

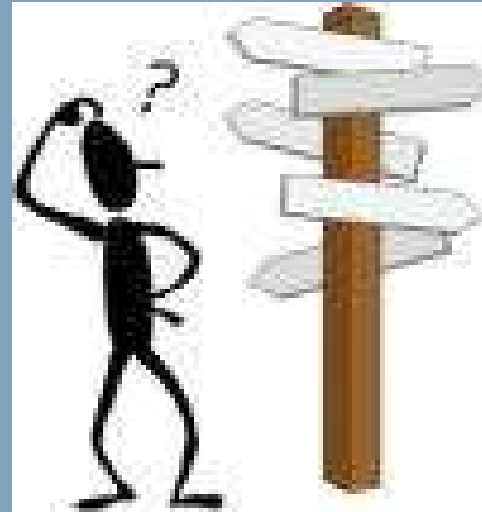


# Applying Model Driven Development to the DDS Domain

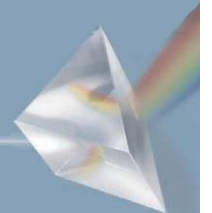
Bruce Trask  
Hans van 't Hag

# Complex Publish Subscribe Systems

- ▶ Object Oriented
- ▶ Component Based
- ▶ Multithreaded/MultiProcess
- ▶ Real-time
- ▶ Embedded
- ▶ C++/Java/Ada/VHDL
- ▶ Platform Independent
- ▶ High Performance
- ▶ Heterogeneous
- ▶ Distributed
- ▶ Vital
- ▶ Secure
- ▶ Fault Tolerant
- ▶ Portable
- ▶ Standardized
- ▶ Declarative
- ▶ Imperative
- ▶ Dynamic

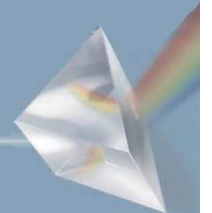


So what's the big deal?  
Each one *by itself* is difficult,  
let alone doing them all *at the same time*



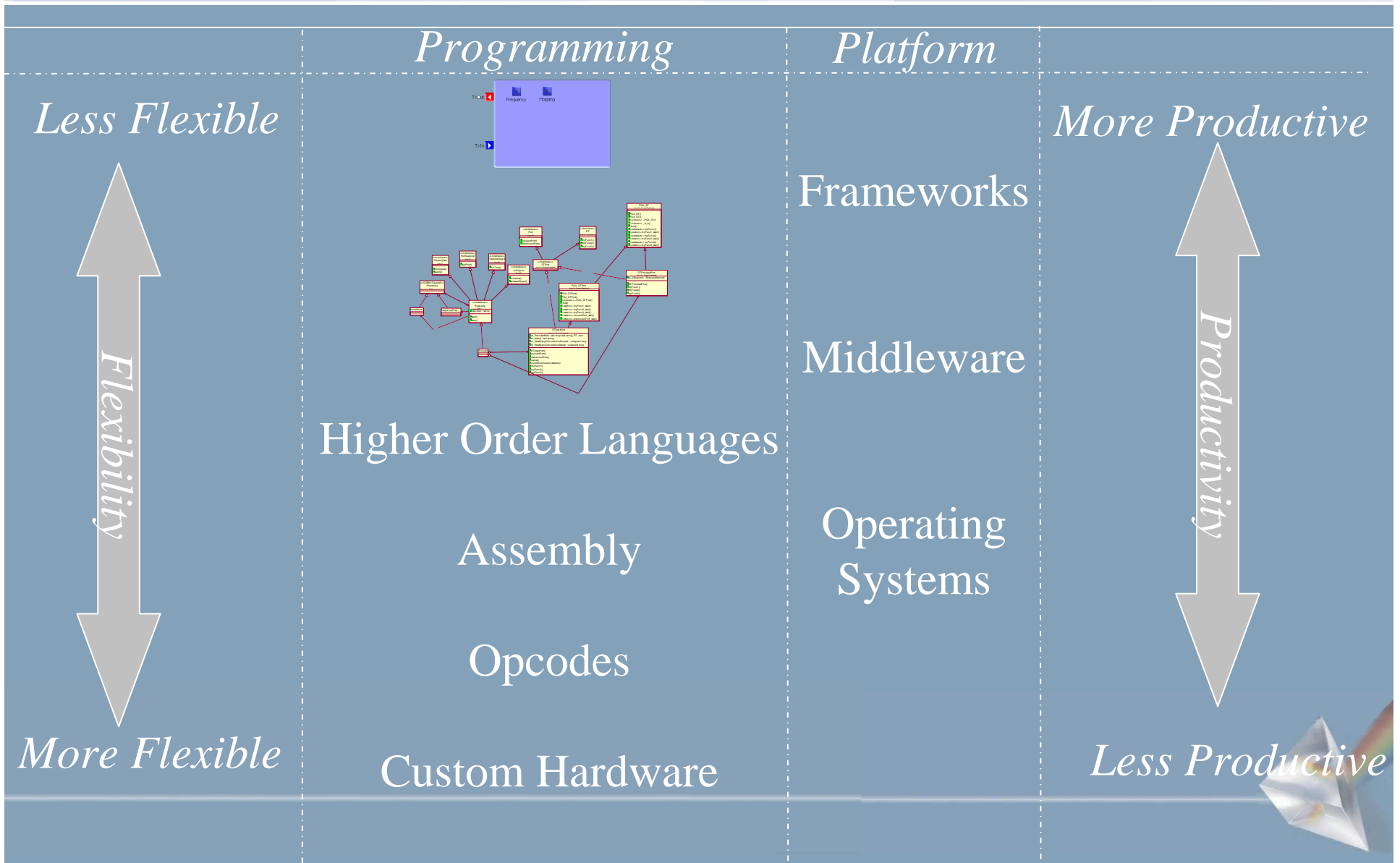
# Computer Science, Software Engineering

- ▶ “Computer Science is the discipline that believes all problems can be solved with one more layer of indirection”<sup>1</sup>
- ▶ Formalize the level of indirection
- ▶ Provide ways to program at this new level
- ▶ Provide automatic ways to get back to executable lower levels of abstraction
- ▶ Standardize
- ▶ Optimize

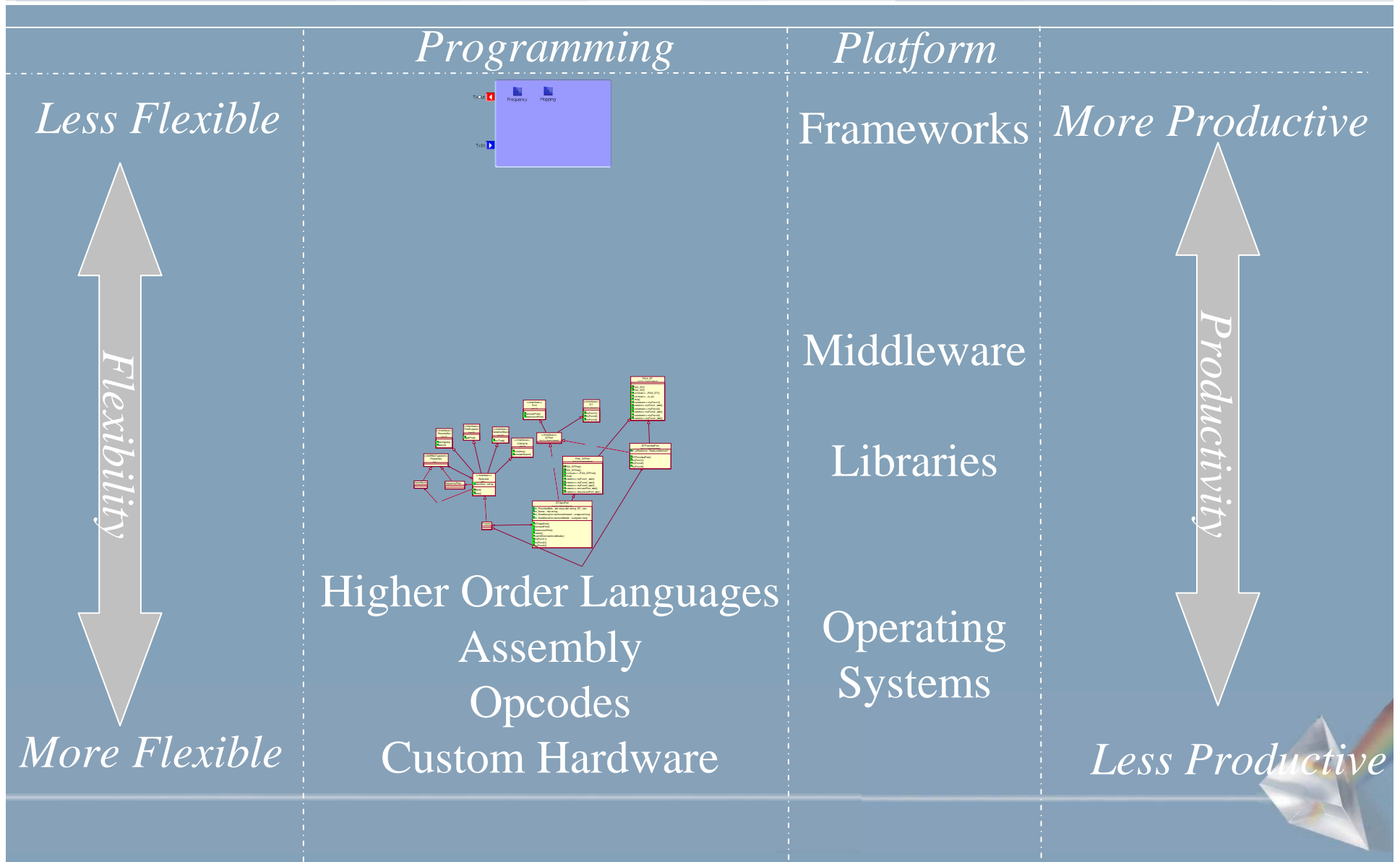


<sup>1</sup> Dennis DeBruler, *Refactoring*, Fowler et. al. Addison Wesley 1999

# Levels of Abstraction



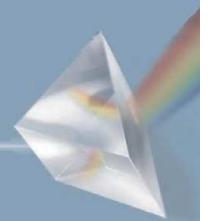
# Levels of Abstraction – more accurate



# The Problem

- ▶ General Purpose programming languages have not kept pace with the growth of platform complexity<sup>1</sup>
- ▶ Families of systems are the order of the day
- ▶ Systems and requirements have become more complex

