UML Profile for DDS

a tutorial for OMG Specification in Government Workshop (Real-Time & Embedded Systems)
July 13, 2009

Prepared by: Sam Mancarella (Sparx Systems)
Presented by: Angelo Corsaro (PrismTech)
Agenda

• Part 1 - Introduction
  – DDS Overview
  – Motivating the UML4DDS

• Part 2 – UML4DDS by Examples
  – DCPS
  – DLRL
  – Application Targets
  – MDA, PIM → PSM

• Part 3 - Conclusion
  – Other Applications
  – Concluding Remarks
  – Discussion
The OMG Data Distribution Service (DDS)

- **DDS v1.2 API Standard**
  - Language Independent, OS and HW architecture independent
  - **DCPS.** Standard API for Data-Centric, Topic-Based, Real-Time Publish/Subscribe
  - **DLRL.** Standard API for creating Object Views out of collection of Topics

- **DDSI/RTPS v2.1 Wire Protocol Standard**
  - Standard wire protocol allowing interoperability between different implementations of the DDS standard
High Performance Pub/Sub

- Fully distributed, Peer-to-Peer Communication
- No Single Point of Failure
- No Single Point of Bottleneck
- Multicast-enabled
- High performance and highly scalable
- High availability

The right data, at the right place, at the right time – All the Time.
Data-Centric Pub/Sub

- Distributed Relational Data Model
- Local Queries
- Continuous Queries / Content Based Subscriptions
- Windows
- Object/Relational Mapping
- Support for a subset of SQL-92

Data-Centric Features are built-in and don’t rely on an external DBMS
- Providing thus performance, scalability, and availability

Perfect Blend of Data-Centric and Real-Time Publish/Subscribe Technologies
Topics and Data-Centric Pub/Sub

- **Topics.** Unit of information exchanged between Publisher and Subscribers.

- **Data Types.** Type associated to a Topic must be a structured type expressed in IDL

- **Topic Instances.** Key values in a datatype uniquely identify a Topic Instance (like rows in table)

- **Content Awareness.** SQL Expressions can be used to do content-aware subscriptions, queries, joins, and correlate topic instances

```
struct TempSensor {
    long tID;
    float temp;
    float humidity;
};

#pragma keylist TempSensor tID

SELECT * FROM TempSensor t WHERE t.temp > 25
```
Distributed Relational Information Modeling

- Topic Keys can be used to identify instances as well as relationships
- Relationships can be navigated by relying on a subset of SQL 92
- One-to-many relationships can be captured using foreign keys
- Many-to-many relationships need to be modeled using a topics
- Keys can be represented by an arbitrary number of Topic fields
Object/Relational Mapping

- Automatically bridges the Object/Relational Impedance Mismatch
- Arbitrary object reconstructions
- Automatic Relationships Management
- Inheritance
- Local Operations
- Local/Distributed State
### Sample QoS Policies

- Rich set of QoS allow to configure several different aspects of data availability, delivery and timeliness
- QoS can be used to control and optimize network as well as computing resource
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
UML 4 DDS

• 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.
UML 4 DDS

• Beta Specification: mars/2008-06-18
• Joint Submission by:
  – PrismTech
  – Real Time Innovations Inc
  – Sparx Systems
• Request For Proposal: mars/2006-09-40
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
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 Report Due: August 2009
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 Entities (+ QoS)
    - Data Local Reconstruction Layer
  - Introduced 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
Designing DDS Systems
DDS Design Steps

- Designing DDS-based system can be decomposed in the following few simple steps:
  - **Step #1**: Define **Information Model**
  - **Step #2**: Associate **QoS** representing **key non-functional invariants** for your system with the **Information Model**
  - **Step #3**: Define Topics / Partition / Domain Mapping
  - **Step #4**: Identify Topic Readers/Writers
  - **Step #5**: Define QoS requirements for Readers/Writers
  - **Step #6**: Bind the model to a specific **PSM**
NetChat Overview

• Hypothetical, peer-to-peer network chat application
• Two Components:
  – “ChatRoom” DDS Dataspase 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
DDS Design Steps

- Designing DDS-based system can be decomposed in the following few simple steps:
  - **Step #1:** Define Information Model
  - **Step #2:** Associate QoS representing key non-functional invariants for your system with the Information Model
  - **Step #3:** Define Topics / Partition / Domain Mapping
  - **Step #4:** Identify Topic Readers/Writers
  - **Step #5:** Define QoS requirements for Readers/Writers
  - **Step #6:** Bind the model to a specific PSM
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
DDS Design Steps

- Designing DDS-based system can be decomposed in the following few simple steps:
  - **Step #1**: Define **Information Model**
  - **Step #2**: Associate **QoS** representing **key non-functional invariants** for your system with the **Information Model**
  - **Step #3**: Define Topics / Partition / Domain Mapping
  - **Step #4**: Identify Topic Readers/Writers
  - **Step #5**: Define QoS requirements for Readers/Writers
  - **Step #6**: Bind the model to a specific **PSM**
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
Designing DDS-based system can be decomposed in the following few simple steps:

