Webinar begins at 2:05PM, London Time

Dr. Angelo Corsaro ***[angelo.corsaro@prismtech.com]***

OpenSplice DDS Product Marketing Manager, PrismTech

Angelo co-chairs the OMG Data Distribution Service (DDS) Special Interest Group and the Real-Time Embedded and Specialized Services (RTESS) Task Force. He is a well known figure in the distributed real-time and embedded systems middleware community and has a wealth of experience in hard real-time embedded systems, large-scale and very large-scale distributed systems, such as defense, aerospace, homeland security and transportation systems. Prior to joining PrismTech, he worked for the SELEX-SI CTO Directorate, a FINMECCANICA company, where his responsibilities included mapping business requirements to technology capabilities, strategic standardization and technology innovation.

Hans van't Hag ***[hans.vanthag@prismtech.com]***
OpenSplice DDS Product Manager, PrismTech

Hans has extensive experience in applying an information approach towards mission-critical and real-time net-centric systems. He is a co-author of the OMG DDS specification and has presented numerous papers on DDS and publish subscribe middleware technologies. Prior to joining PrismTech he worked for 23 years at Thales Naval Netherlands (TNN) where he was responsible as Product Manager for the development of the data-centric real-time middleware (SPLICE) as applied in TNN's TACTICOS combat system in service with 15 Navies worldwide.

PRISMTECH
Productivity Tools & Middleware

Taming the Data Centric Design with OpenSplice DDS and OpenSplice Powertools

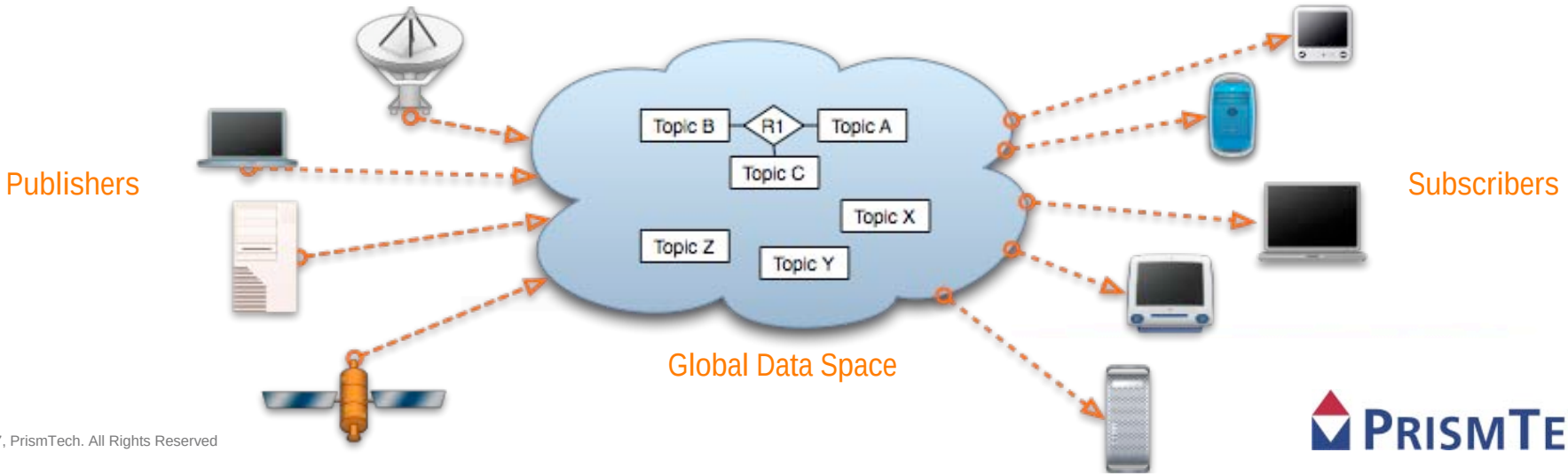
Agenda

- OpenSplice DDS Overview
- Information modeling
- DDS by Example
- Modeling the Example
- OpenSplice Powertools™
- Whats next

Dr. Angelo Corsaro

OpenSplice DDS

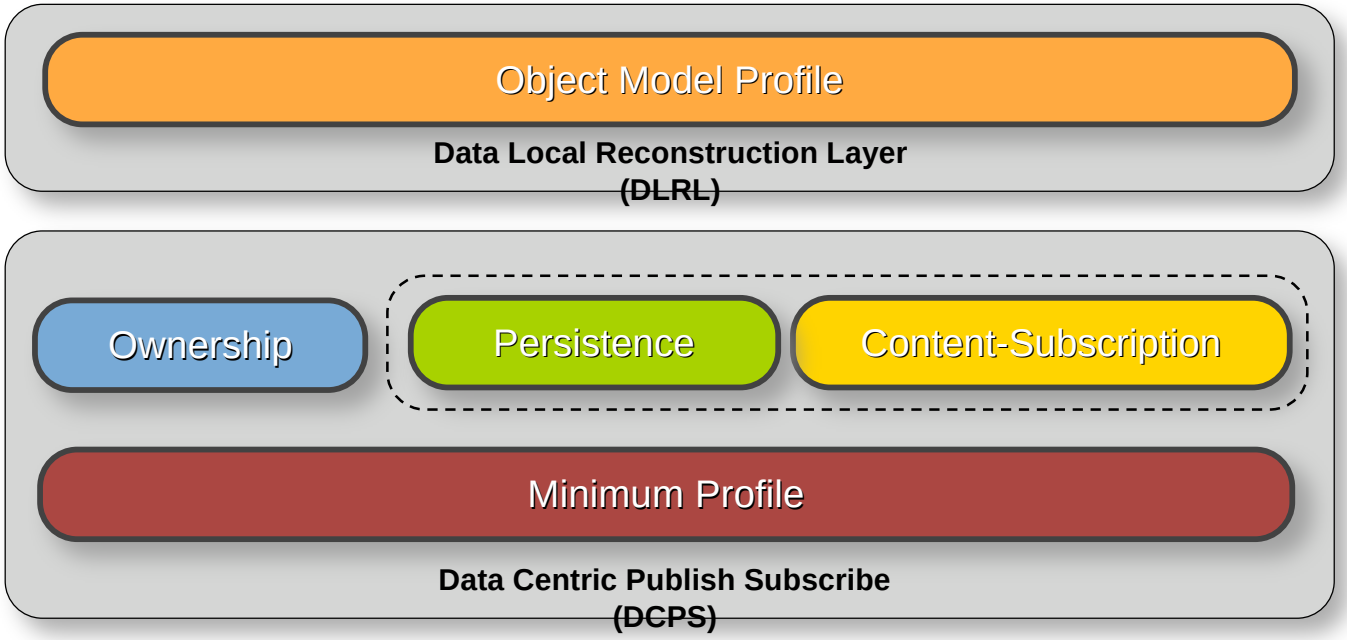
- **An High Performance Real-Time Data-Centric Publish/Subscribe Middleware**
  - *The right data, at the right place, at the right time -- all the time!*
  - *Fully distributed, high performance, highly scalable, and high availability architecture*
- **Unparalleled support for Data-Centric & real-time Publish/Subscribe features**
  - *Content based subscriptions, queries and filters, DLRL*
  - *Fine grained tuning of resource usage and data delivery and availability QoS*
  - *Optimal networking and computing resources usage*
- **Loosely coupled**
  - *Plug and Play Architecture with Dynamic Discovery*
  - *Time and Space Decoupling*
- **Open Standard**,
  - *Complies with the full profile of the OMG DDS v1.2*



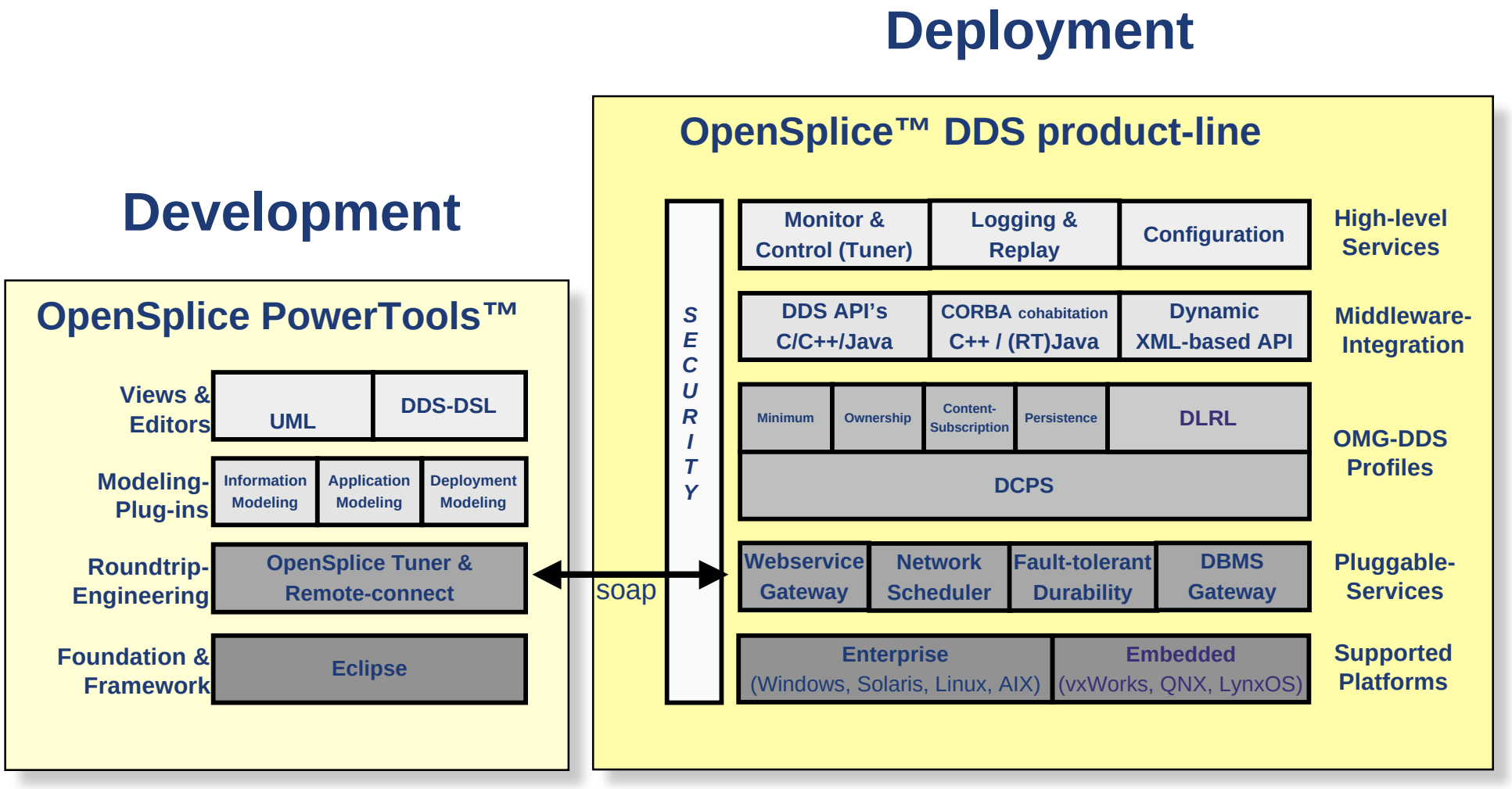
PRISMTECH

Standard Compliance

OpenSplice DDS is compliant with the full profile specified in the OMG DDS Specification v1.2



OpenSplice™ DDS Overview



OpenSplice™ DDS

PRISMTECH

OpenSplice™ DDS v3 - In Summary

OpenSplice™ DDS

Functionalities

- Full OMG-DDS specification coverage *(DCPS and DLRL)*
- Provision of a true 'fault-tolerant information backbone' *(content-aware and FT-durability)*
- Wide Cohabitation and Connectivity with other Technologies *(Corba, RT-Java, DBMS, SOAP, XML)*
- Availability of (remote) deployment tools *(Tuner™ offering total & remote control)*
- Support for Information/application/deployment modeling *(DCPS/DLRL-specific roundtrip development)*

Performance

- **Scalability** w.r.t. number of applications as well as computing nodes and topics
- **Real-time determinism** by urgency (latency-budget) & importance (priority) based network-scheduling
- **Fault-tolerance** by FT-durability and reliable network-service shielding faulty applications from the network

Pedigree

- **Maturity.** Product proven, fielded, In service in 15 Navies world-wide
- **Fractal Architecture.** Large-scale, real-time, fault-tolerant, embedded, all in one system!
- **High Standard of Quality Assurance.** Process/procedures, QA-artefacts and regression testing w.r.t. number of applications as well as computing nodes and topics

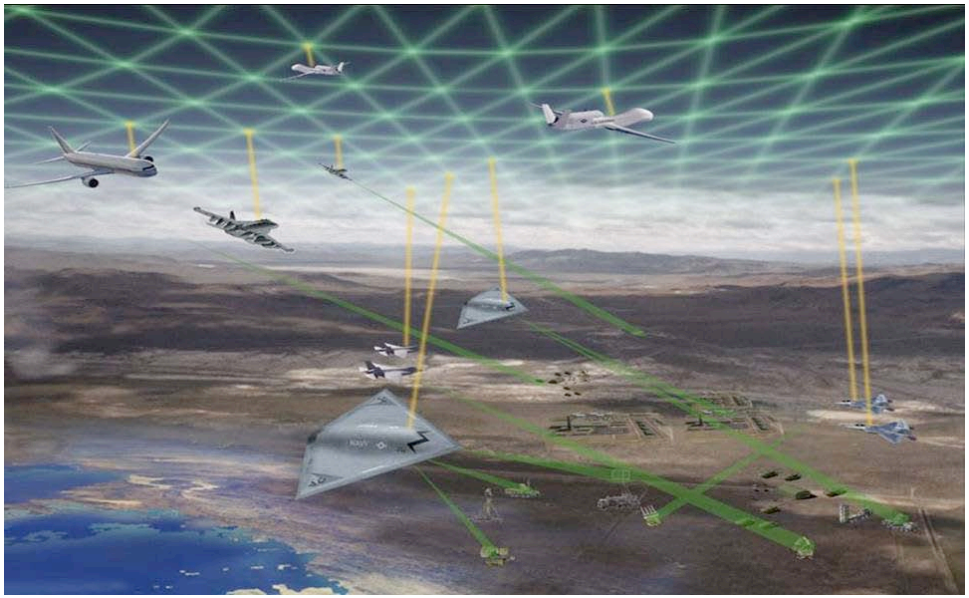
PRISMTECH

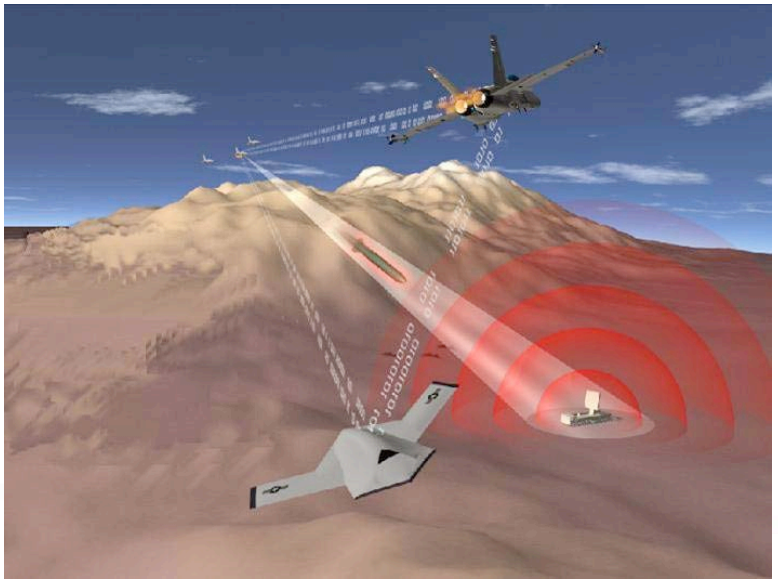
Agenda

- OpenSplice DDS Overview
- **Information modeling**
- DDS by Example
- Modeling the Example
- OpenSplice Powertools™
- Whats next

Hans van't Hag

More Complex Systems and Requirements

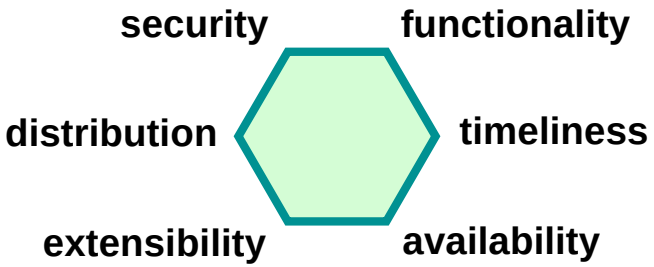




Distributed Systems: The Problem

Problem: engineering (-cost) of distributed systems

- too complex
- not reactive
- not future-proof
- not fault tolerant



Because 'multi-dimensional engineering' is needed:

What about the current 'state-of-the-art'?

- architectures: client/server, message-passing, SOA
- most efforts fall short in a number of dimensions:
- typically:
  - limited RT performance (high-volume & low-latency balance)
  - exploding complexity (dependencies in many dimensions)
  - costly evolution (impact of changes & extensions)

Overall Architectural Requirements

System design
- *provide a stable basis to operate upon by applications*
- *enhance component autonomy*
- *allow transparent and global QoS assurance*

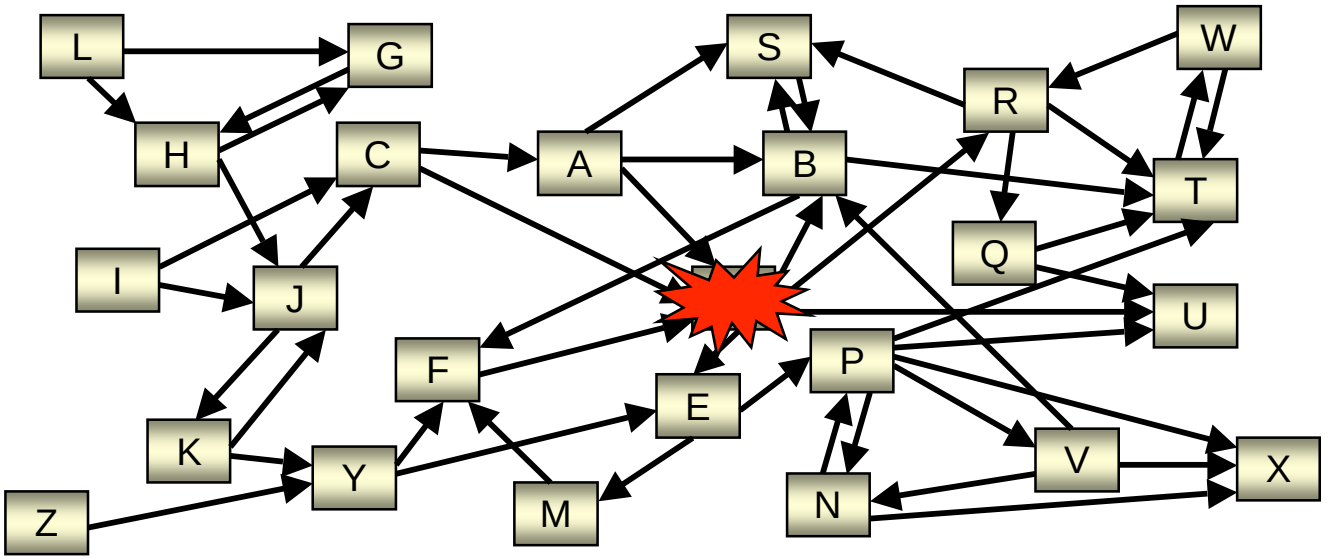
System development
- *reduce complexity and enhance re-usability*
- *provide shared/guaranteed properties*
- *small learning effort and flat learning curve*