<sup>1</sup>*Model Driven Engineering, Douglas C Schmidt IEEE Computer Feb 2006*

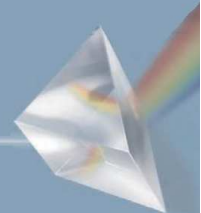
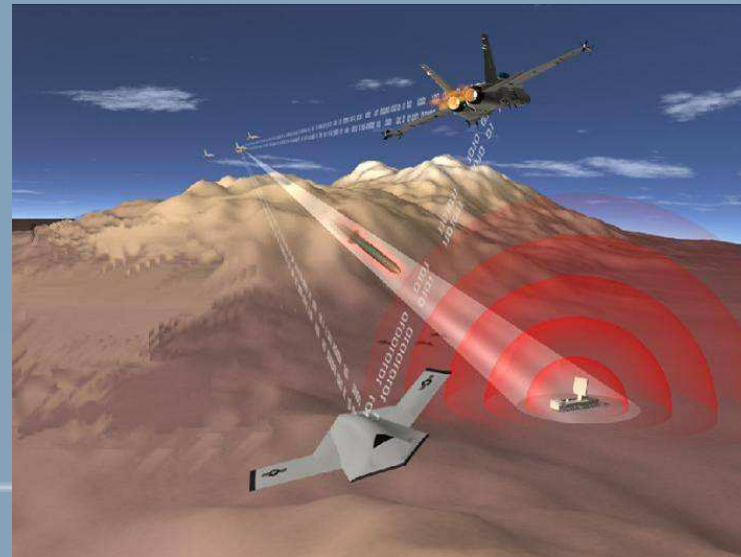


# Families of Systems



▶ <http://www.army.mil/fcs/>

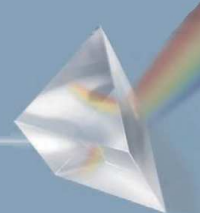
# More Complex Systems and Requirements





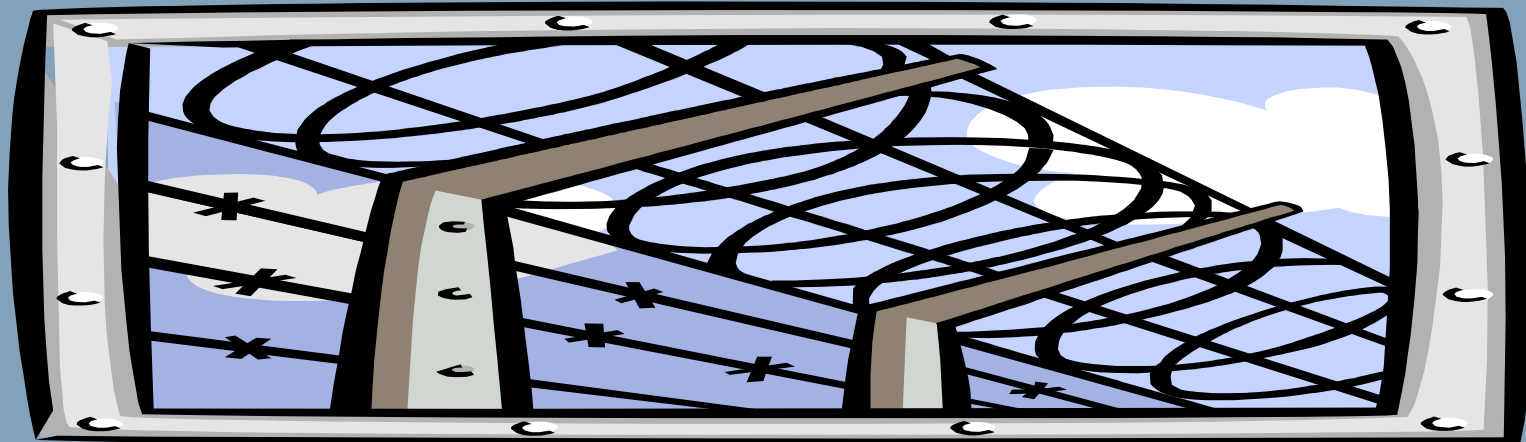
# Traditional tools

- ▶ UML
  - ▶ Insufficiently domain specific
  - ▶ Some say not semantically precise enough to be machine processed
  - ▶ Not much different than programming with OO languages
- ▶ 3GL
  - ▶ C++, Java, Ada etc.
  - ▶ Insufficiently domain specific
  - ▶ Not at a sufficient level of abstraction
- ▶ Frameworks, libraries and components
  - ▶ So many
  - ▶ Complicated
  - ▶ Manipulated via 3GL languages
- ▶ DDS-tools
  - ▶ IDL pre-processing: Insufficiently domain specific

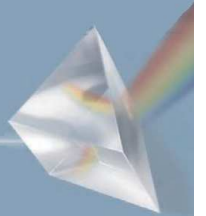


# Traditionally – a separation

*modeling*

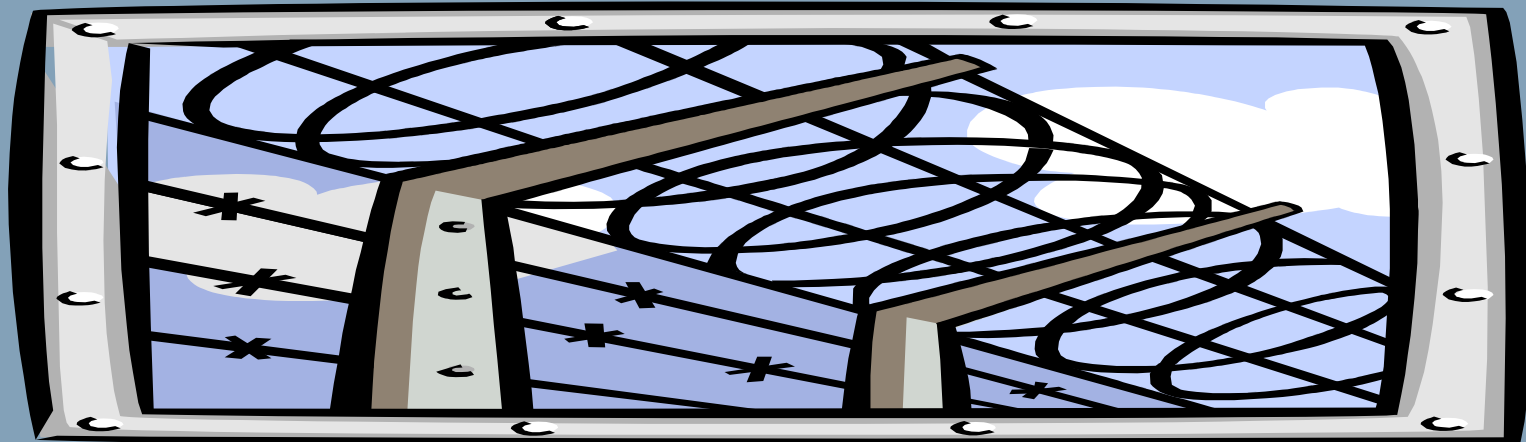


*Programming*

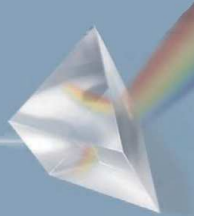


## Traditionally – a separation

*Systems Engineering*



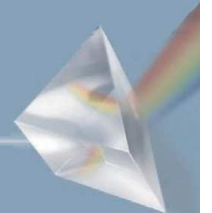
*Software Engineering*



# Critical Innovations

- ▶ Technologies that enable systematic vs. ad hoc reuse and enable the systematic development of families of systems
- ▶ Component technologies
- ▶ Domain Specific Languages<sup>1</sup>

<sup>1</sup>*Software Factories, Greenfield et. al Wiley 2005*



# Model Driven Development

*Bringing modeling and programming together*

*modeling*



*Programming*

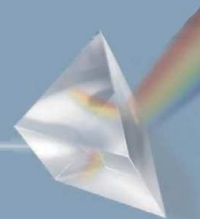
*“Instead of forcing a separation of the high-level engineering/modeling work from the low-level implementation programming. It brings them together as two well-integrated parts of the same job”<sup>1</sup>*

Eclipse modeling Framework, Budinsky et.al Addison Wesley 2004

## Some ramifications

- ▶ modeling and programming blend into one activity
- ▶ Models become *development* artifacts as well not just *design* artifacts
- ▶ Domain Specific Models can be machine processed and thus integrate well with generators
- ▶ Developers can “program in terms of their design intent rather than the underlying, computing environment”<sup>3</sup>

<sup>3</sup> IEEE Computer Magazine, February 2006 Model Driven Engineering, Douglas C. Schmidt



