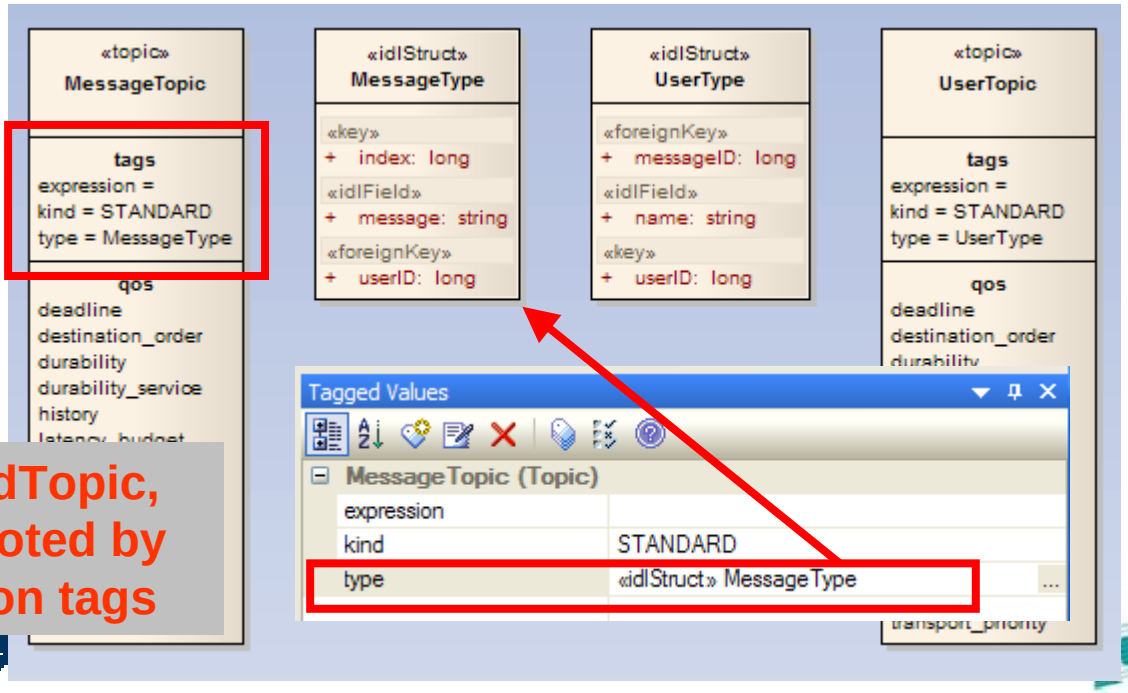
DCPS Topics & Data Types

- DDS Topics
  - Describes the DCPS characteristics of the published/subscribe data type, constrained to QoS Policy



DCPS Topics & Data Types

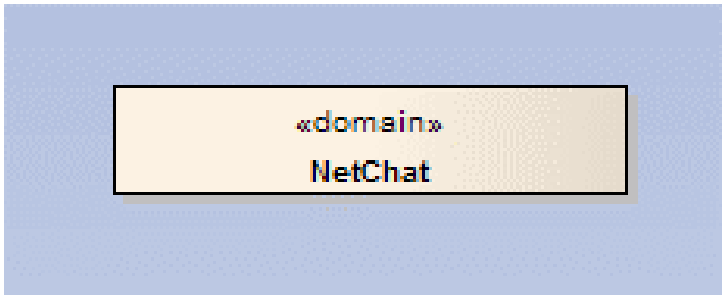
- DDS Topics
  - Describes the DCPS characteristics of the published/subscribe data type, constrained to QoS Policy



