OMG-DDS “Exploiting the Potential”
[ Proven suitability in the Naval Combat System Domain ]

OMG Real-time & Embedded Systems Workshop
July 2005
Hans van ’t Hag
hans.vanthag@nl.thalesgroup.com
(2) Combat systems: ‘If your life depends on it …’
(3) Architecture requirements: ‘Challenge your middleware …’
(4) CMS Use-Case: ‘DDS by example …’
(5) Conclusion: ‘SPLICE-DDS: Proven suitability …’
(1) ARCHITECTURE TRENDS

“Is there life after client-server ...”
**Key Trend**
- DoD system requirements are increasingly more dynamic, diverse, & demanding

**Problems**
- Existing architectures
- Existing COTS
- Existing multiple technology bases
- brittle & configured statically
- too big, slow, buggy, incapable, & inflexible
- proprietary & limit effectiveness by impeding
  - **Assurability** (of QoS),
  - **Adaptability**
  - **Affordability**

**Today, each system brings its own:**
- networks
- computers
- displays
- software
- people

**Consequences**
- Hard to meet required performance levels
- Hard to control distributed resources
- High software lifecycle costs
  - e.g., many “accidental complexities” & low-level platform dependencies
Open Architecture Computing Environment (OACE)

**Unique**

- Navy Controls
  - Standards
  - Specifications
  - Interfaces

- Industry Builds Unique Systems
  - NAVY Controls Interfaces

**Common**

- **COMMON FUNCTIONS ACROSS ALL SHIP CLASSES**

- **Open Architecture Computing Environment** (Common Standards & Guidance)

- **Architecture, Technologies, Standards, Specifications and Products**

- **Validation**

**Design Guidance**

- Collaborate With Defense Industry Via Ipts For:
  - Common Component Specifications for Industry to Build
  - Interface Specification

- Collaborate With Commercial Computing Industry For:
  - Computing Standards
  - Technology Base
### TRENDS: © SUN

<table>
<thead>
<tr>
<th>Catch Phrase</th>
<th>Client-Server</th>
<th>3/N-Tier</th>
<th>Net Apps</th>
<th>Net Services</th>
<th>Next</th>
<th>After that</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Network Is the computer</td>
<td>The Network Is the computer</td>
<td>Objects</td>
<td>Legacy to the Web</td>
<td>The Computer Is the Network</td>
<td>Network of embedded things</td>
<td>Network of Things</td>
</tr>
<tr>
<td>System</td>
<td>Scale</td>
<td>When/Peak</td>
<td>Leaf Protocol(s)</td>
<td>Directory(s)</td>
<td>Session</td>
<td>Schematic</td>
</tr>
<tr>
<td></td>
<td>100s</td>
<td>1000s</td>
<td>1000000s</td>
<td>10000000s</td>
<td>100000000s</td>
<td>1000000000s</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>+HTTP (+/VM)</td>
<td>+LDAP (*)</td>
<td>+XML</td>
<td>Unknown</td>
</tr>
<tr>
<td></td>
<td>NIS, NIS+</td>
<td>+CDS</td>
<td>+LDAP (*)</td>
<td>+CORBA, RMI</td>
<td>+SOAP, XML</td>
<td>+RMI/Jini</td>
</tr>
<tr>
<td></td>
<td>RPC, XDR</td>
<td>+CORBA</td>
<td>+CORBA, RMI</td>
<td>+CORBA, RMI</td>
<td>+SOAP, XML</td>
<td>+RMI/Jini</td>
</tr>
</tbody>
</table>

**“OMG-DDS ??!!”**

Corba Web Java Information Grids
(2) Combat Management Systems

“If your live depends on it ...”
OPERATIONAL REQUIREMENTS

- Vast amounts of information
- Thousands of tracks
- High degree of automation
- Automatic multi-sensor data fusion
- Automatic threat evaluation
- Automatic sensor + weapon pairing
- Extensive Command Support
- Easy to Operate
- Easy to Maintain
- Flexible in Missions, and Manning
- On-line Simulation & Training
- Total System Integration
Data-traffic: 4,000 publications per second over the system-data bus
Programs: 2,200 programs allocated over 150 processors
Accuracy: 100 us. time-alignment accuracy within the distributed system
CMS ARCHITECTURE: Starting points

Fault-tolerant: High combat survivability & maintainability (no single-point-of-failure)
Flexible: Mission-based configuration, on-board training & simulation
Evolvable: Evolutionary upgrading based on COTS & Open Standards
Scalable: From patrol-boats up to destroyers

Applications dynamically distributed over a “Pool of COTS Computers”
CMS: Dynamic Resource Management

- ‘pool-of-computers’ utilization (Allocation schemes)
  - Allocation driven by resource-needs and availability (CPU-power, Memory)
  - ‘Dispersion groups’ to force geographical separation (battle-damage resistance)
  - ‘Preference allocation groups’ to advice co-location (efficiency)
  - Degradation driven by ‘Functional Priorities’ (importance)
DDS - Information Backbone