# Model Driven Development

*Bringing Systems and Software together*

*Systems  
Engineering*



*Software  
Engineering*

*“Instead of forcing a separation of the high-level engineering/modeling work from the low-level implementation programming. It brings them together as two well-integrated parts of the same job”<sup>1</sup>*

Eclipse modeling Framework, Budinsky et.al Addison Wesley 2004

## Some ramifications

- ▶ Increased cross discipline collaboration
- ▶ Speaking the same language
- ▶ Flattens learning curves

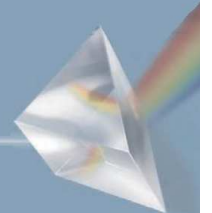
<sup>3</sup> IEEE Computer Magazine, February 2006 Model Driven Engineering, Douglas C. Schmidt



## How is this done?

- ▶ Domain Specific modeling Languages<sup>2</sup>
- ▶ Domain Specific Editors
- ▶ Transformation engines and generators
- ▶ Combine the above three – Called Model Driven Development (a.k.a Model Driven Engineering, or Model Integrated Computing)

<sup>2</sup> IEEE Computer Magazine, February 2006 Model Driven Engineering, Douglas C. Schmidt



# Domain Specificity



*Domain Independent*



*Domain Specific - Tools*



*The task at hand*



# The steps

## In general

*Isolate the abstractions and how they work together*

*Create a formalized grammar for these - DSL*

*Create a graphical representation of the grammar – GDSL*

*Provide domain-specific constraints – GDSCL, DSCL*

*Attach generators for necessary transformations*

## In our domain

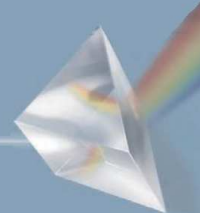
*The DDS specification*

*Create a formalize DDS meta-model*

*Create a DDS specific graphical tool*

*Program into the tool the constraints*

*C++, C and Java generators*

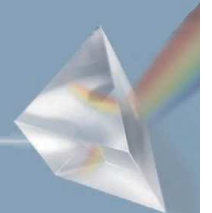




# Applying MDD to the DDS Domain

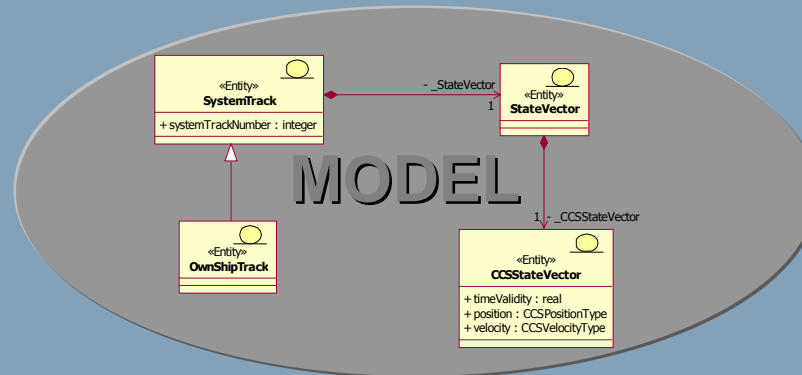
# Contents

- ▶ DDS – An Information Centric Approach
- ▶ Introducing the Example
- ▶ Information modeling
- ▶ Application modeling
- ▶ Deployment modeling



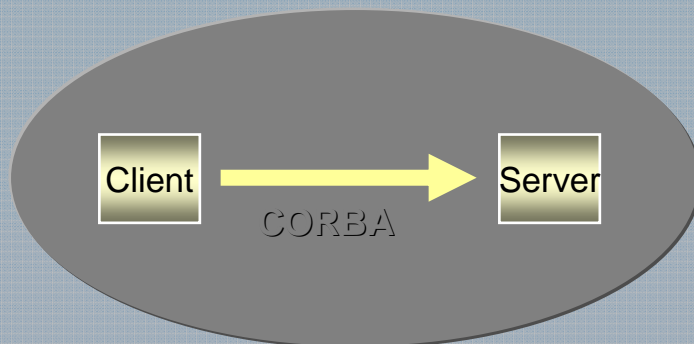
# DDS: An Information Centric Approach

## Information MEANING



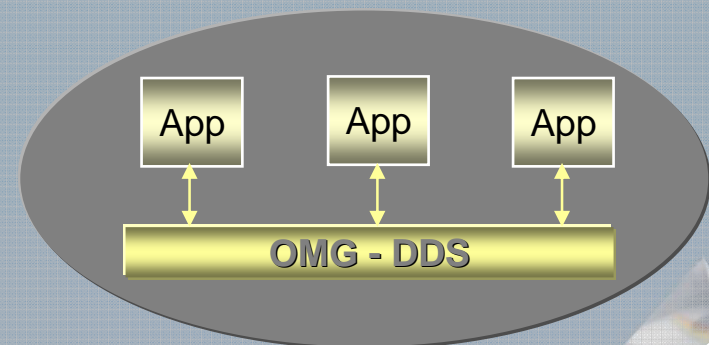
Analysis

## Server BEHAVIOR

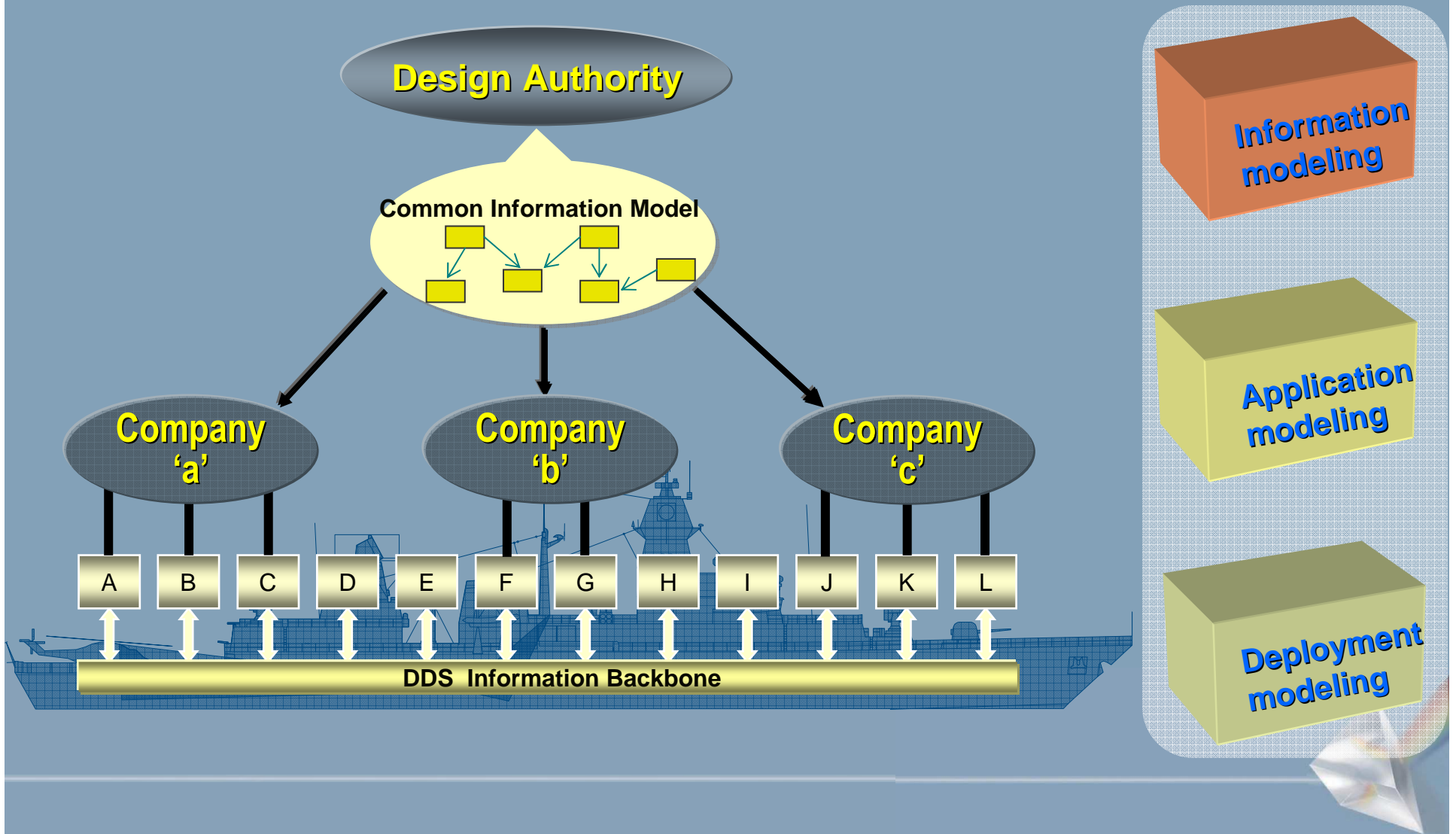


Design

## Data AVAILABILITY



# MDD-aspects in 'Coalition Based Development'



# Introducing The Example



## SENSOR PROCESS

- ▶ Optical sensor
- ▶ Scans the environment
- ▶ Produces 'Tracks'
- ▶ Position of 'objects'
- ▶ Reports '*pointTrack*'