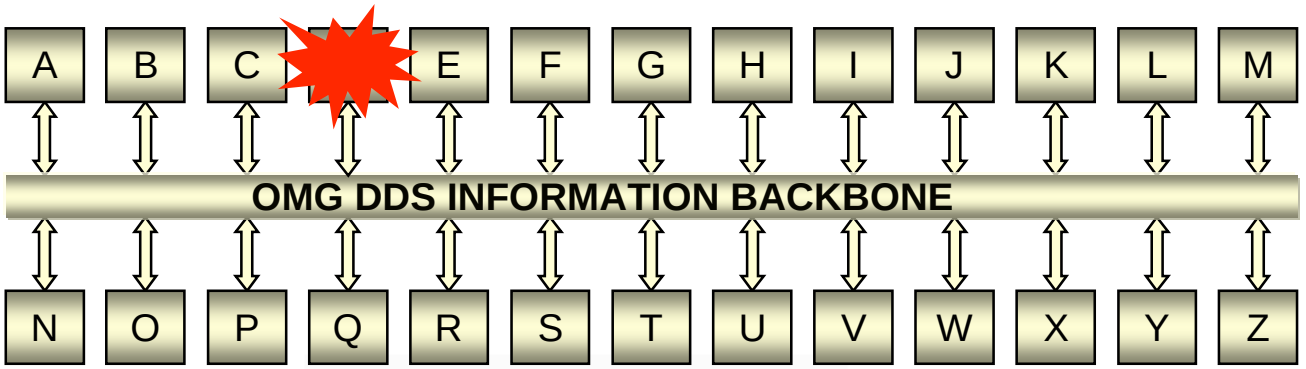
System integration
- *support effortless component integration*
- *provide easy monitor & control*
- *shift ratio between design and integration effort*

System deployment
- *guaranty QoS for reliability, latency and persistency*
- *allow applications to join the system at any time*
- *allow runtime migration/restart of applications*

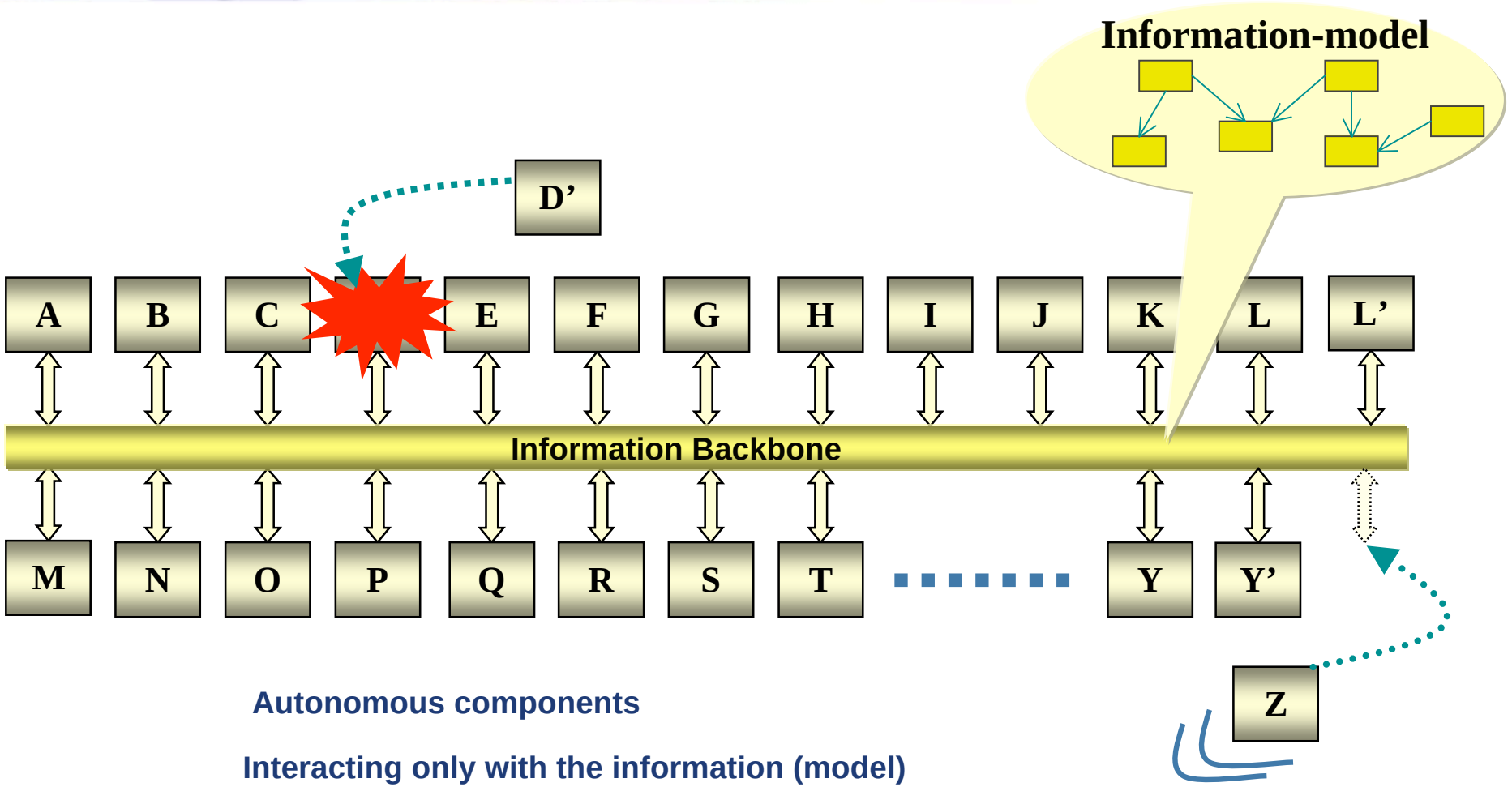
System maintenance & evolution
- *allow runtime replacement and evolutionary upgrading*
- *support for logging & replay of information*
- *provide future-proof, re-usable, robust and scalable system*

Data centric systems: A mind shift





DDS: AN INFORMATION-CENTRIC ARCHITECTURE



Autonomous components

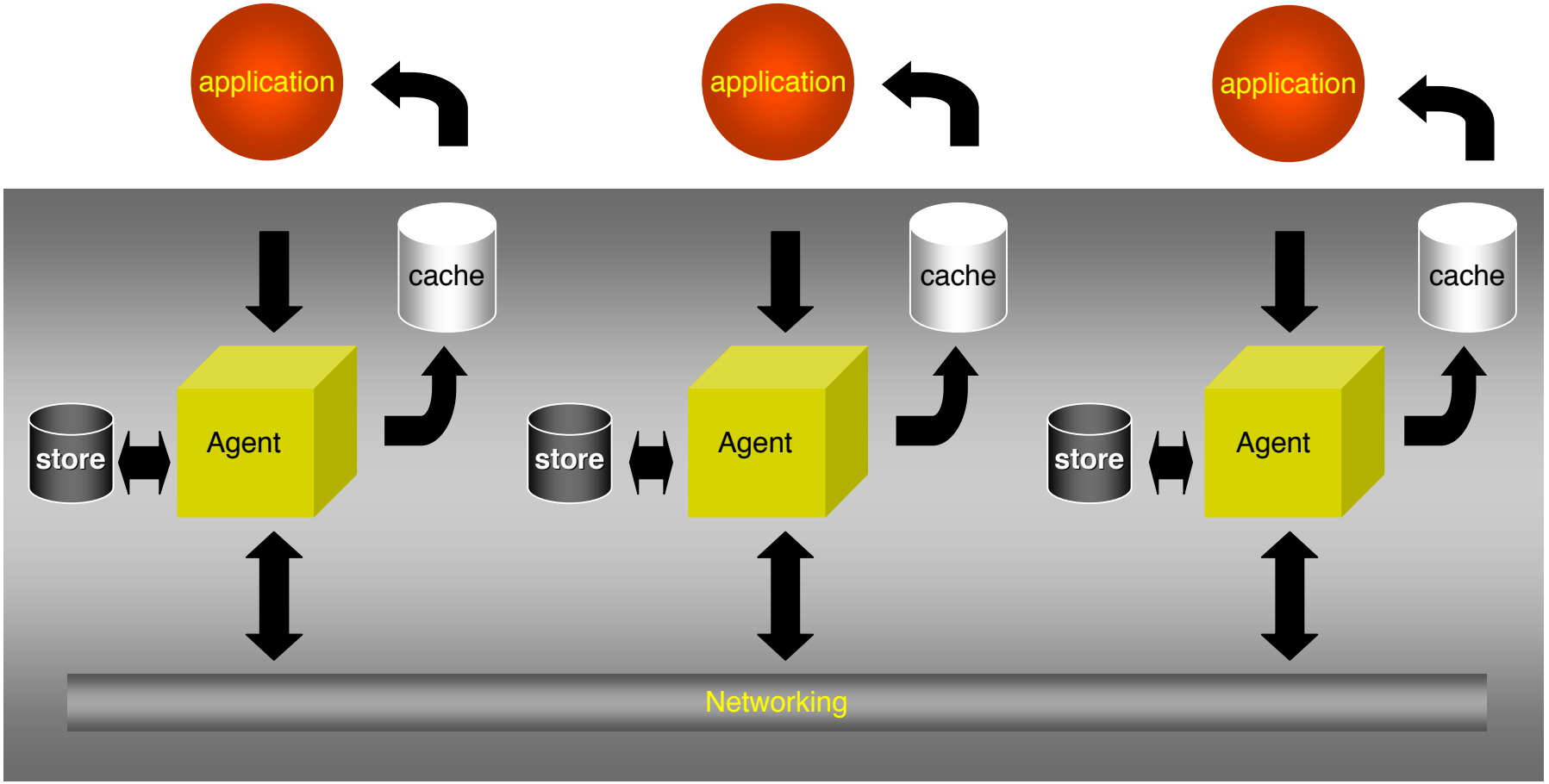
Interacting only with the information (model)

Spontaneous: Z, Self-healing: D'

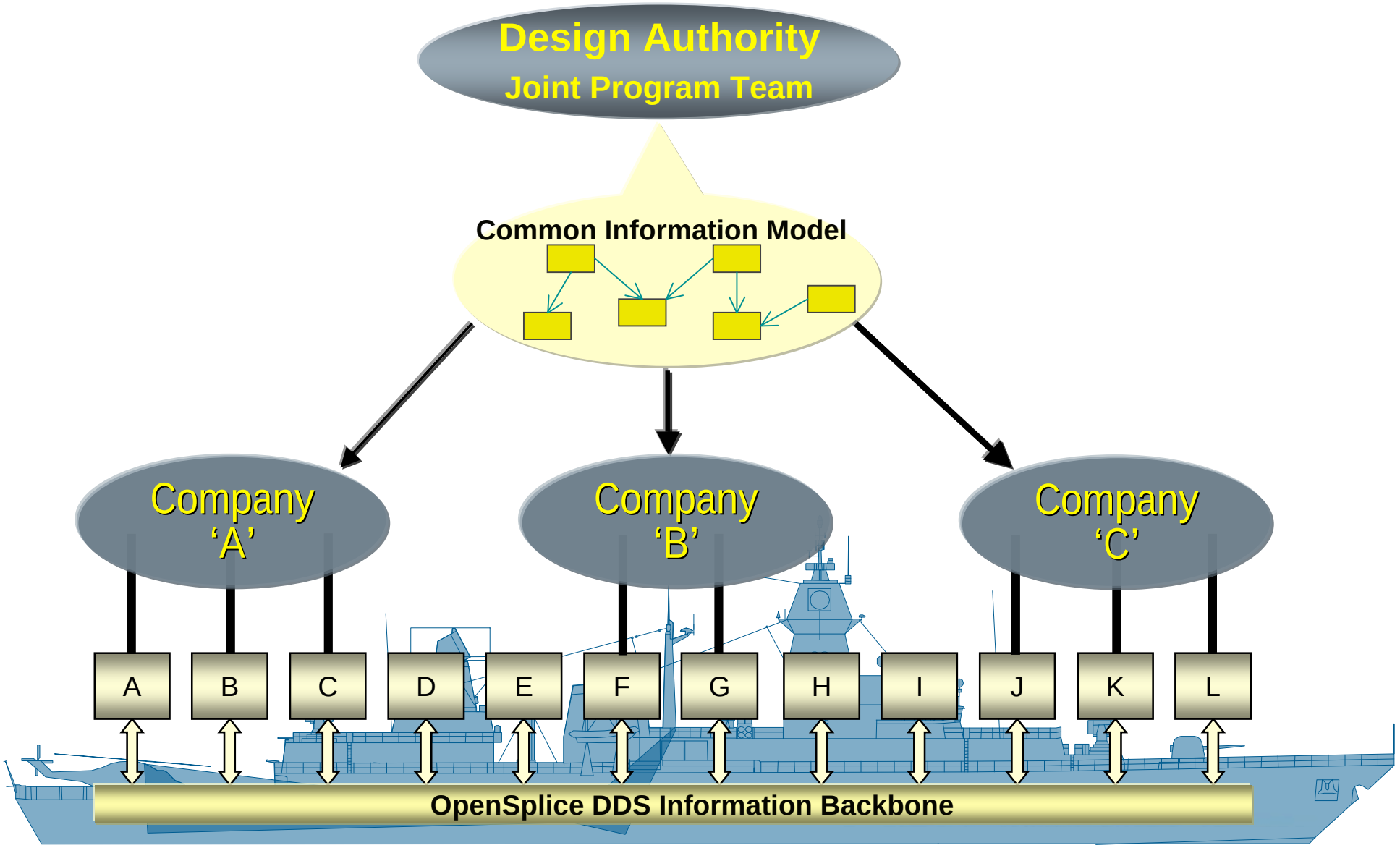
Redundant & Replicated: , L' Y'

QOS-driven Data Distribution Service (urgency, importance, durability): DDS

Self-Healing: Fault-tolerant persistence



EXAMPLE: Coalition-based development (German F124 frigate)

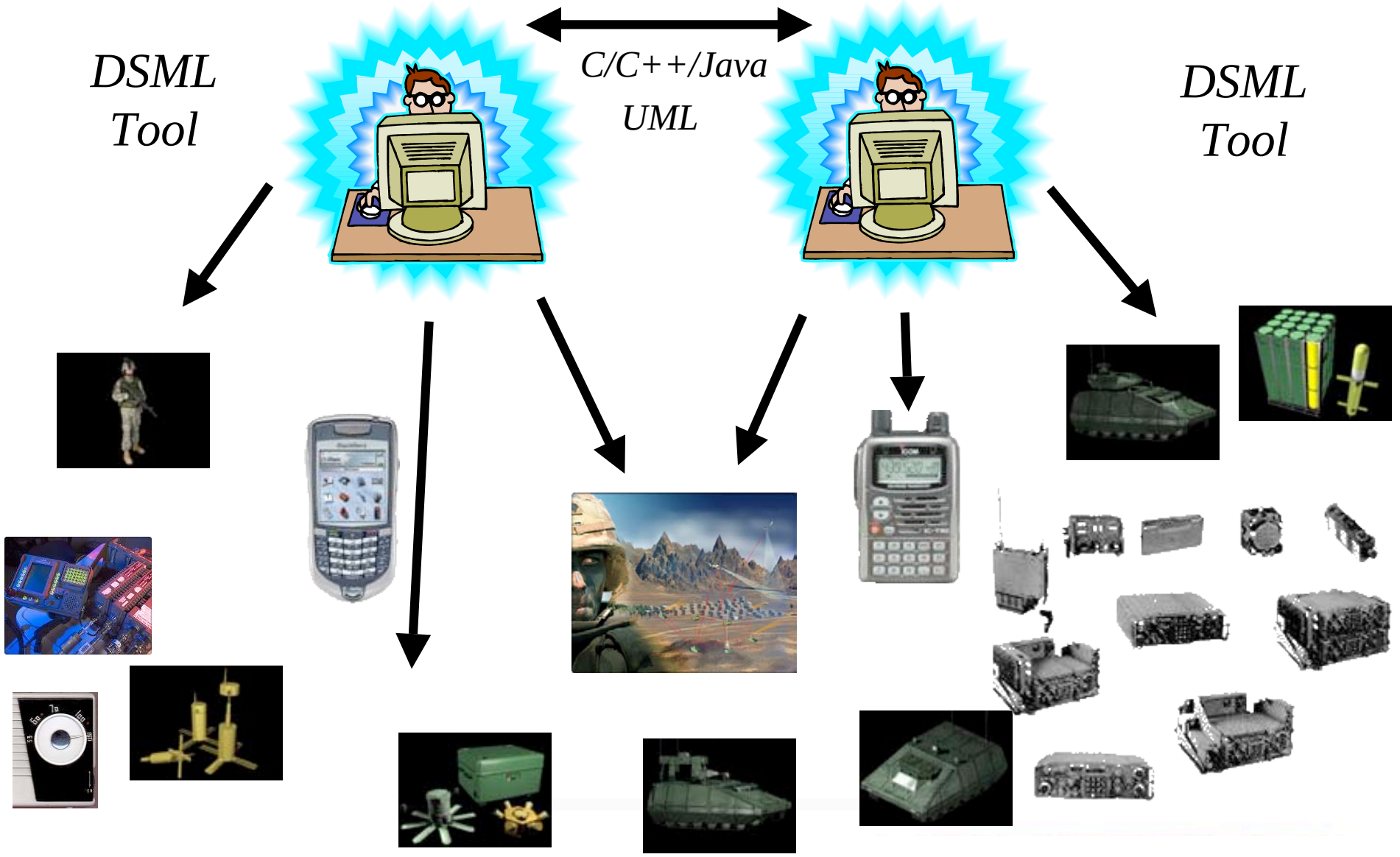


Complex Publish Subscribe Systems

- 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*

Providing sufficient tools to do the job



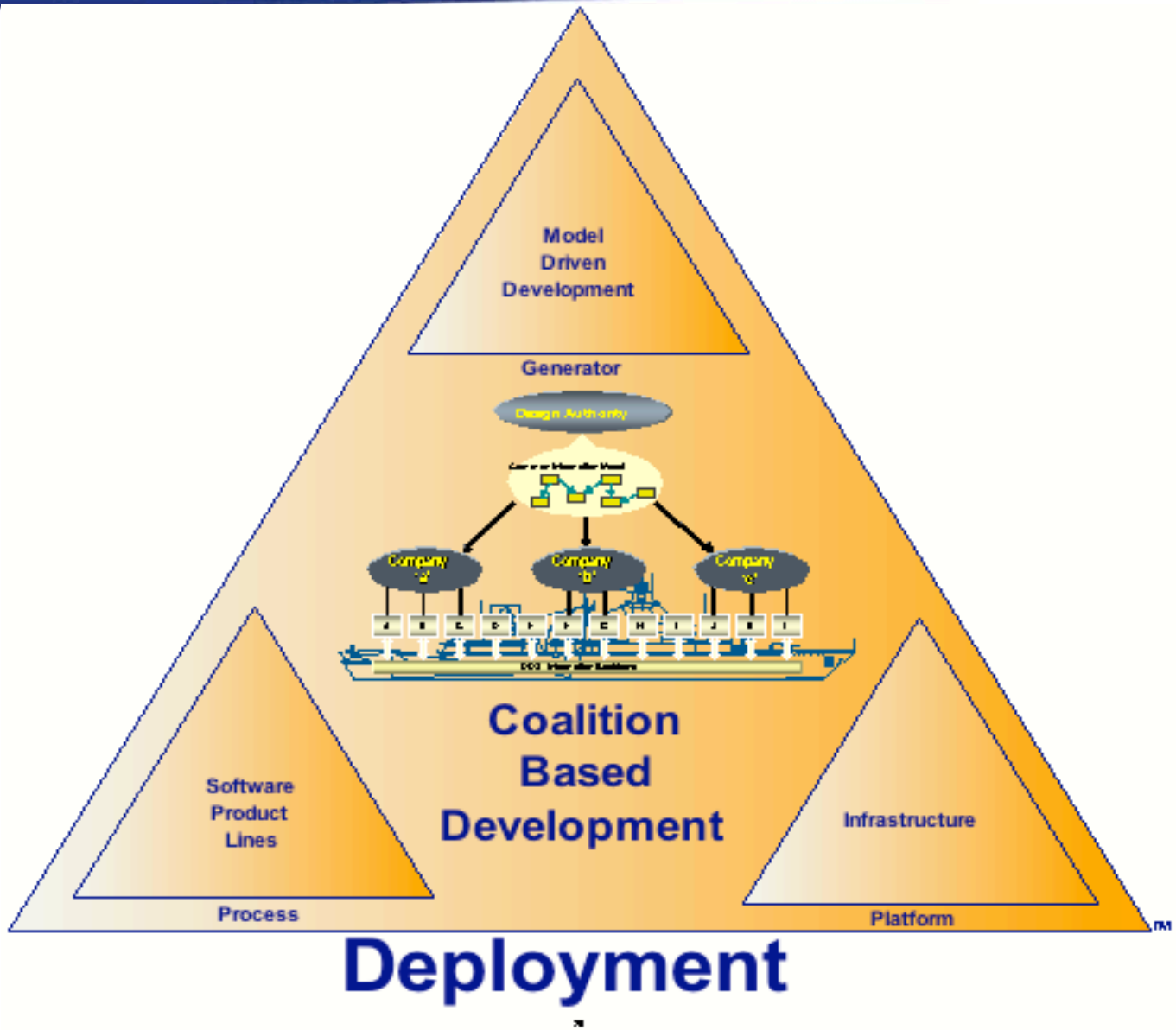
Domain Specificity

Domain Independent

Domain Specific - Tools

The task at hand

The “Holy Grail” ??....