ContentFilteredTopic, MultiTopic denoted by kind, expression tags

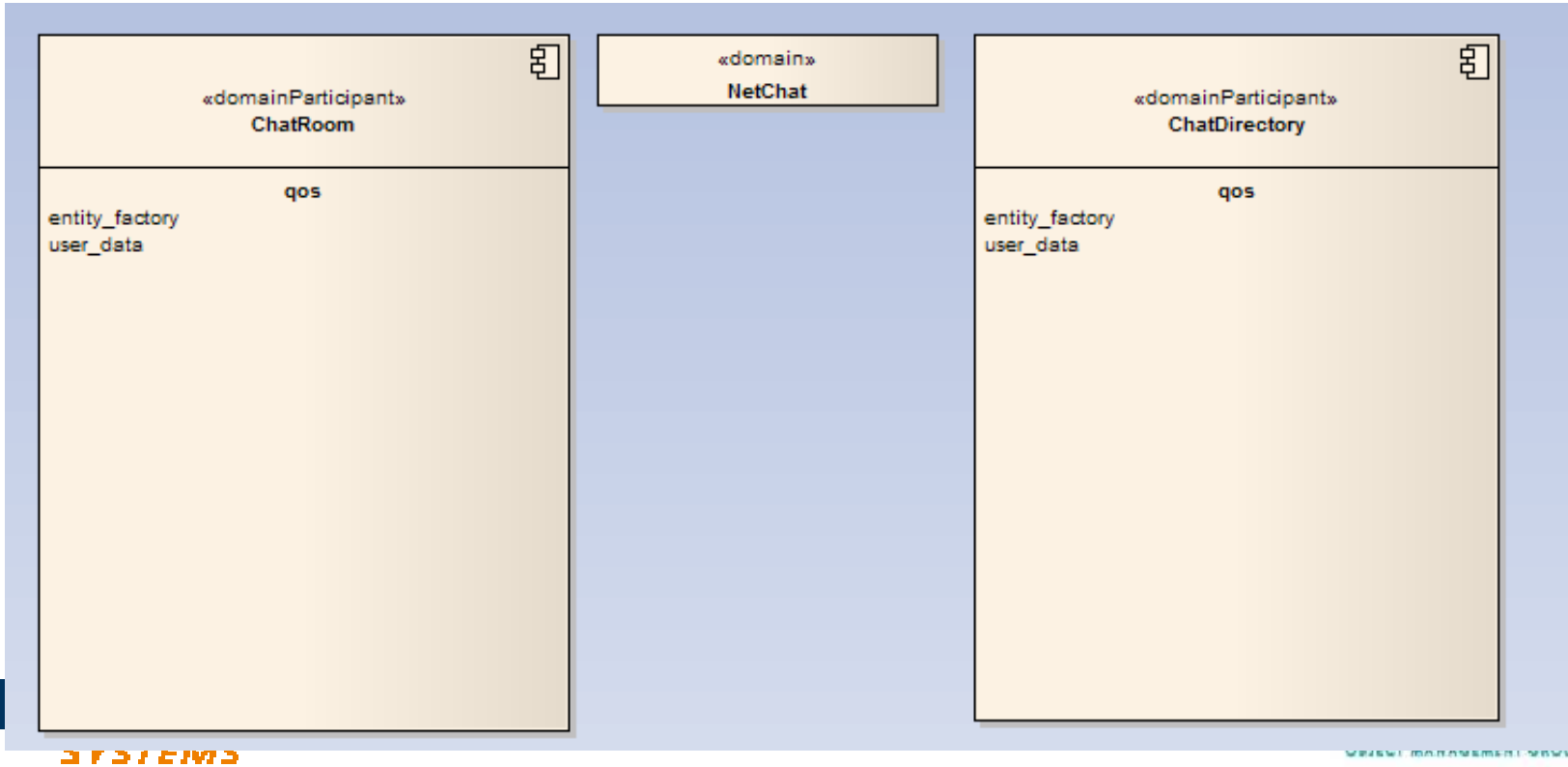
DCPS Domain & Entities

- Domain Entity
  - Logical ‘grouping’ of DCPS Topics, & DomainParticipants



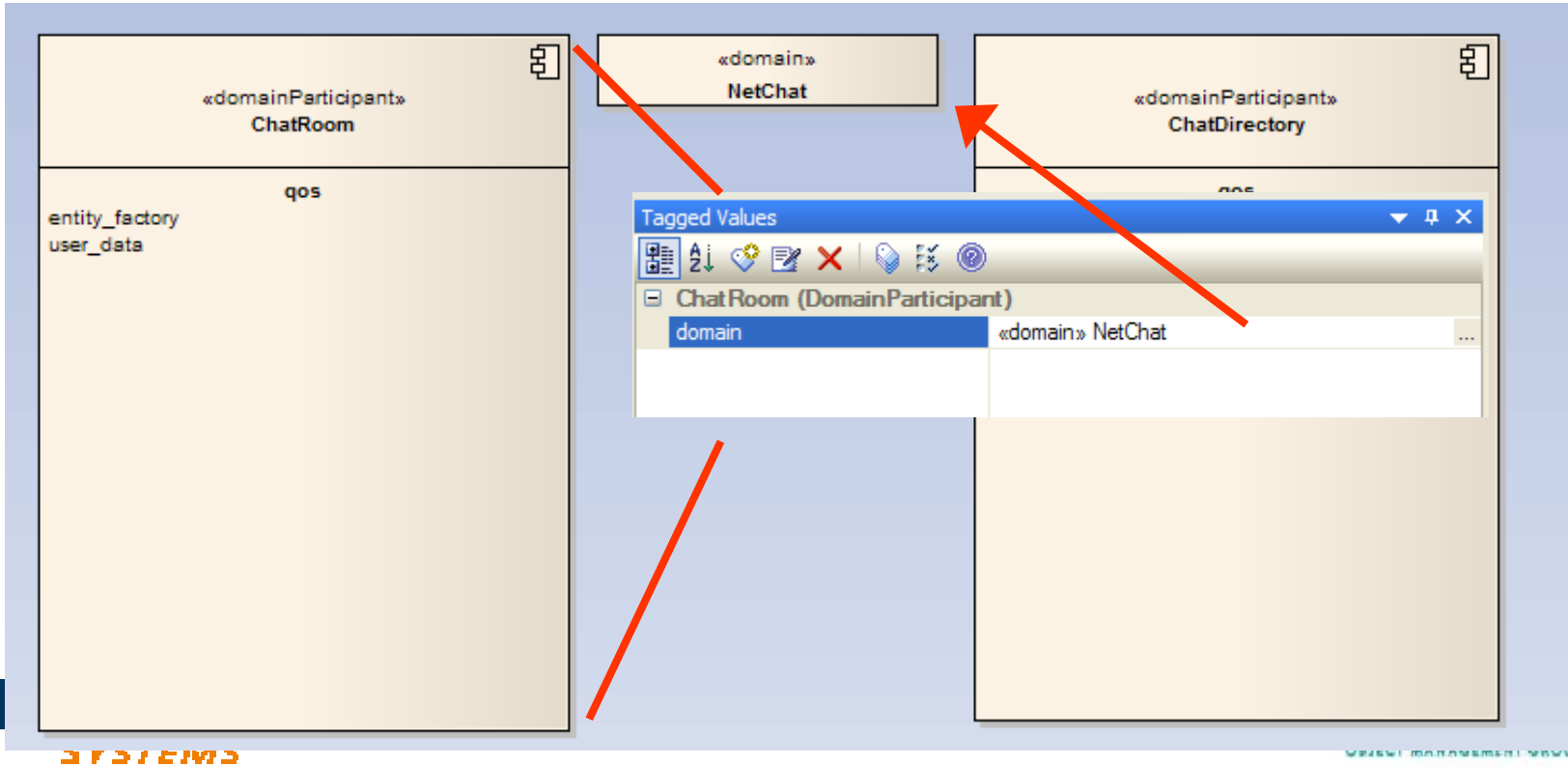
DCPS Domain & Entities

- DomainParticipant Entity
  - DDS Publish/Subscribe entity



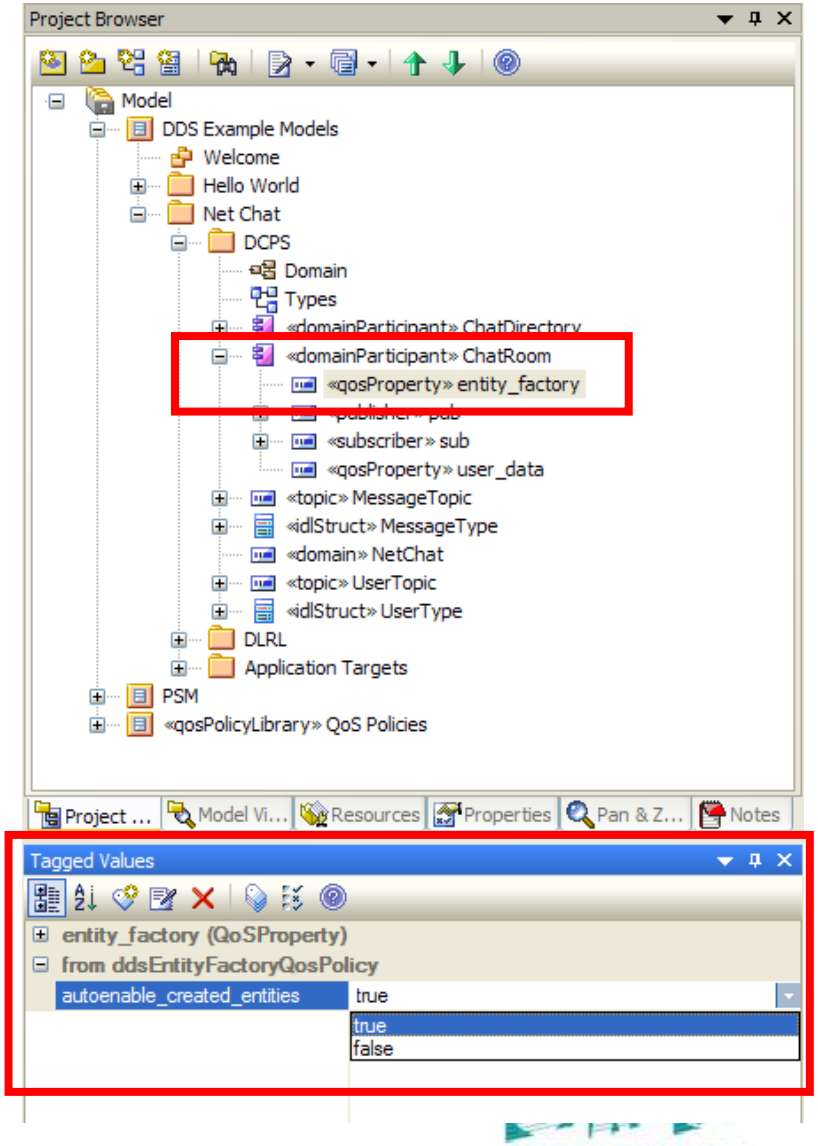
DCPS Domain & Entities

- DomainParticipant Entity
  - Participate in domain nominated by tagged value



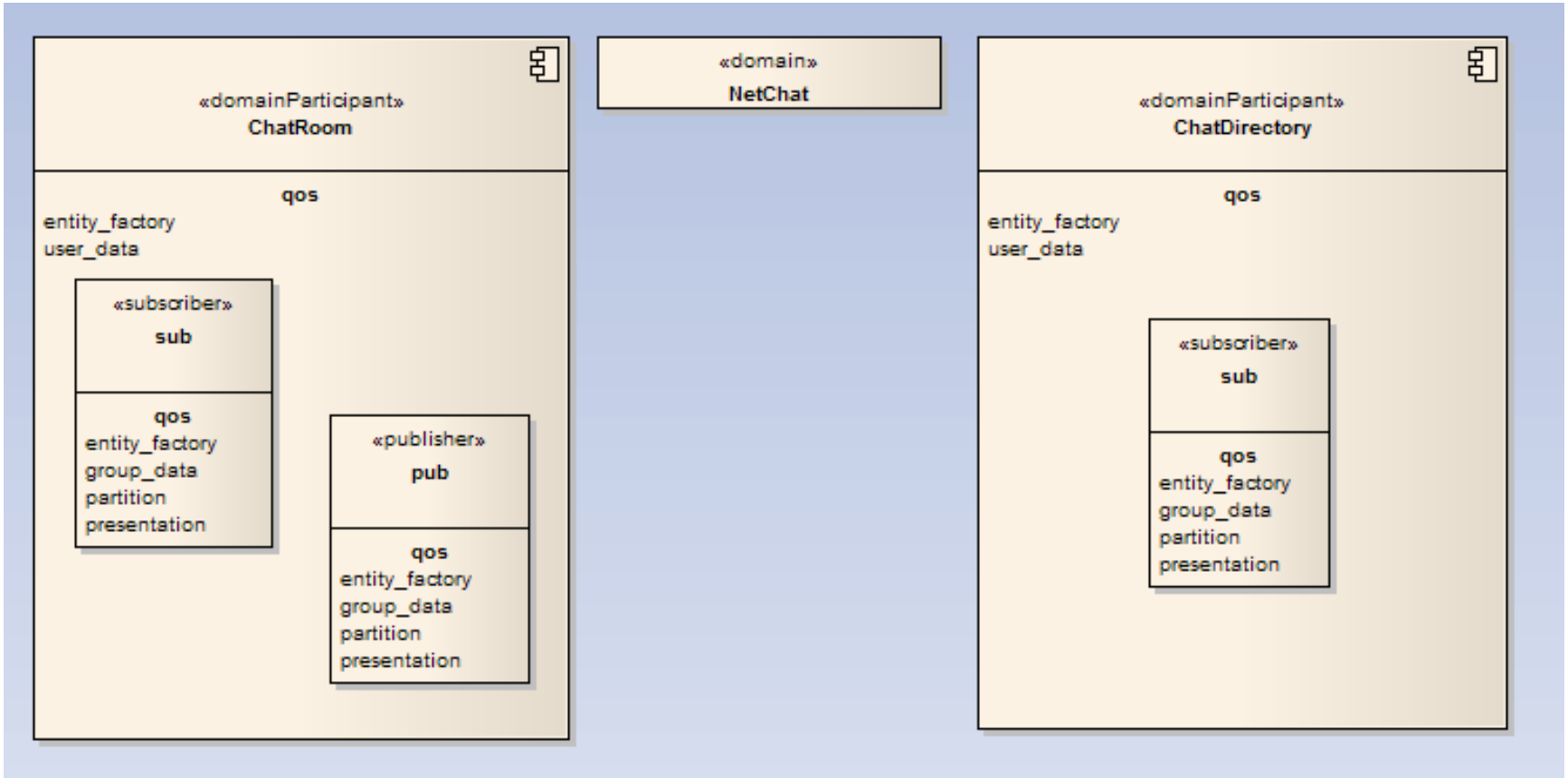
DCPS Domain & Entities

- DomainParticipant Entity
  - Qos applied as properties, typed by the QoS Policy types in the qosPolicyLibrary