– **Step #1:** Define **Information Model**
– **Step #2:** Associate **QoS** representing **key non-functional invariants** for your system with the **Information Model**
– **Step #3:** Define Topics / Partition / Domain Mapping
– **Step #4:** Identify Topic Readers/Writers
– **Step #5:** Define QoS requirements for Readers/Writers
– **Step #6:** Bind the model to a specific **PSM**
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
DDS Design Steps

- Designing DDS-based system can be decomposed in the following few simple steps:
  - **Step #1**: Define **Information Model**
  - **Step #2**: Associate **QoS** representing **key non-functional invariants** for your system with the **Information Model**
  - **Step #3**: Define Topics / Partition / Domain Mapping
  - **Step #4**: Identify Topic Readers/Writers
  - **Step #5**: Define QoS requirements for Readers/Writers
  - **Step #6**: Bind the model to a specific **PSM**
DCPS Domain & Entities

- Added Publisher, Subscriber Entities
DCPS Domain & Entities

• Added DataReaders, DataWriters Entities
DDS Topics connected to DataReaders & DataWriters
DDS Design Steps

- Designing DDS-based system can be decomposed in the following few simple steps:
  - **Step #1**: Define **Information Model**
  - **Step #2**: Associate **QoS** representing **key non-functional invariants** for your system with the **Information Model**
  - **Step #3**: Define Topics / Partition / Domain Mapping
  - **Step #4**: Identify Topic Readers/Writers
  - **Step #5**: Define QoS requirements for Readers/Writers
  - **Step #6**: Bind the model to a specific **PSM**
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.
Worked Example

DLRL
Object/Relational Mapping

- Automatically bridges the Object/Relational Impedance Mismatch
- Arbitrary object reconstructions
- Automatic Relationships Management
- Inheritance
- Local Operations
- Local/Distributed State
Concepts

The mechanism at the foundation is a managed Object Cache:

- An Object Cache can be populated by different types (classes) of Objects.
- Each object class has its own manager called an ObjectHome.
  - They can inform the application about object creation/modification/deletion.
- Classes may contain navigable relationships to other classes.
- Each Object class may inherit from 1 other Object class.
DLRL: How does it Work?

Processing Updates

- ‘vanilla DDS’: updates arrive as separate samples at separate times.
- DDS Object Technology: updates are processed in ‘update rounds’:
  - ObjectHomes read all available samples from the DDS information backbone and update their corresponding objects in the Cache accordingly.
  - Objects are allocated once and their state is ‘overwritten’ on subsequent updates.
  - Therefore an Object always contains the latest available state.
  - Push mode: update rounds start when new data arrives. The application gets notified by Listeners.
  - Pull mode: the application can determine the start of each update round manually.
Notification Patterns

Notifying the application
The Object Caches offer two ways to notify an application of incoming information:

- Listeners can be triggered for each modification of an object’s state.
  - Listeners registered to the Cache indicate the start and end of each update round.
  - Listeners registered to the ObjectHome pass each modification back as a callback argument.
  - With a simple mechanism that can be translated into callbacks for Listeners on individual objects.
- It is possible to get a separate list of all objects that have been created/modifed/deleted in the current update round.
Using snapshots
Some applications want to be able to store temporal ‘snapshots’:

- A CacheAccess can be used to contain a temporal graph of objects.
  - The graph is identified by a so-called ‘cloning contract’.
- Objects must physically be cloned from Cache to CacheAccess.
- A CacheAccesses is not automatically kept in sync with the main Cache.
- A ‘refresh’ operation can be used to resync the contents of CacheAccess with the contents of the main Cache.
Some applications want to be able to modify or create certain objects:

- An initial set of Objects may be cloned into a writeable CacheAccess.
- Available objects may then be modified locally.
- New objects can be created in the CacheAccess as well.
- The ‘write’ operation instructs the ObjectHomes to write any modifications into the system.
Creating and managing Selections
A Selection mechanism can keep track of subsets of information:

- Selections are created and managed by the ObjectHomes.
- A Criterion plugged into a Selection determines the boundaries of a subset:
  - A QueryCriterion determines boundaries based on an SQL statement.
  - A FilterCriterion determines boundaries based on user-defined callback filters.
- Selections can notify the application when objects enter and leave it.
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
Other Applications

• Not just a DDS architecture description
• Not just a PIM
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
Concluding Remarks

• Next Steps
  – Complete the FTF submission
  – Unleash to the world – promote industry adoption, drive market demand

• For More information
  – Contact us
    • sam.mancarella@sparxsystems.com
    • angelo.corsaro@prismtech.com
  – Visit the Sparx exhibit for more information & demo
Thank you for your attention!