Agenda

- OpenSplice DDS Overview
- Information modeling
- DDS by Example
- Modeling the Example
- OpenSplice Powertools™
- Whats next

Introducing The Example

SENSOR PROCESS

- Optical sensor
- Scans the environment
- Produces ‘Tracks’
- Position of ‘objects’
- Reports ‘***pointTrack’***

IDENTIFICATION PROCESS

- Classifies Tracks
- Determines their identity
- Analyses the trajectories
- Deduces hostile intent
- Reports ‘***trackState’***

DISPLAY PROCESS

- Displays track info
- Both Position & Identity
- Raises alerts
- Requires ‘***pointTrack’***
- Requires ‘***trackState’***

Defining the Example-system architecture

Importance of the information model

- In Data-centric systems, it's a key-asset for customer-interaction and system-design
- In a DDS-based system, it's actually THE (only) interface between an application and the rest of the system

Complexity reduction: Process Autonomy

- ***Sensor Process:***
  - only knows about observed object-locations which it should publish at 'his' rates
  - Shouldn't bother about 'classifications' or any other subscriber to its information
- ***Classification/Display Process:***
  - Also autonomous applications without dependencies

Assuring Non-Functional properties: Performance & Fault-tolerance

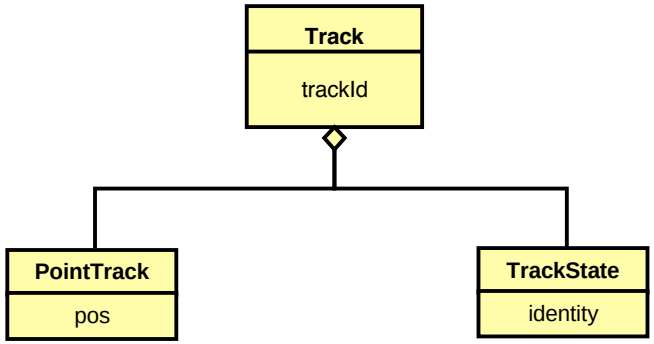
- ***Performance***
  - Periodic measurements (pointTrack objects): *best-effort* delivery, *volatile* persistence
  - A-periodic state-data (trackState objects): *reliable* delivery, *transient* persistence
- ***Fault tolerance***
  - **Sensor**: replicated sensors shouldn't increase system complexity
  - **Identification**: state should be preserved to allow quick re-start after any errors
  - **Display**: should be able to 'join' the system at any time

DDS DATA DEFINITION

Information modeled as "TOPICS"

Each TOPIC has an associated *name* and *data type*

- Type in IDL
- 'Key' fields for unique identification
- Relational Data Model (keys)



Topics can be annotated with QoS policies

- Driving system-wide behavior w.r.t. delivery, durability, priority, urgency, etc.
- Topic-level QoS policies can be used as defaults for Readers/Writers

Topic ***"PointTrack"***

```
Struct PointTrackType {
  long  trackId;

  Position pos;
}
```

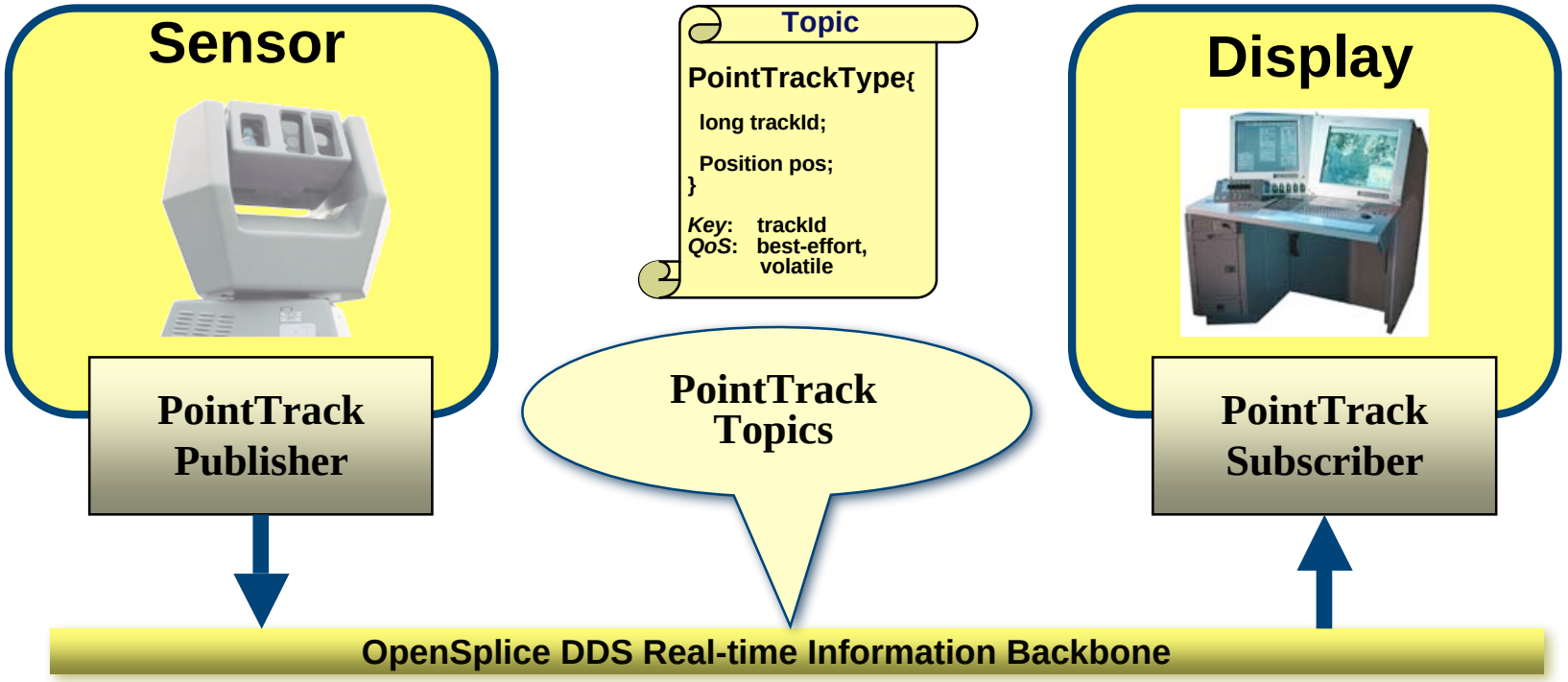
Key fields

Topic ***"TrackState"***

```
Struct TrackStateType {
  long  trackId;

  Id identity;
}
```

Utilizing the 'minimum profile' features



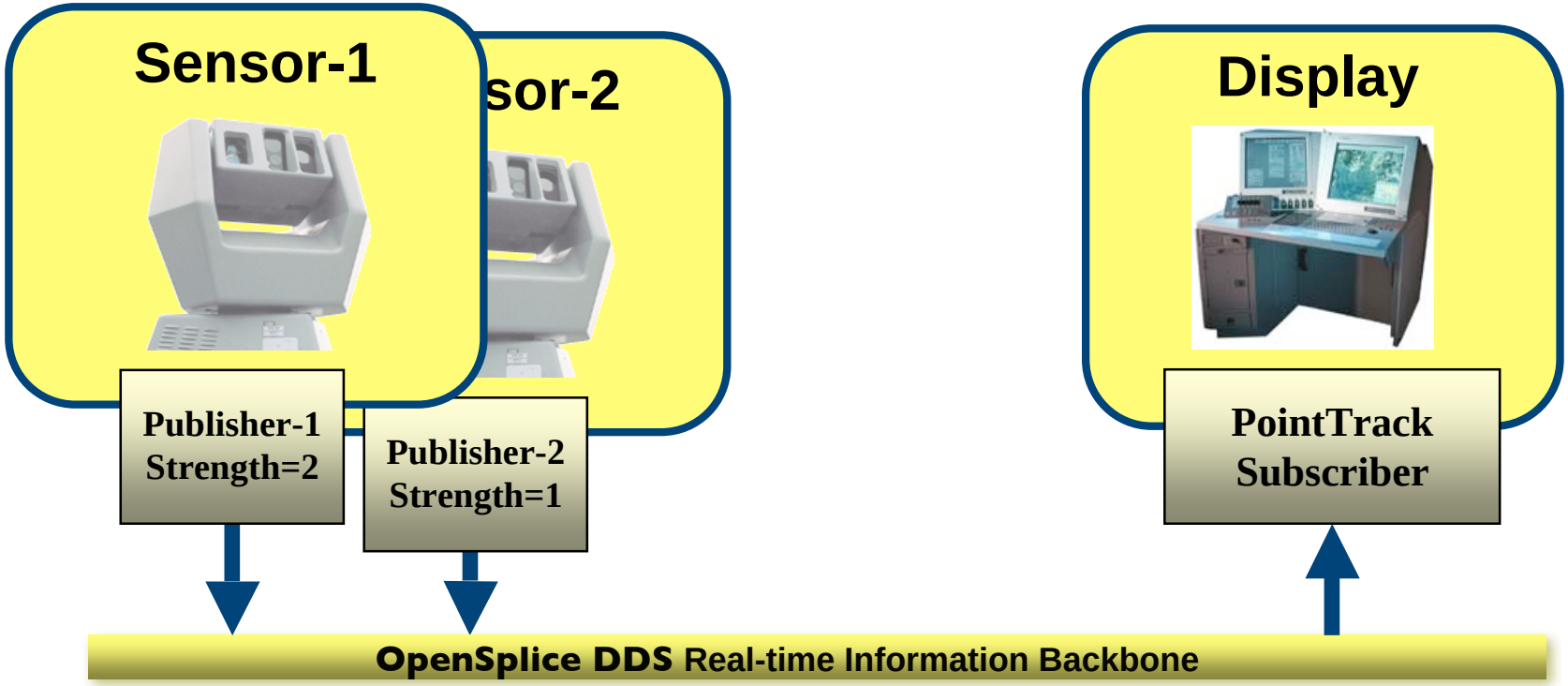
System Requirements:

- Autonomous applications
  - decoupled in space & time
- Easy integrate-able
- Re-usable

Information Model impact (Sensor example)

- *PointTrack* Topic as 'only' interface of the sensor
  - Type: position of each identified 'Track'
  - Keys: TrackId as key-field
  - QoS: *Best-effort* delivery, *Volatile* persistence

Utilizing the 'ownership profile' features



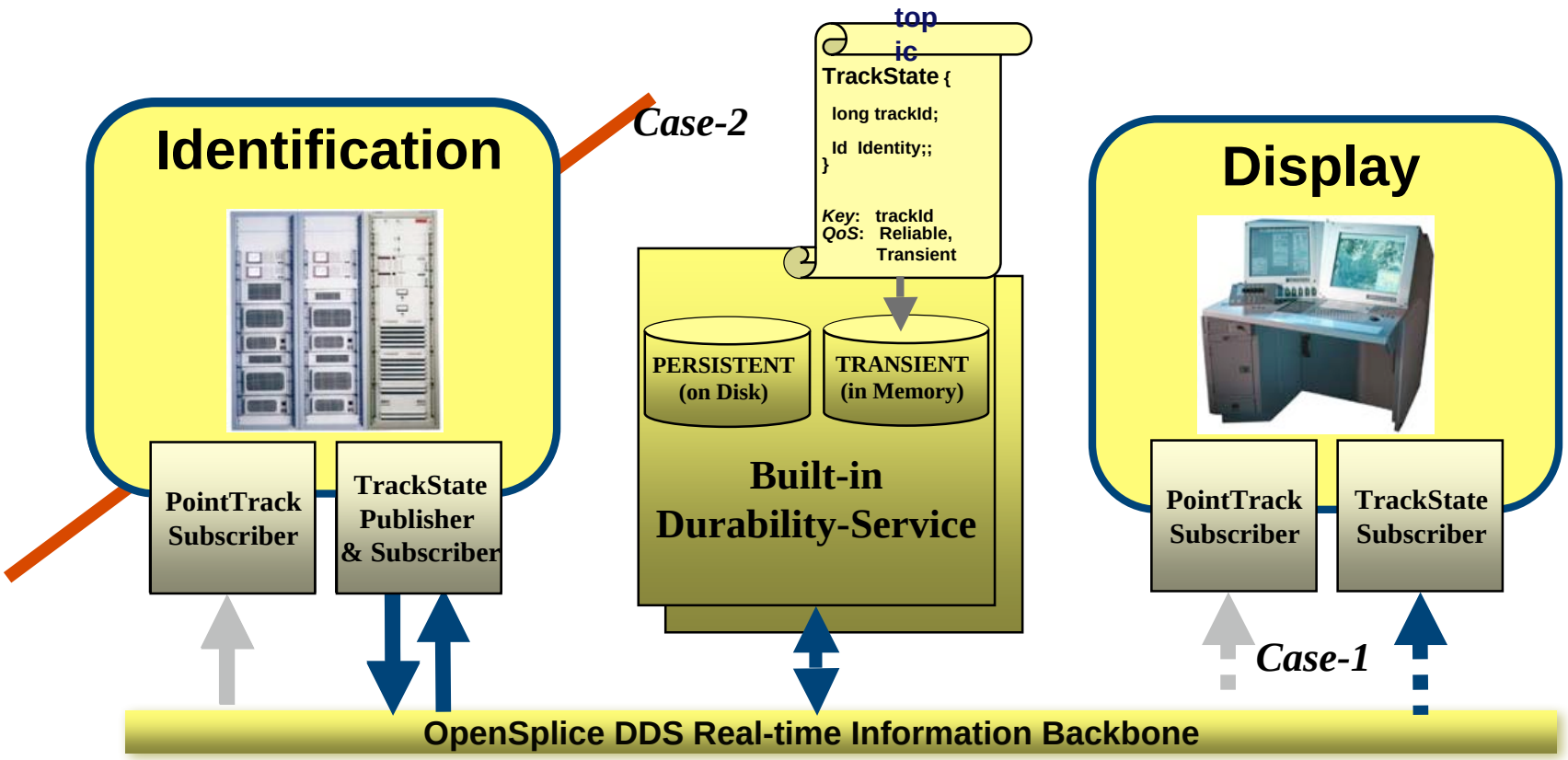
System Requirements

- Replicated sensors to increase fault-tolerance
- Shall not increase sensor or system complexity
- Tolerable to assign each replica a 'strength'

Modeling impact

- Transparent Fault-tolerance by replication
  - Only highest-strength data will be received
- Alternative solution = content-awareness
  - e.g. query for highest quality (data-attributes)

Utilizing the 'Persistence Profile' features



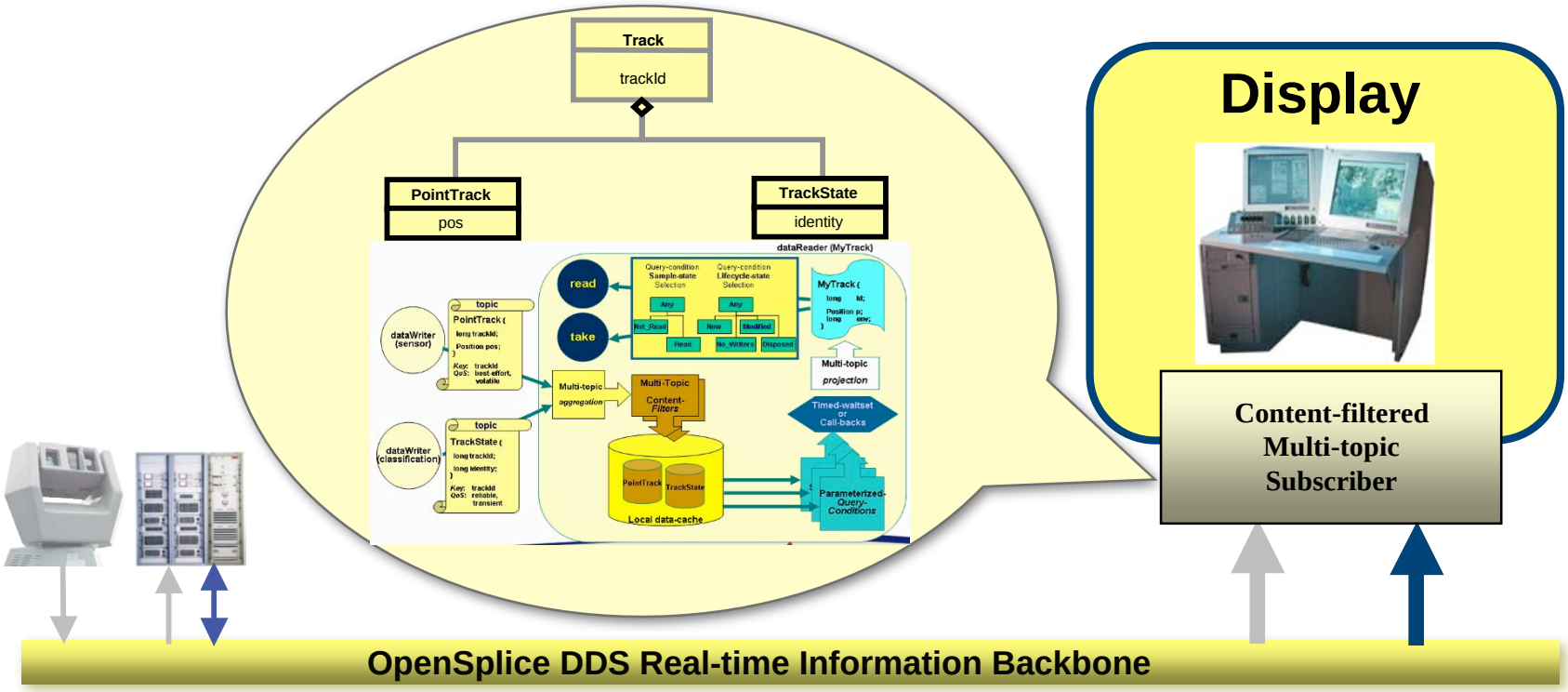
System Requirements

- **Case-1**: *late-joining* of Display process
  - Previously produced TrackStates must be available
- **Case-2**: *restart* of failed Classification process
  - Internal state (already classified tracks) must be available upon restart

Information Modeling Impact

- Information *durability* QoS
  - Volatile: data only available at production time
  - Transient: FT-availability for late-joining app's
  - Persistent: data persisted on (replicated) disks
- Information *reliability* QoS: resends of missed data