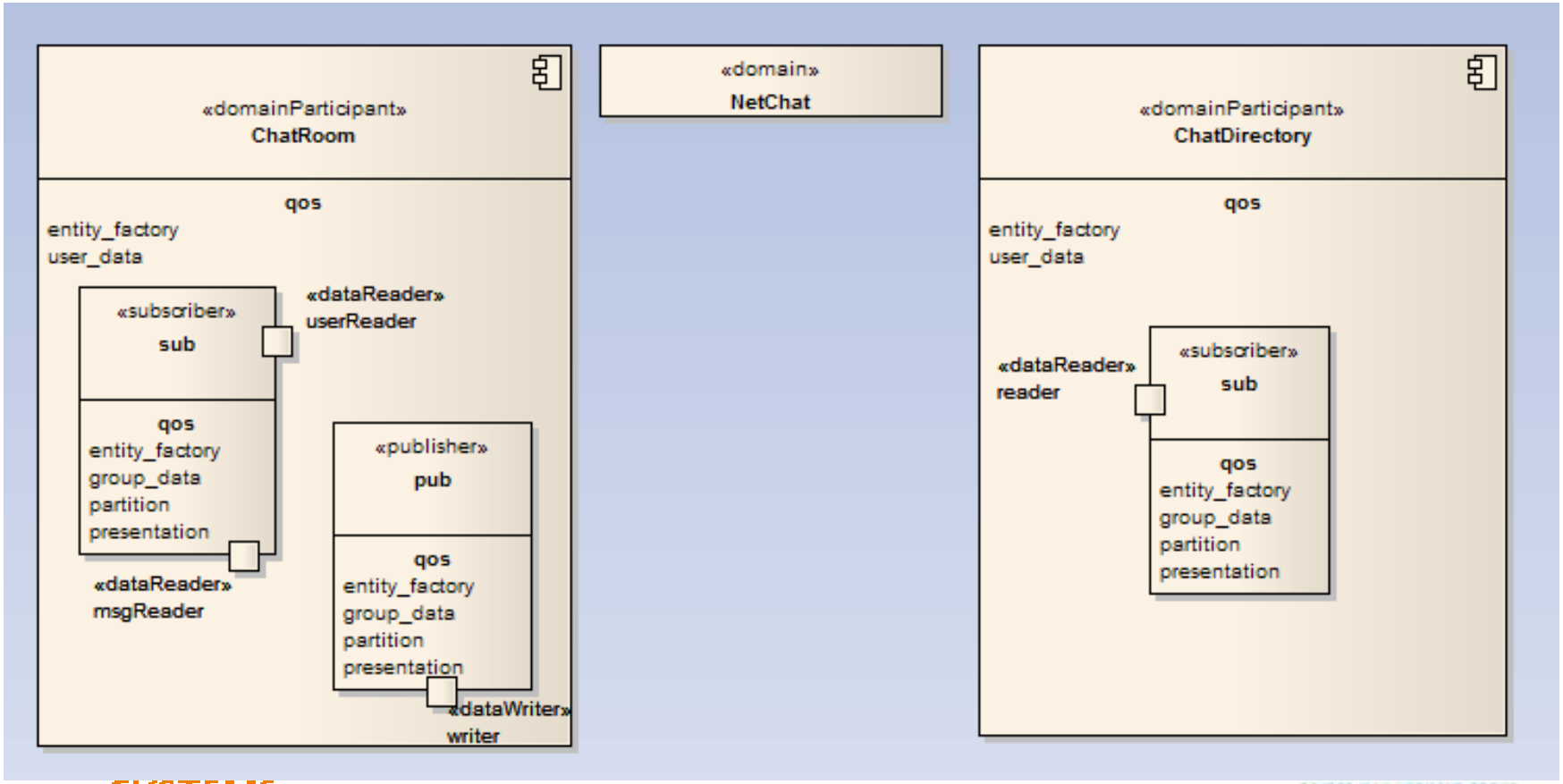
DCPS Domain & Entities

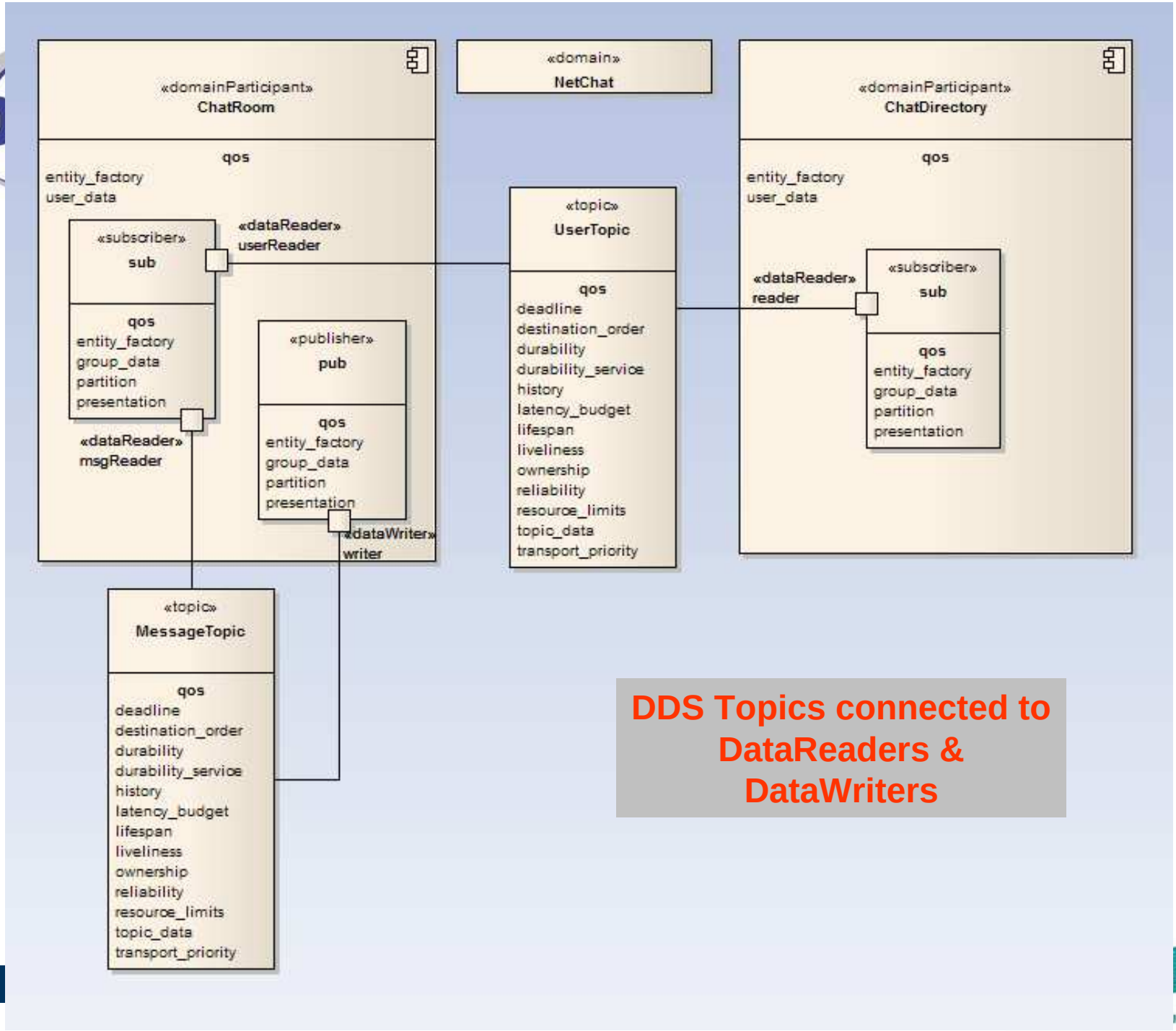
- Added Publisher, Subscriber Entities



DCPS Domain & Entities

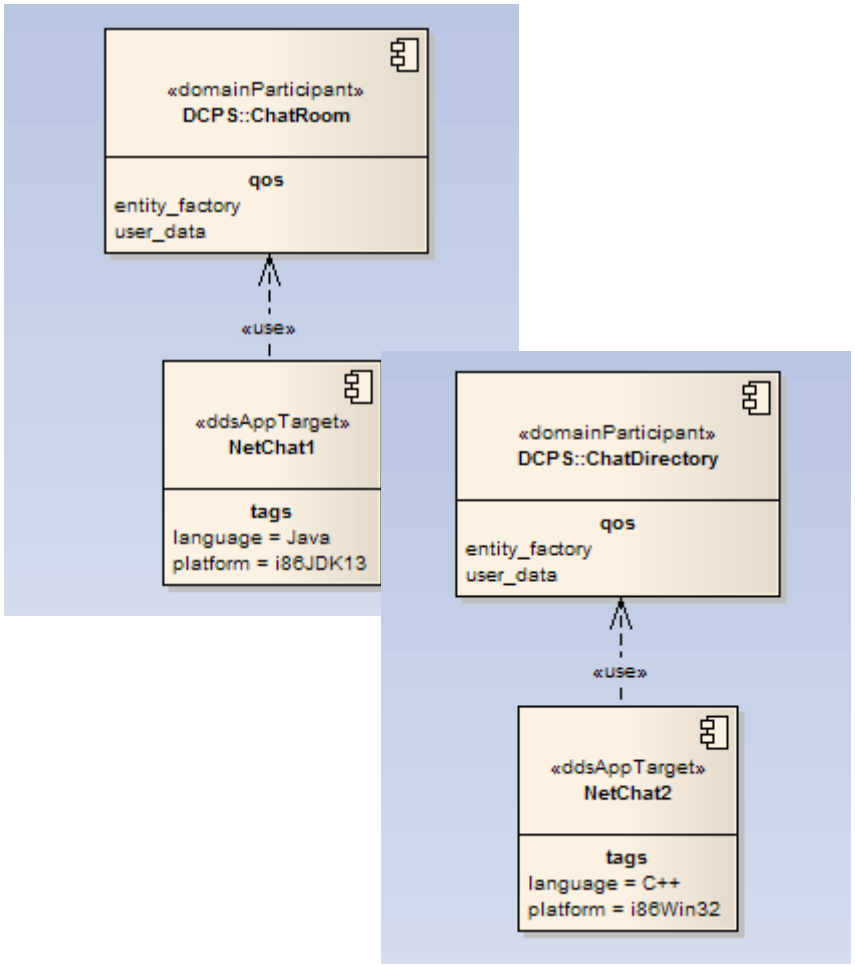
- Added DataReaders, DataWriters Entities




«domainParticipant»
ChatRoom
qos
entity_factory
user_data
«subscriber»
sub
«dataReader»
userReader
qos
entity_factory
group_data
partition
presentation
«dataReader»
msgReader
«publisher»
pub
qos
entity_factory
group_data
partition
presentation
«dataWriter»
writer
«domain»
NetChat
«topic»
UserTopic
qos
deadline
destination_order
durability
durability_service
history
latency_budget
lifespan
liveliness
ownership
reliability
resource_limits
topic_data
transport_priority
«domainParticipant»
ChatDirectory
qos
entity_factory
user_data
«dataReader»
reader
«subscriber»
sub
qos
entity_factory
group_data
partition
presentation
«topic»
MessageTopic
qos
deadline
destination_order
durability
durability_service
history
latency_budget
lifespan
liveliness
ownership
reliability
resource_limits
topic_data
transport_priority
DDS Topics connected to DataReaders & DataWriters