Autonomous components
Interacting only with the information-bus
Spontaneous: Z, Self-healing: D'
Redundant & Replicated: L', Y'
QOS-driven Data Distribution Service (reliability, persistency, latency): DDS
COMMON ENTITY MODEL: Deployment – F124

Design Authority
Joint Program Team

Joint Gov/Industry team

Common Entity Model

ATP1, STANAGs, ADaTP3

THALES

EADS

Raytheon

REAL-TIME (OMG-DDS/SPLICE) Information Backbone
COMMON ENTITY MODEL: The "CMS Glossary"
Business Demands: re-use, flexibility, scalability

CHARACTERISTICS

Many different customers: 12 Navies world-wide use 1 CMS product-line
Many different ships/missions: 20 Ships classes (from FPB’s up to Destroyers)
(3) ARCHITECTURE REQUIREMENTS

“Challenge your middleware ...”
Summary: 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: message-passing, client/server, OO
- most efforts fall short in a number of dimensions:
  - typically:
    - limited performance (high-volume & low-latency balance)
    - exploding complexity (dependencies in many dimensions)
    - costly evolution (impact of changes & extensions)
Architectural Requirements: Lifecycle focus

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 **runtime migration** of applications
- allow applications to **join** the system at **any time**

System maintenance & evolution
- allow runtime replacement and **evolutionary upgrading**
- support for **logging & replay** of information
- provide **future-proof, reusable, robust and scalable** system
Towards a solution:
- make development effort more simple
- develop less
- develop solutions only once

How:
- minimize component dependencies (‘simple’)
- maximize component autonomy (‘re-use’)
- normalize component interactions (‘only once’)

The clue:
- share the stable properties, localize the unstable ones
- information is what matters most, not how it is processed
- properly modeled data is stable, processing often is not
- so focus on data first and then on the processing of it
OMG-DDS: A Data-Centric Solution

Object Orientated information view
- Local object-model extending the distributed DCPS data-model
- Manages relationships and supports native language constructs

Distributed QoS-driven information management
- Fault tolerant and global persistence of selected data
- Guaranteed data availability supports application fault-tolerance
- Content-aware filtering and dynamic queries:
  - reducing application-complexity
  - improving system-performance

Real-time pub/sub messaging:
- Asynchronous ‘one-to-many’ real-time data communication
- Dynamic data-flow based on ‘current-interest’ (pub/sub)
- Platform independent data-model (IDL)
- Strong-typed interfaces for multiple languages
- Information Ownership management for replicated publishers
(4) CMS USE-CASE

“DDS by example …”
INTRODUCING: THE EXAMPLE

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

- Classifies tracks
- Determines their identity
- Analyses the trajectories
- Determines hostility
- Reports ‘trackState’

- Displays track info
- Both position & identity
- Raises alerts
- Requires ‘pointTrack’
- Requires ‘trackState’
Data Definition

- Information modeled as "TOPICS"
  - Basic units of information
  - Individually produced and/or consumed
  - Support autonomous & decoupled programs

- Each TOPIC has an associated name and data type
  - Data-definition in IDL
  - ‘Key’ fields for unique identification
  - Relational Data Model (keys)

- Our example:

```
// Topic "PointTrack"

struct PointTrackType {
    long trackId;  // key
    Position pos;
}

// Topic "TrackState"

struct TrackStateType {
    long trackId;  // key
    Id identity;
}
```
Sensor

PointTrack Publisher

Display

PointTrack Subscriber

SPLICE-DDS Real-time Information Backbone

**PointTrack**

*long trackId;
Position pos;*

*Key: trackId
QoS: best-effort, volatile*

**PointTrack Topics**

**Characteristics**
- Basic publish/subscribe data distribution
- Topics (types) specified in IDL
- QoS regarding: reliability, urgency & lifeliness

**Features / Advantages**
- Autonomous & loosely coupled applications
- Pub/Sub & QoS driven communication
- Strong-typed interfaces
- Smart networking based on priority & latency-budget
THE OWNERSHIP PROFILE

Sensor-1
Publisher-1
Strength=2

Sensor-2
Publisher-2
Strength=1

Display
PointTrack Subscriber

SPLICE-DDS Real-time Information Backbone

Characteristics
- Replicated publishers of data (with own ‘strength’)
- Only highest-strength will be received
- On failure, next highest-strength will ‘take-over’

Features / Advantages
- Fault-tolerance by replication
- Notes:
  - Requires a lot of resources
  - Quality must be expressible as an ‘integer’
THE PERSISTENCE PROFILE

Characteristics
- Built-in persistence for non-volatile data
- State preservation for transient publishers
- Settings persistence surviving system downtime
- Replicated durability service for maximal fault-tolerance

Features / Advantages
- Case-1: late-joining of Display process
  - Previously produced TrackStates readily available
- Case-2: restart of failed Classification process
  - Internal state (already classified tracks) regained
THE CONTENT-SUBSCRIPTION PROFILE (1)

Characteristics
- Adds ‘content awareness’
  - Content-filtered Topics & query-conditions
  - Supports ‘compound interest’
    - Multi-topics (combine/filter/re-arrange topics)

Features / Advantages
- Reduced application complexity
  - Automatic ‘reconstitution’ (join) of related data
- Improved system performance
  - Only receive/process what is of interest

SPLICE-DDS Real-time Information Backbone

Display

Content-filtered Multi-topic Subscriber

Thales Netherlands B.V. and/or its suppliers. Subject to restrictive legend on title page.