Utilizing the ‘content subscription profile’ features



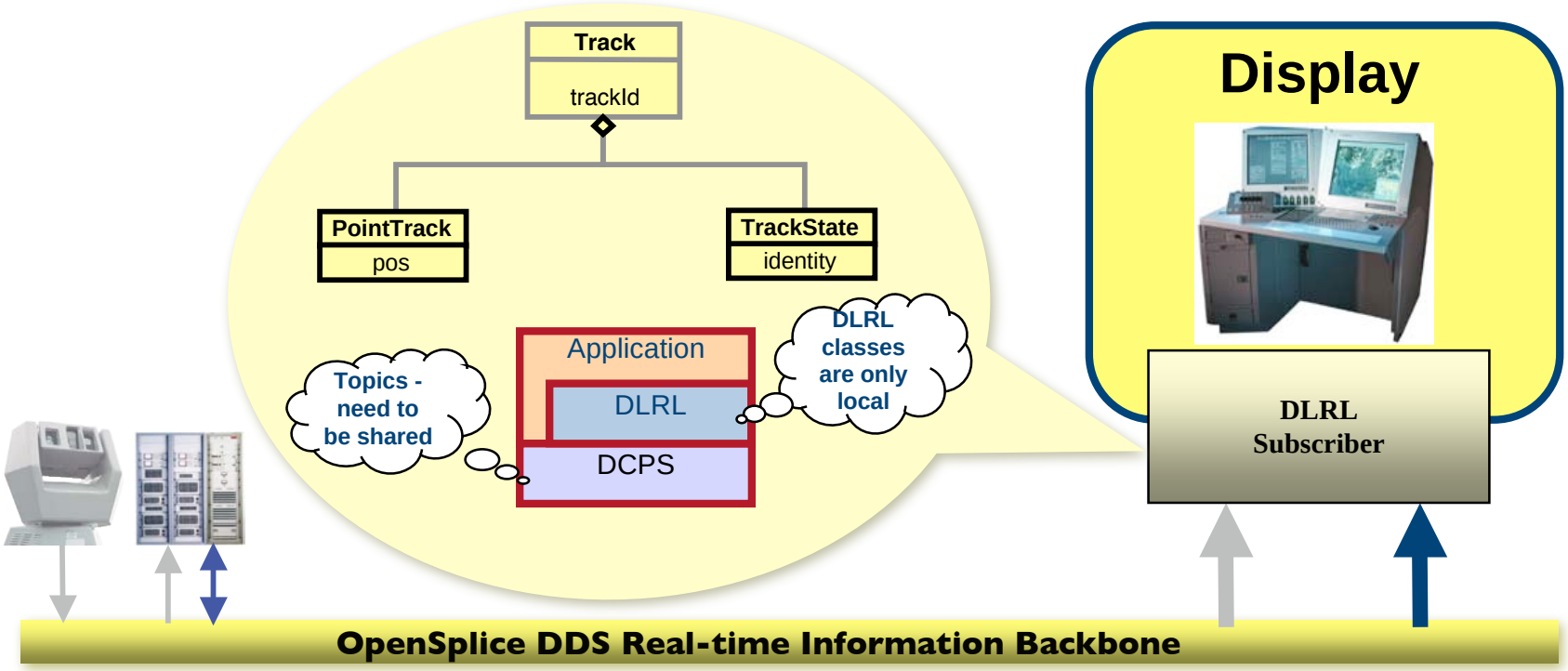
Display Requirements

- Use ‘content-aware’ information backbone
  - To reduce application complexity
  - To increase system performance

Information Modeling Impact

- SQL-like filtering and/or querying on topic-attributes
  - Application can express fine-grained interest
  - Only relevant information needs to be processed

Utilizing the 'DLRL profile' features



System Requirements

- Reduced application complexity
  - Intuitive OO-interface (navigation)
  - Automatic change-management

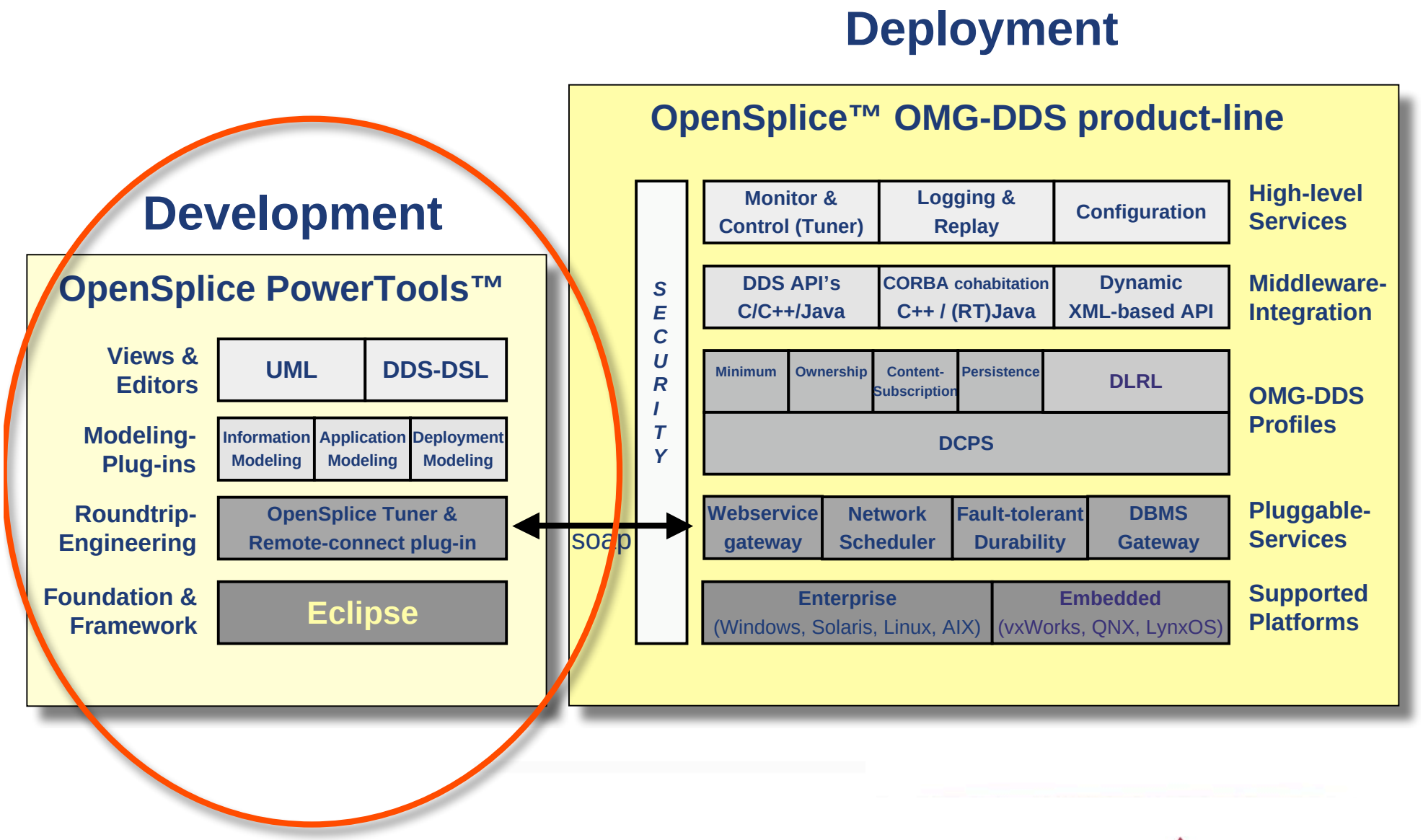
Information Modeling impact

- Model an 'Track' object model on top of the DCPS topics
  - DLRL objects have user-defined assessor methods
  - DLRL objects have fine-grained listener mechanism
  - DCPS-topics will still be distributed

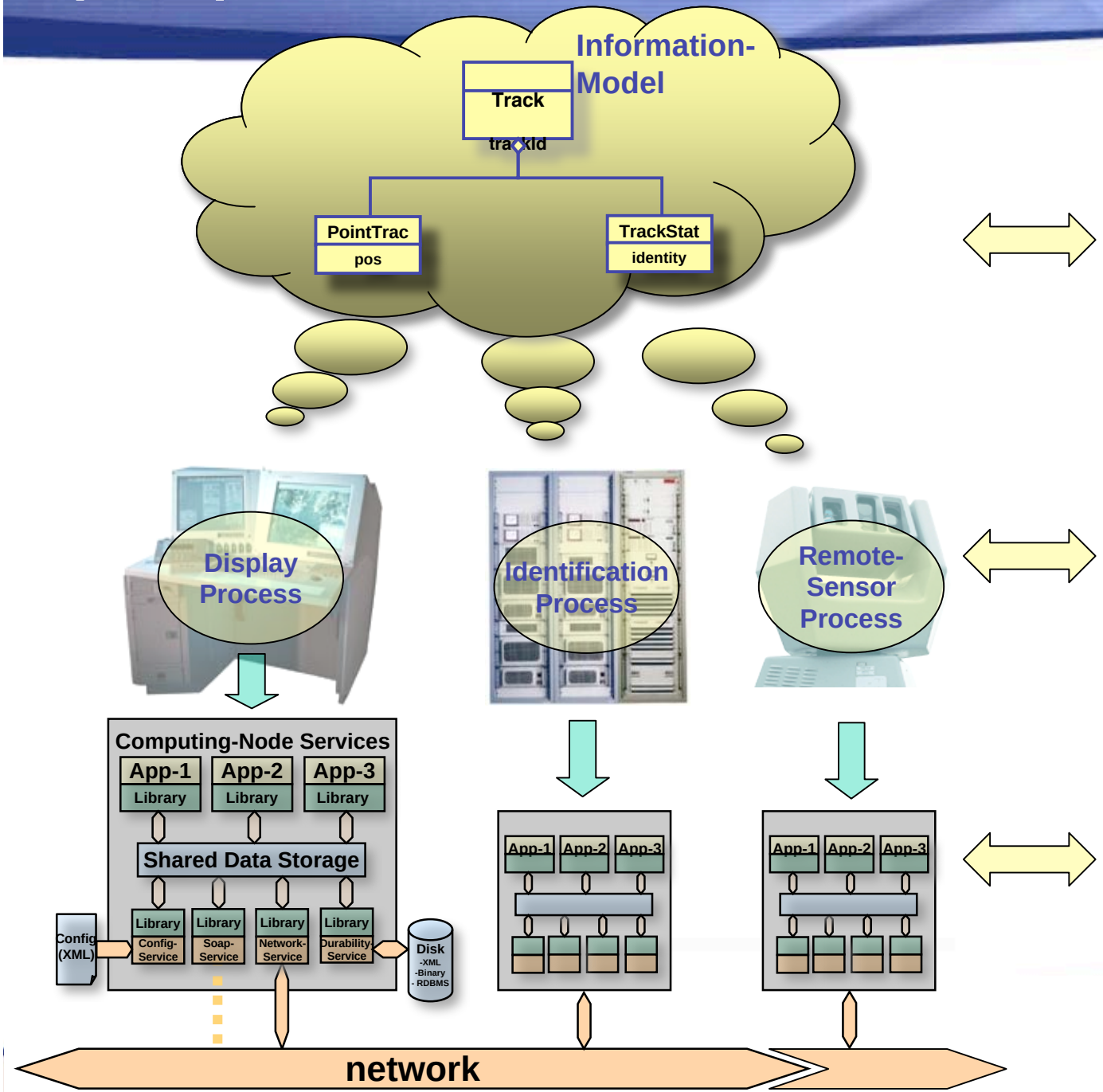
Agenda

- OpenSplice DDS Overview
- Information modeling
- DDS by Example
- **Modeling the Example**
- OpenSplice Powertools™
- Whats next

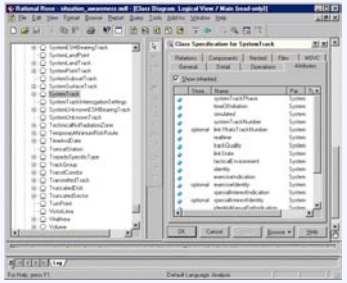
OpenSplice DDS Development Support



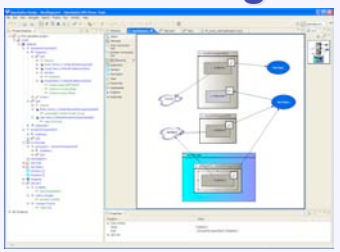
OpenSplice DDS PowerTools™ MDE Suite



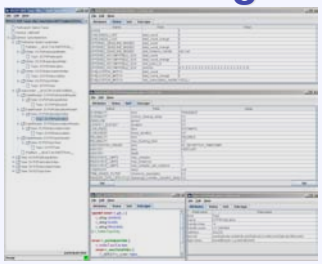
Information modeling



Application modeling

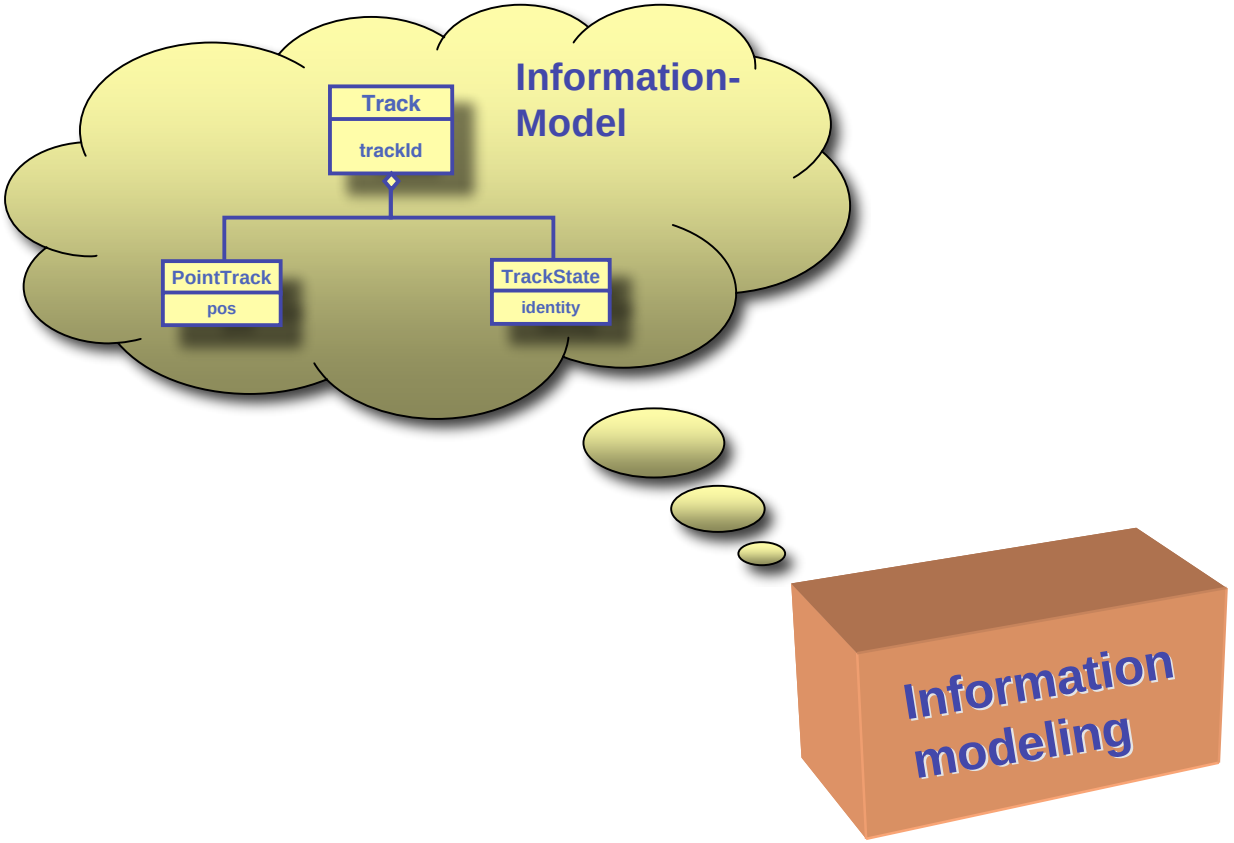


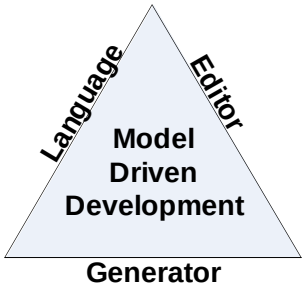
Deployment modeling



PRISMTECH

Information modeling

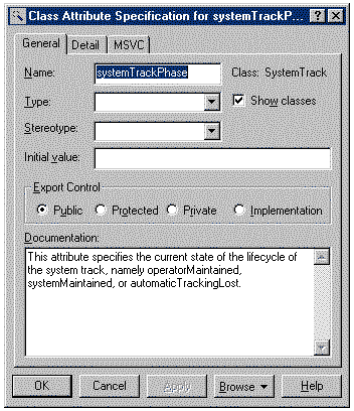




Information modeling: Scope & Purpose

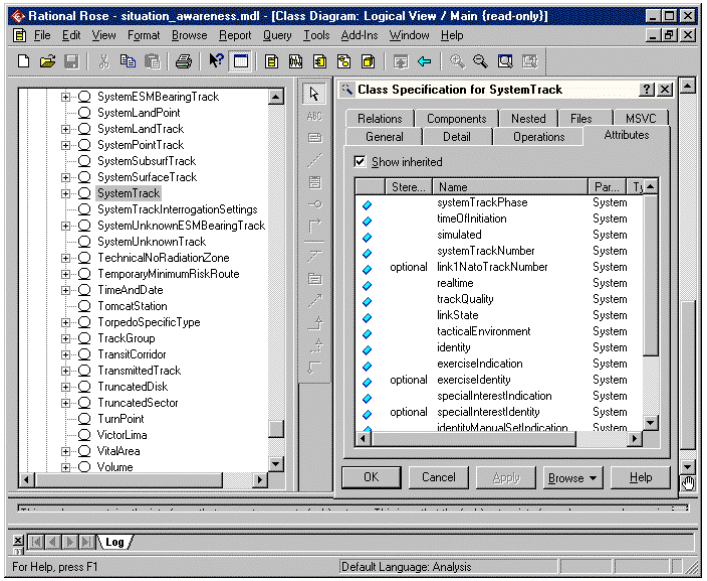
Language

- 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 + keys + QoS)



Editor

- Semantics: UML (‘annotated’)
- Syntax: IDL (can be imported)
- Behaviour: QoS (reliability, urgency, durability)

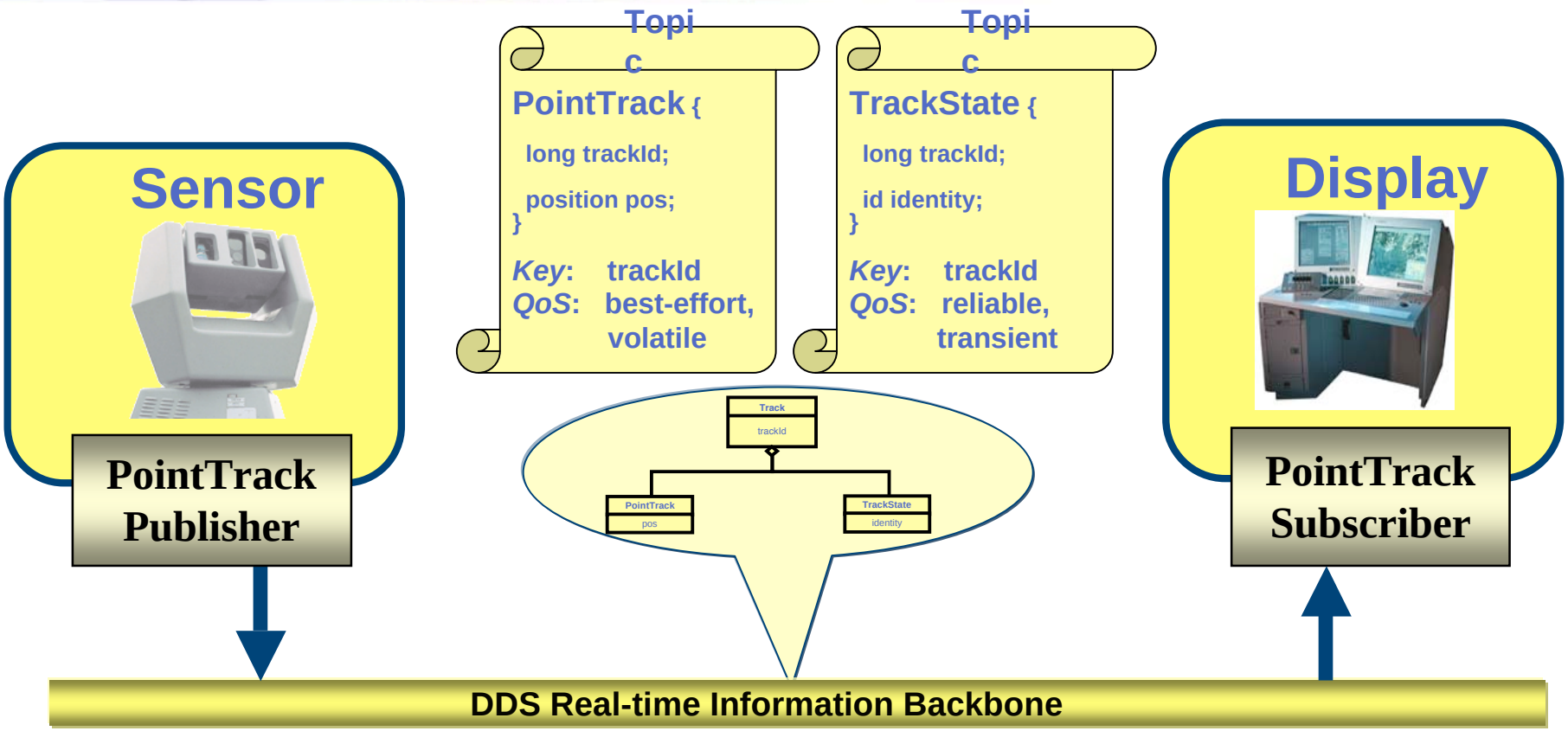


Generator

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

PRISMTECH

Information modeling (example)