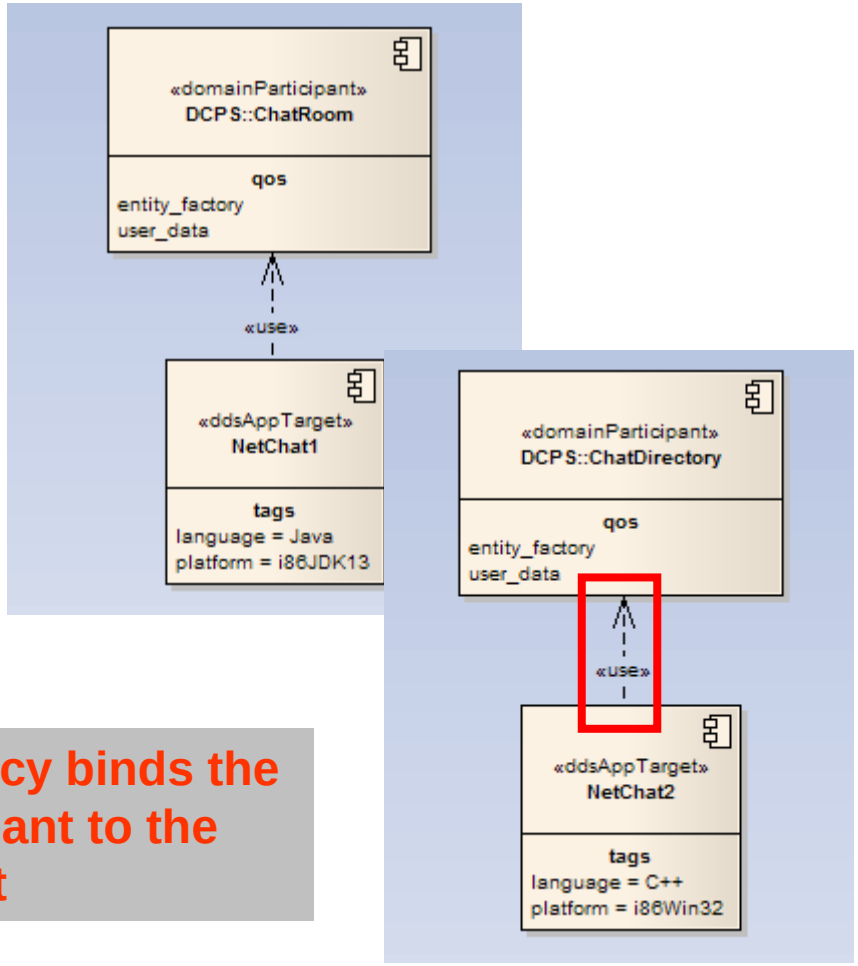
Application Targets

- ddsAppTarget
  - Binds one or more DomainParticipants to a PSM configuration



Application Targets

- ddsAppTarget
  - Binds one or more DomainParticipants to a PSM configuration

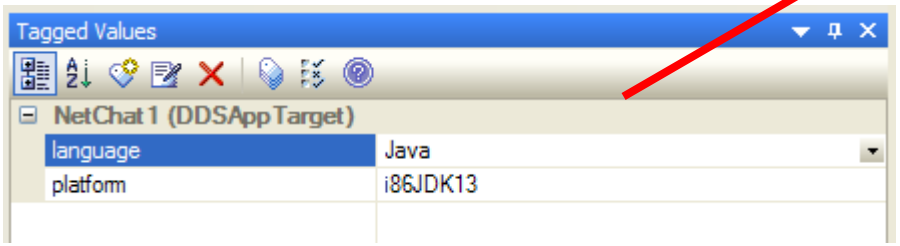


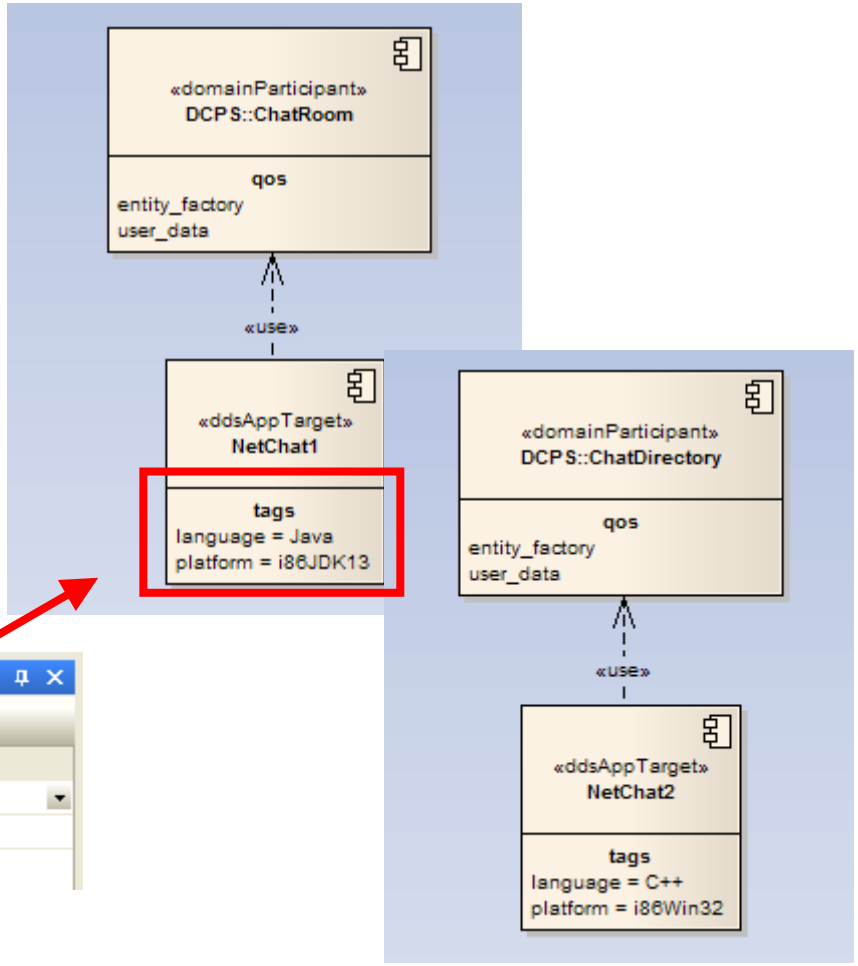
'usage' Dependency binds the DomainParticipant to the Target

Application Targets

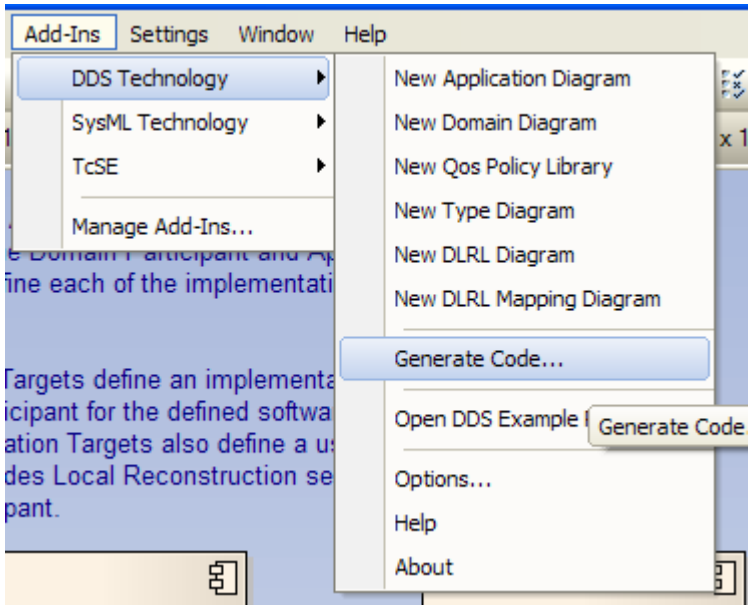
- ddsAppTarget
  - Binds one or more DomainParticipants to a PSM configuration

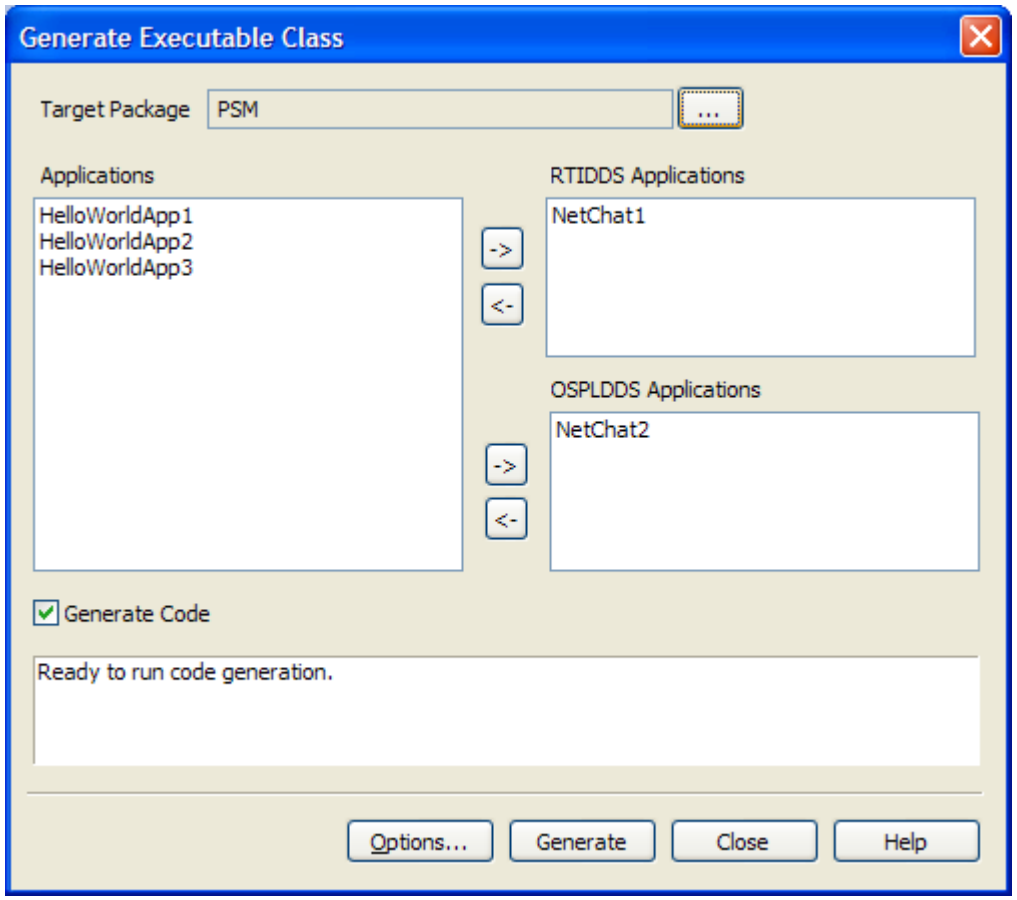
Tagged values specify the desired PSM output