UNCLASSIFIED

UNCLASSIFIED

Defence
Content Filtered Multi-Topics

dataWriter (sensor)

PointTrack {
  long trackId;
  Position pos;
}
Key: trackId
QoS: best-effort, volatile

dataWriter (classification)

TrackState {
  long trackId;
  long identity;
}
Key: trackId
QoS: reliable, transient

dataReader (MyTrack)

MyTrack {
  long Id;
  Position p;
  long env;
}

Multi-topic projection
Timed-waitset or Call-backs
Parameterized-Query-Conditions

THE CONTENT-SUBSCRIPTION PROFILE (2)
**THE OBJECT-MODEL PROFILE**

**Characteristics**
- **Local** Object Oriented Data-Access Layer
- Supports ‘OO’ features:
  - Inheritance, aggregation, composition
- Uses DCPS to distribute **state** by ‘mapped topics’

**Features / Advantages**
- Ease of Management of (related) data
  - Object oriented ‘graphs of objects’ (value-types)
- Supports ‘native language constructs’ (i.e. navigation)
  - Automatic ‘change-management’ of objects

---

**SPLICE-DDS Real-time Information Backbone**

**Display**
- DLRL Subscriber

**Topics - need to be shared**
- Application
  - DLRL
  - DCPS

**Track**
- trackId

**PointTrack**
- pos

**TrackState**
- identity

---

**Real-time & Embedded Systems Workshop July’05**

© Thales Group 2005 - All Rights Reserved - Subject to re-use & republish only under written agreement.
(5) CONCLUSION

“Proven suitability…”
SUMMARY: Requirements & Realization

Requirement: Realized by:

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 runtime migration of applications
- allow applications to join the system at any time

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

- shared Information Model
- state-based information-centric system
- Information classification (QoS topic-defaults)
- minimized component dependencies
- standardized (DDS-) interaction-environment
- intuitive concept, simple/powerful features
- maximized component autonomy
- globally accessible information (data+metadata)
- focus on info-model & decoupled applications
- realtime “DDS” information backbone
- global & FT availability of transient state data
- dynamic discovery and data persistence
- de-coupled & autonomous components
- global availability of all (time-stamped) data
- highly adaptive associative data-model
SPLICE-DDS “Proven & Complete DDS implementation”

OMG main Contributions

- OMG/CORBA
  - IDL for data-definition
  - Object orientation

- SPLICE-extra’s
  - Content awareness
  - Information Management

- NDDS/SPLICE
  - pub/sub messaging
  - real-time networking

DDS Compliance-profiles

- Object-Model
- Persistence
- Content-Subscription
- Ownership
- Minimum-Profile

SPLICE-DDS compliance

- DLRL Module
  - Persistence Module
  - Cont. Sub. Module
  - Basic Module

SPLICE-DDS v1.3

(‘06)

THALES
SPLICE-DDS: *Pluggable Service Architecture*

**Computing-Node**

- **App-1**
  - Splice-lib
  - Config-Service
  - Soap-Service

- **App-2**
  - Splice-lib
  - Soap-Service
  - Network-Service

- **App-3**
  - Splice-lib
  - Network-Service
  - Durability-Service

**Shared memory**

**Disk** (XML/Binary)

**Config (XML)**

**network**

**SPLICE-DDS TUNER (100% Java)**

**SOAP**
SPLICE-DDS, a FULL OMG-DDS Implementation

- **SPLICE-DDS**, developed by **TNL**, Marketed & Supported by **PrismTech**
  - SPLICE-DDS v1.3 supports **ALL** DCPS profiles
  - Check [http://www.prismtech.com](http://www.prismtech.com) for product-details and SPLICE-DDS webcast

- **PrismTech** brings its wide experience as a middleware ISV for the marketing and support SPLICE-DDS worldwide:
  - Product evaluation
  - Ports and customisations to customer’s special environments
  - Support & training

- **Thales Netherlands (TNL)** is in charge of product evolutions, with a dedicated team of over 15 experienced engineers and a long term commitment to:
  - Continued contribution to OMG DDS specification process
  - Committed road map for full implementation of the DDS standard
  - Expert level consulting
“DDS SAILS THE SEVEN SEAS”
QUESTIONS ?????