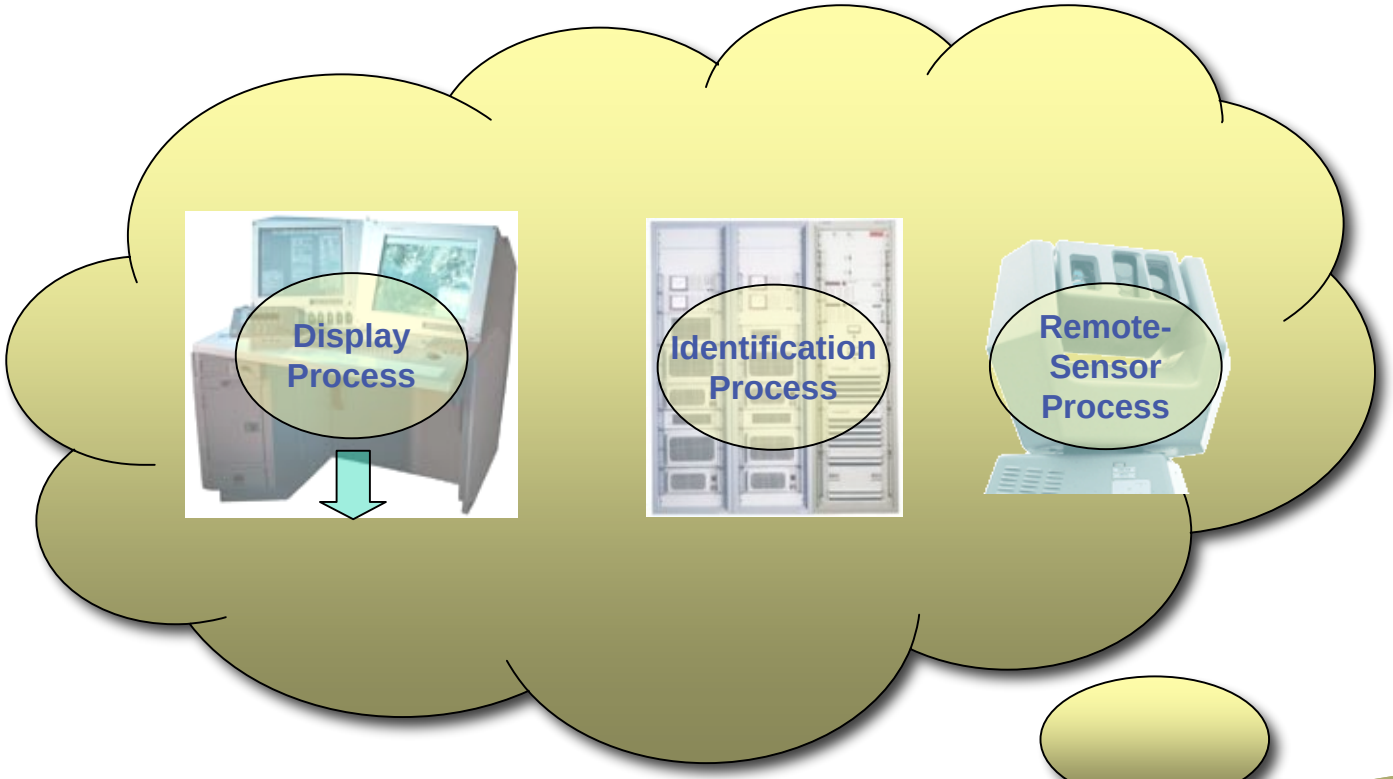
Task: Information modeling

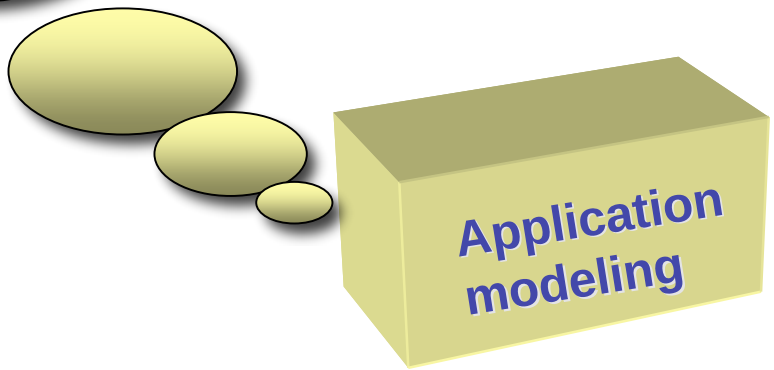
- Model the 'PointTrack' and 'TrackState' topics
- Model the system-wide behaviour of topics (QoS)
- Model the relationships between the Topics (keys)
- Separate these concerns from applications

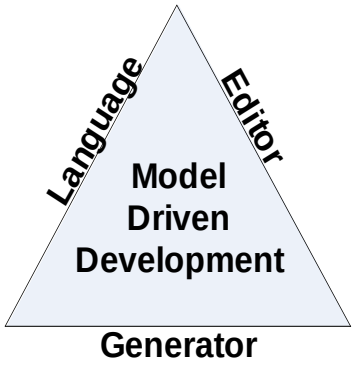
MDE: Features / Advantages

- 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

Application modeling







Application modeling: Scope & Purpose

Language

- Processing-language
- DDS ‘keywords’
  - publishers/writers, subscribers/readers,..
  - Content-filters, queries, waitsets, listeners,…

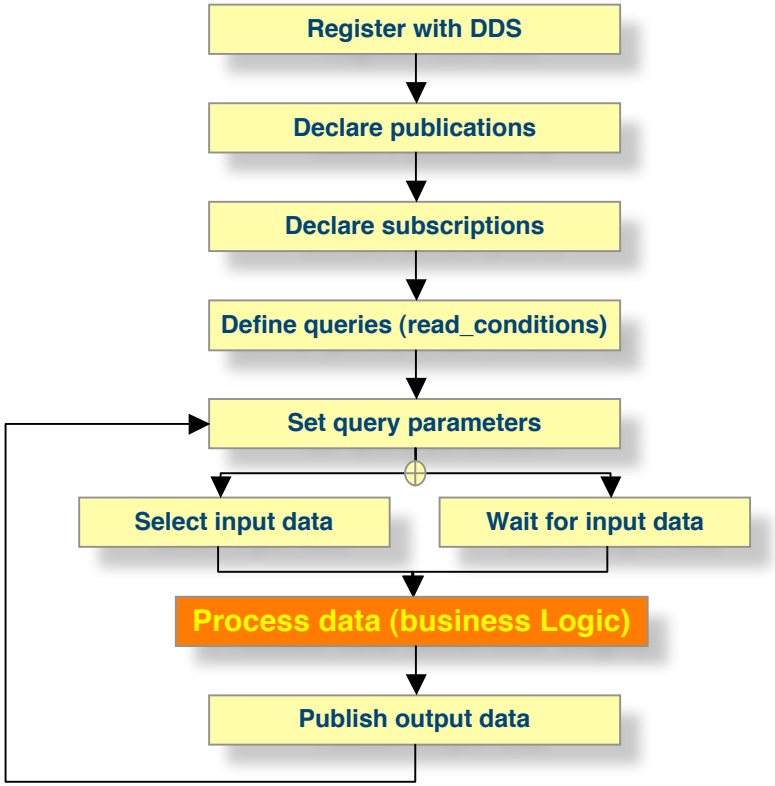
Editor

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

Generator

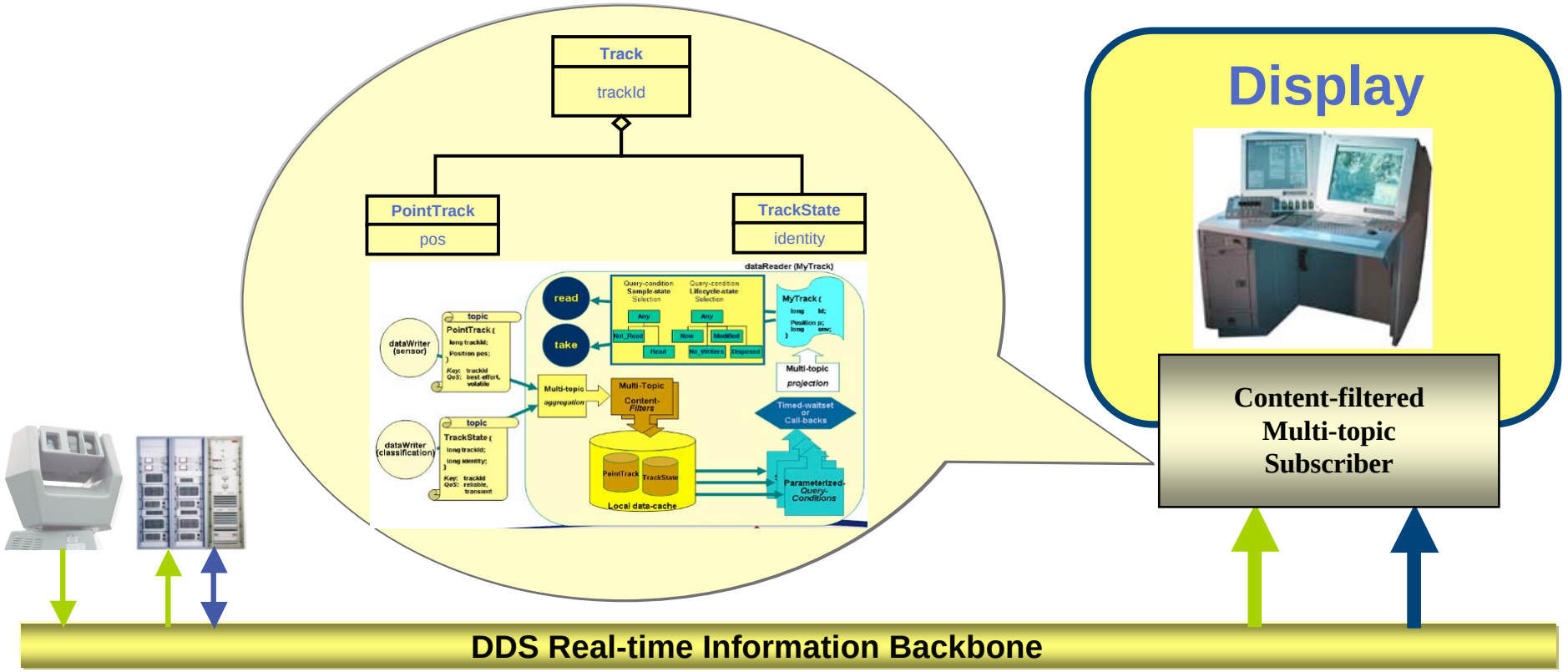
- Application-framework from templates
- DDS entities from (graphical) model
- Annotated application- and test-code

Simple DDS-Loop Pattern



PRISMTECH

Application modeling (example)



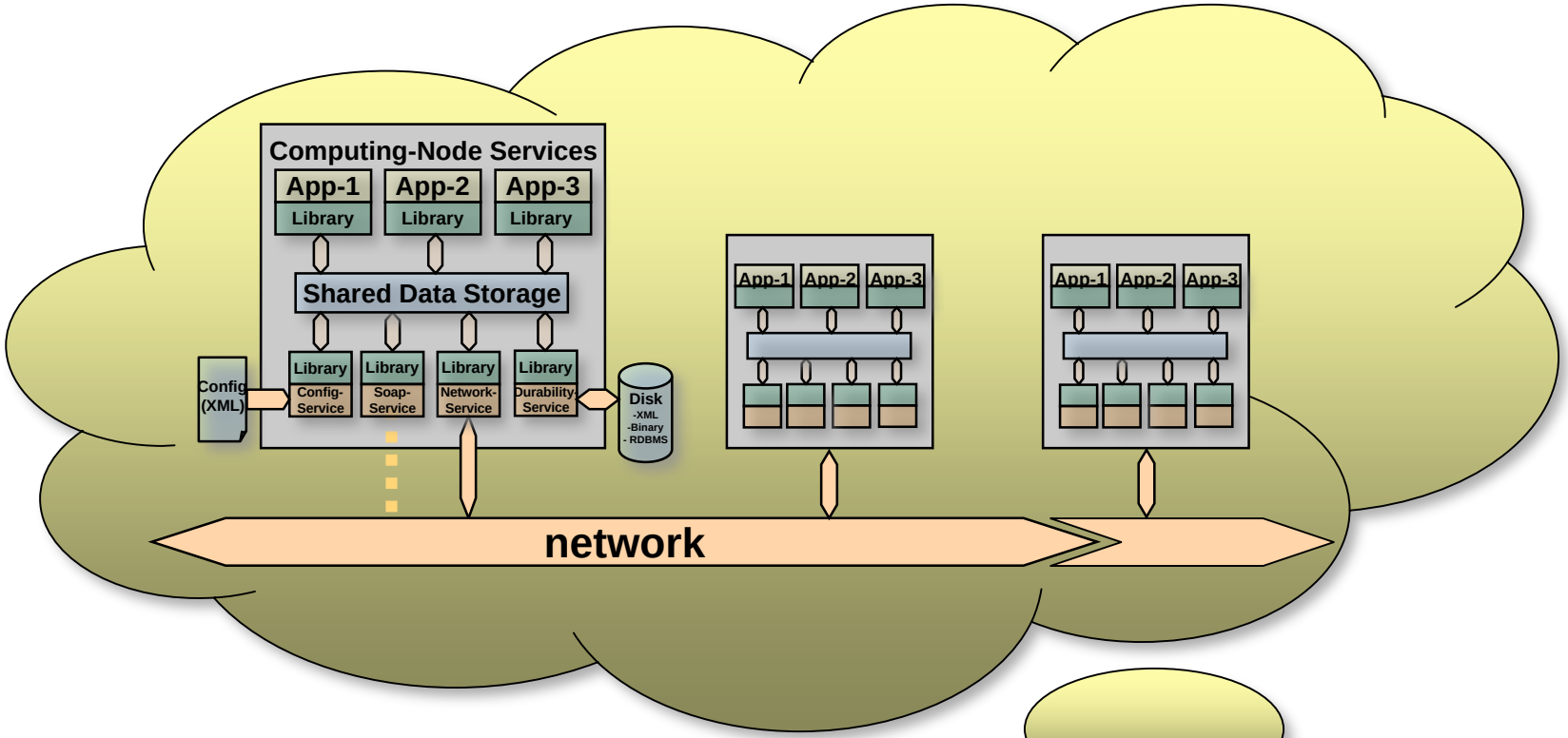
Task: Display Process modeling

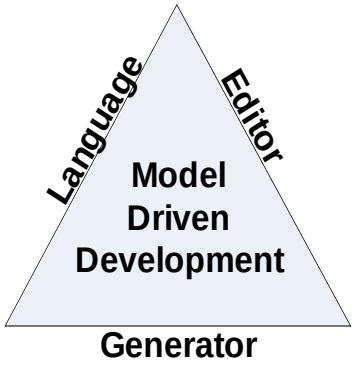
- Model 'aggregate interest'
  - 'multi-topic' or 'DLRL-object' with local 'QoS'
- Model display process (periodic loop)
  - Handle first-appearances of hostile tracks with prio

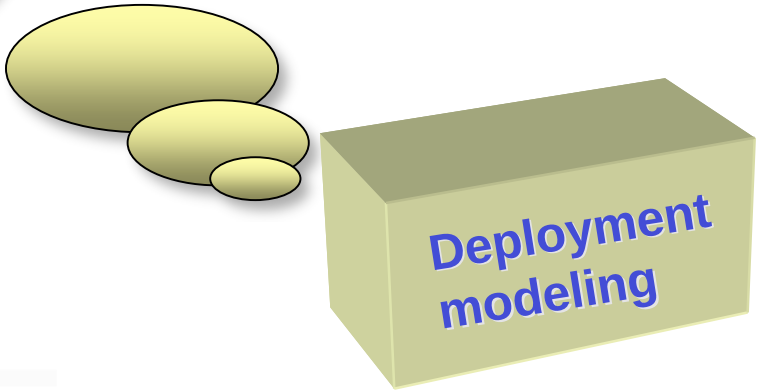
MDE: Features / Advantages

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

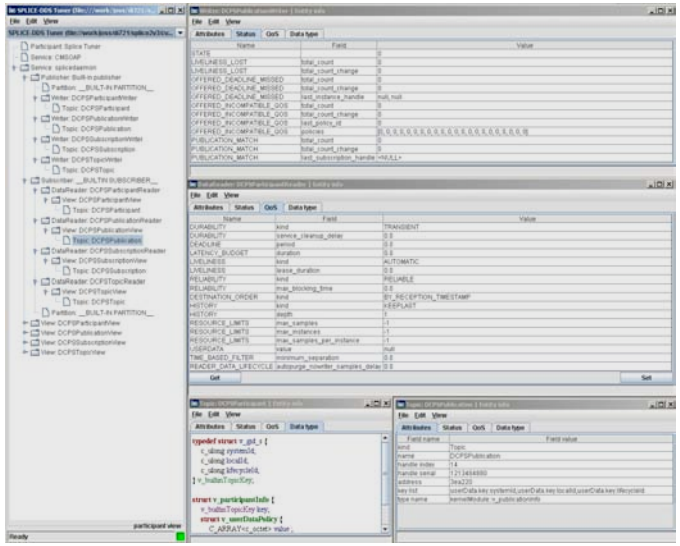
Deployment modeling







Deployment modeling: Scope & Purpose

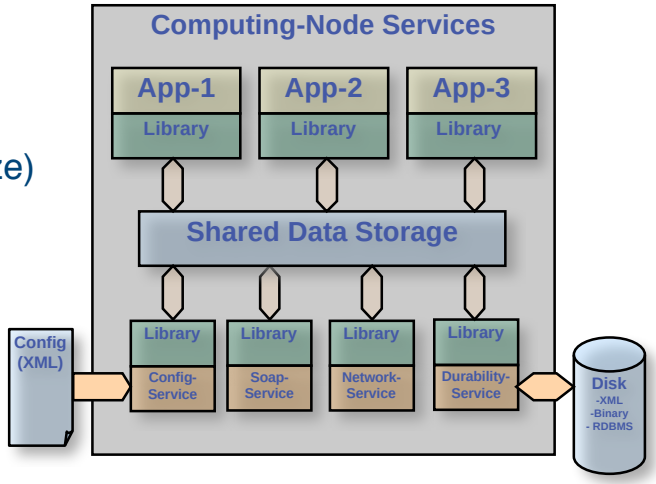


Language

- Deployment-language
- DDS ‘keywords’
  - Participants, Partitions, Resource limits,..
  - Latency-budget, Transport-priority,…

Editor

- Configure ‘physical’ DDS system
  - Networking, Durability, Resource-limits, …
- Map dynamic DDS Policies to deployment properties
  - DDS ‘partitions’ to *Network-partitions* (multicast groups)
  - DDS ‘transport-priority’ to *Network-channels* (OS-prio/diffserv/burstsize)