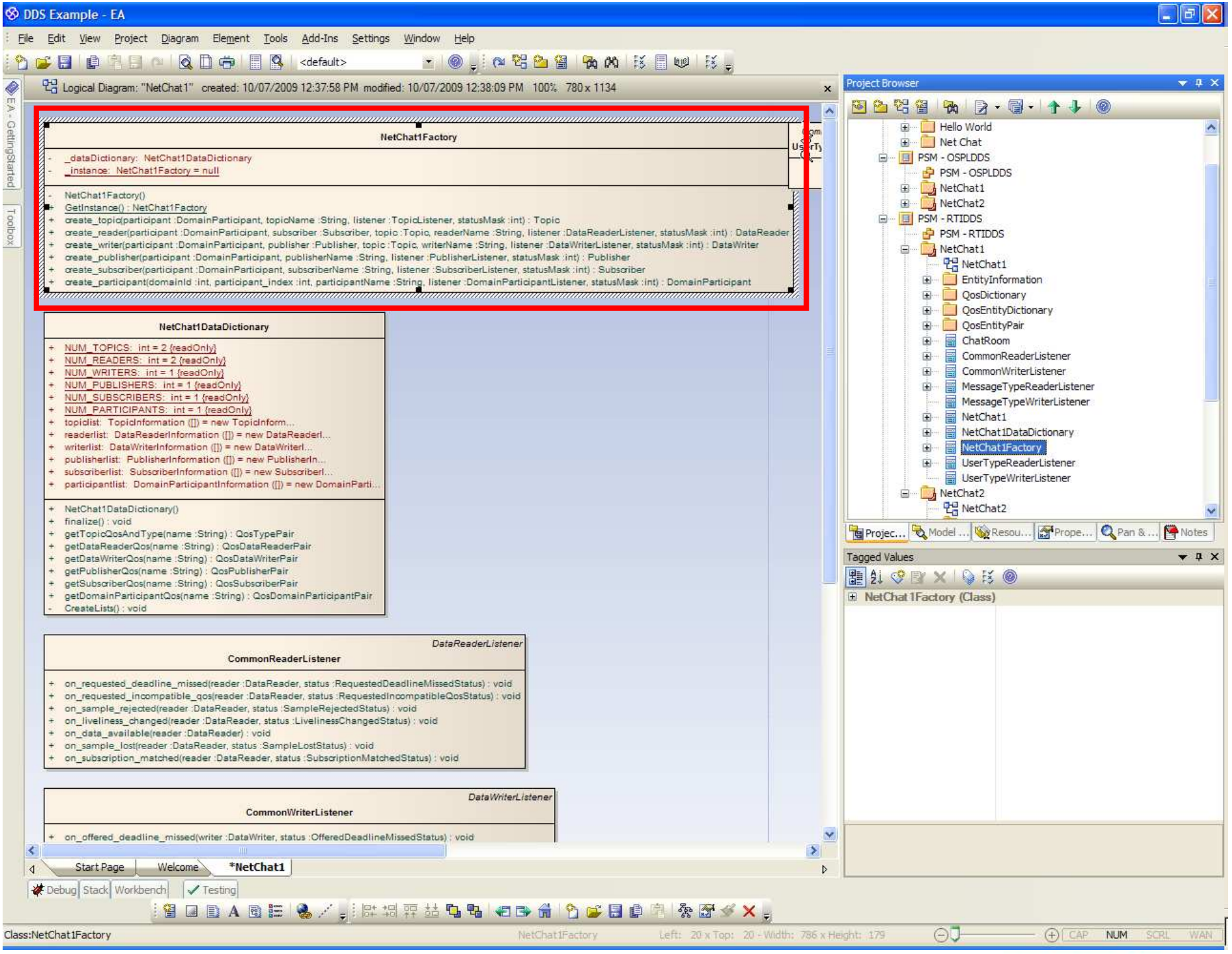


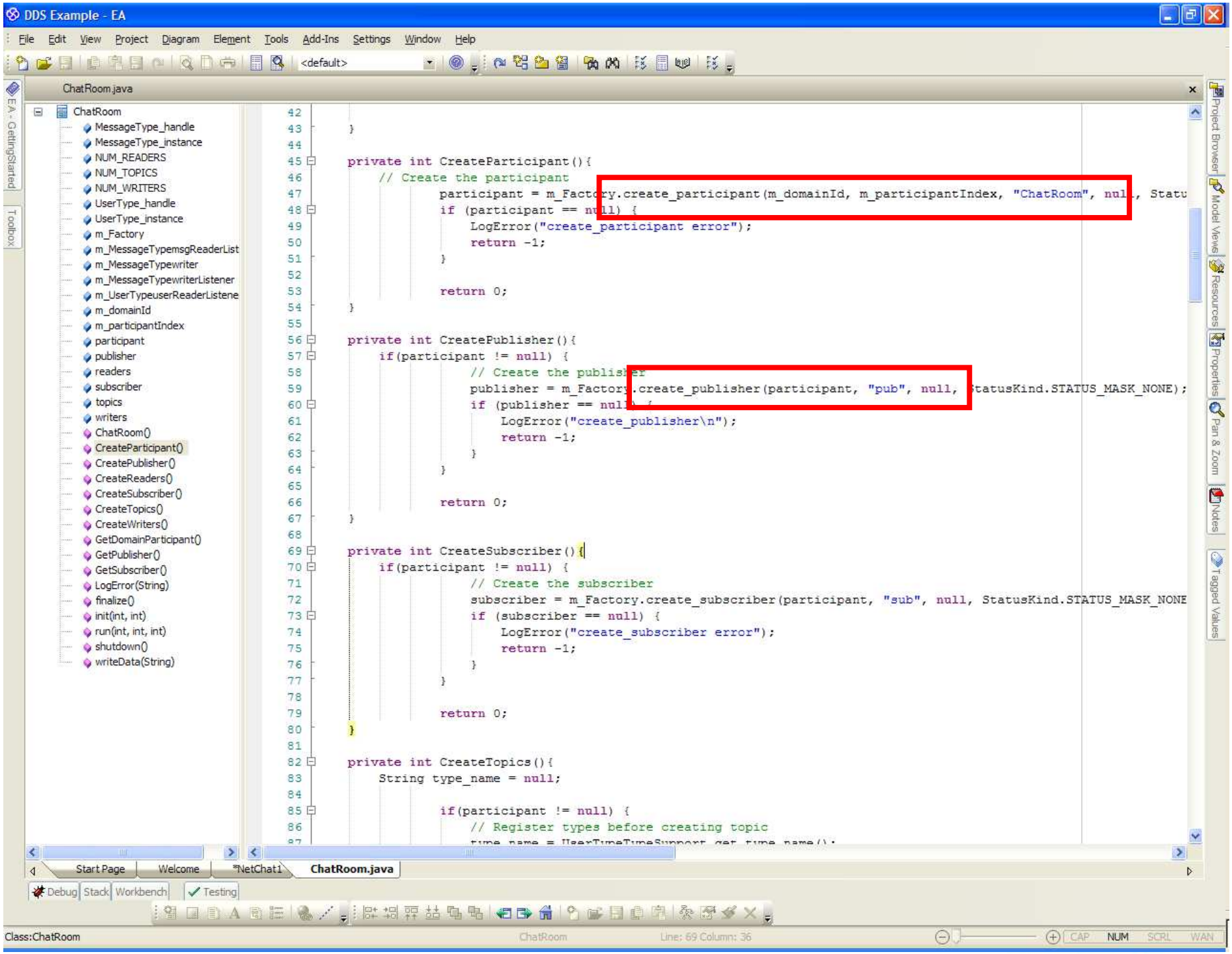
Code (PSM) Generation





Tool Specific: Enterprise Architect prompts the user to designate the application targets to a specific DDS output platform


DDS Example - EA
File Edit View Project Diagram Element Tools Add-Ins Settings Window Help
Logical Diagram: "NetChat1" created: 10/07/2009 12:37:58 PM modified: 10/07/2009 12:38:09 PM 100% 780 x 1134
NetChat1Factory
- _dataDictionary: NetChat1DataDictionary
- _instance: NetChat1Factory = null
- NetChat1Factory()
+ GetInstance() : NetChat1Factory
+ create_topic(participant :DomainParticipant, topicName :String, listener :TopicListener, statusMask :int) : Topic
+ create_reader(participant :DomainParticipant, subscriber :Subscriber, topic :Topic, readerName :String, listener :DataReaderListener, statusMask :int) : DataReader
+ create_writer(participant :DomainParticipant, publisher :Publisher, topic :Topic, writerName :String, listener :DataWriterListener, statusMask :int) : DataWriter
+ create_publisher(participant :DomainParticipant, publisherName :String, listener :PublisherListener, statusMask :int) : Publisher
+ create_subscriber(participant :DomainParticipant, subscriberName :String, listener :SubscriberListener, statusMask :int) : Subscriber
+ create_participant(domainId :int, participant_index :int, participantName :String, listener :DomainParticipantListener, statusMask :int) : DomainParticipant
NetChat1DataDictionary
+ NUM_TOPICS: int = 2 {readOnly}
+ NUM_READERS: int = 2 {readOnly}
+ NUM_WRITERS: int = 1 {readOnly}
+ NUM_PUBLISHERS: int = 1 {readOnly}
+ NUM_SUBSCRIBERS: int = 1 {readOnly}
+ NUM_PARTICIPANTS: int = 1 {readOnly}
+ topiclist: TopicInformation ([]) = new TopicInform...
+ readerlist: DataReaderInformation ([]) = new DataReaderI...
+ writerlist: DataWriterInformation ([]) = new DataWriterI...
+ publisherlist: PublisherInformation ([]) = new PublisherIn...
+ subscriberlist: SubscriberInformation ([]) = new SubscriberI...
+ participantlist: DomainParticipantInformation ([]) = new DomainParti...
+ NetChat1DataDictionary()
+ finalize() : void
+ getTopicQosAndType(name :String) : QosTypePair
+ getDataReaderQos(name :String) : QosDataReaderPair
+ getDataWriterQos(name :String) : QosDataWriterPair
+ getPublisherQos(name :String) : QosPublisherPair
+ getSubscriberQos(name :String) : QosSubscriberPair
+ getDomainParticipantQos(name :String) : QosDomainParticipantPair
- CreateLists() : void
DataReaderListener
CommonReaderListener
+ on_requested_deadline_missed(reader :DataReader, status :RequestedDeadlineMissedStatus) : void
+ on_requested_incompatible_qos(reader :DataReader, status :RequestedIncompatibleQosStatus) : void
+ on_sample_rejected(reader :DataReader, status :SampleRejectedStatus) : void
+ on_liveliness_changed(reader :DataReader, status :LivelinessChangedStatus) : void
+ on_data_available(reader :DataReader) : void
+ on_sample_lost(reader :DataReader, status :SampleLostStatus) : void
+ on_subscription_matched(reader :DataReader, status :SubscriptionMatchedStatus) : void
DataWriterListener
CommonWriterListener
+ on_offered_deadline_missed(writer :DataWriter, status :OfferedDeadlineMissedStatus) : void
Start Page
Welcome
*NetChat1
Debug
Stack
Workbench
Testing
Project Browser
Hello World
Net Chat
PSM - OSPLDDS
PSM - OSPLDDS
NetChat1
NetChat2
PSM - RTIDDS
PSM - RTIDDS
NetChat1
NetChat1
EntityInformation
QosDictionary
QosEntityDictionary
QosEntityPair
ChatRoom
CommonReaderListener
CommonWriterListener
MessageTypeReaderListener
MessageTypeWriterListener
NetChat1
NetChat1DataDictionary
NetChat1Factory
UserTypeReaderListener
UserTypeWriterListener
NetChat2
NetChat2
Projec...
Model ...
Resou...
Prope...
Pan & ...
Notes
Tagged Values
NetChat1Factory (Class)
Class:NetChat1Factory
NetChat1Factory
Left: 20 x Top: 20 - Width: 786 x Height: 179
CAP
NUM
SCRL
WAN

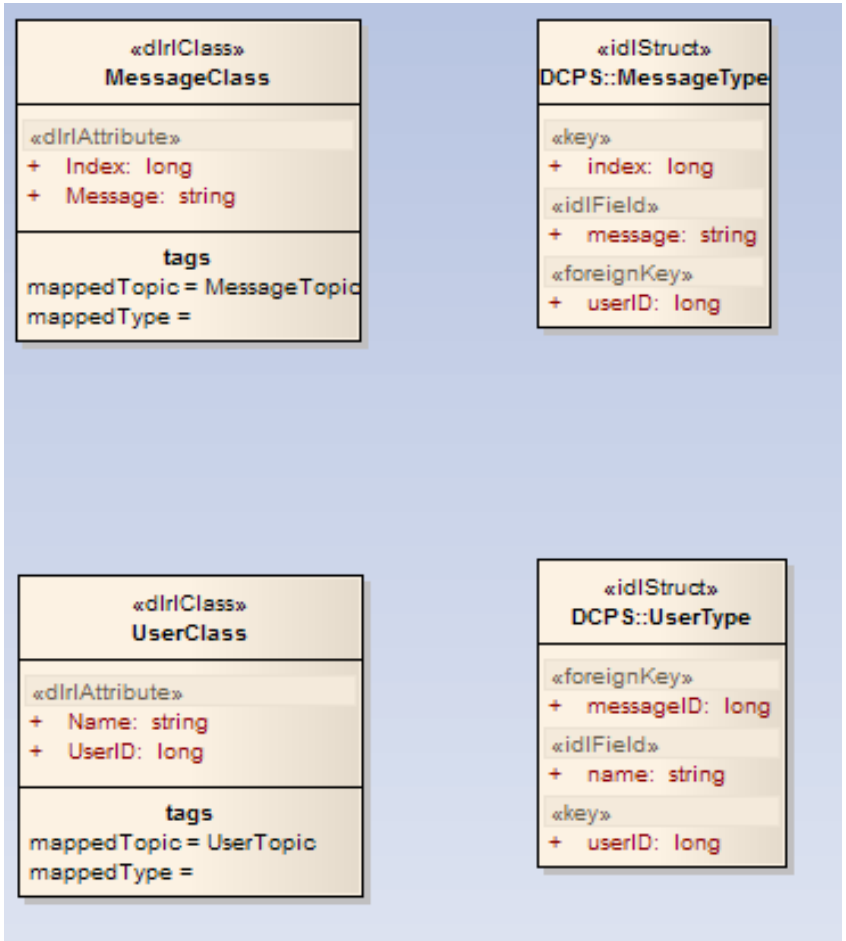

DDS Example - EA
File Edit View Project Diagram Element Tools Add-Ins Settings Window Help
<default>
ChatRoom.java
ChatRoom
MessageType_handle
MessageType_instance
NUM_READERS
NUM_TOPICS
NUM_WRITERS
UserType_handle
UserType_instance
m_Factory
m_MessageTypemsgReaderList
m_MessageTypewriter
m_MessageTypewriterListener
m_UserTypeuserReaderListene
m_domainId
m_participantIndex
participant
publisher
readers
subscriber
topics
writers
ChatRoom()
CreateParticipant()
CreatePublisher()
CreateReaders()
CreateSubscriber()
CreateTopics()
CreateWriters()
GetDomainParticipant()
GetPublisher()
GetSubscriber()
LogError(String)
finalize()
init(int, int)
run(int, int, int)
shutdown()
writeData(String)
42 }
43
44
45 private int CreateParticipant(){
46 // Create the participant
47 participant = m_Factory.create_participant(m_domainId, m_participantIndex, "ChatRoom", null, Statu
48 if (participant == null) {
49 LogError("create_participant error");
50 return -1;
51 }
52
53 return 0;
54 }
55
56 private int CreatePublisher(){
57 if(participant != null) {
58 // Create the publisher
59 publisher = m_Factory.create_publisher(participant, "pub", null, StatusKind.STATUS_MASK_NONE);
60 if (publisher == null) {
61 LogError("create_publisher\n");
62 return -1;
63 }
64 }
65
66 return 0;
67 }
68
69 private int CreateSubscriber(){
70 if(participant != null) {
71 // Create the subscriber
72 subscriber = m_Factory.create_subscriber(participant, "sub", null, StatusKind.STATUS_MASK_NONE
73 if (subscriber == null) {
74 LogError("create_subscriber error");
75 return -1;
76 }
77 }
78
79 return 0;
80 }
81
82 private int CreateTopics(){
83 String type_name = null;
84
85 if(participant != null) {
86 // Register types before creating topic
Start Page
Welcome
*NetChat1
ChatRoom.java
Debug
Stack
Workbench
Testing
Class:ChatRoom
ChatRoom
Line: 69 Column: 36
CAP
NUM
SCRL
WAN