## CLASSIFICATION PROCESS

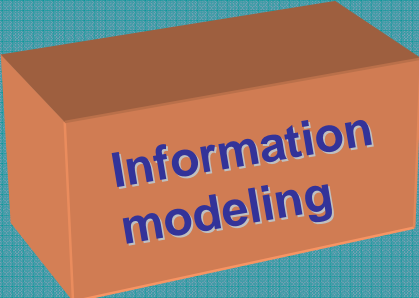
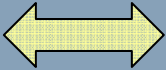
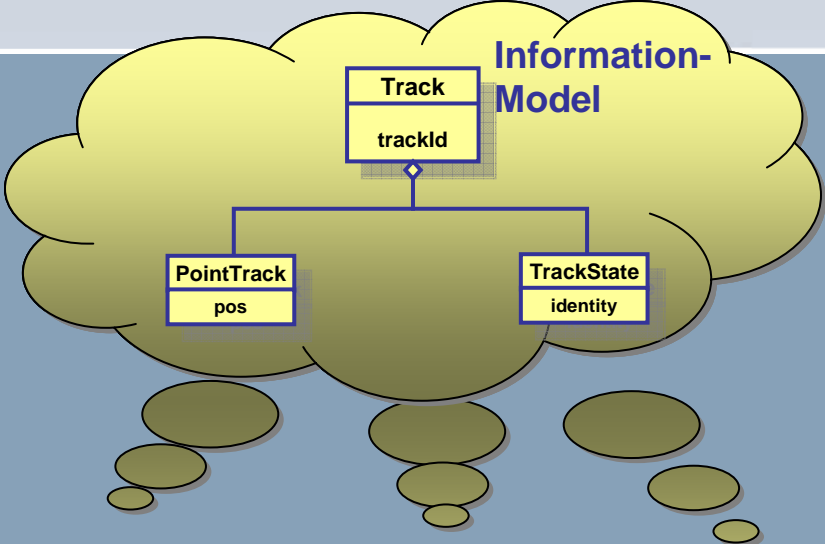
- ▶ Classifies tracks
- ▶ Determines their identity
- ▶ Analyses the trajectories
- ▶ Determines hostility
- ▶ Reports '*trackState*'



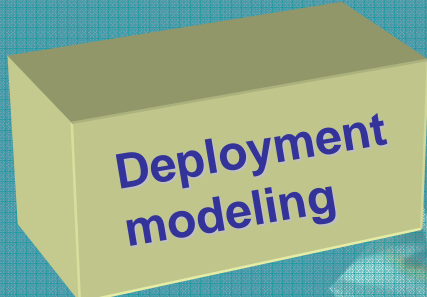
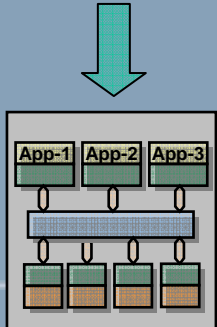
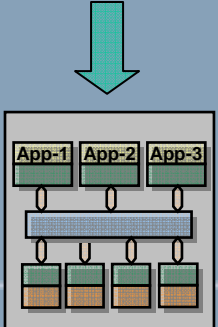
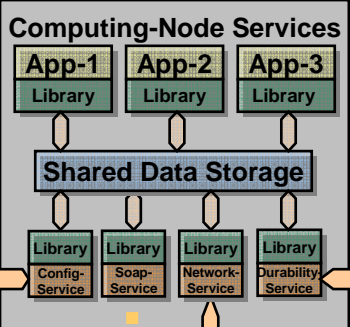
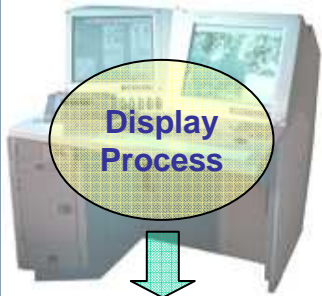
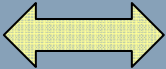
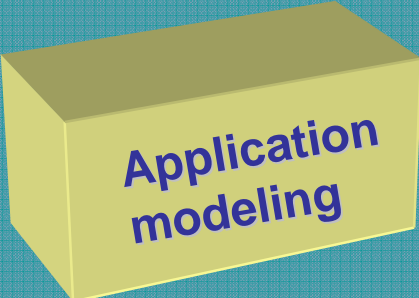
## DISPLAY PROCESS

- ▶ Displays track info
- ▶ Both position & identity
- ▶ Raises alerts
- ▶ Requires '*pointTrack*'
- ▶ Requires '*trackState*'

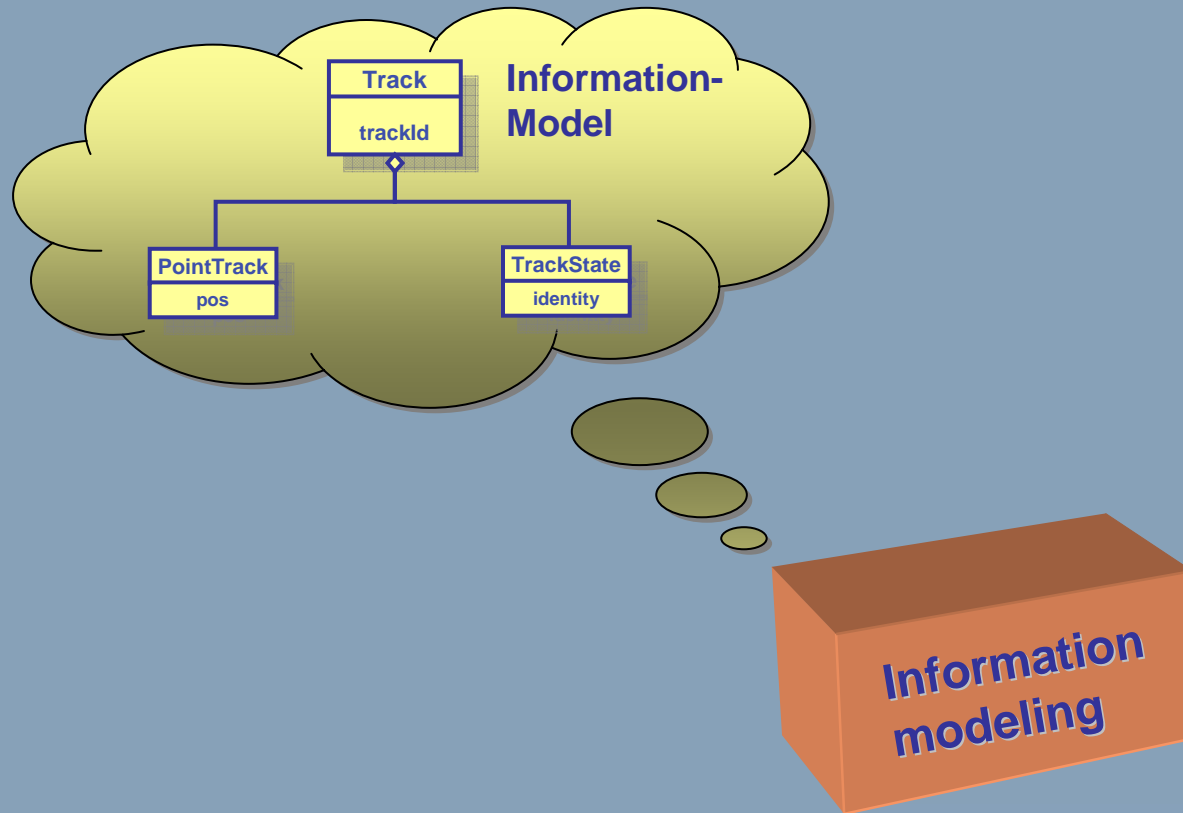
# MDD and the Example



MDD Tool suite



# (1) Information modeling



# Information modeling: Scope & Purpose

## ▶ Language

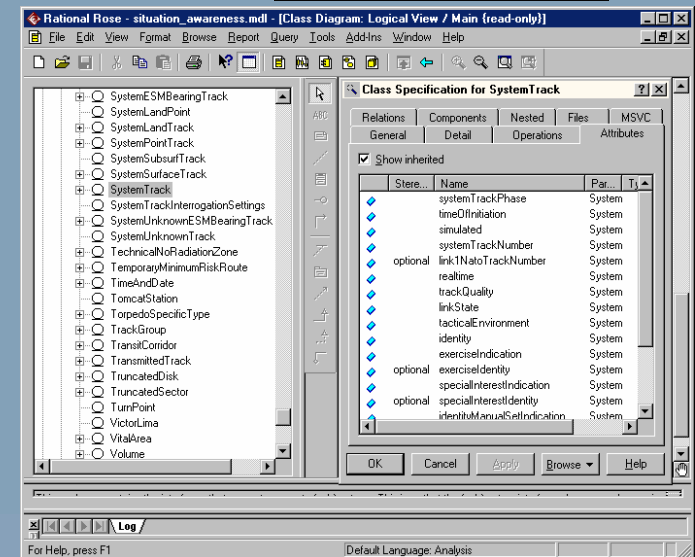
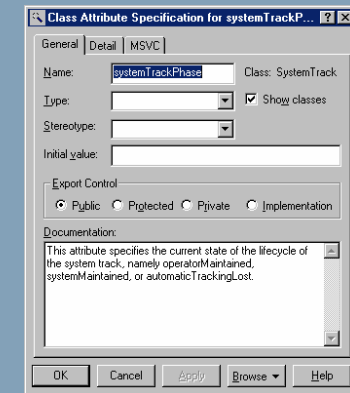
- ▶ Define the 'spoken language' in the system
  - ▶ Requirements phase: to talk with the customer
  - ▶ Development phase: to talk between components
- ▶ DDS-Domain specific 'Topics' (Type + QoS)