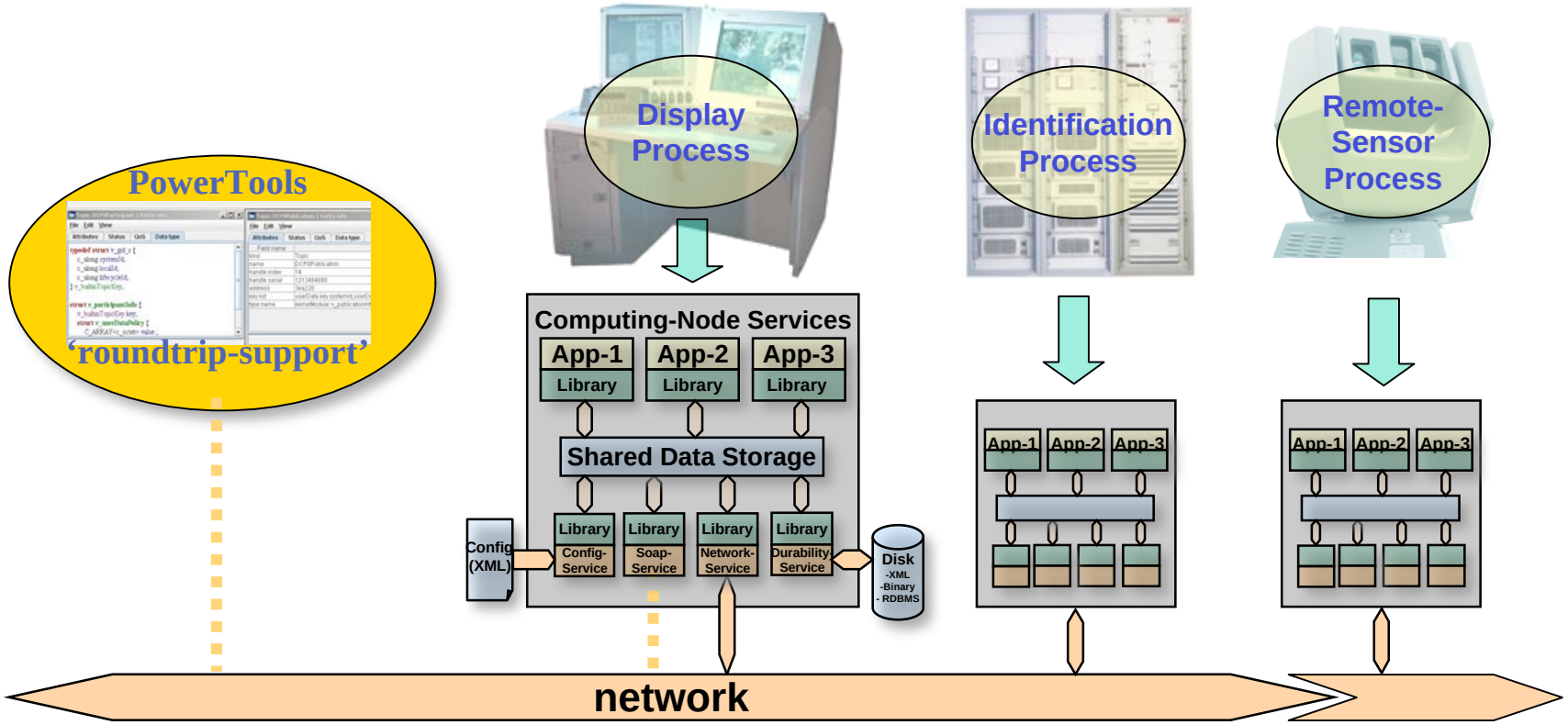


‘Generator’ (Control & Monitoring)

- (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,…)

PRISMTECH

Deployment modeling (example)



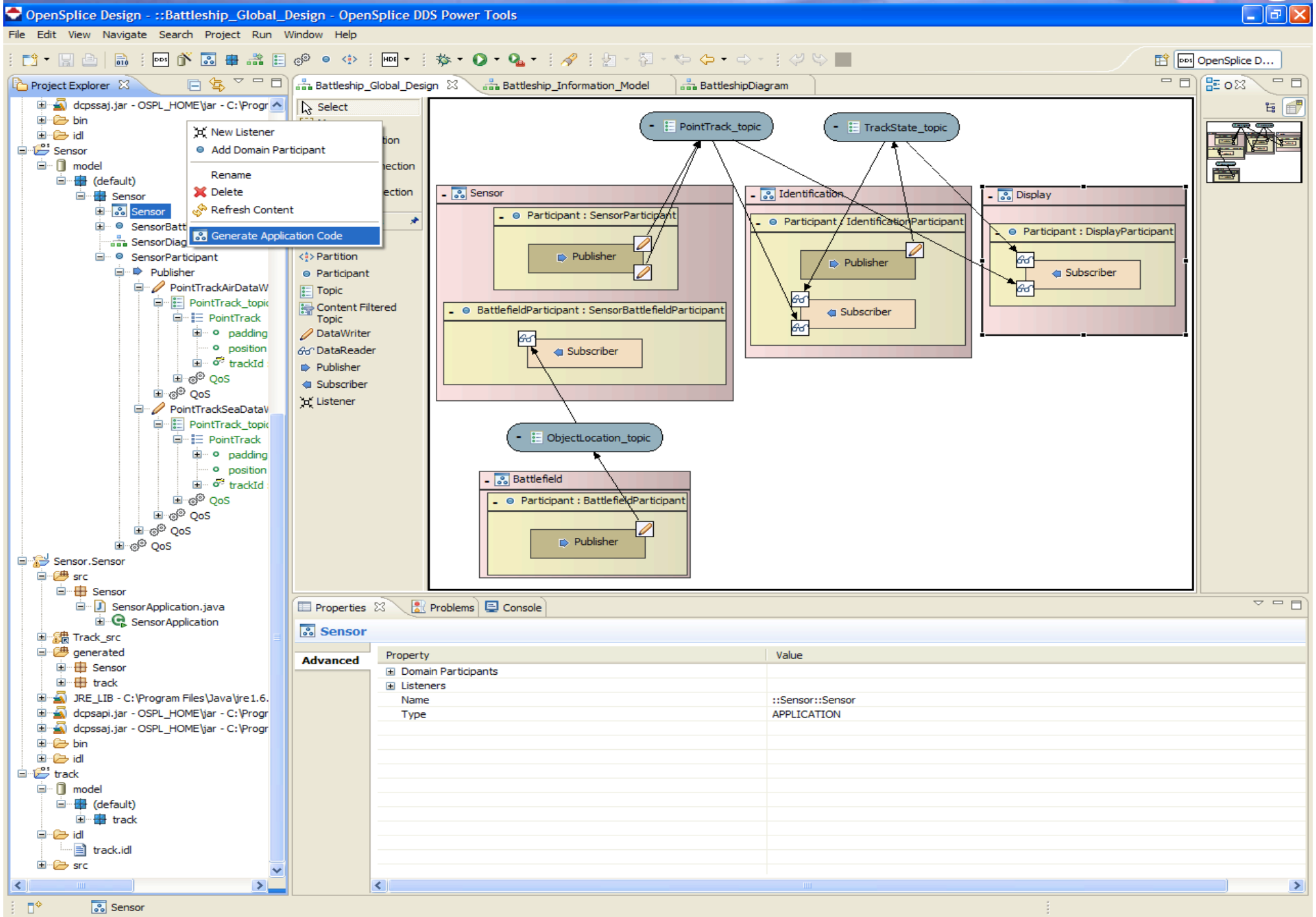
Task: Deployment modeling ('Control & Monitoring')

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

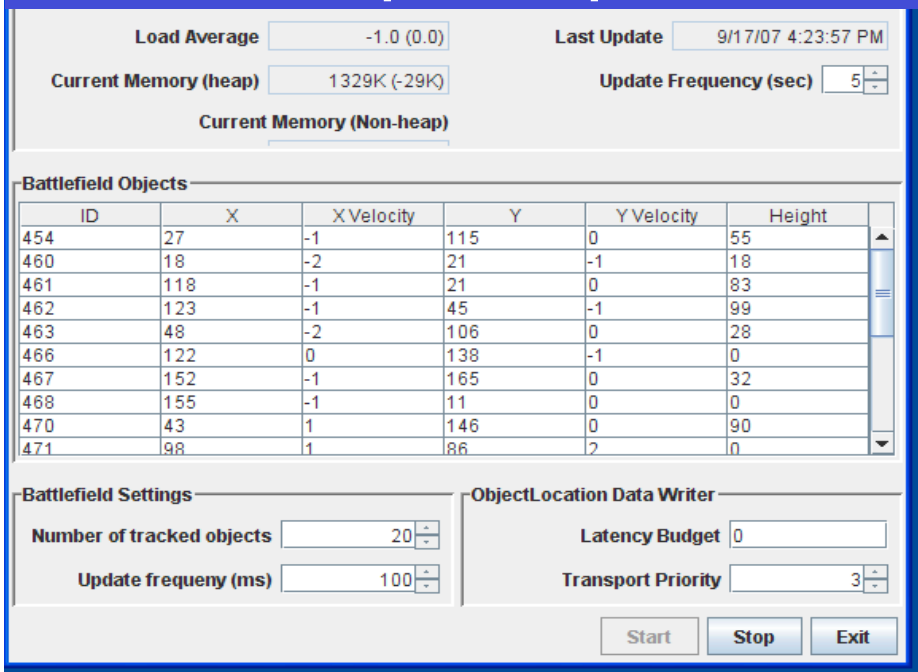
MDE: Features / Advantages

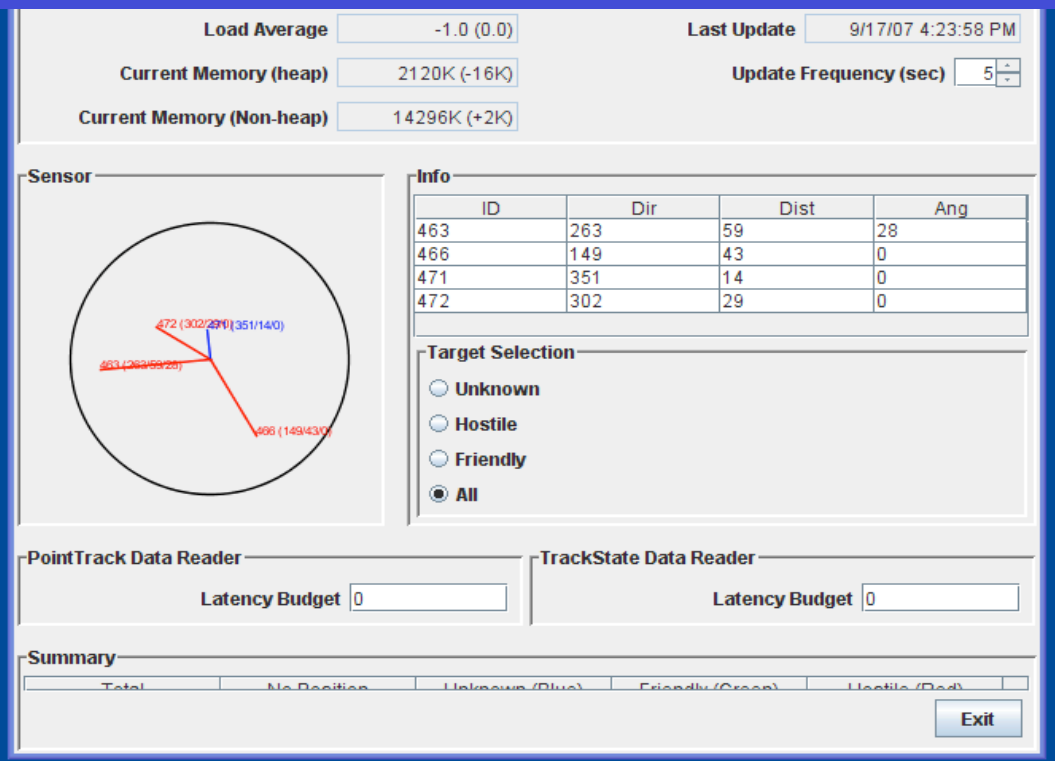
- Round-trip engineering
  - Deploy & tune models (info & application)
- Control & Monitor deployed system
  - Using the domain specific 'DDS-entity language'

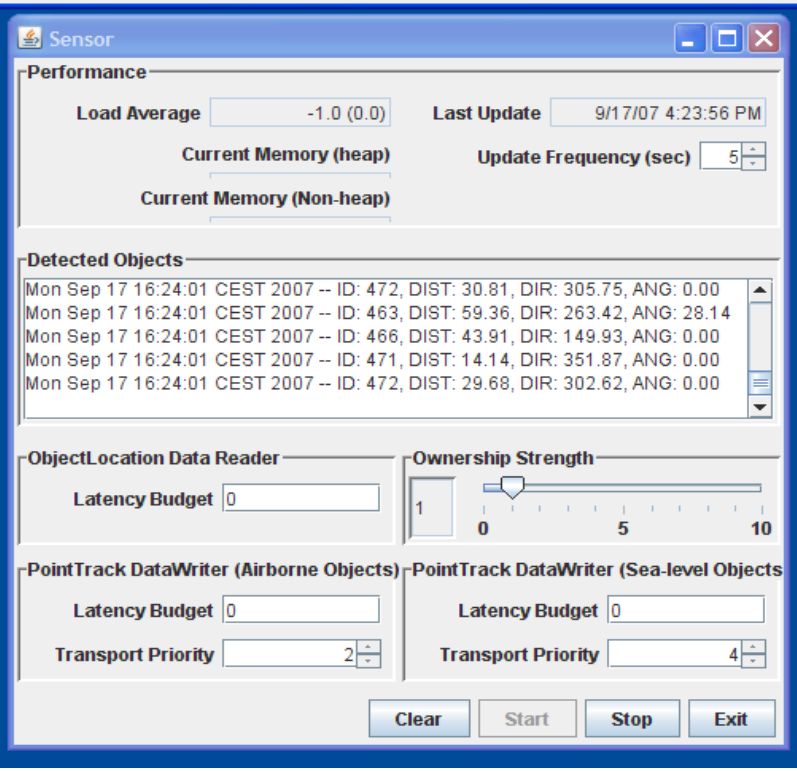
Modeling the Example

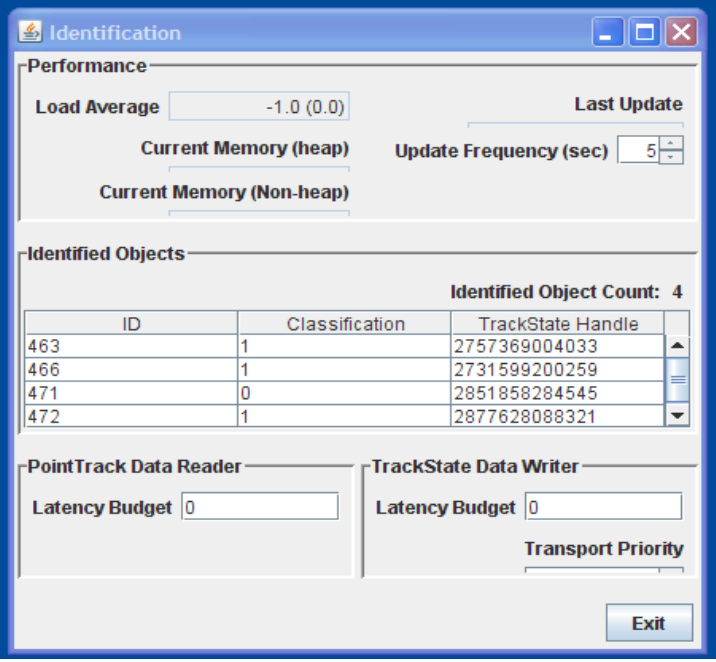


The “Battleship” example ‘at work’





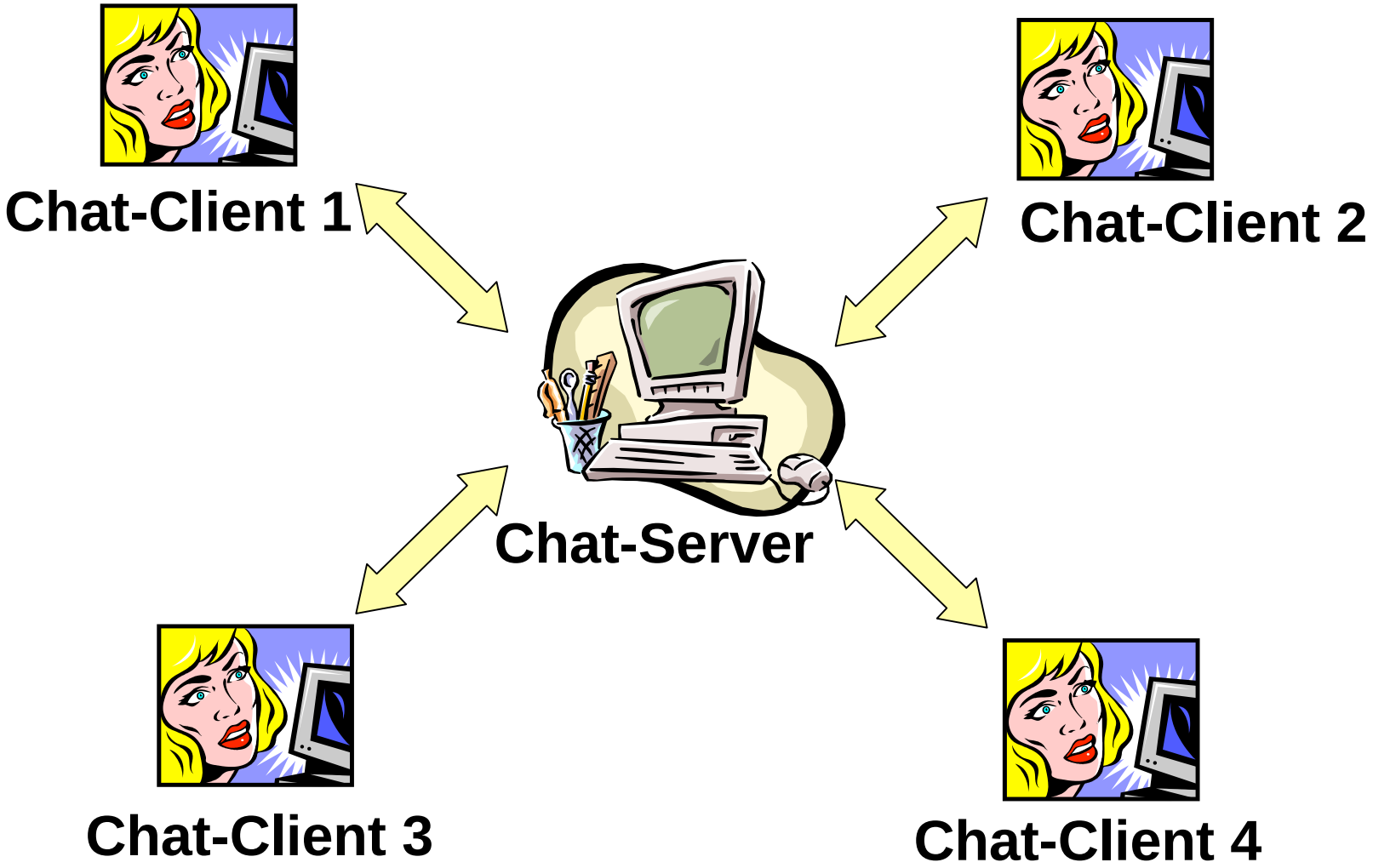




Agenda

- OpenSplice DDS Overview
- Information modeling
- DDS by Example
- Modeling the Example
- OpenSplice Powertools™
- Whats next