Worked Example

DLRL

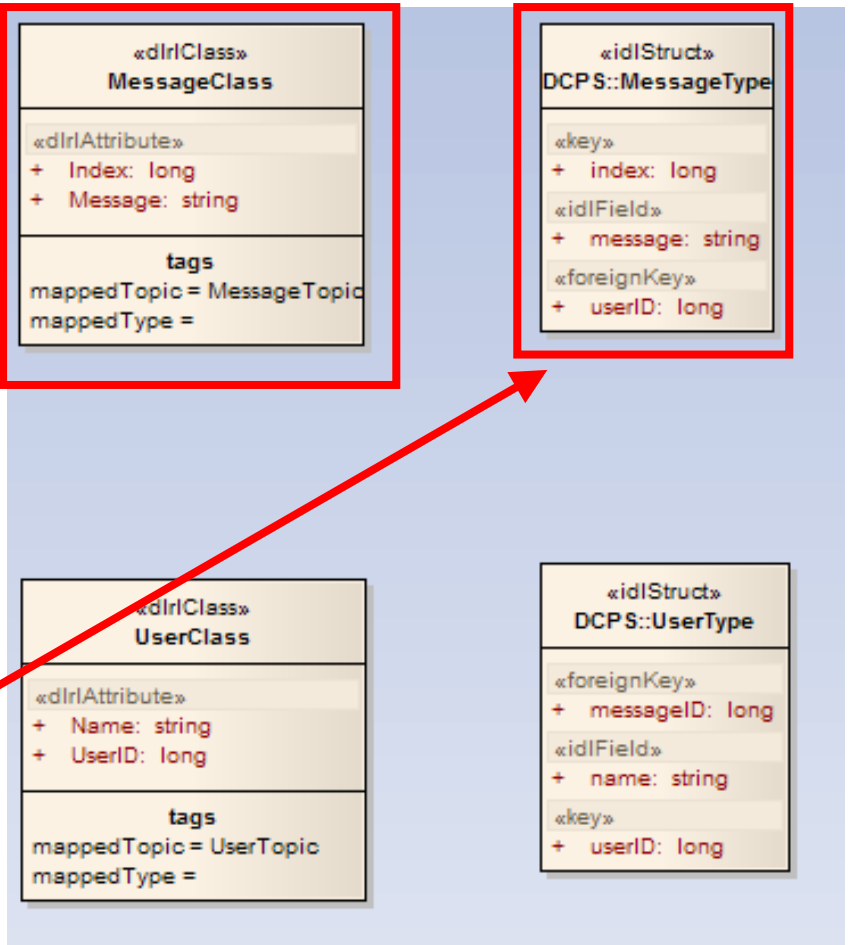
DLRL – Class & Type Mapping

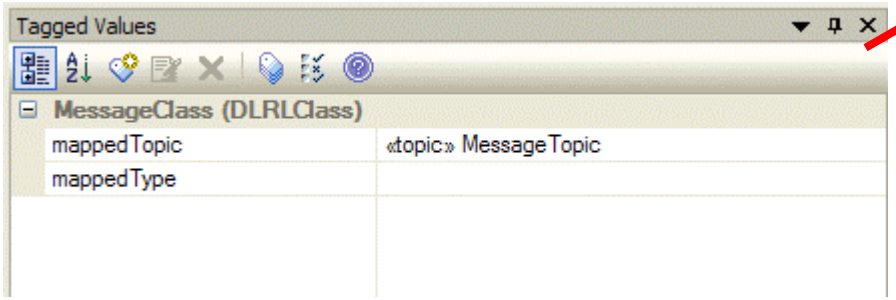
- dlrlClass
  - DLRL Class representing a subscribed DCPS Topic Type



DLRL – Class & Type Mapping

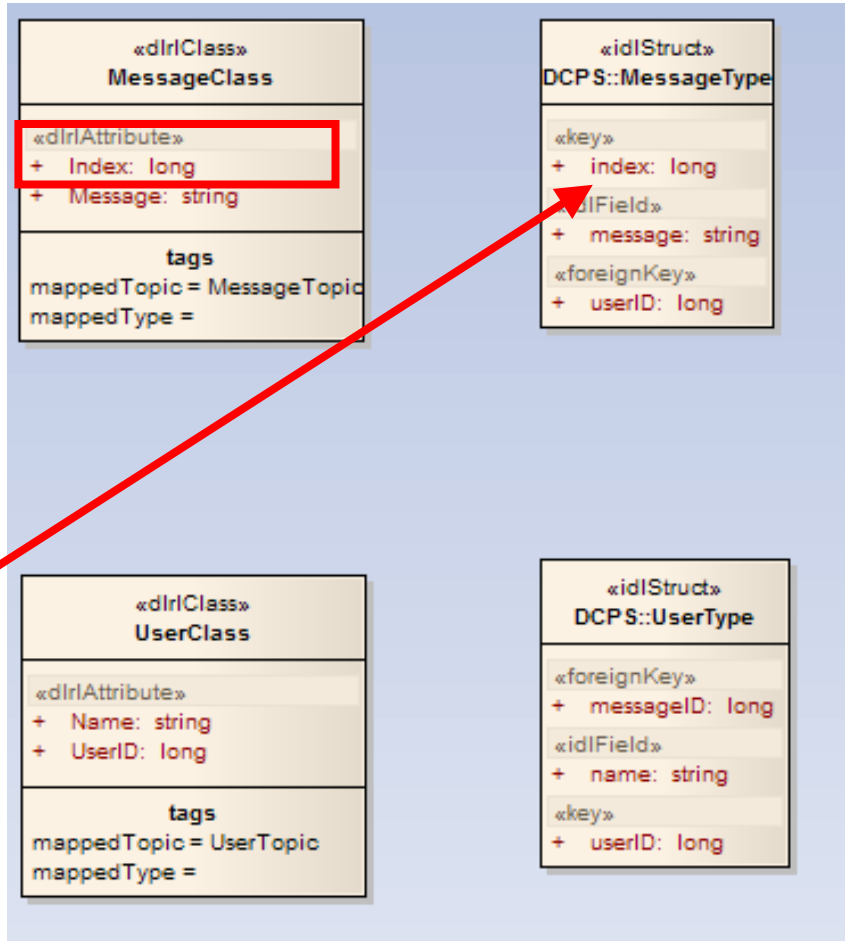
- dlrlClass
  - DLRL Class representing a subscribed DCPS Topic Type

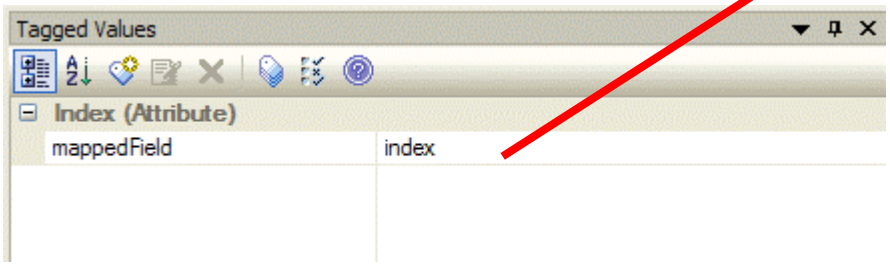




DLRL – Class & Type Mapping

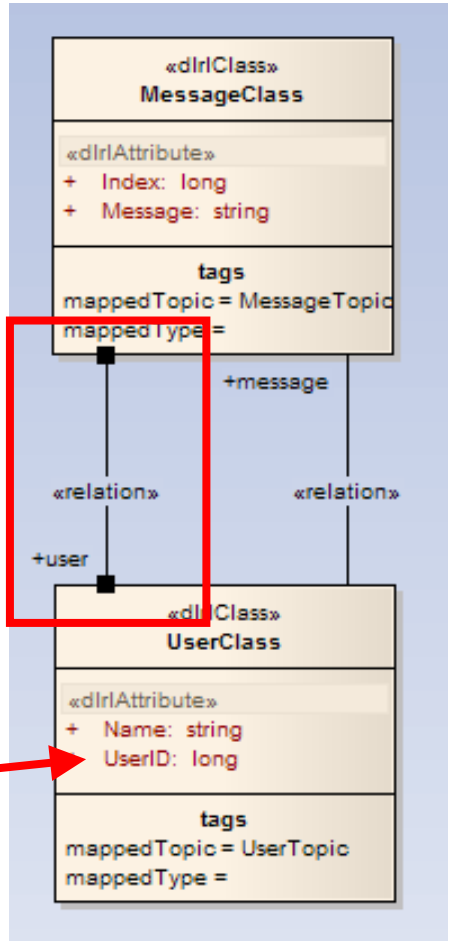
- dlrlAttribute
  - DLRL Attribute representing mapped DCPS Type fields

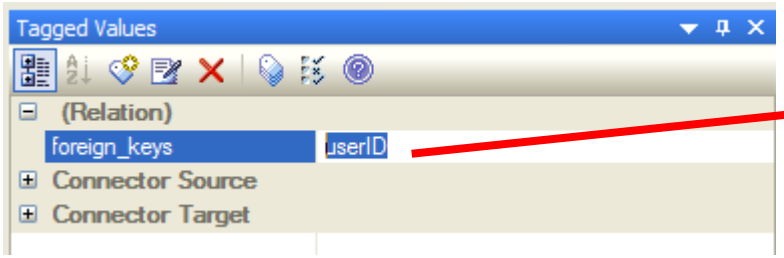




DLRL – Class & Type Mapping

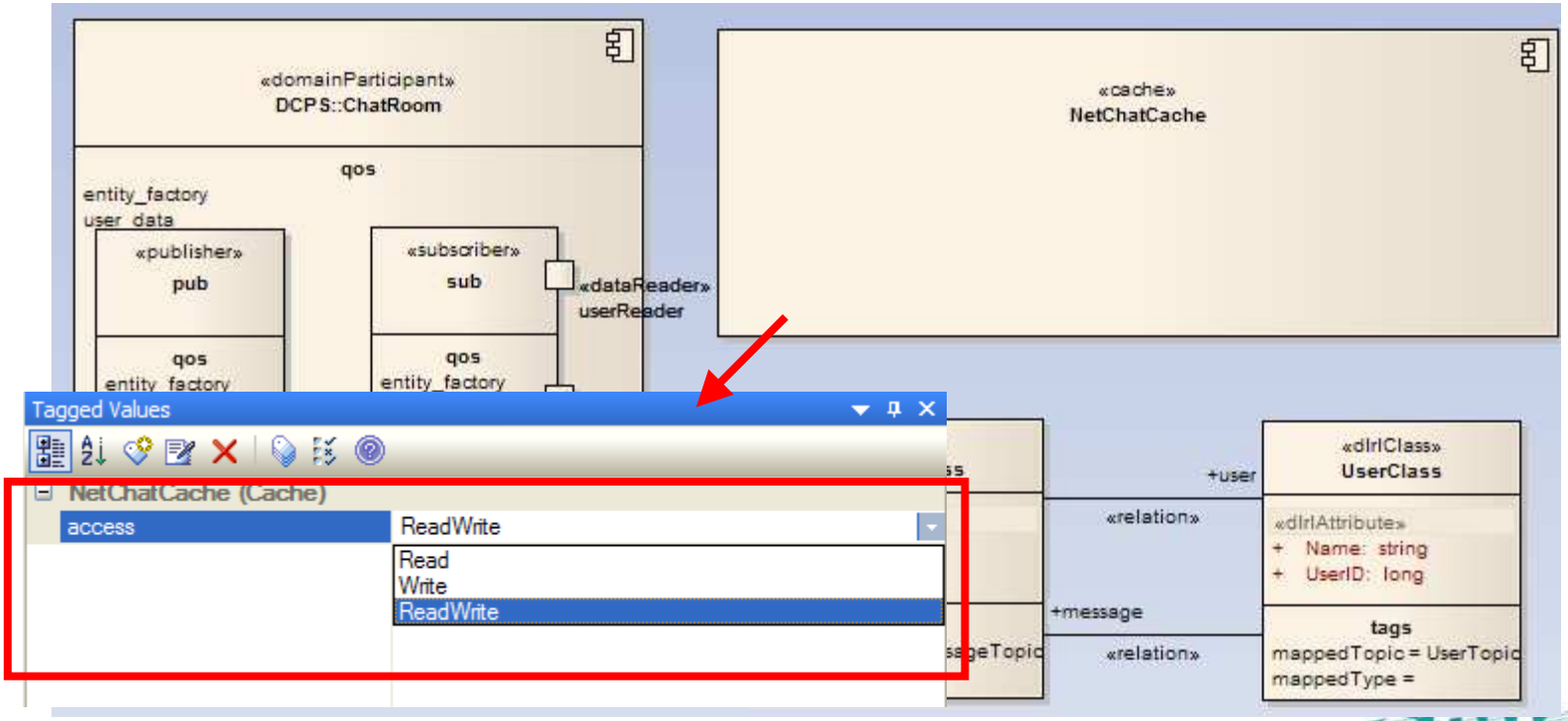
- relation
  - Association used to aggregate multiple classes using DLRL foreign keys