## ▶ Editor

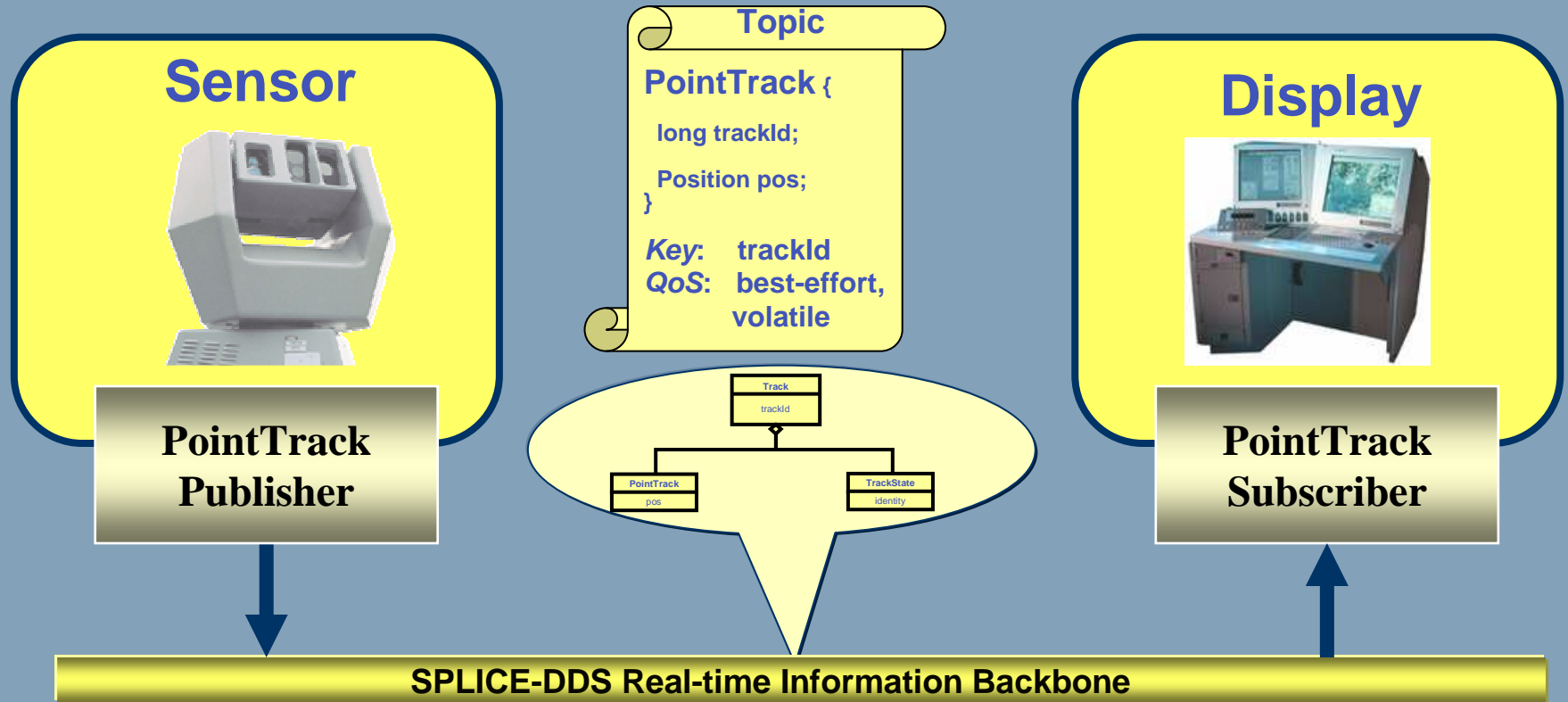
- ▶ Semantics: UML ('annotated')
- ▶ Syntax: IDL (optional MDD-input if predefined)
- ▶ Behavior: QoS (reliability, urgency, durability)

## ▶ Generator

- ▶ Types (IDL) from UML (or direct IDL-import)
- ▶ Topics from Types
- ▶ DDS-entities
  - ▶ Topic-creation, QoS-setup, typed readers/writers



# Information modeling: The Example



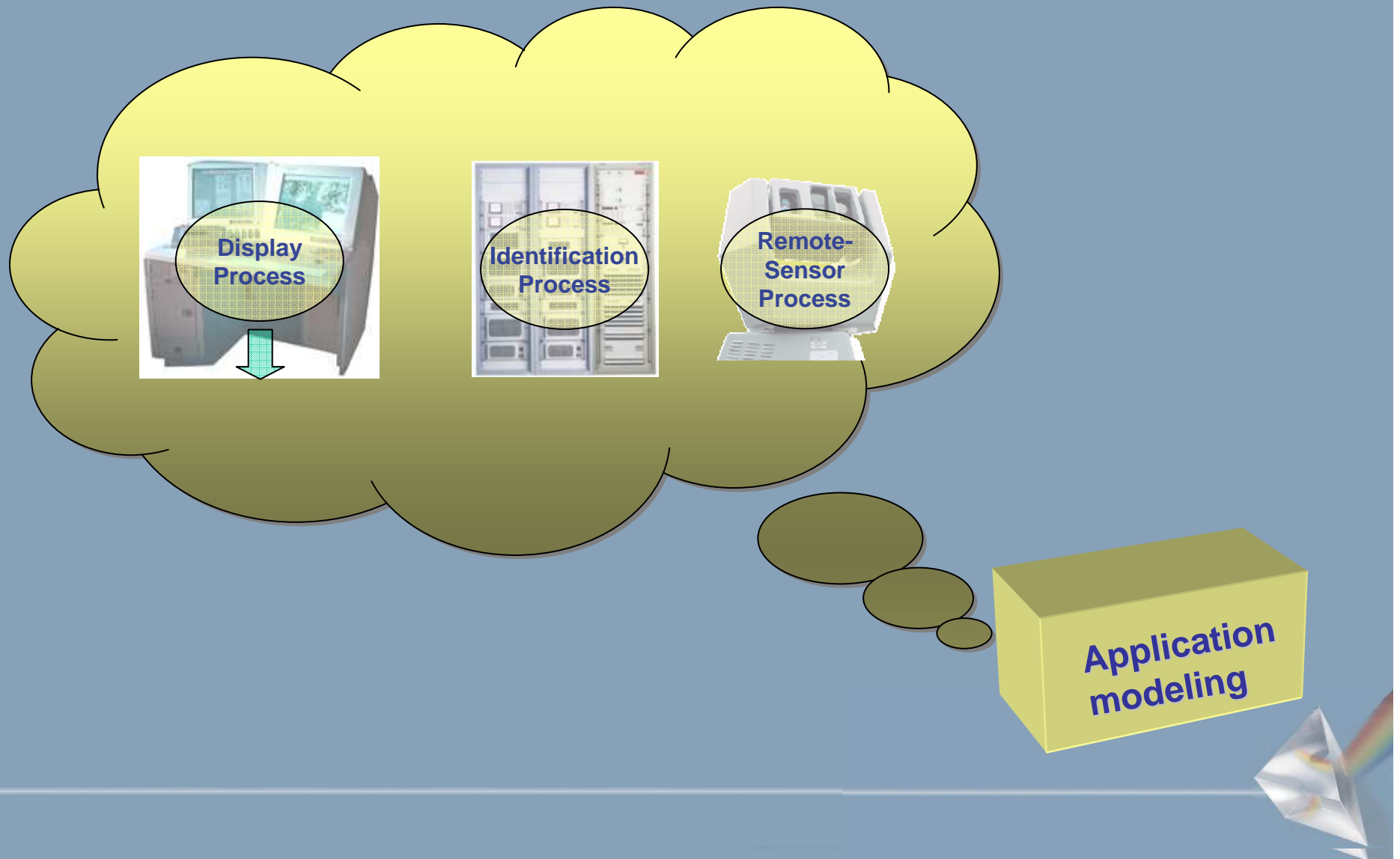
## Task: Information modeling

- ▶ Model the 'PointTrack' and 'TrackState' topics
- ▶ Model the system-wide behaviour of topics
- ▶ Model the relationships between the Topics
- ▶ Separate these concerns from applications

## MDD: Features / Advantages

- ▶ Graphical modeling of structure and QoS (intuitive, fast)
- ▶ DDS-entity code generation (topics, interfaces)
- ▶ Allowing for direct utilization (by applications or tool)
- ▶ Documented packages of re-usable topic-sets

## (2) Application modeling



# Application modeling: Scope & Purpose

## ▶ Language

- ▶ Processing-language
- ▶ DDS 'keywords'
  - ▶ publishers/writers, subscribers/readers,..
  - ▶ Content-filters, queries, waitsets, listeners,...