Chatroom Example – Traditional architecture



Chatroom Example – DDS based architecture

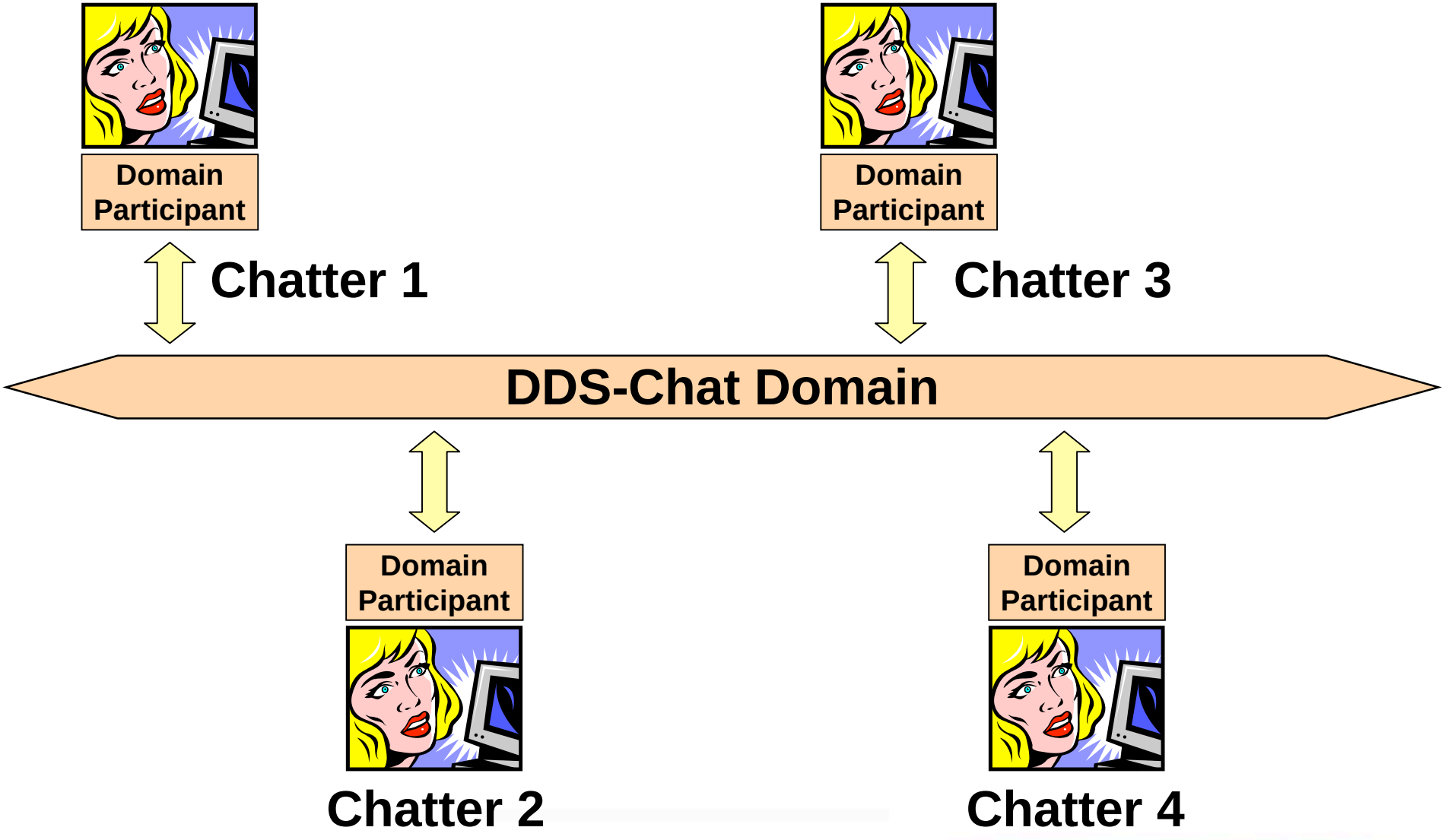
Typical sequence of events on a **traditional** Chat-application:

- **Connect** to the Chat-Server.
- **Transmit** your identity.
- **Download** the identities of the other chatters.
- **Receive** chat messages from other users.
  - These messages are **forwarded** to you by the server.
- **Write** your own chat-messages.
  - These messages are **forwarded** by the server to all the other users.

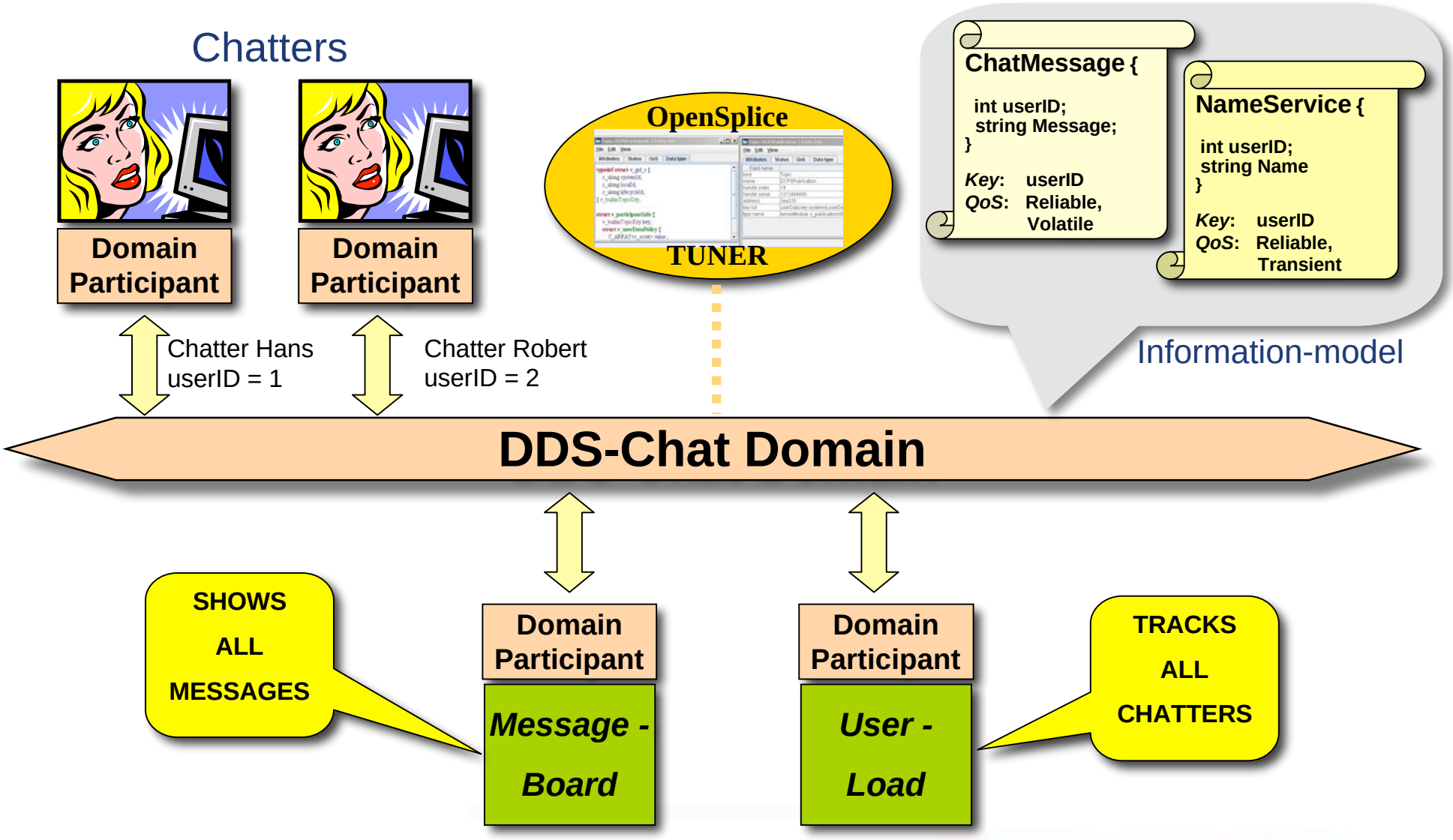
Typical sequence of events on a **DDS-based** Chat-application:

- **Participate** in the Chat-domain.
- **Publish** your identity.
- **Subscribe** yourself to the identities of all other chatters
- **Subscribe** yourself to all chat-messages in the Chat-domain.
  - All messages are delivered to you **directly** by their respective writers.
- **Publish** your own chat-messages.
  - Your messages are **directly** delivered to all the other interested users.

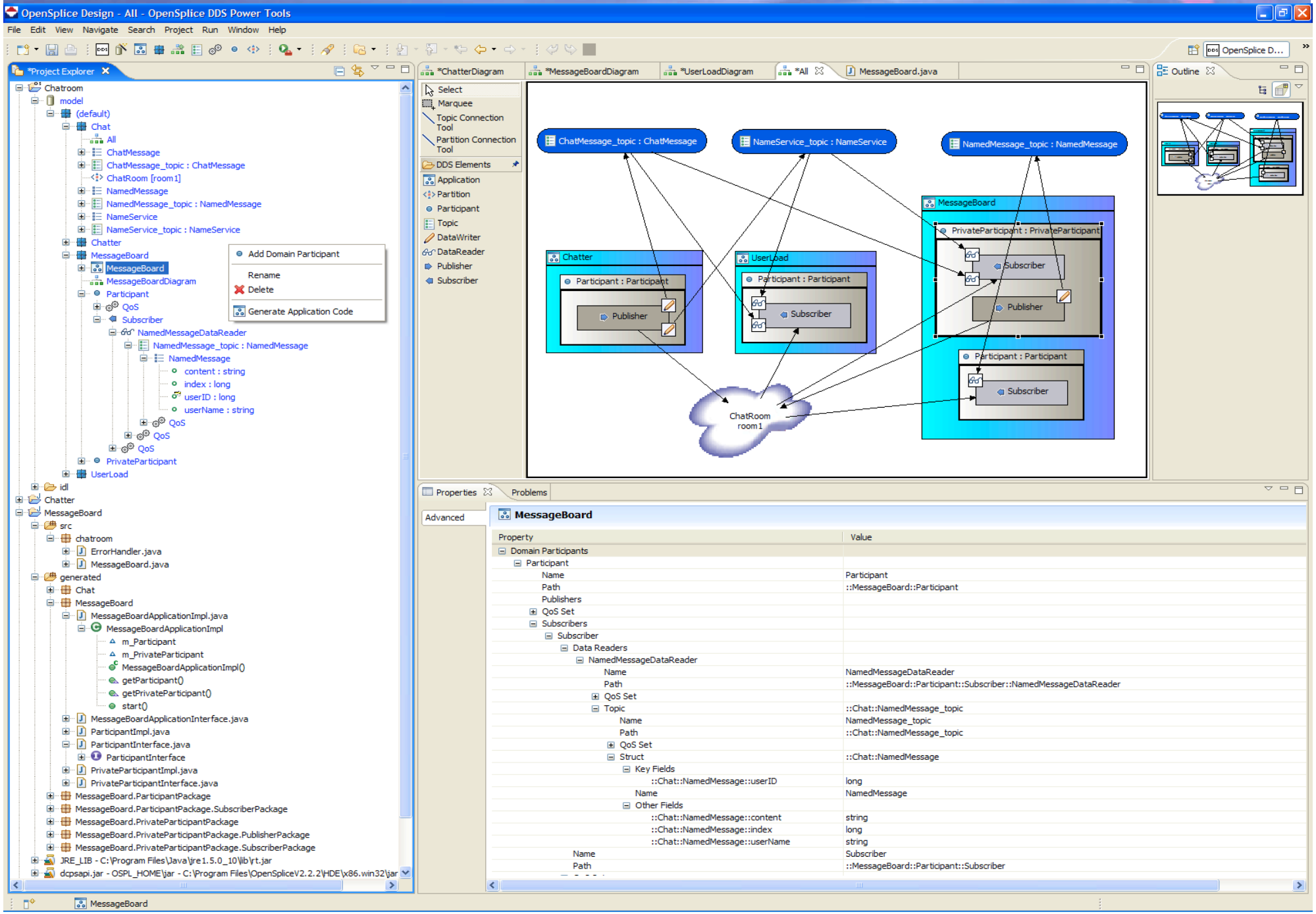
Chatroom Example – DDS based architecture



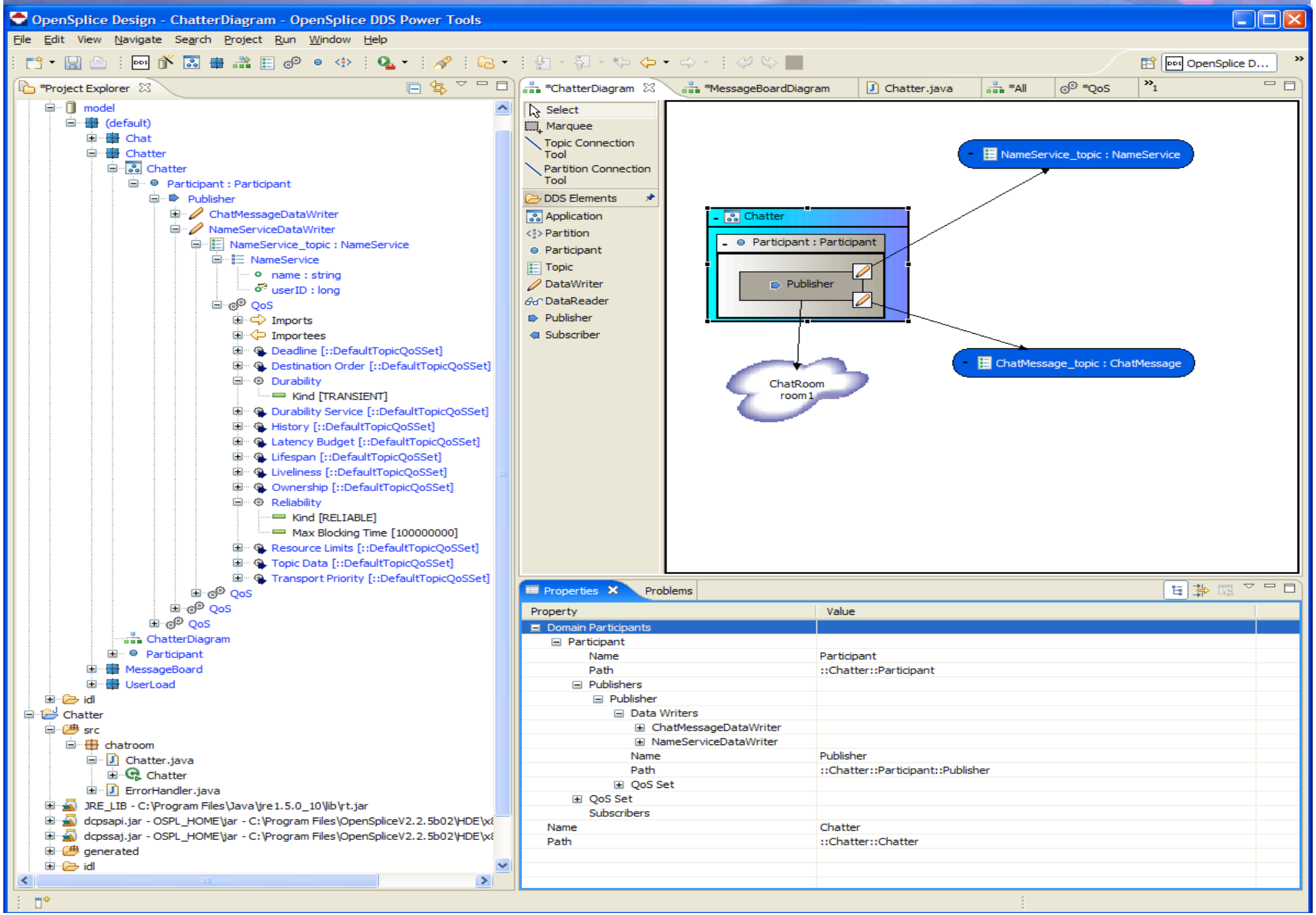
Chatroom Example – Applications



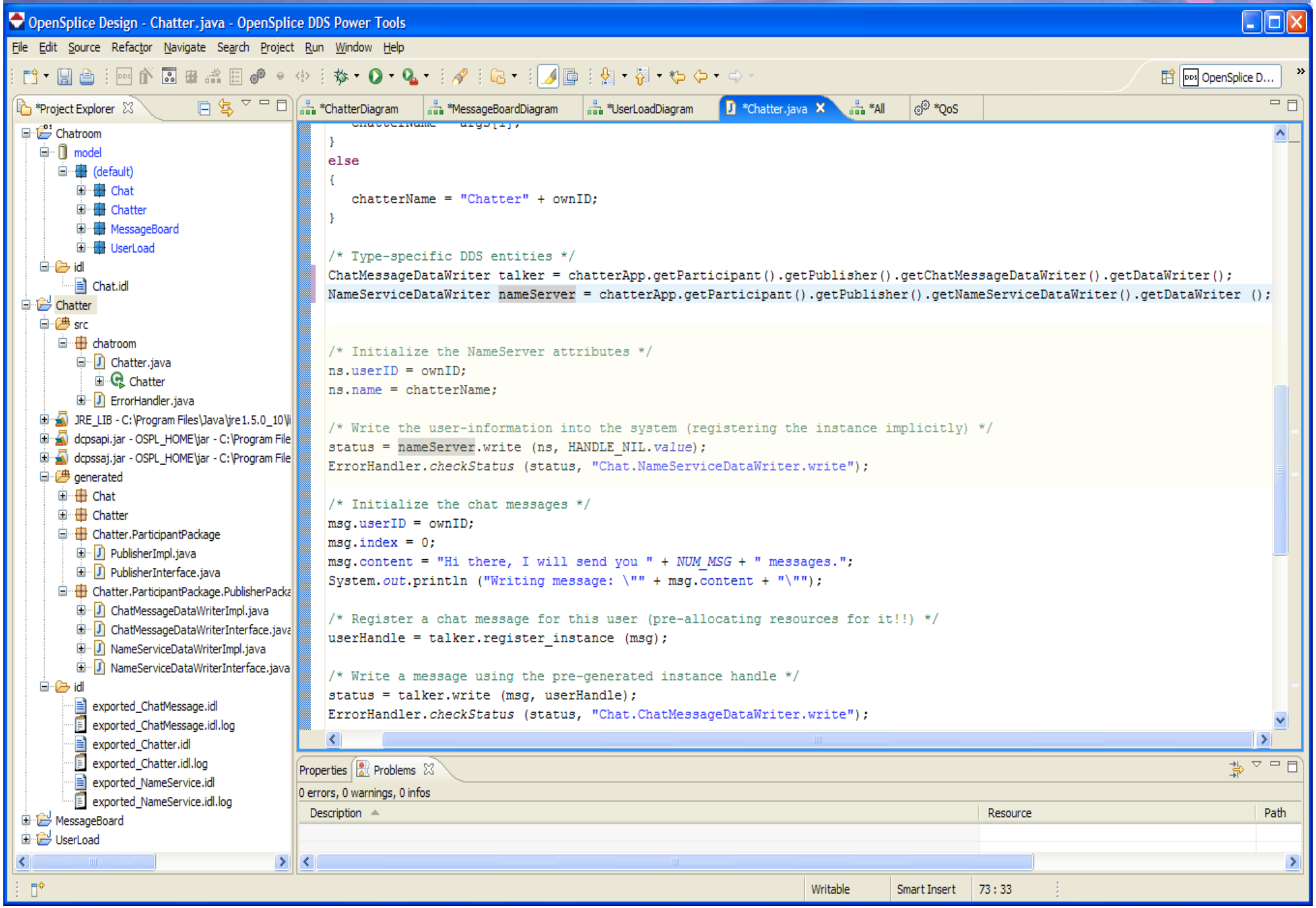
OpenSplice Power Tools™: Eclipse based MDE-suite