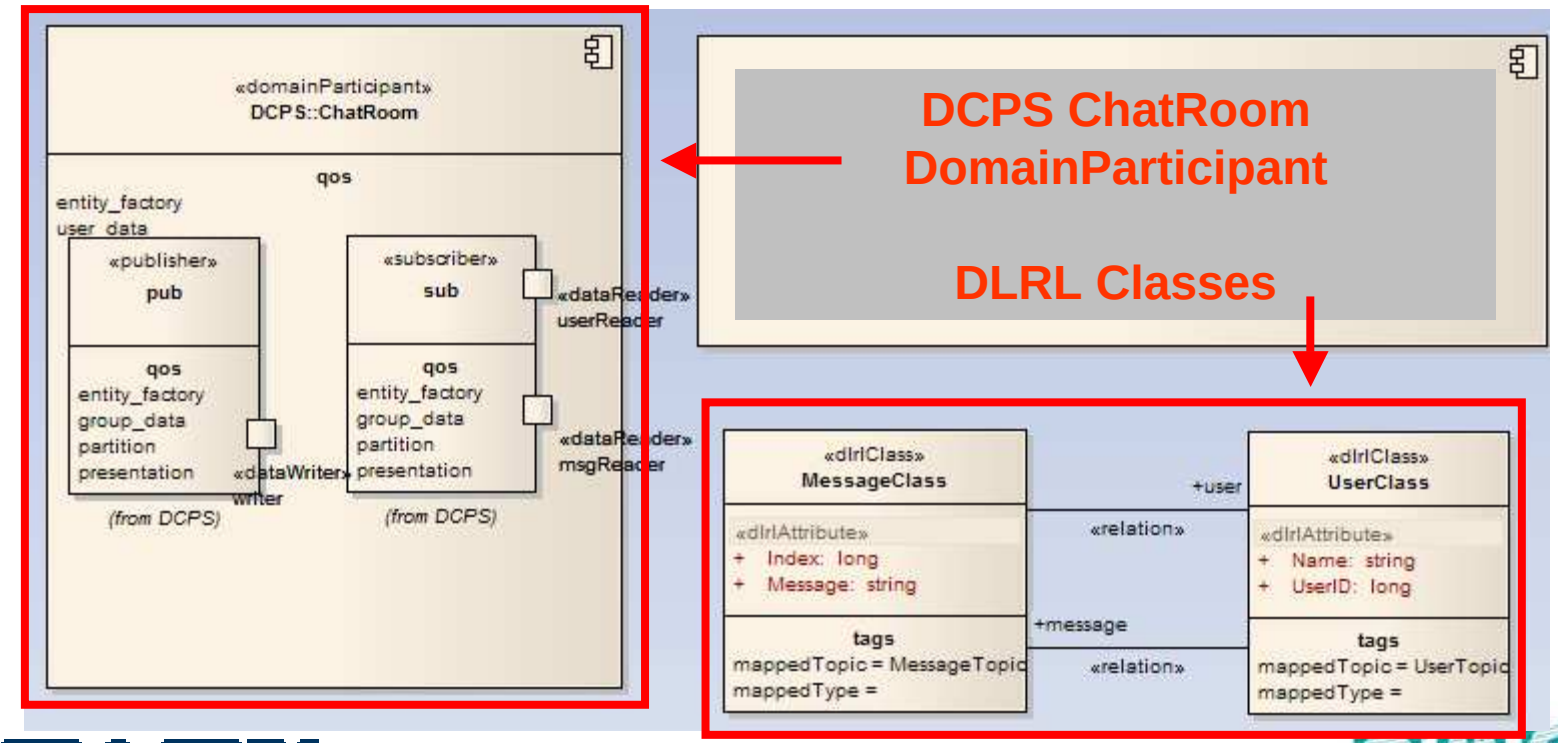
DLRL – Local Reconstruction

- Cache
  - Describes a DLRL cache entity used to provide dlrl class access to the user



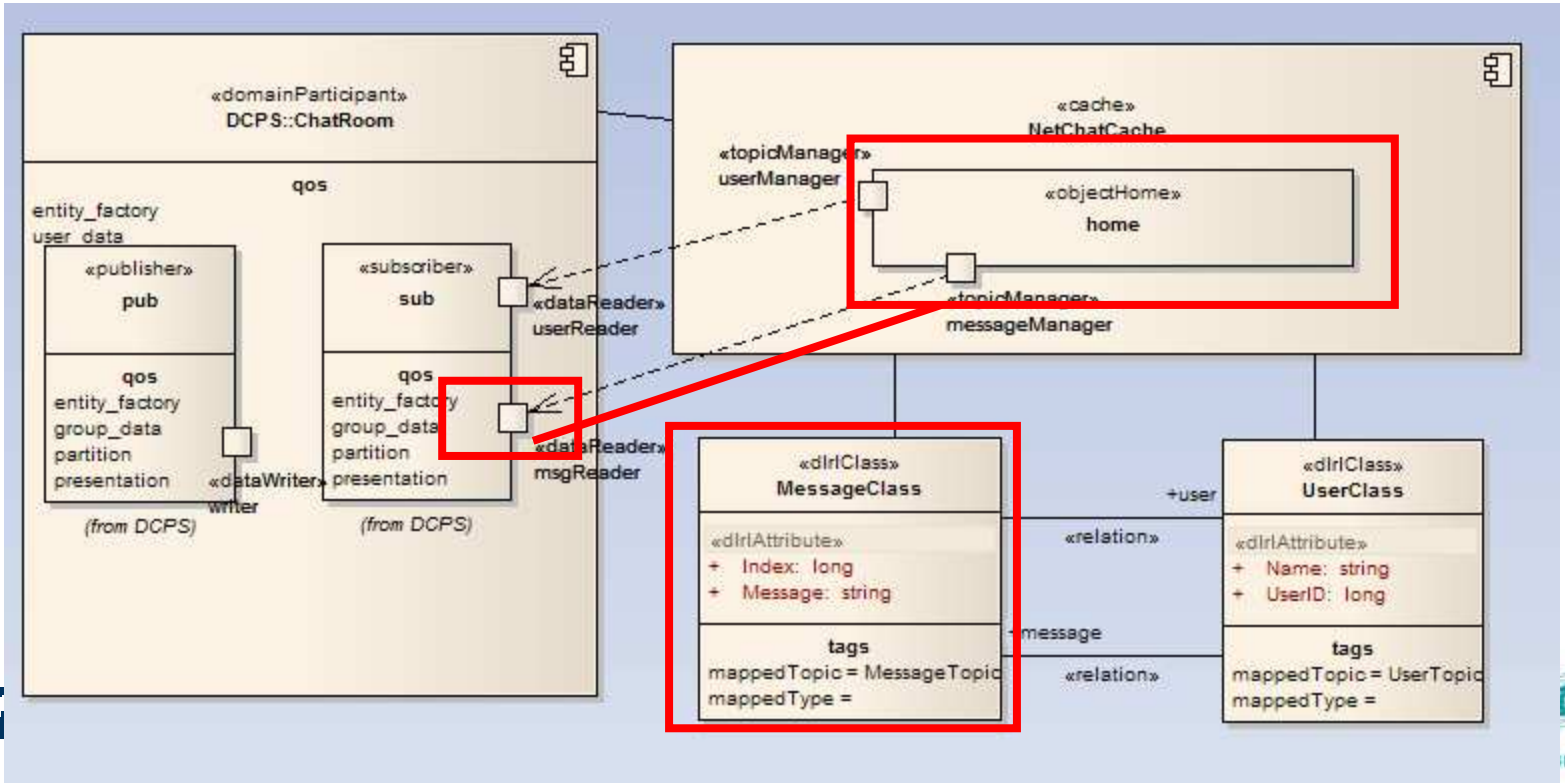
DLRL – Local Reconstruction

- Cache
  - Describes a DLRL cache entity used to provide dlrl class access to the user



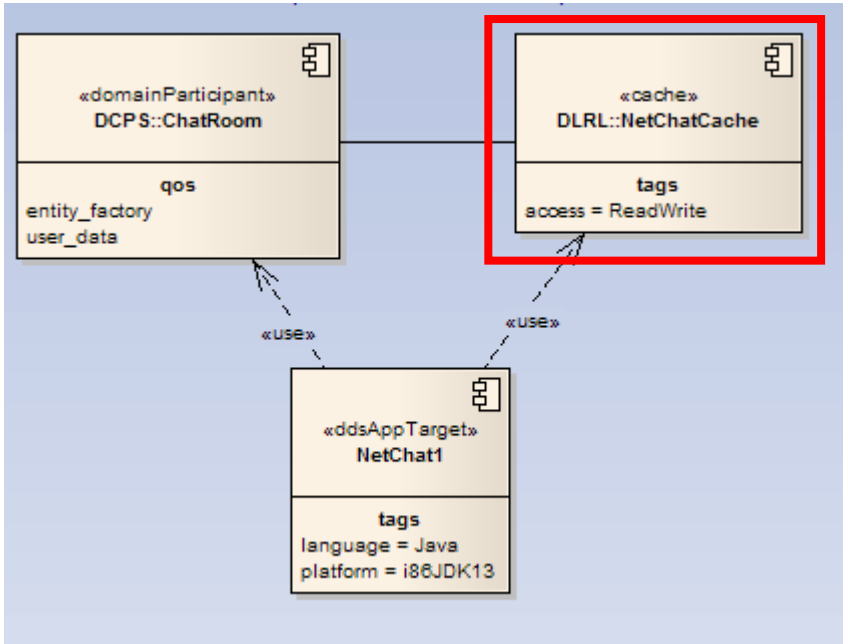
DLRL – Local Reconstruction

- objectHome, topicManager
  - Binds the cache to DataReaders to access the specific DCPS Topic, Types