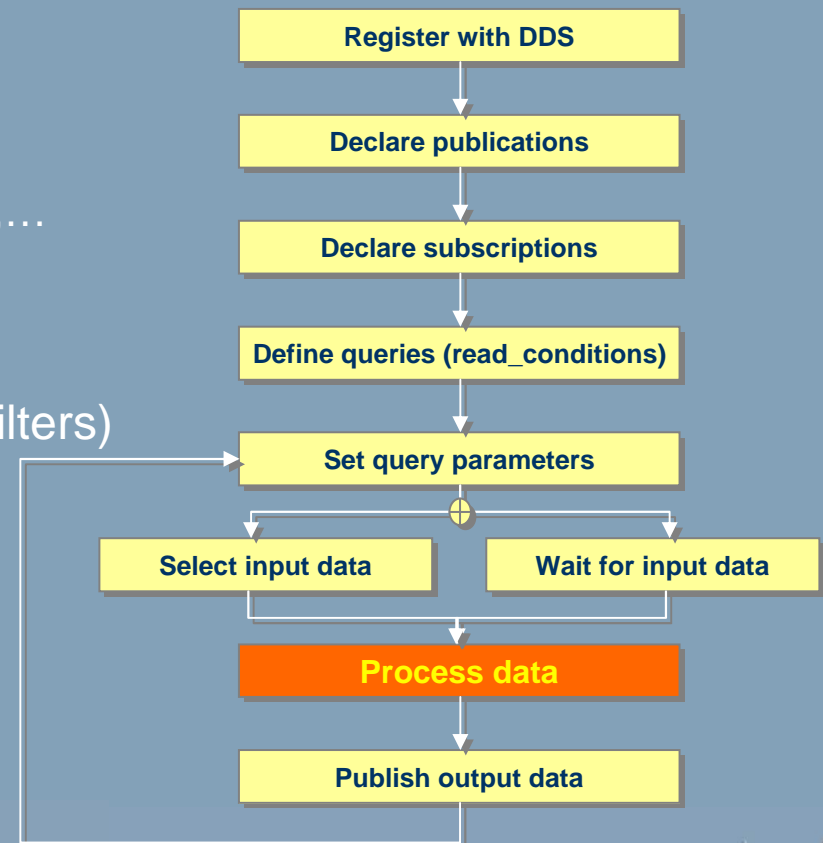
## ▶ Editor

- ▶ Templates & Patterns (loop, MVC, ...)
- ▶ Application-level QoS modeling (history, filters)
- ▶ Process modeling (waitsets, listeners)

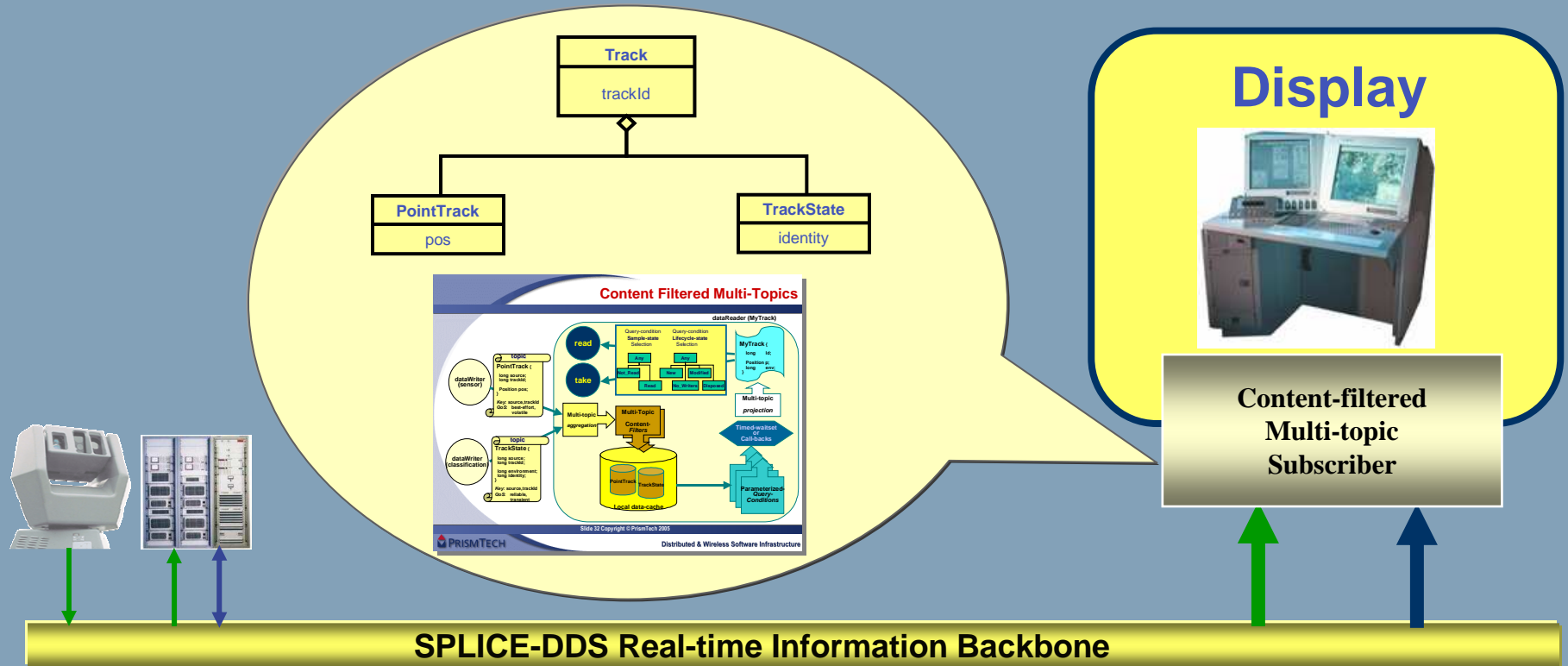
## ▶ Generator

- ▶ Application-framework from templates
- ▶ DDS entities from (graphical) model
- ▶ Annotated application- and test-code

### *Simple DDS-Loop Pattern*



# Application modeling: The Example



## Task: Display Process modeling

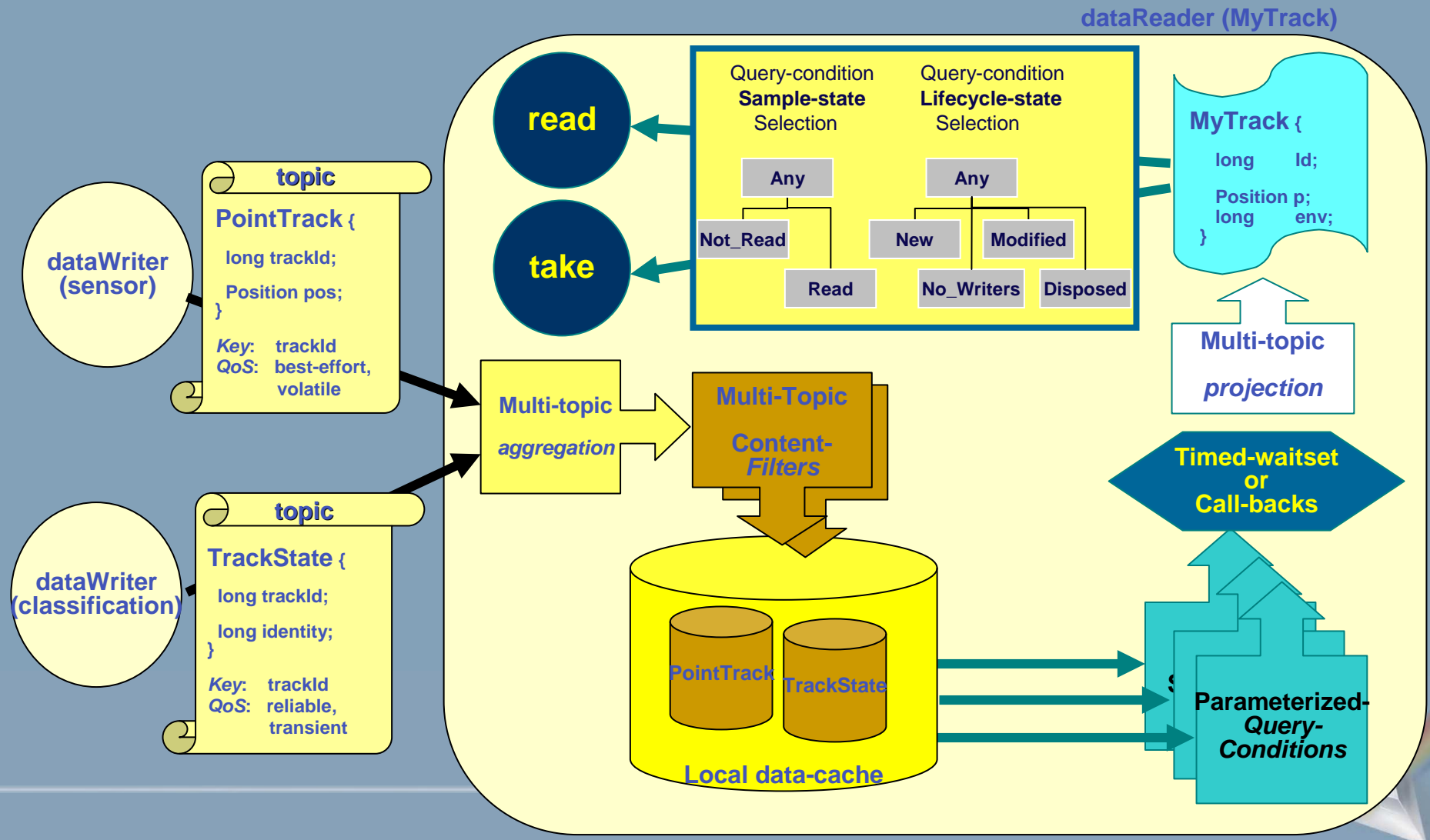
- ▶ Model 'aggregate interest'
  - ▶ 'multi-topic' or 'DLRL-object' with local 'QoS'
- ▶ Model display process (periodic loop)
  - ▶ Handle first-appearances of hostile tracks with prio