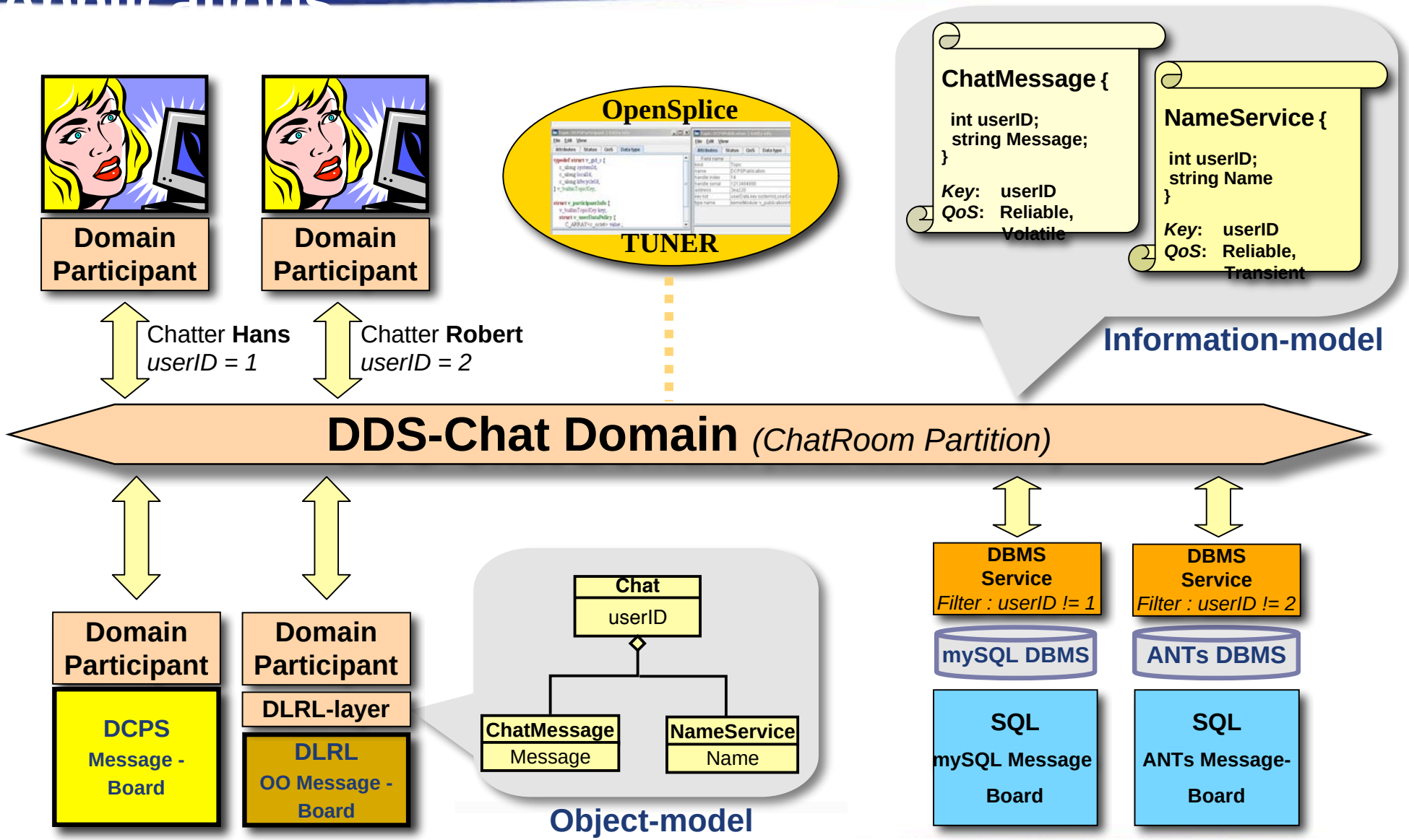
PowerTools™ Graphical modeling: Intuitive, Easy and Fast



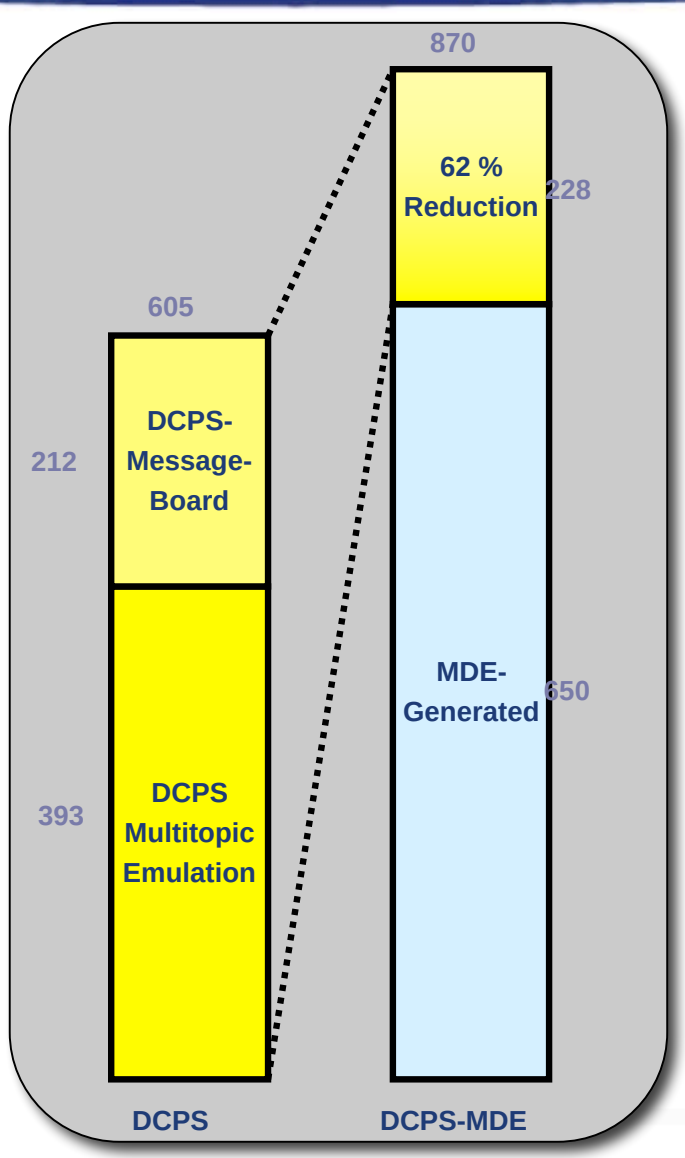
PowerTools™ Code Generation: Reduced complexity



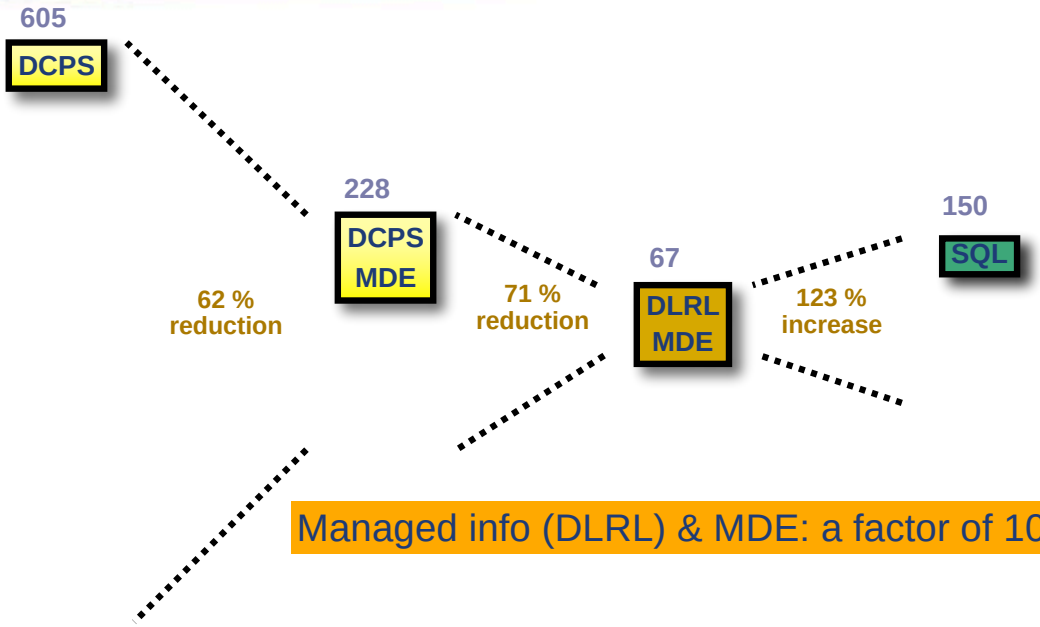
Chatroom Example – DCPS/DLRL & SQL Applications



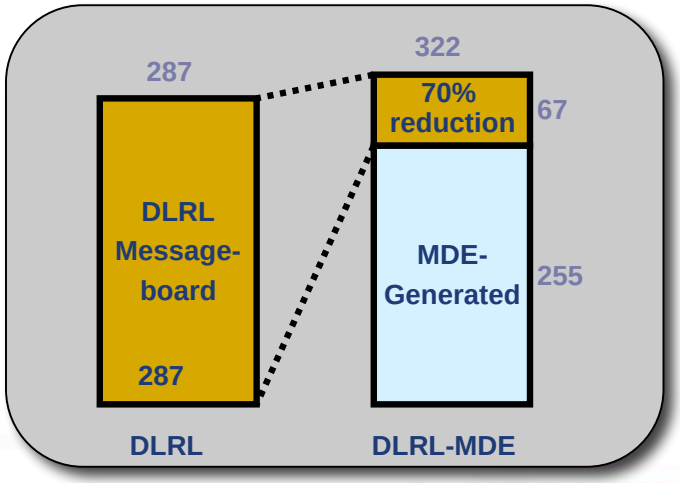
MessageBoard Example – Comparing architectures by size



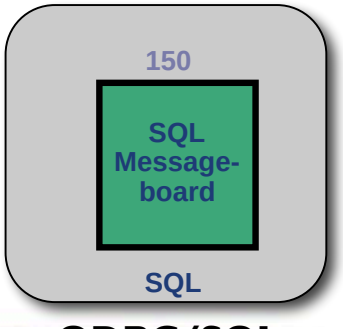
DDS/DCPS



Managed info (DLRL) & MDE: a factor of 10 !



DDS/DLRL

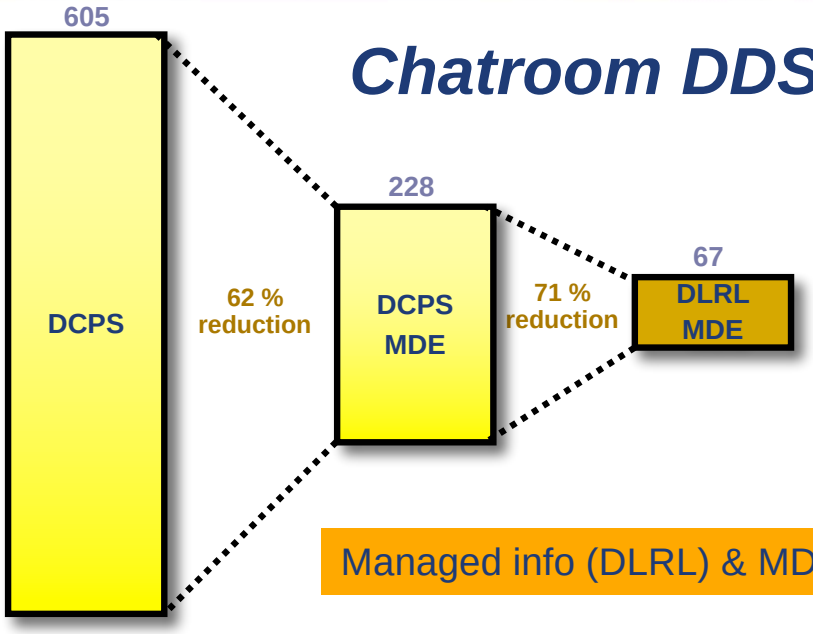


ODBC/SQL

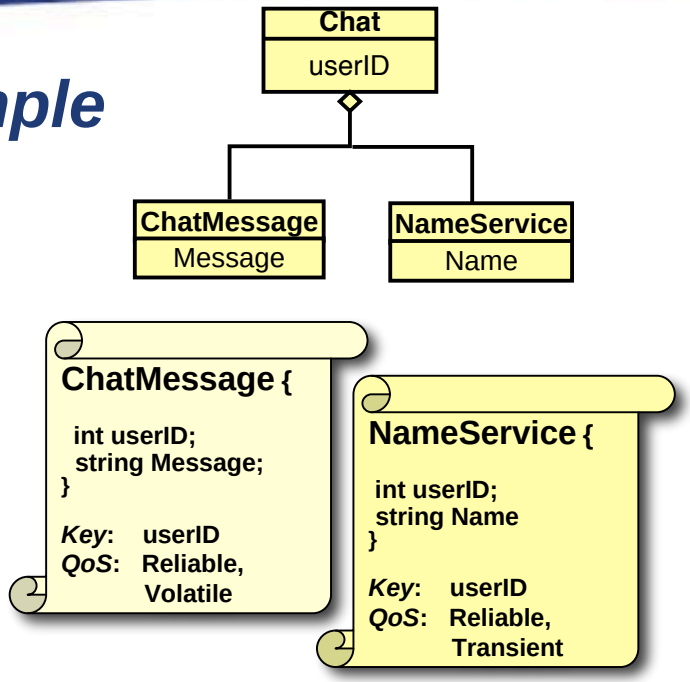
OpenSplice DDS

PRISMTECH

DCPS/DLRL Modeling – Summary



Chatroom DDS tutorial example



Managed info (DLRL) & MDE: a factor of 10 !

OpenSplice DDS, PowerTools™ MDE-suite for modeling & code-generation

- *Supports information/topic modeling: IDL-import and QoS annotation*
- *Supports application modeling: DDS entity/interaction modeling and code-generation*
- *Supports deployment modeling: OpenSplice Tuner™ for remote monitor & control (incl. QoS tuning)*

OpenSplice DDS, DLRL API

- *Object relationship-management greatly reducing application complexity*
- *Extensive selection and fine-grained listener mechanisms ease application design*
- *High-performance/low-overhead due to DLRL-support by DCPS-kernel in-memory OO-database*

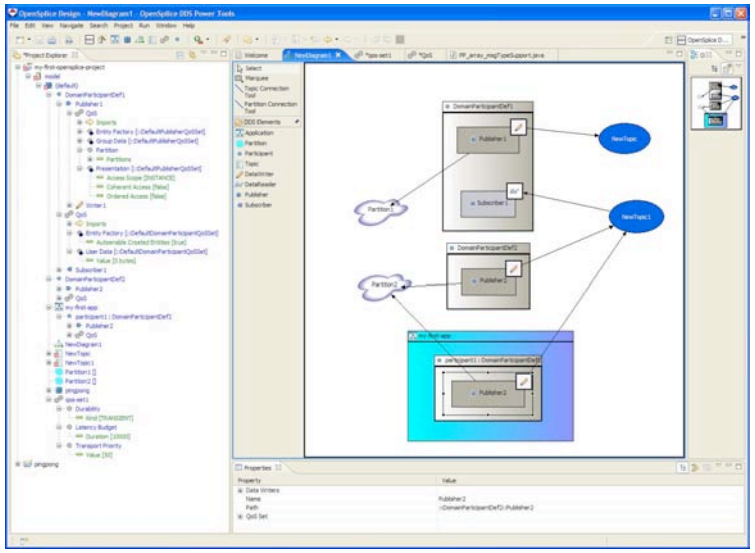
Application size

- *Chatroom example (simple application) shows a **62%** LOC reduction **by MDE** another **71%** LOC reduction **by DLRL***

MDE tool-suite: Summary & Advantages

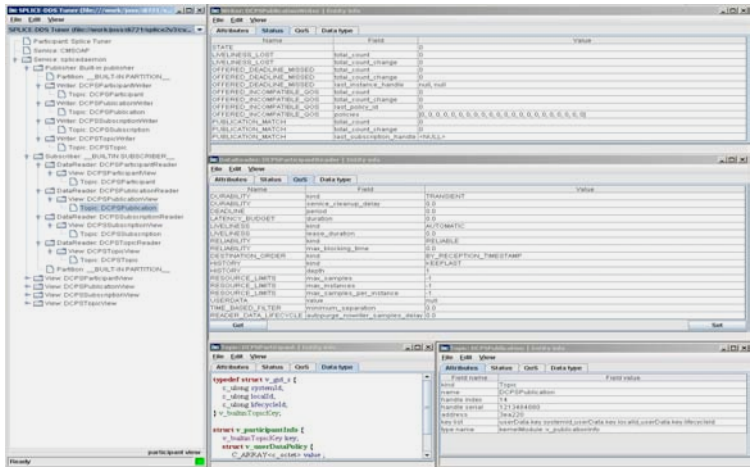
Complete modeling of system design cycle

- Information/application/deployment Modeling
- Context aware guidance / well defined steps
- **Fast, intuitive, correct**



Information Modeling

- Graphical system-wide information + QoS modeling
- DDS code-generation of topics and typed readers/writers
- **Documented packages of re-usable topic-sets**



Application Modeling

- Information-model (parts) import
- Graphical Application Modeling
- **Code-generation from patterns (listener/waitset/MVC)**

Deployment Modeling

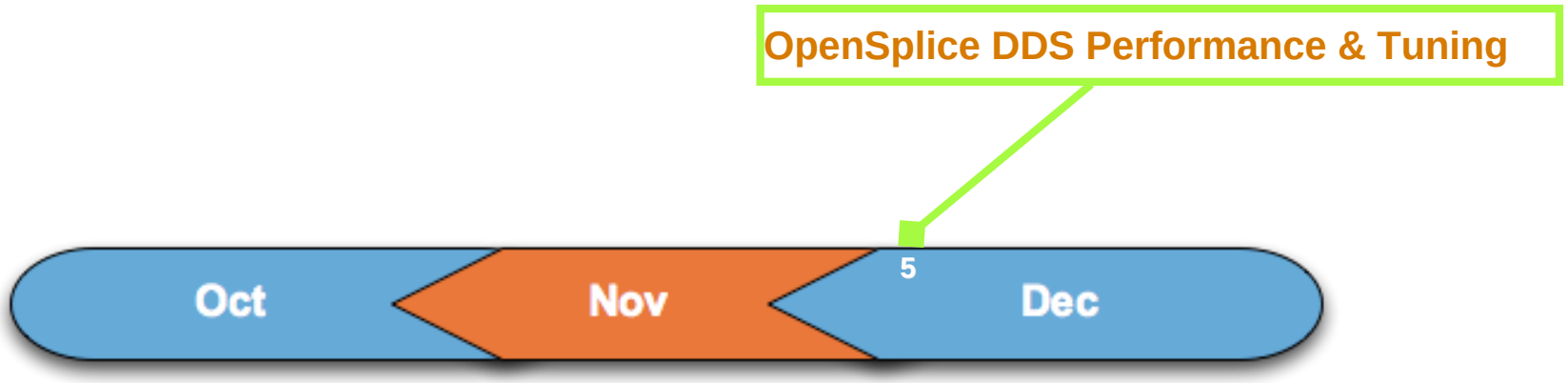
- Modeling of DDS-configuration
- Service configuration (networking, durability)
- **Runtime control (& round-trip engineering) by OpenSplice Tuner™**

Agenda

- OpenSplice DDS Overview
- Information modeling
- DDS by Example
- Modeling the Example
- OpenSplice Powertools™
- Whats next

Dr. Angelo Corsaro

Upcoming Webinars



2007

Registration: http://www.prismtech.com/section-item.asp?id=731&sid=29&sid2=15&sid3=289

Concluding Remarks

Architecture

- OpenSplice DDS™ uniquely implements the full OMG DDS v1.2 standard combining Real-time Pub/Sub with Data-centricity
- OpenSplice DDS™ can therefore significantly reduce system complexity and enhance component re-use

MDE Tool suite

- OpenSplice PowerTools™ optimally support data-centric system engineering
- OpenSplice PowerTools™ Promote a clear separation of concerns between information-modeling, application-modeling/code-generation and deployment
- Java/Eclipse based toolsuite for development, deployment as well as remote maintenance

OpenSplice DDS is the best solution available on the market for solving your data distribution problems!

Contact Us

OpenSplice™ DDS

- OpenSpliceDDS Home Page
  - http://www.prismtech.com/opensplice-dds/
- For Information on OpenSplice DDS, or for evaluation licenses, contact:
  - opensplicedds@prismtech.com -or-
  - sales@prismtech.com
- OMG DDS Information
  - http://www.dds-forum.org/
  - http://portals.omg.org/dds/