Application Targets

- ddsAppTarget
  - Binds one or more DomainParticipants to a PSM configuration
  - Binds at most one DLRL cache to the PSM configuration



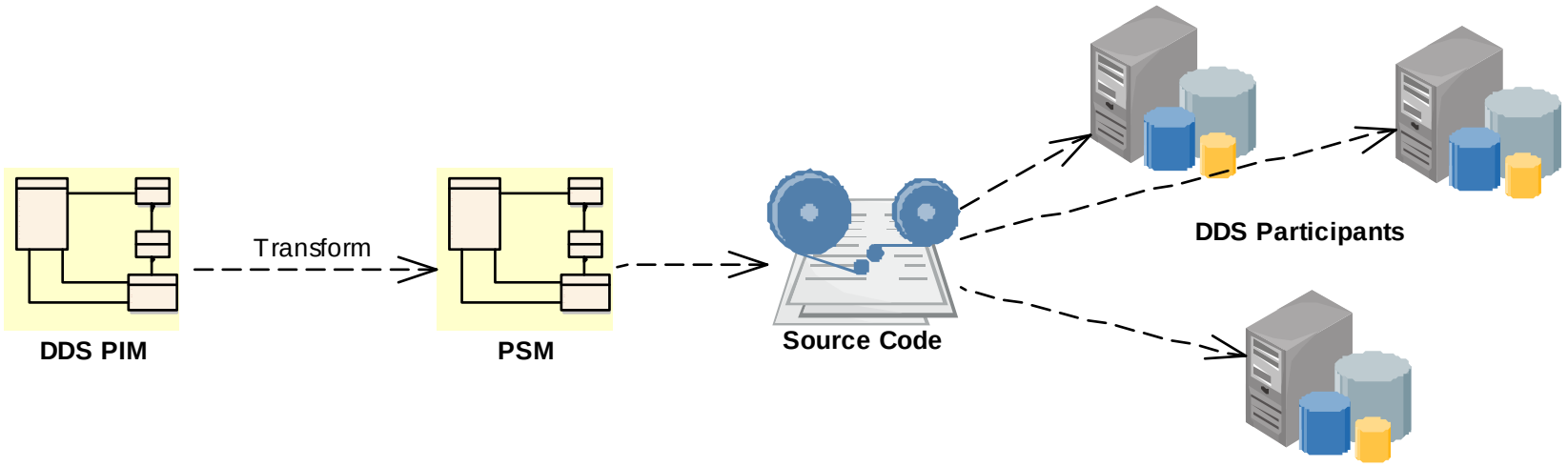
Conclusion & Wrap Up

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

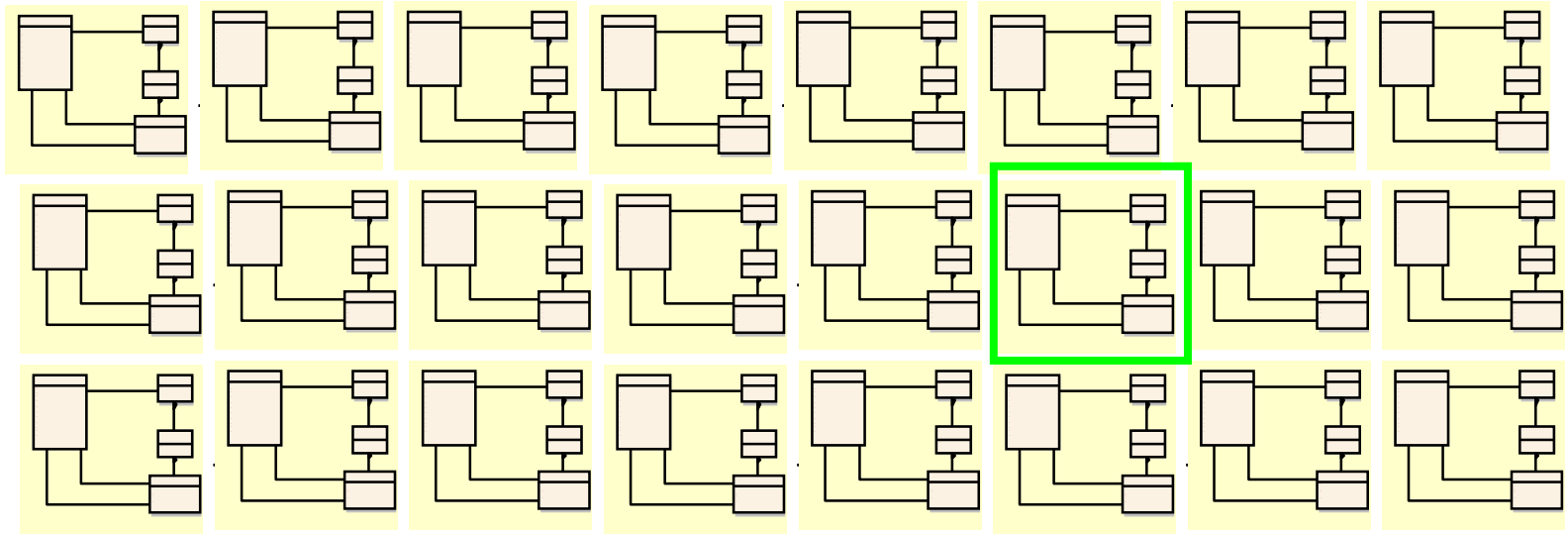
Other Applications

- Not just a DDS architecture description
- Not just a PIM



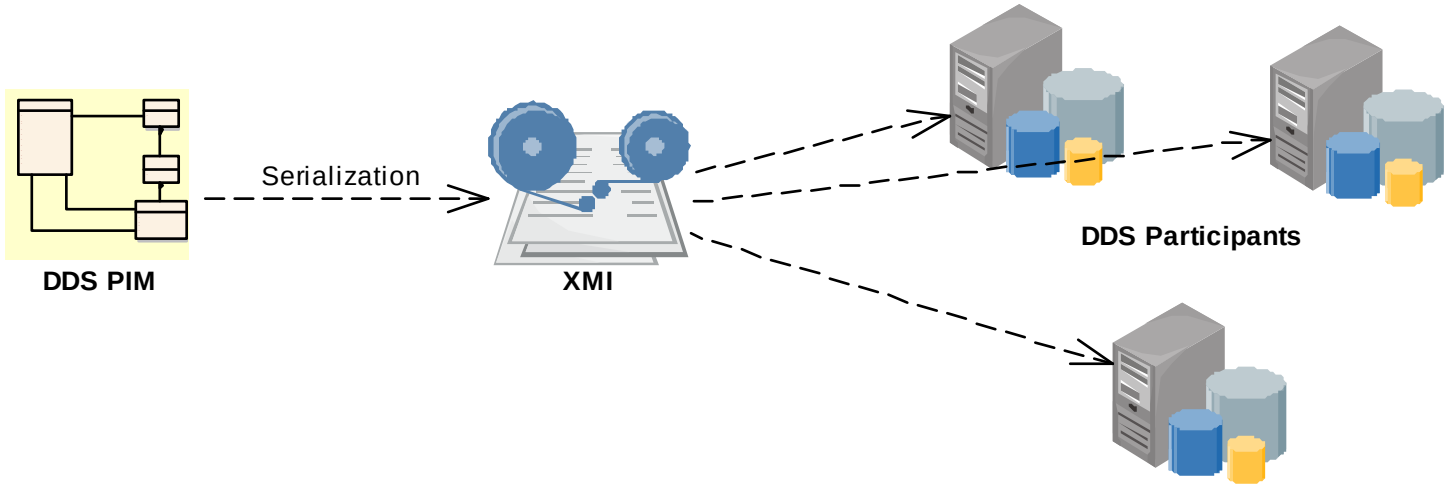
Other Applications

- Model Driven Architecture
  - UML based DSL
  - Interoperable with SysML, SOAML, UPDM, MARTE, etc…
  - Integrated, traceable ‘architectable’
  - Part of the ‘big picture’



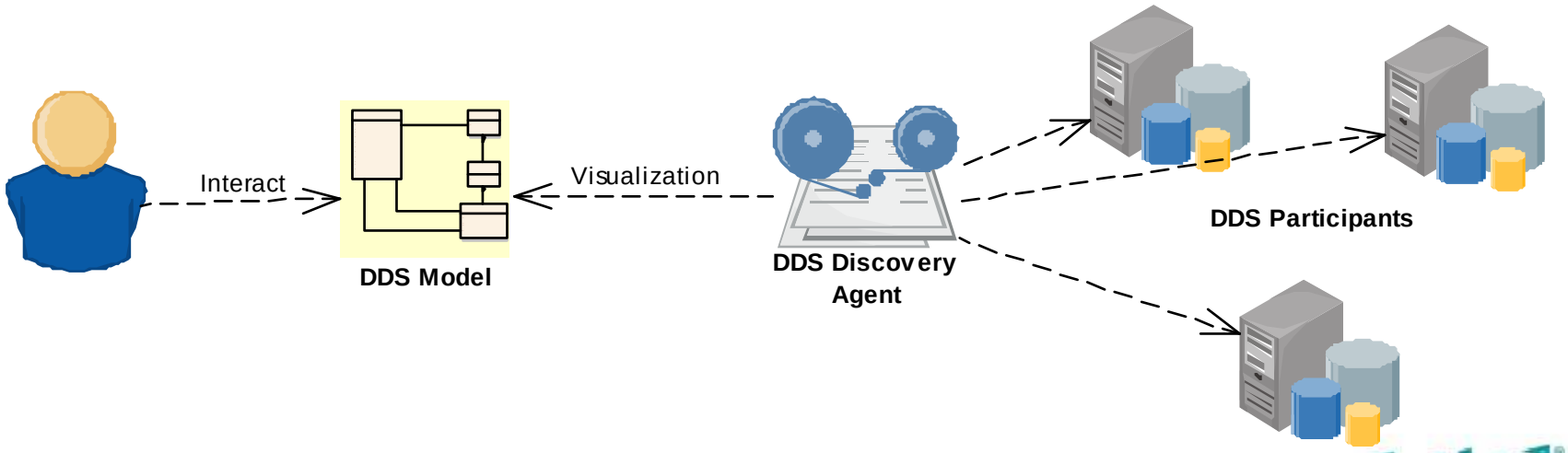
Other Applications

- XMI Serialization → Direct Deployment
  - XMI Document describes the DDS application configuration with Participants, Topics, QoS, etc
  - Configuration loaded by runtime to configure nodes
  - No source code



Other Applications

- Visual Deployment Interface
  - DDS discovery to create a DDS model which visualizes a running deployment
  - Field Engineers interact with the DDS model to make changes to the deployment
  - Maintenance, re-engineering, documentation applications



Concluding Remarks

- UML Profile for DDS exemplifies the co-operation of multiple OMG standards to:
  - Overcome the real-world challenges of design complexity management
  - Provide turnkey rapid-development solutions for DDS applications
- Culmination of OMG's
  - Real-time distributed data middleware technology
  - UML extensibility (domain-specific languages)
  - Model-Driven Development / Architecture
  - XML Metadata Interchange specifications

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Concluding Remarks

- Next Steps
  - Complete the FTF submission
  - Final Publication & market adoption
  - RTF
- For More information
  - Contact us
    - sam.mancarella@sparxsystems.com
    - http://sparxsystems.com/dds
  - Visit the Sparx exhibit for more information & demo

SPARX SYSTEMS

OMG OBJECT MANAGEMENT GROUP

Thank you for your attention!

SPARX SYSTEMS

OMG
OBJECT MANAGEMENT GROUP