## MDD: Features / Advantages

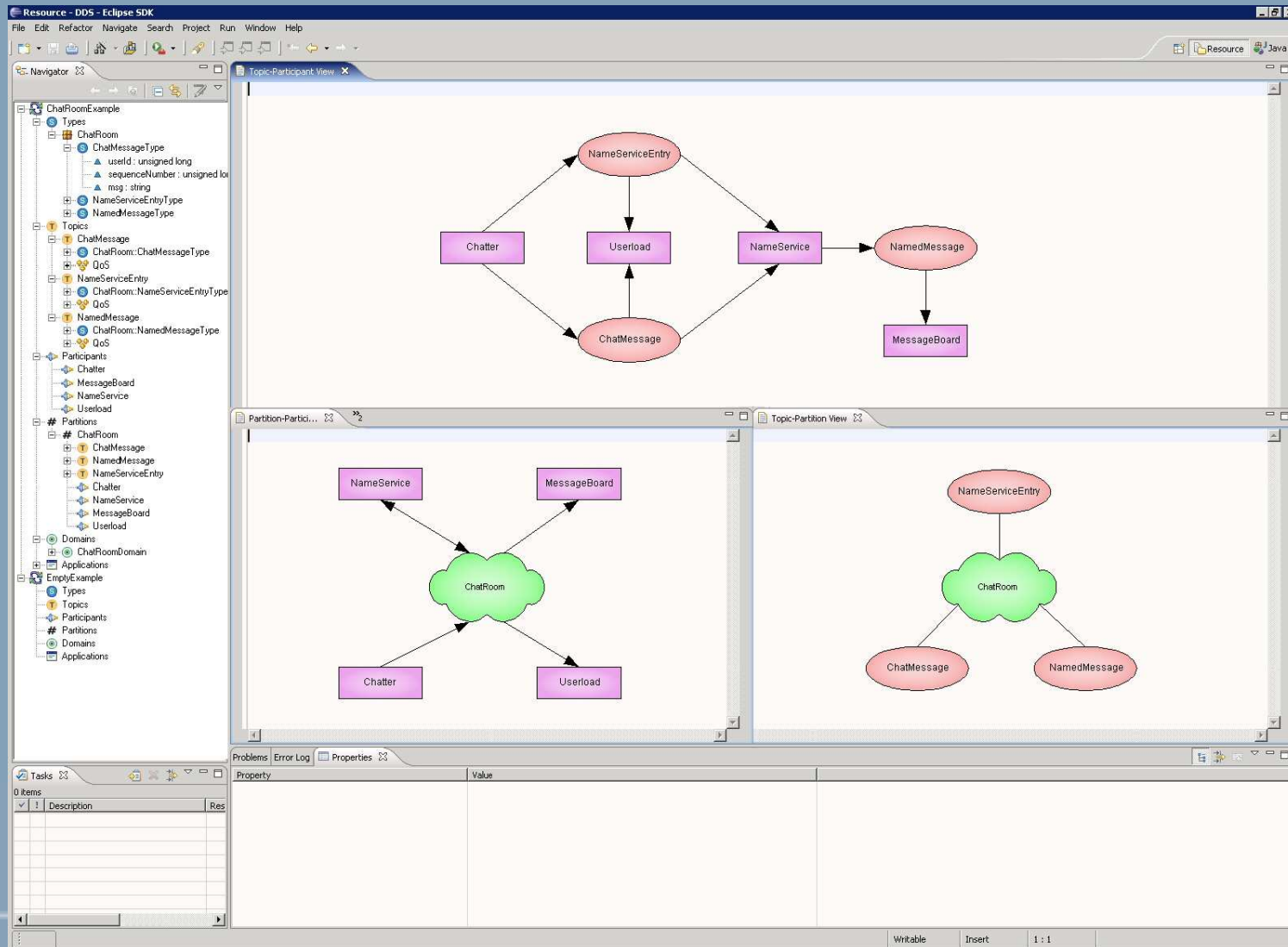
- ▶ Import information-model
- ▶ Provide Application-framework (from 'loop' template)
- ▶ Model DLRL-objects, Multi-Topics, Queries, Waitsets,...
- ▶ Developer can concentrate on 'business-logic' (GUI)

# THE CONTENT-SUBSCRIPTION PROFILE (2)

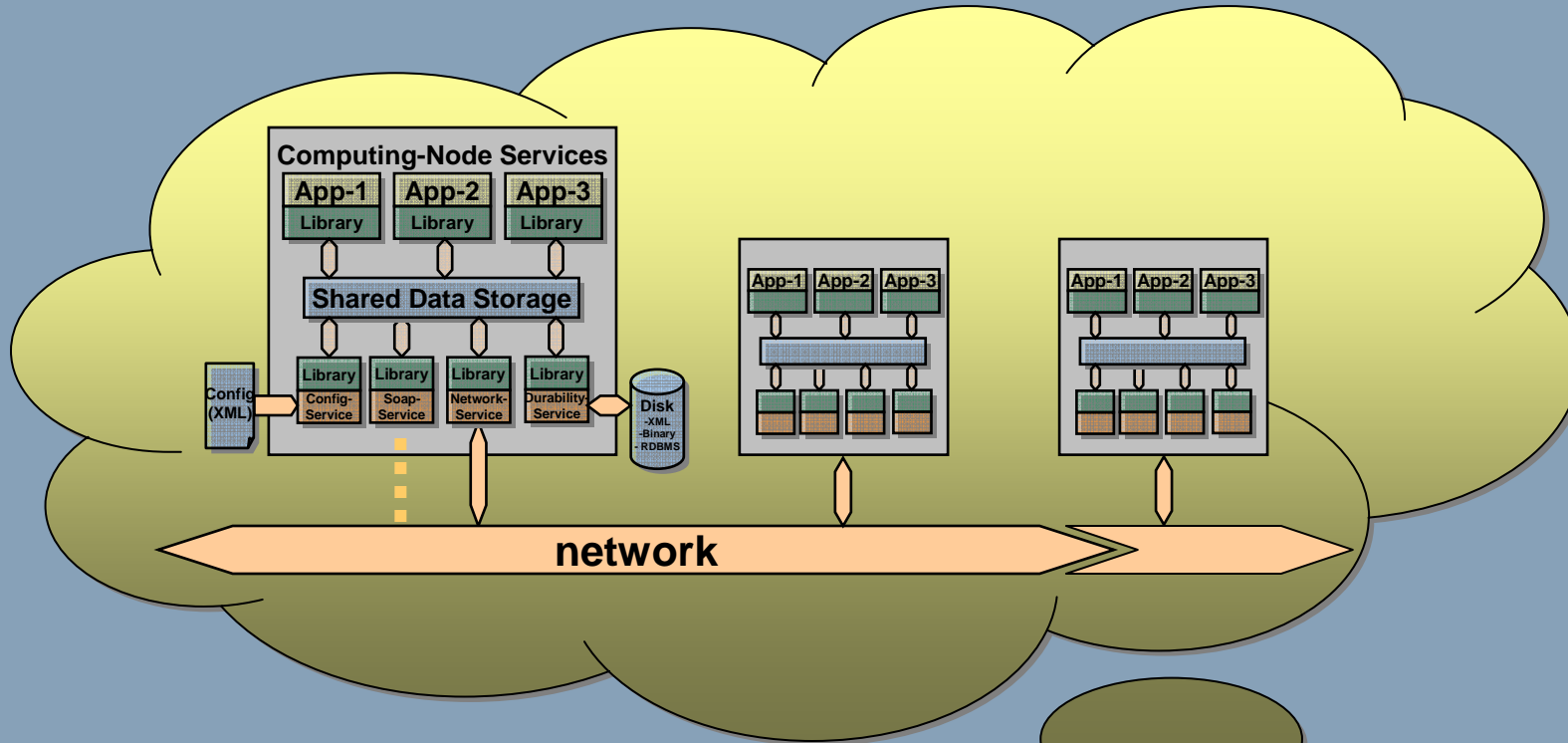
## Content Filtered Multi-Topics



# Application modeling: MDD-Tool 'Sneak-Peak'



# (3) Deployment modeling



Deployment modeling

# Deployment modeling: Scope & Purpose

## ▶ Language

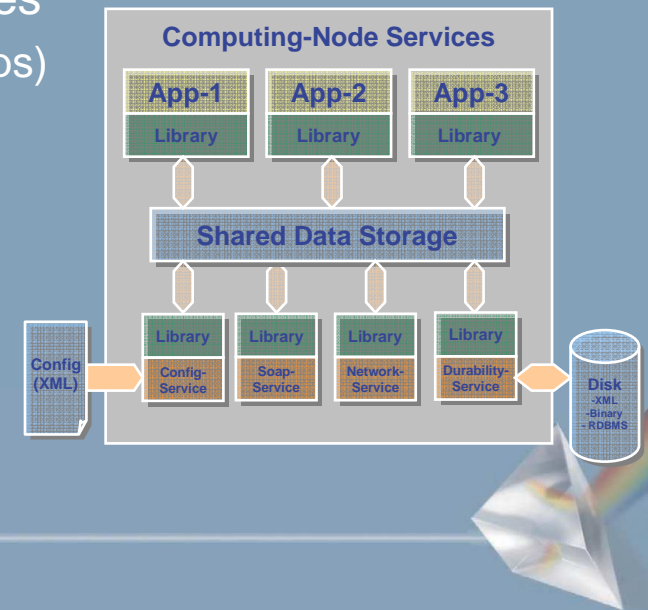
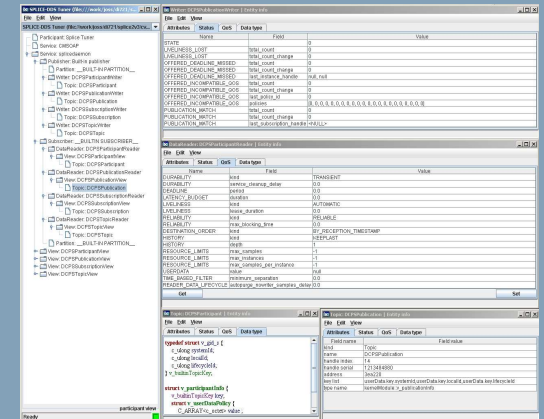
- ▶ Deployment-language
- ▶ DDS 'keywords'
  - ▶ Participants, Partitions, Resource limits,...
  - ▶ Latency-budget, Transport-priority,...

## ▶ Editor

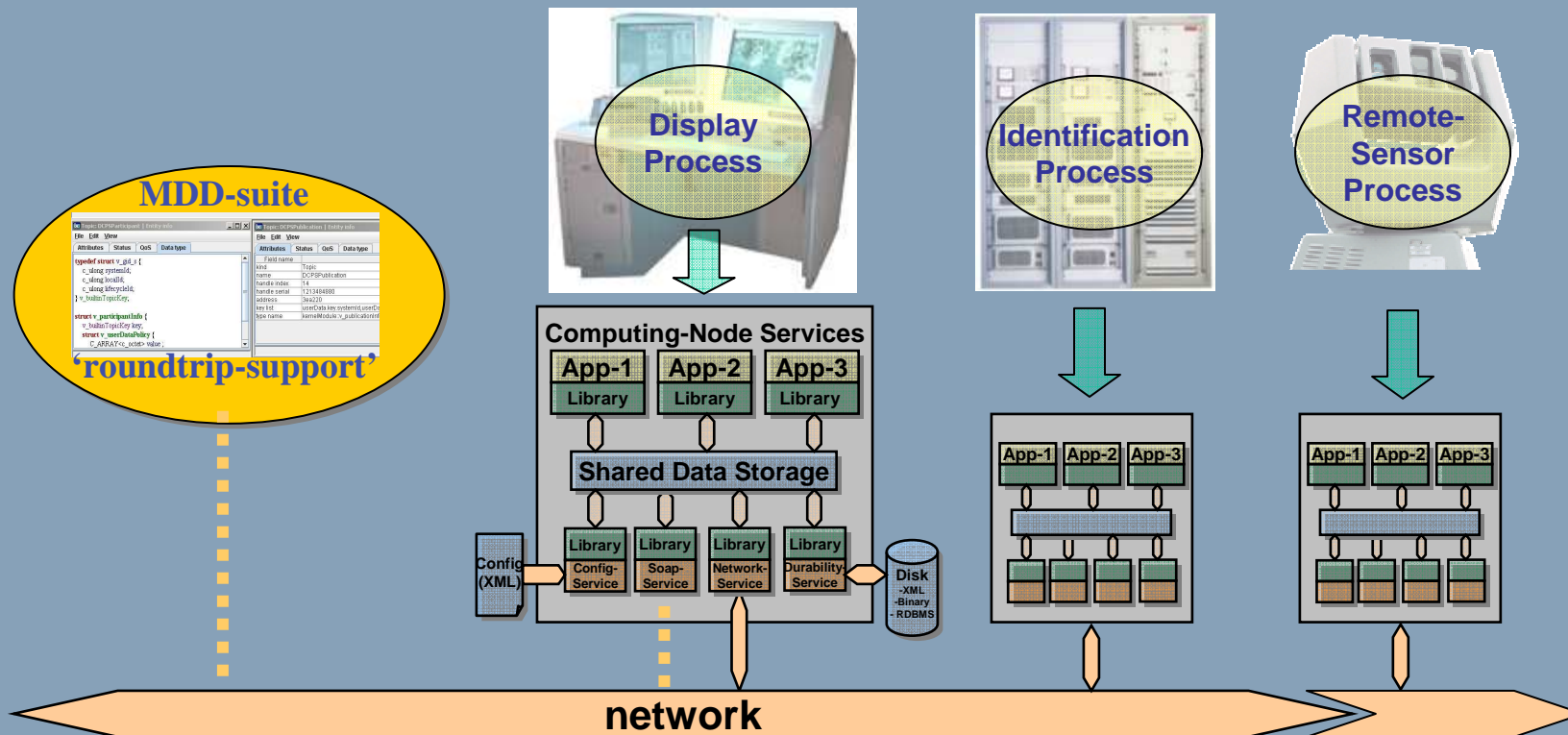
- ▶ Configure 'physical' DDS system
  - ▶ Networking, Durability, Resource-limits, ...
- ▶ Map DDS (QoS) properties to Deployment properties
  - ▶ DDS 'partitions' to Network-partitions (multicast groups)
  - ▶ 'transport-priority' to Network-channels (prio/diffserv)
  - ▶ 'latency-budget' to Network-dispatcher (efficiency)

## ▶ 'Generator' (Control & Monitoring)

- ▶ (remote) set-up, configure and monitor DDS-system
- ▶ Deploy information-models (create readers/writers)
- ▶ Deploy applications (monitor, QoS-tuning)
- ▶ Deploy systems (logging/replay, QoS tuning,...)



# Deployment modeling: The Example



## Task: Deployment modeling ('Control & Monitoring')

- ▶ Configure Durability-service (transient TrackState topic)
- ▶ Configure Network-channels (priority, reactivity)
- ▶ Configure Resources (resource-limits)
- ▶ Remote control & Monitoring of deployed system

## MDD: Features / Advantages

- ▶ Round-trip engineering
  - ▶ Deploy & tune models (info & application)
- ▶ Control & Monitor deployed system
  - ▶ Using the domain specific 'DDS-language'

# Deployment modeling: MDD-tool 'sneak-peak'

SPLICE-DDS Tuner (file:///work/joss/di721/s...)

File Edit View

SPLICE-DDS Tuner (file:///work/joss/di721/splice2v3/cv...)

- Participant: Splice Tuner
- Service: CMSOAP
- Service: spliced daemon
  - Publisher: Built-in publisher
    - Partition: \_\_BUILT-IN PARTITION\_\_
      - Writer: DCPSParticipantWriter
        - Topic: DCPSParticipant
      - Writer: DCPSPublicationWriter
        - Topic: DCPSPublication
      - Writer: DCPSSubscriptionWriter
        - Topic: DCPSSubscription
      - Writer: DCPSTopicWriter
        - Topic: DCPSTopic
- Subscriber: \_\_BUILTIN SUBSCRIBER\_\_
  - DataReader: DCPSParticipantReader
    - View: DCPSParticipantView
      - Topic: DCPSParticipant
  - DataReader: DCPSPublicationReader
    - View: DCPSPublicationView
      - Topic: DCPSPublication
  - DataReader: DCPSSubscriptionReader
    - View: DCPSSubscriptionView
      - Topic: DCPSSubscription
  - DataReader: DCPSTopicReader
    - View: DCPSTopicView
      - Topic: DCPSTopic
- Partition: \_\_BUILT-IN PARTITION\_\_
  - View: DCPSParticipantView
  - View: DCPSPublicationView
  - View: DCPSSubscriptionView
  - View: DCPSTopicView

Writer: DCPSPublicationWriter | Entity info

File Edit View

Attributes Status QoS Data type

Name	Field	Value
STATE		0
LIVELINESS_LOST	total_count	0
LIVELINESS_LOST	total_count_change	0
OFFERED_DEADLINE_MISSED	total_count	0
OFFERED_DEADLINE_MISSED	total_count_change	0
OFFERED_DEADLINE_MISSED	last_instance_handle	null, null
OFFERED_INCOMPATIBLE_QoS	total_count	0
OFFERED_INCOMPATIBLE_QoS	total_count_change	0
OFFERED_INCOMPATIBLE_QoS	last_policy_id	0
OFFERED_INCOMPATIBLE_QoS	policies	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
PUBLICATION_MATCH	total_count	0
PUBLICATION_MATCH	total_count_change	0
PUBLICATION_MATCH	last_subscription_handle	<NULL>

DataReader: DCPSParticipantReader | Entity info

File Edit View

Attributes Status QoS Data type

Name	Field	Value
DURABILITY	kind	TRANSIENT
DURABILITY	service_cleanup_delay	0.0
DEADLINE	period	0.0
LATENCY_BUDGET	duration	0.0
LIVELINESS	kind	AUTOMATIC
LIVELINESS	lease_duration	0.0
RELIABILITY	kind	RELIABLE
RELIABILITY	max_blocking_time	0.0
DESTINATION_ORDER	kind	BY_RECEPTION_TIMESTAMP
HISTORY	kind	KEEPLAST
HISTORY	depth	1
RESOURCE_LIMITS	max_samples	-1
RESOURCE_LIMITS	max_instances	-1
RESOURCE_LIMITS	max_samples_per_instance	-1
USERDATA	value	null
TIME_BASED_FILTER	minimum_separation	0.0
READER_DATA_LIFECYCLE	autopurge_nowriter_samples_delay	0.0

Get Set

Topic: DCPSParticipant | Entity info

File Edit View

Attributes Status QoS Data type

```
typedef struct v_gid_s {
    c_ulong systemId;
    c_ulong localId;
    c_ulong lifecycleId;
} v_builtinTopicKey;

struct v_participantInfo {
    v_builtinTopicKey key;
    struct v_userDataPolicy {
        C_ARRAY<c_octet> value;
    };
};
```

participant view

Topic: DCPSPublication | Entity info


File Edit View

Attributes Status QoS Data type

Field name	Field value
kind	Topic
name	DCPSPublication
handle index	14
handle serial	1213484880
address	3ea220
key list	userData.key.systemId,userData.key.localId,userData.key.lifecycleId
type name	kernelModule::v_publicationInfo

Ready

Slide 39 Copyright © PrismTech 2006



# MDD For DDS

*QUESTIONS ?*

